October 2022
Geoff Huston

IPv6 Extension Headers Revisited

The topic of the robustness of IPv6 Extension Headers has experienced a resurgence of interest in recent months. I'd like to update our earlier report on this topic with some more recent measurement data.

These extensions to the basic IPv6 packet header have their antecedents in the IPv4 option fields. The IPv4 option fields included options to control loose source routing, route recording, timestamps and about 25 other functions. The notion of performing per-hop per-packet processing of packets date from a time when the processing capability within routers generally exceeded the packet presentation rates of contemporary IPv4 circuits, so spending some router processing time handling these options was not seen to represent a significant operational issue. Of course, since then kilobit per second circuits have become hundreds of gigabits per second, and the billion-fold potential increase in packet presentation rates has not been matched by a billion-fold increase in processing capabilities of routers. Some of these options represented a security risk, others were a potential drain on router resources, and it turned out that others performed no useful function at all! It's no surprise that these IPv4 options have largely fallen into disuse in the public IPv4 internet, and they are essentially unmourned in their passing!

However, this effective demise of IPv4 Packet header options in the public Internet did not stop the IPv6 design from including a similar collection of IP-level packet options, positioned in the IPv6 packet header between the common IPv6 header and the transport section. These extension headers include a set of options that need to be examined by all IP-level forwarding devices on the packet's end-to-end path (Hop-by-Hop, or HBH), options that need to be examined only by the destination of the packet (the remote "end") (Destination Option, or DST), a packet header for fragmentation control, routing directives, security options, and specialised support for the HIP and SHIM6 protocols.

In terms of the requirements of the public Internet much the same could be said of these IPv6 Extension Headers as their predecessors in IPv4. While they have their uses in limited contexts, there has no clear imperative to support these options in the public internet, except perhaps with one exception, namely the Fragmentation control extension header.

There was little in the way of testing whether these IPv6 extension header options worked within the public IPv6 Internet for many years, and it was a surprise for many in 2016 to read RFC 7872 and learn that there were significant levels of transmission failure of IPv6 packets with various forms of IPv6 Extension Headers.

The published results in RFC7872 for interactions with IPv6 web servers is summarised in Table 1.

	DST	HBH	FRAG
World IPv6 Day Data Set	11.88%	40.70%	30.51%
Alexa Top 1M	10.91%	45.45%	28.26%

Table 1 – Measured IPv6 Extension Header Drop rate from RFC 7872

The test apparently involved opening an IPv6 session to a web server, and then adding an IPv6 Extension Header into the session and seeing whether the session continued, or whether persistent packet loss caused the session to fail. They used three tests:

- The test of the DST Extension Header used a header length of a total of 8 bytes, which includes 4 bytes containing the Next Header and the Extension Header Length, and a PadN padding field of 4 bytes.

- The test of the HBH Extension Header also used a header length of a total of 8 bytes, which includes 4 bytes containing the Next Header and the Extension Header Length, and a PadN padding field of 4 bytes.

- The Fragmentation Test involved sending two fragments of 512 bytes each in size.

This work has prompted other researchers to investigate this behaviour using different test scenarios and the results of these various tests are, at the very least, confusing.

A group at the Université de Liège has recently set up a measurement platform that uses a smaller set of test nodes (21), where the measurement involves sending crafted IPv6 packets between these test nodes. with a far broader set of Extension Header variants. Their test is a modified traceroute probe, where the probe packets have the extension header, and they look for ICMPv6 hop limit exceeded messages to indicate the likely point of packet drop. They have recorded a minimum of a 91% drop rate for small (8 byte) HBH extension headers, rising rapidly to 100% as the size of the HBH extension header was increased. DST options fared far better, and they have observed that some 90% of packets with DST options reach their intended destination for DST options of 48 bytes or smaller, whereas larger options trigger a more prevalent drop behaviour with a minimum of a 90% drop rate for DST options of 128 bytes or larger.

As well as the somewhat different nature of the tested network path (probe-to-probe as distinct from client-to-server), this test is different in that it apparently measures the network ("Did this packet arrive at its intended destination?"), while the earlier exercise measured both the network and the destination host behaviour as an end-to-end supported operation ("Did the original sender receive an indication that the packet was received by the destination host and was processed as part of the data stream?").

Another group has been looking at the use of a DST option to carry end-to-end performance and diagnostics information. Their experiment used FreeBSD platforms in six locations and performed an FTP operation between these platforms with the DST option added to the IPv6 packet stream. They reported that these FTP operations "worked" indicating that the IPv6 packets with this option were successfully passed to the destination. There were no metrics given with this report.

Earlier work was presented at the IETF 108 meeting in July 2020 by researchers from the University of Aberdeen. Their work used the Alexa top 1M and use DNS queries over IPv6 in an end-to-end traversal test. It appears that the HBH and DST options tested were a PadN option, presumably of 4 bytes in length. The HBH drop rate appeared to be of the order of 80%-85%, while the DST drop rate was approximately 50%.

APNIC's Measurements

APNIC has also been conducting a measurement exercise on IPv6 extension headers, looking at the end-to-end Fragmentation, HBH and DST extension headers on networks paths toward the client side, rather than toward the server side. The measurement uses scripts within an online ad campaign and executes some 4M different measurements per day, spanning a large proportion of the IPv6 client base. A recent (March 2022) presentation on this work can be found online.

Constructing the Measurement Environment

For this measurement we are using Google's online advertisement framework to enrol end hosts to conduct the measurement experiment. This means that we have a necessarily restricted repertoire of techniques that we can use at the end host, namely being limited only those techniques that we can incorporate into a scripted retrieval of a web object. The only end-to-end component of this measurement are the HTTPS sessions used for retrieval of the web objects. To measure fragmentation impact, we need to fragment the TCP data packets being sent from the server towards the end host.

The approach being used here is to set up a conventional web server as a back-end service and set up a front end that performs packet fragmentation or insertion of extension headers on outbound packets as required. To achieve this, we will use the raw socket interface on Linux server platforms and bypass all the normal kernel and NIC card packet processing steps on the front-end system.

To allow the greatest level of flexibility, this front-end was constructed using IPv6 network address translation. This allows the two components (web server and outbound packet editing) to be provisioned on separate platforms, allowing greater scalability. Incoming IPv6 packets addressed to TCP port 80 or port 443 at the front-end server have their IPv6 and TCP headers rewritten as they are passed towards the back-end web server. The translated packet's source address is the IPv6 address of the front-end server, and the destination address is that of the back-end web server, and a locally selected port number used as the source port. This port number is the lookup key in the translator's table of active connections, so that packets received from the back end addressed to this front-end can have their addresses and port values translated back into packets that are to be destined to the original remote endpoint.

In addition to this IPv6 NAT function, the front end also performs packet editing on outbound packets. If fragmentation is being used, then all TCP packets passed across the back-end unit towards the Internet that contain a TCP payload larger than 1200 bytes are fragmented. The size of the initial fragmented packet is a random selection from range 1,200 to 1,416 bytes The fragmentation controls in IPv6 limit the payload size of initial fragments to be a multiple of 8 bytes, so the IPv6 packet sizes used here are 1,200, 1208, 1,216 and so on through to 1,416 bytes.

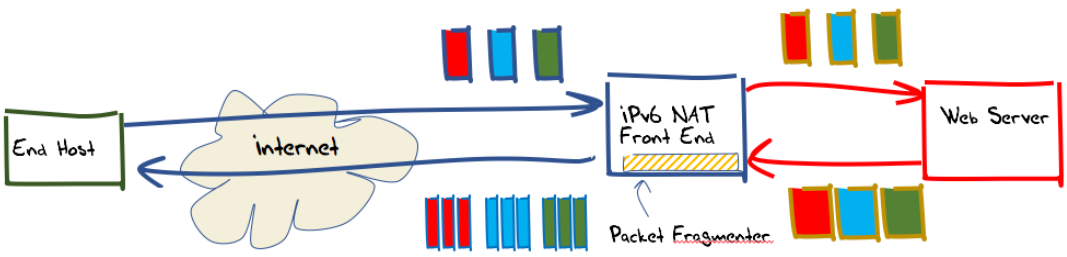


Figure 1 – Experiment Configuration

Alternatively, an Extension Header can be added to the outgoing packet. This can be either a Hop-by-Hop extension header or a Destination extension header. The size of the header is one of 8, 16, 32, 64 or 18 bytes., and the header code is the Experimental Use code of 1E. To ensure that the header can fir ono outbound packets, the incoming MSS value in the TCP handshake is adjusted by the intended size of the extension header.

Unfortunately, there are many aspects of today's Linux platforms that don't make this easy. We can't use a regular TCP socket connection on this front-end, as we are relying on a packet level interface to perform the necessary processing of the packet headers.

The *Berkley Packet Filter* (BPF) drivers and the associated *libpcap* routines allow us the necessary flexibility to pull in all incoming packets into the front-end process at this "raw" packet level, but it's not quite as easy as it may sound. Many operating systems respond to incoming TCP packets that are addressed to port numbers that have no associated active listener with a TCP reset (RST) packet. This has to be turned

off. Also, many physical interface cards are now "smart" interfaces, and rather than sending to the BPF driver exactly the packets as received on the wire, they may join a number of successive packets together and present a synthetic TCP packet that is the join of these successive packets. Again, these functions need to be disabled. The BPF drivers have no internal buffer, so the processing system that receives packets from the BPF driver is required to process the packet within the minimum inter-packet arrival times. Our solution to this was to use the Linux shared memory services to implement a buffer between the processes performing the packet arrival processing step and the NAT and fragmentation packet processing steps, extending the processing time budget closer to the average inter-packet arrival time.

The IPv6 specification requires that all network paths be capable of passing an IPv6 packet of up to 1,280 bytes without requiring packet fragmentation. What it fails to specify is the minimum fragmented packet size that an end host can receive. It appears that we can fragment almost any packet, irrespective of its size, and that implies we can fragment small packets as well as large ones. Conveniently, over at the receiver, the TCP stack should be unaware of any packet fragmentation that may have occurred. Packet fragmentation and reassembly is an IP layer function, and TCP is a byte streaming protocol. This means that our NAT unit can perform both packet fragmentation and TCP re-sequencing as required, and these packet transforms will be invisible to the remote TCP process.

The subsequent data analysis detects if the end host has received and successfully reassembled the set of fragments by looking at the front-end's packet capture log. Where an incoming TCP ACK numbers for this TCP session has an ACK sequence number that encompasses the sending sequence number of outbound fragments, then we have evidence that the remote end has successfully reassembled the fragmented packet.

IPv6 Fragmentation

The fragmentation test has been broken down into 28 different fragmentation subtests, using an initial fragment size that is randomly selected between 1,200 and 1,416 bytes. Each tested client will perform one of these tests. The average fragmentation drop rate is currently at some 7% with weekend values up to 1.5% lower that weekday values. Fragments of total size of up to 1,368 bytes show a drop rate somewhat lower than 7%. Packets of size 1,376 bytes and larger show a 7% drop rate and packets of total size 1,416 bytes show a 14% drop rate (Figure 2).

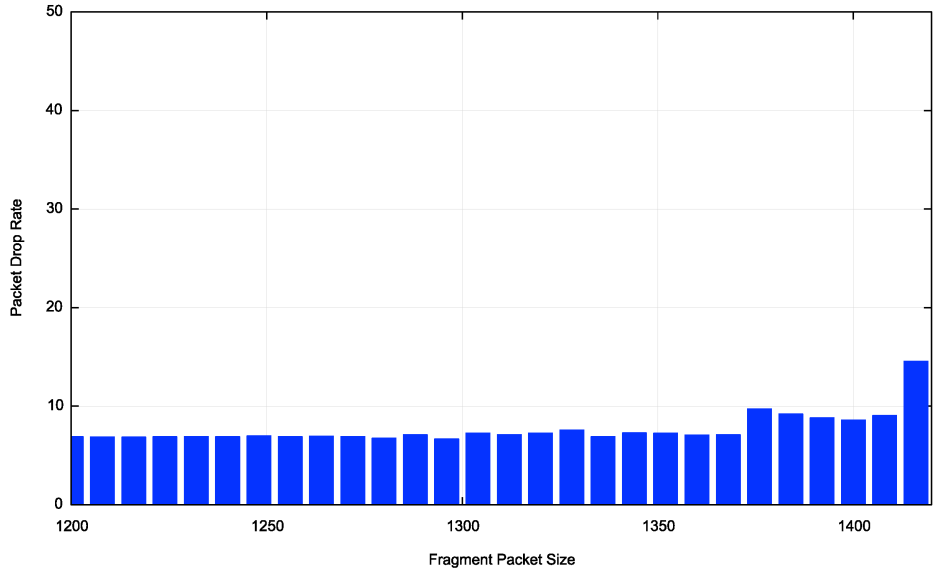
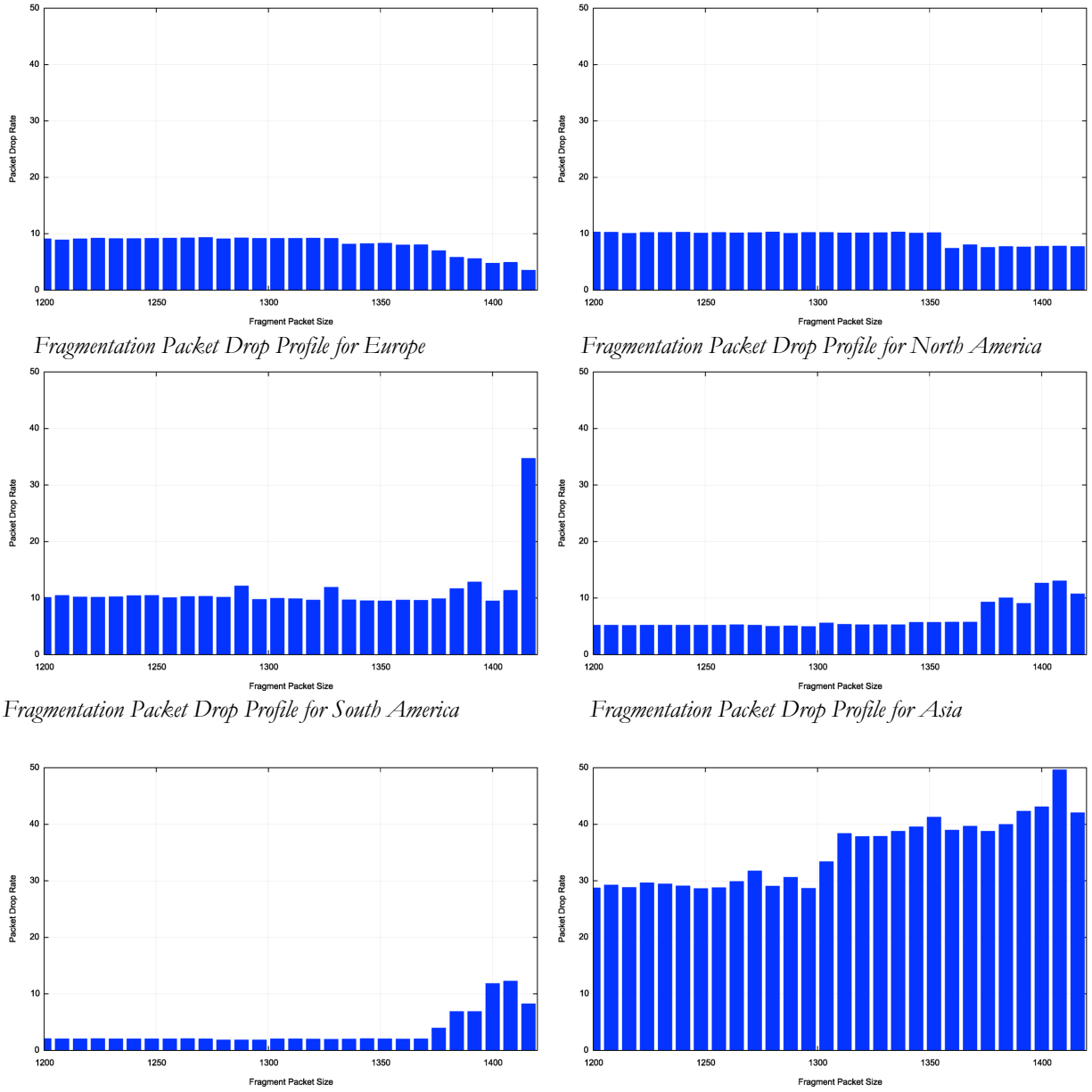


Figure 2 – IPv6 Fragmentation Drop Profile – from https://stats.labs.apnic.net/v6frag/XA

This figure suggests a uniform behaviour in the IPv6 network with respect to the drop behaviour of various sized initial fragments. This is not the case, as there are very marked regional variations here. The average for Europe and North America shows an overall higher drop rate for smaller packets, declining at 1,336 bytes and larger. The Asian profile is closer to the world average, influenced heavily by client or network behaviour in India (Figure 3)



Fragmentation Packet Drop Profile for Europe

Fragmentation Packet Drop Profile for North America

Fragmentation Packet Drop Profile for South America

Fragmentation Packet Drop Profile for Asia

Fragmentation Packet Drop Profile for India

Fragmentation Packet Drop Profile for China

Figure 3 – IPv6 Fragmentation Drop profile for various regions and economies

It is challenging to attribute these differing fragmentation pack drop profiles to a single factor. In terms of network profiles, it appears that the locales with more recent IPv6 deployment (such as India) have an overall better response to fragmented packets than Europe or North America, although the ongoing IPv6 deployment in China is an exception to this.

Larger fragmented packets appear to experience a higher drop rate in most parts of the Ipv6 network, with the exceptions being Europe and North America. It could possibly be due to a higher incidence of IPv6-in-IPv4 tunnelling in certain markets.

The other question here is whether this packet drop behaviour is due to packet handling of fragmentation per se, or packet handling of packets with Extension Headers. In an effort to differentiate between these two cases we have also looked at packet drop profiles for so-called *Atomic Fragments*. These are IPv6 packets with the Fragmentation Extension Header present, but the fields in the extension header indicate that that the entire packet is present, and no packet fragmentation has taken place. If the issue with fragmentation packet drop is the use of an Extension Header, then we would expect the atomic fragment drop rate to be high, but this is not the case then we would expect it to be low.

If we just look at the total picture the results appear to indicate that there is a Fragment Extension header problem out there, and the atomic fragment drop rate appears to be around 50% of the fragmented packet drop rate (Figure 4).

The picture swings in both directions when we narrow the focus to individual economies. Figures 5a and 5b shows the Atomic Fragment drop rate for the United States and China. In the case of the United States the Atomic Fragment drop rate is equal to, or slightly higher than the fragmented packet drop rate. In China, where the fragmented packet drop rate is high, the atomic fragment drop rate is very low. The data tends to suggest that this behaviour is a network behaviour, rather than a host behaviour.

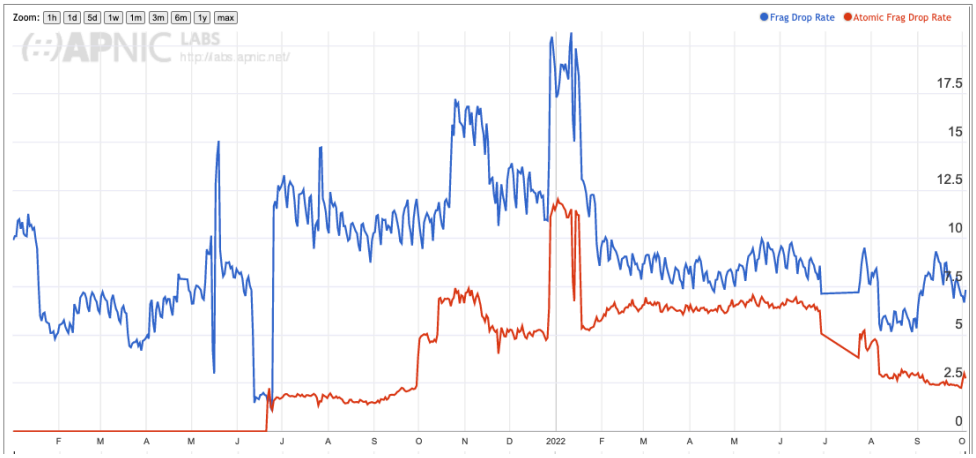


Figure 4 – Atomic Fragment Drop Rate for the IPv6 Internet

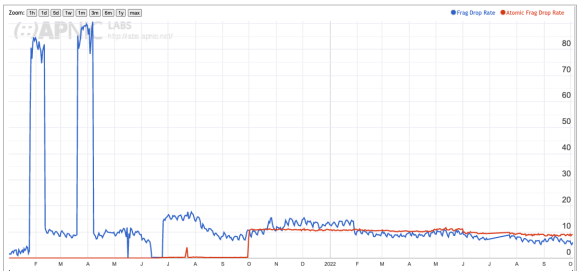


Figure 5a – Atomic Fragment Drop Rate for the US

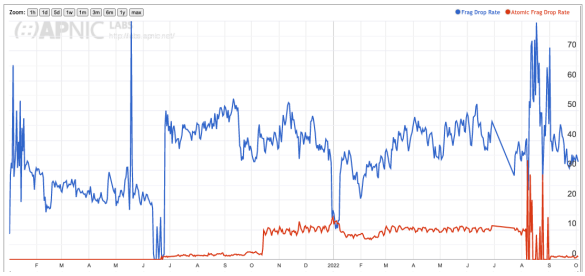


Figure 5b – Atomic Fragment Drop rate for China

The atomic fragment packet drop rate of 2.5% globally is unacceptably high, and it would suggest that IPv6 packet fragmentation is incompatible with a robust IPv6 environment. However, it is unclear whether this unacceptably poor behaviour is due to the use of IPv6 Extension Headers to carry the packet fragmentation signalling or due to the various security issues with packet fragmentation handling. The data tends to suggest that both of these factors are present in different parts of the IPv6 network.

This is an unanticipated outcome. We are used to a level of homogeneity in the network's behaviour and we are seeing something quite different here. It is highly likely that there are basic behavioural differences in the broadband access networks and mobile access networks, and also potential differences in behaviour as a result of the various IPv6 transition technologies being used.

Irrespective of the cause, the conclusion is invariant: **IPv6 packet fragmentation is best avoided in the public IPv6 network.**

Destination Extension Headers

In the first phase of this measurement work we were also testing clients using a single PadN HBH option or a PadN DST option, filling the option field with all 1's as padding. The DST option was experiencing a 94.5% drop rate and the HBH option was higher, at an average of 99.5%.

After reviewing the Université de Liège work on different drop responses for differently sized extension headers, we altered the experiment to alter the padding size, testing HBH and DST options using padding sizes of 8, 16, 32, 64 and 128 bytes, still filling the field with all 1's. The HBH drop rates were over 99% in all cases, while the DST drop rates were at 95% for all bar the 128-byte PadN option, which was at 99%.

At this stage we were pointed to the Linux source code that handles incoming IPv6 packets with Padding Extension Headers (https://elixir.bootlin.com/linux/latest/source/net/ipv6/exthdrs.c#L152)

```
if (nh[off] == IPV6_TLV_PADN) {
        /* RFC 2460 states that the purpose of PadN is
         * to align the containing header to multiples
         * of 8. 7 is therefore the highest valid value.
         * See also RFC 4942, Section 2.1.9.5.
         */
        padlen += optlen;
        if (padlen > 7)
                goto bad;
        /* RFC 4942 recommends receiving hosts to
         * actively check PadN payload to contain
         * only zeroes.
         */
        for (i = 2; i < optlen; i++) {
                if (nh[off + i] != 0)
                        goto bad;
        }
```

Whoops. That's two problems with our test right there. We've been padding with all 1's, while the Linux code base is expecting only 0's and will discard a packet with non-zero padding. Secondly, in testing larger extension headers we used larger padding fields. The Linux code uses a very strict interpretation of RFC2460, and it discards packets with padding field sizes greater than 7 bytes. This code is used in many Linux-derived platforms, including Android, so its perhaps unsurprising that the drop rates we've been seeing have been so high. If we want to test larger extension headers, then PadN using 1's is clearly the wrong choice.

We've changed the test to use 0x1E, an experimental code point as listed by the IANA (https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml), and we fill the field with all 0's.

Firstly, let's look at the Destination Header drop rate when we switch over from using PadN to the Experimental code point. It is an open question as to what constitutes a safe behaviour option for the network. Forwarding on a packet where the router does not 'understand' the packet contexts in the IP header could be construed as promulgating a potential problem. On the other hand, a destination option is not a network problem, and networks could confidently expect that hosts need to fend for themselves.

Figure 6 shows a 'before and after' picture of the change from PadN to the experimental code point for the Destination Header option. The experiment used 5 extension header lengths (8, 16, 32, 64 and 128 bytes), and in our case the 8, 16 and 32 byte headers had the greatest success rates, while the two larger sizes experienced greater drop rates. There is nothing obvious in the Linux source code that could explain this behaviour, unlike the PadN issues. That tends to indicate that the size-related differential response for DST Extension header handling might be due to network equipment behaviours rather than host platform behaviours.

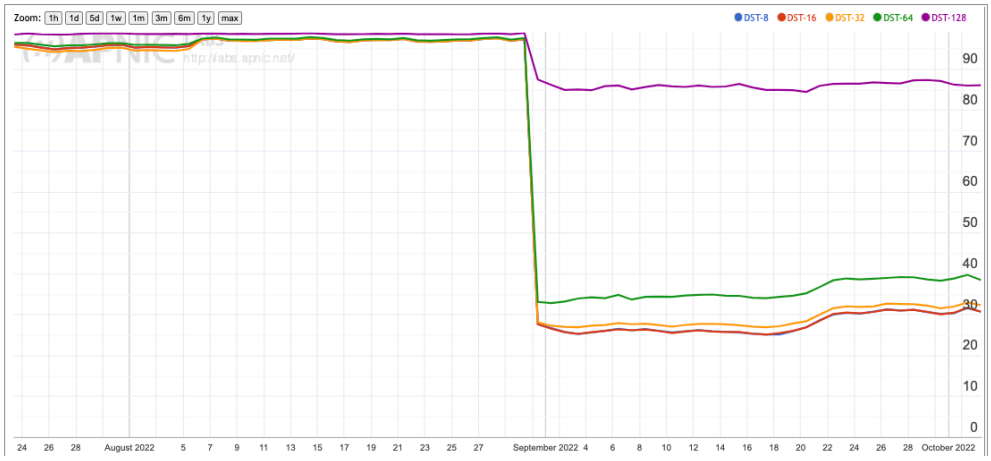


Figure 6 – DST option Extension Header Drop Rate

Once more national profiles vary greatly here. A good example of a very low drop rate for the Destination Extension Header is Greece, with a drop rate of below 7% for all bar the largest extension header option (Figure 7a), while the drop rate in the United States remains above 70% for all sizes (Figure 7b).

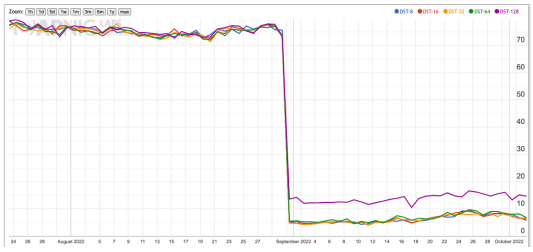


Figure 7a – DST option Drop Rate for Greece

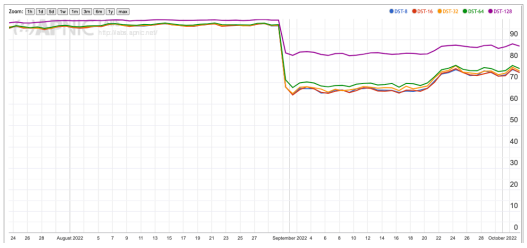


Figure 7b – DST option Drop Rate for the United States

The conclusion from this data is that there is no single Destination Option drop rate for the public IPv6 Internet that we can observe. Different regions of the public Internet exhibit very different responses to IPv6 packets with Destination Options, as shown in Figure 8.

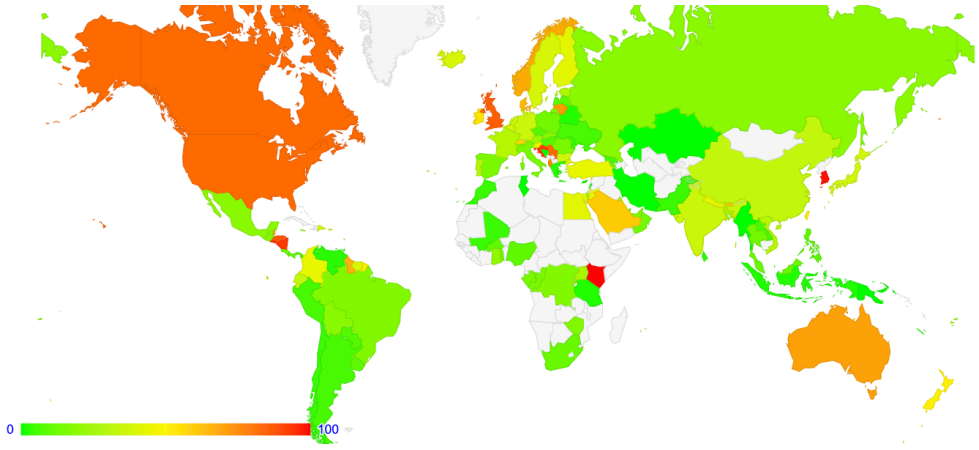


Figure 8 – DST Option Drop Rate – October 2022

If you are looking for a general rule of thumb about what makes a robust IPv6 service on the public Internet then firstly large Destination Extension Header options fare somewhat more badly than smaller ones, but small ones do not work that well either!

The highly level of variance on observed behaviours tends to lead to the same conclusion as the Fragmentation Extension Header: **IPv6 Destination Option Extension Headers are best avoided in the public IPv6 network.**

Hop-by-Hop Extension Headers

In the first phase of this measurement work we were also testing clients using a single PadN HBH option, filling the option field with all 1's as padding. The HBH option was experiencing an average packet drop rate of 99.5% across all HBH option sizes.

As with the Destination Option header we changed the HBH option to use the same experimental code point, 0x1E, and used all zeros as the code value.

In this case the change in the code value and the change to the value contained in the field had no appreciable effect on the measured drop rate (Figure 9).

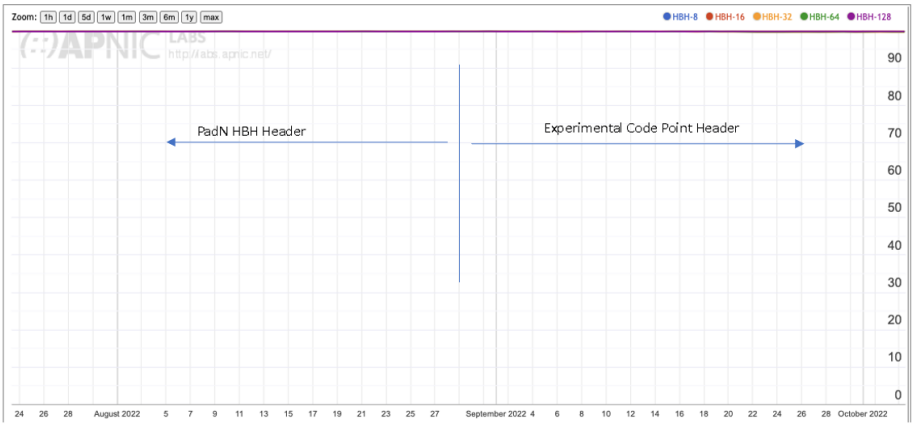


Figure 9 – IPv6 Packet drop with HBH Headers

The global map of drop rates shows that this comprehensive packet drop for HBH headers is experienced in a relatively uniform manner across most of the IPv6 network (Figure 10).

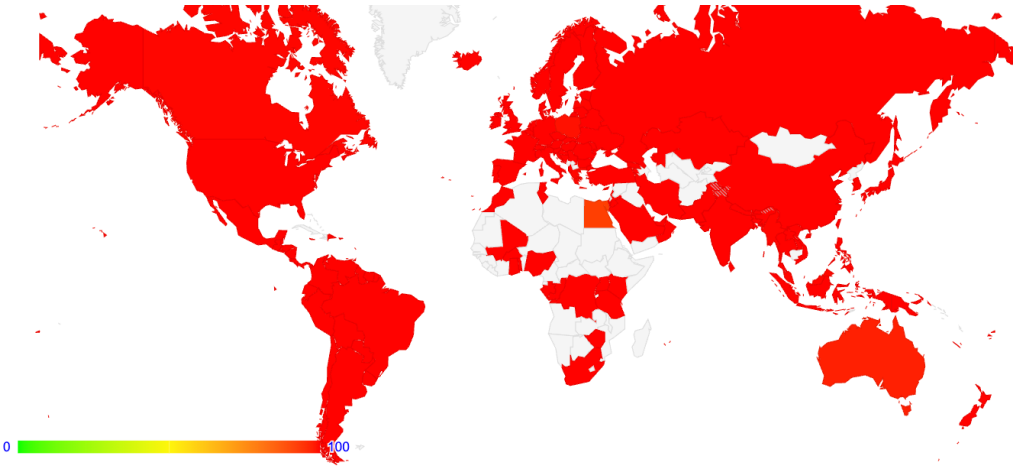


Figure 10 – HBH Option Drop Rate – October 2022

This result differs markedly from RFC7872, where the HBH drop rate for a header using PadN with a total length of 8 bytes was reported as 41% - 45%, whereas we are seeing a 99% or greater drop rate for all HBH options. We altered the 8 byte HBH header to use PadN, and still saw no change in the drop profile. So why do we see an almost comprehensive drop rate for HBH packets whereas the earlier work saw a drop rate slightly under one half of all tests.

There are is one significant difference here between the two measurements. This is the relative direction of the HBH packets. In the case of the APNIC measurement the packets are passing towards the client devices, traversing the IPv6 access networks to reach the destination, while the RFC7872 measurement was directed to service platforms, which are generally connected within the core of the network infrastructure rather than at the edge of an access network. In our case the packets are generated on virtual servers hosted by Linode, so the packets enter the network at the core rather than at the edge. So, the APNIC measurement is a "core-to-many edges" measurement. RFC7872 does not record the source of the packets. It is not specified whether the results were obtained using packet generator located in the core of a network or on an edge-connected platform, but it seems reasonable to assume that this measurement was performed on servers located in the core of the network rather than from hosts attached to one or more access networks. The RFC782 measurement appears to be the result of a "core-to-many-cores" measurement. If HBH drop is more prevalent in edge networks, then this could go some way to explain the different results we see here.

Packets with HBH Extension Headers, in this case the padding and experimental code points, are observed to be universally discarded in our measurements. The conclusion is the same as the Fragmentation and Destination Option Extension Headers: **IPv6 Hop by Hop Option Extension Headers are best avoided in the public IPv6 network.**

Network or Host?

In this measurement framework we can only instrument one side of the connection, namely the sender. We cannot instrument the end host, and therefore cannot tell is the packet loss is due to the receiving IPv6 protocol stack rejecting the incoming packet, or the network rejecting the packet and never attempting delivery.

It's likely that fragmentation drop occurs close to the client edge of the end-to-end path. IP fragments are generally regarded as a security vulnerability and the CPE firewall configurations are often configured to drop fragmented packets. A similar stance is seen in mobile deployments, where the mobile provider is providing the equivalent of the edge firewall function. The higher loss rates for larger fragments may possibly be due to the particular IPv6 transition mechanism being used by the access provider, although we have no clear measurements to substantiate this assumption.

The Destination Option drop rate is probably due to both the network and the host. The resources allocated for router functions performed by hardware are intentionally limited, packets are passed to a software switching path when the extension header is greater than 64 bytes for some router platforms, and slightly lower for other platforms. This router behaviour would explain the very high drop rate for 128-byte DST options shown in Figure 5.

There is also a significant variation in the baseline drop rate for DST options between 8 to 64 bytes in size. It is likely that this may be attributed to the differences in the network equipment used to support IPv6. In the US Comcast (AS79223), a broadband cable-based provider shows a 100% drop rate for 128-byte options, and an 80% drop rate for DST options of size 8 to 64 bytes (Figure 11).

DST Drop Measurement for AS7922: COMCAST-7922

Figure 11 – HBH Option Drop Rate for Comcast (AS7922)

The predominately mobile service provider, T-Mobile (AS21928) has a completely different drop profile, which a 35%-45% drop rate for all DST option sizes (Figure 12).

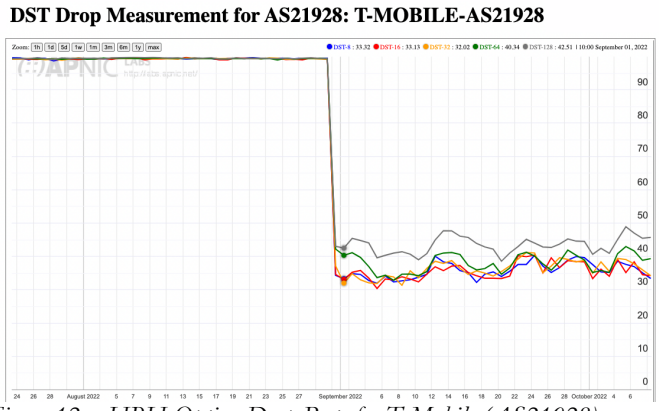


Figure 12 – HBH Option Drop Rate for T-Mobile (AS21928)

The inference from these two figures is that the DST packets, including the 128-byte DST option packets are making it out of the Linode data centres that house the servers for this measurement, and are being passed to the access ISPs, but the various IPv6 equipment used by the various access ISPs have different packet drop profiles.

The Hop-by-Hop profile is one of close to uniform drop. Close, but not complete. The measurement from AS36992 in Egypt (Figure 13) shows an 80% drop rate for all but the 16-byte HBH header, which is somewhat higher at 95%.

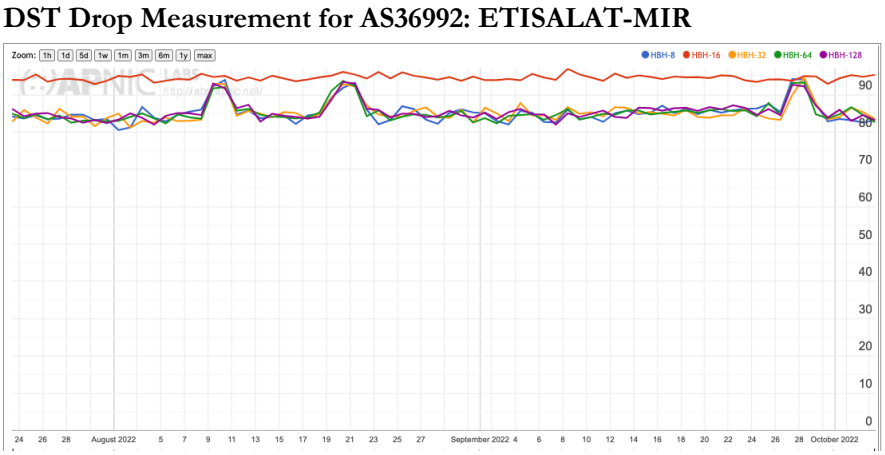


Figure 13 – HBH Option Drop Rate for ETISALAT-MIR (AS36992)

This shows that at least some packets with HBH options are leaving the Linode server data centre, and that the HBH packet drop appears to be occurring elsewhere in the network, at least in part. It is not known why the 16-byte header experiences the higher drop rate.

Conclusions

In closing, however, it should be remembered that we are using a "core-to-many-edge devices" form of measurement here, and the drops we are seeing appear to be related in many ways to to the edge access networks and the edge devices, and in both cases the access networks and client hosts have a significant proportion of mobile access. In this case it appears that the particular IPv6 transitional technology used in the mobile access network matters a lot. This would not be a significant factor in other measurements of EH loss, particularly in the "core-to-core" scenarios and even in the "core to broadband-access-edge" scenarios.

More generally, we are used to measuring the Internet as a largely homogenous environment. Centrality in so many aspects of the Internet means that there is little in the way of diversity of technology and

diversity of behaviour. Edge devices are generally based on a Linux operating system (Android is highly dominant here), and client browsers are predominately Chrome. Internal switching devices tend to be sourced from Cisco or Juniper or behave in a manner that is very similar to units supplied by these two vendors. From this perspective, the assumptions about homogeneity in the network appear to have some level of justification. However, the IPv6 situation is nowhere near as uniform. Access networks can support full independent dual stack operation or can encapsulate IPv4-in-IPv6 or IPv6-in-IPv4. Other access networks can perform various permutations of translation and encapsulation. What this implies is that different measurements of EH drop can see quite different results. Smaller scale measurements see a more limited set of network behaviours, while larger scale measurements tend to see an averaged result. The variance in behaviours that are observable shows that Extension Headers in the public IPv6 network leads to the conclusion that no developer can count on the coherency and utility of IPv6 Extension Headers in the public IPv6 Internet when looking across the complete environment of servers and clients.

Selective sub-environments might show more consistent results, but in general the conclusion that we draw from these results are: **IPv6 Extension Headers are best avoided in the public IPv6 network.**

Acknowledgement

Setting up the IPv6 Extension Header testbed has been an extensive software development effort undertaken over a number of years. I'd like to acknowledge the work of my colleague, Joao Damas, in maintaining this testbed code and adding the various tweaks and refinements needed to help us refine our understanding the behaviours of networks and hosts in responding to IPv6 packets with Extension Headers.

Disclaimer

The above views do not necessarily represent the views or positions 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