April 2021
Geoff Huston,
Joao Damas

IPv6 Fragmentation Loss

Committees should never attempt to define technology There always seems to be a point in the process where there is a choice between two quite different options, and there is no convincing case that one choice is outstandingly better than the other. Committees find it terribly hard to decide at this point, and this can lead to stupid, or even fatally flawed outcomes. Look at the definition of the cell size in ATM. Who would've thought that 53 was a sensible answer? Even if you recognise that the answer is constructed from cells that have 48 bytes of data and 5 bytes of cell header it's still a silly number. Why not 32 bytes of payload? Or 64 bytes? It was said that the design committee couldn't reach a clear answer between these two options and decided to define an outcome halfway between the two. At least, goes the story, this was an equally unsuitable answer! Or in the OSI process there was no clear consensus whether the transport service should be based on virtual circuits (connection-based) or stateless packet forwarding (connectionless). So, the design committee picked both at once. Obviously these two transport protocols cannot interoperate. The result? Right down at the very fundamentals of the OSI protocol stack was this schism. This was not perhaps the only reason why OSI failed, but it was surely a big factor in its failure.

So why are we pondering the weaknesses of a process of technology-design-by-committee? In the Internet context, and the IETF in particular, there are many contenders for similar committee-think brokenness. The routing security framework that is built upon RPKI, and DNSSEC are both good examples where there are some fundamental flaws that in all likelihood are an outcome of the committee process used by the IETF where the working groups were unable to make a set of consistent and coherent design decisions. A more significant example of this weakness is IPv6. In retrospect it's hard to understand how we managed to get it wrong. IPv6 was intended to be "just like IPv4, but with 128-bit source and address fields". Just make IP addresses 96 bits longer and add no more magic. But it didn't turn out that way. The design process made a few more changes, and the bits that were changed are still causing confusion and mishaps. The variably sized IP options from the fixed IPv4 packet header were removed and it its place was created an explicitly labelled options section (also variably sized) between the basic IPv6 packet header and the end-to-end transport protocol header. This new section, which was optionally present in IPv6 packets, was called the "Extension Header Set". So far this looks like a pretty cosmetic change. However, the Fragmentation Control section, which was present in all IPv4 packets as part of the fixed header, was redefined as an optionally present Extension Header. And the IPv4 "don't fragment" control bit was removed altogether! That does not mean that IPv6 packets cannot be fragmented. They can. But what is does mean is that IPv6 packets cannot be fragmented on the fly by routers. Only the packet source can generate fragmented IPv6 packets.

There are a couple of operational issues with this approach that continue to prove problematical for operational IPv6 deployments. The first is the presence of variable sized components of the IP packet header. When we optimise routers for very high packet processing capacity it's a challenge for the router to unravel variably sized header chains in a fixed processing time budget. In IPv4 the common approach used by routers is to use the fast-processing path for IPv4 packets that do not contain IPv4 options and pass all other IPv4 packets to a slower path that will unravel the IPv4 options. Perhaps this behaviour is not noticed these days because we stopped relying on IPv4 options decades ago! When we come to IPv6 we have a similar issue. IPv6 packets with extension headers may be treated as second class packets that have slower travel times through the network with a higher drop probability.

This issue was studied in 2015 and the results of a measurement exercise was published as RFC 7872 ("Observations on the Dropping of Packets with IPv6 Extension Headers in the Real World," published in June 2016). At that time the packet drop probability for IPv6 packets with extension headers was measured at rates between 11% and 55% of cases. For the subset of cases where the Fragmentation Extension Header was used the drop rate was measured between 28% and 55%.

It's not just the use of Extension Headers to contain IPv6 Fragmentation Control parameters that was the problem here. The entire issue of IP fragmentation in IPv6 was being questioned at the time, and an earlier internet-draft document ("IPv6 Fragment Header Deprecated", from 2013 and available at https://tools.ietf.org/html/draft-bonica-6man-frag-deprecate-02) describes some related thoughts on the issues that fragmentation presents within the protocol.

At APNIC Labs we've looked into the robustness of IPv6 Fragmentation Extension Headers and drop rates a few times and look at both end-to-end drop rates in TCP sessions and UDP-related drop rates in the context of DNS responses. (https://www.potaroo.net/ispcol/2017-08/xtn-hdrs-2.html).

This 2017 result measured an IPv6 Fragmentation drop rate for TCP packets at 21%, which prompted the challenging conclusion that "Whatever the reasons, the conclusion is here is unavoidable: IPv6 fragmentation is just not a viable component of the IPv6 Internet."

In this report I would like to revisit this measurement of packet drop for IPv6 Fragmented packets and see if the picture has changed over the intervening four years.

Much has happened over this time. The number of users with IPv6 capability has risen dramatically, from an estimated 300M IPv6-capable users at the start of 2017 to 1.1B IPv6-capable users in April 2020. Not only has the user base quadrupled, the deployment of IPv6-capable network infrastructure has expanded in a comparable manner. The question here is whether today's IPv6 infrastructure shows the same issues with packet drop of IPv6 packets that have fragmentation headers that we saw in 2017? Or, has the situation improved?

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 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 'mangling') 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 fragmentation. All TCP packets passed across the back-end unit towards the Internet that contain a TCP payload larger than 1200 octets are fragmented. The size of the initial fragmented packet is a random selection from range 1,200 to 1,416 octets The fragmentation controls in IPv6 limit the payload size of initial fragments to be a multiple of 8 octets, so the IPv6 packet sizes used here are 1,200, 1208, 1,216 and so on through to 1,416 octets.

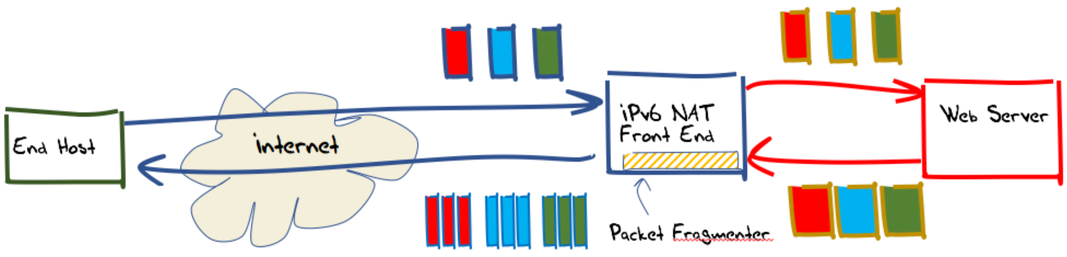


Figure 1 – Experiment Configuration

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 function of this NAT front-end can be described in a couple of rules:

- For TCP packets received from the Internet, if a translation table exists that maps the triplet of the source address, source port and destination port in the packet to a local port value, then replace the source address of the packet with the local host source address, replace the destination address with that of the back end, replace the source port with the local port address and send the packet out to the back end. If no translation table exists, and the packet contains a TCP SYN flag, take the oldest translation table entry and re-use the local port value. Otherwise drop the packet.

- For TCP packets received from the back end, the processing is similar. The source and destination port numbers are used to look up the translation table. If a translation table entry is found, then the packet's destination address and destination port is replaced with those contained in this entry, and the source address is replaced with the local address. If the packet length is greater than the fragmentation threshold then packet is broken into a number of outbound TCP packets, and the initial fragment size is chosen at random from the possible set of initial packet sizes.

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.

How "real" is this Experiment?

Before looking at the results, it may be useful to ask whether this experiment represents a "real" scenario that is commonly encountered on the Internet.

It's certainly the case that in TCP over IPv6 we don't expect to see packet fragmentation. A TCP sender should ensure that all outbound TCP segments fit within the local interface MSS size, so in the absence of network path MTU issues a sender should not be fragmenting outbound TCP packets before sending them.

What about the case where the path MTU is smaller than the local interface MTU? When a packet encounters a network path next hop where the packet is larger than the next hop MTU, then the IPv6 router constructs an ICMPv6 Packet Too Big message, noting the size of the next hop, and also including the original packet headers as the payload of this ICMPv6 message. It sends this ICMPv6 diagnostic message back to the original sender and discards the original packet. When a sending host receives this ICMPv6 message it also has the TCP packet header. This information can be used to find the TCP control entry for this session, and the outbound MSS value of this TCP session can be updated with the new value. In addition to the updated size information, the TCP header in the ICMPv6 message payload also contains the sequence number of the lost packet. The sending TCP process can interpret the ICMPv6 message as an implicit NACK of the lost data, and resend the discarded data, using the updated MSS size. Again, no packet fragmentation is required.

All this sounds like a blatant case of "layer violation" and we should call in the protocol police! But before we do so, maybe we should think about the hypothetical situation where the host did not pass the Packet Too Big message to the TCP control block.

This is analogous to the case where the ICMPv6 Packet Too Big message is not passed to the host at all, where, for example, some unhelpful piece of network filtering middleware is filtering out all ICMPv6 messages.

In this case, the sending TCP session has sent a TCP segment and is waiting to receive an ACK. The receiver will not get this packet, so it cannot ACK it. The sender might have a retransmission timer and it might try to resend the offending large packet, but that too will get lost, so it will never get the ACK.

This results in a wedged TCP state, or a "Path MTU Black Hole" condition.

Hiding ICMPv6 Packet Too Big messages from the TCP controller, either because of local processing rules within the host or because some network element has decided to drop them, is invariably harmful!

In that sense, we have constructed a somewhat "unreal" experiment, and we should not expect to see applications that critically depend on the correct working of packet fragmentation in TCP experiencing the same network conditioned as those we've set up here.

On the other hand, fragmentation is an IP function, not a function performed by an end-to-end transport protocol, and the question of whether a host can receive a fragmented UDP packet is essentially the same question as whether a host can receive a fragmented TCP packet, at least from the perspective of the host itself. In both cases the real question is whether the IPv6 process on the host can receive fragmented IPv6 packets.

In summary, while the experiment itself uses conditions that are essentially an artifice, the result, namely the extent to which IPv6 Extension Header drop occurs when passing fragmented IPv6 packets towards end hosts, is nevertheless a useful and informative result.

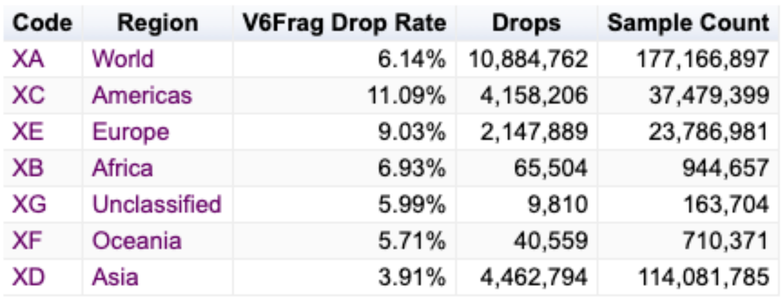
IPv6 Fragmentation Loss Rates

We observed a V6 fragmentated packet drop rate of **21%** in 2017.

What do we see in 2021?

The bottom-line answer is that current average drop rate for IPv6 Fragmented packets is **6%**. Obviously, this is a considerable improvement over a relatively short period of time.

There are considerable levels of variation across IPv6 networks and across geographical regions. Figure 2 shows the average drop rate across the various regions.



Code	Region	V6Frag Drop Rate	Drops	Sample Count
XA	World	6.14%	10,884,762	177,166,897
XC	Americas	11.09%	4,158,206	37,479,399
XE	Europe	9.03%	2,147,889	23,786,981
XB	Africa	6.93%	65,504	944,657
XG	Unclassified	5.99%	9,810	163,704
XF	Oceania	5.71%	40,559	710,371
XD	Asia	3.91%	4,462,794	114,081,785

Figure 2 – IPv6 Fragmentation Drop Rate per region in 2021

We can map loss rates to geography and build a world map of average loss rates per economy. This map is shown in Figure 3. In this case the map considers all loss rates greater than 10% as undesirable and uses red, and the colour keying shifts towards green as the national average loss rates get closer to 0%. This does not take into account the number of IPv6 users in each economy, but simply looks at the fragmented packet loss rate. There are some surprises in this, where one of the larger national IPv6 deployments is India, which has a very low loss rate for the larger ISPs (under 2%). The other major IPv6 deployment in this region is in China, and here the average loss rate is some 20%. Again, there is some variation here, and one of the larger providers is AS4808 with a 60% loss rate, with some other larger providers showing loss rates between 20% to 30% while others are under 10%. The parts of the map in Figure 3 that are not coloured have insufficient IPv6 samples to perform a measurement, which applies generally to Africa and Central Asia.

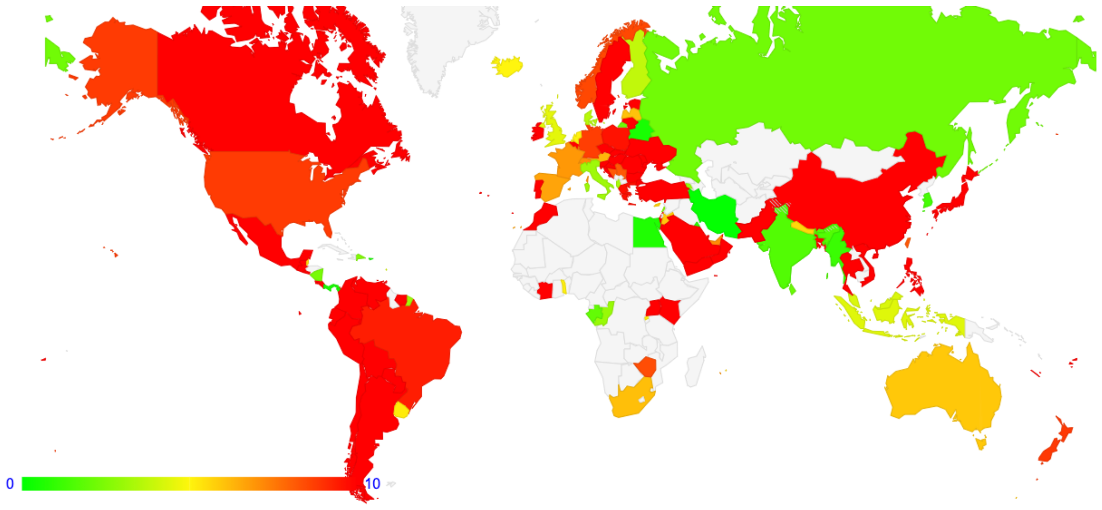


Figure 3 – Distribution of V6 Frag Drop Rates per Economy

There is also a considerable variance across various IPv6 service providers within each national context. For example, users within AS852 in Canada experience a IPv6 fragmentation drop rate in excess of 80%, while users in AS812 in Canada experience a drop rate of below 8%.

It is possible that the variation in the ability to handle fragmented packets is related to the IPv6 transition technology being used in various networks, but within the scope of this measurement exercise we have no direct knowledge of which networks use which transition technology, so we are unable to substantiate or contradict this supposition.

We were interested to understand if the fragmentation loss is related to some form of packet size constraint. If this were the case, then we would observe that larger fragments would likely have a higher loss probability than smaller fragments. If the issues are based around the use of the fragmentation extension header, or fragmentation itself then the loss rates would be the same across the range of packet sizes. We measured fragmentation loss across 28 different fragmented packet sizes, in the series from 1,200 octets, 1,208, 1,216, … and so on through to 1,416 octets. The plot of the loss rate against packet size is shown in Figure 4.

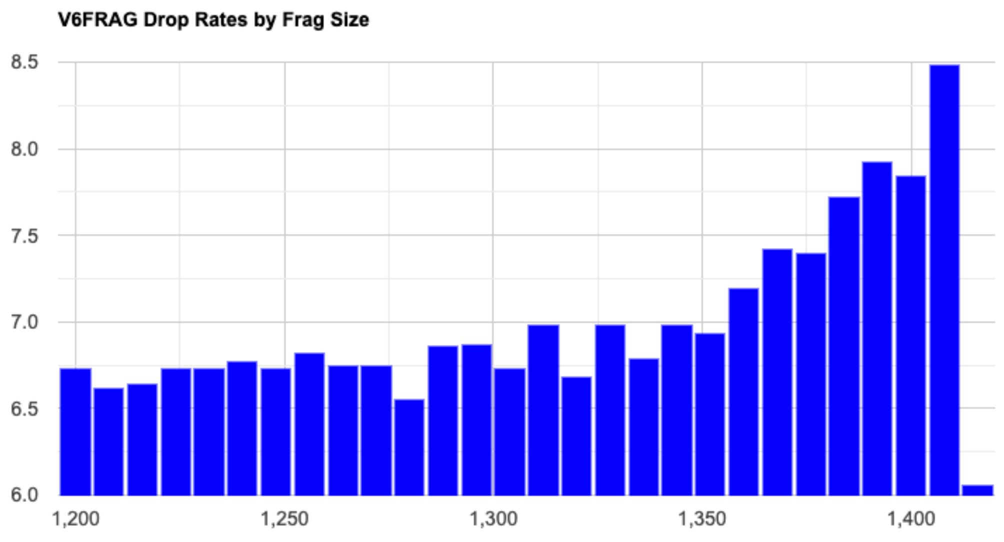


Figure 4 – Fragmented IPv6 Packet Loss Rates vs Packet Size

The data shows a clear rise in the loss rates for packets of size 1,360 octets and larger, rising to a peak at 1,408 octets. We have no explanation why the measurement for packets of size 1,416 octets is anomalously low, but we suspect that this is an artefact of the experiment configuration. The rise in loss rates for larger packets is itself quite curious. We do not see the same pattern across most of the regional data, and this pattern is only evident in Asia. In this data set IPv6 is most prevalent in Asia, with some 60% of measurement points coming from Asia (notably India and China), so it is not surprising that data patterns observed in Asia (Figure 5) are visible in the global data as well.

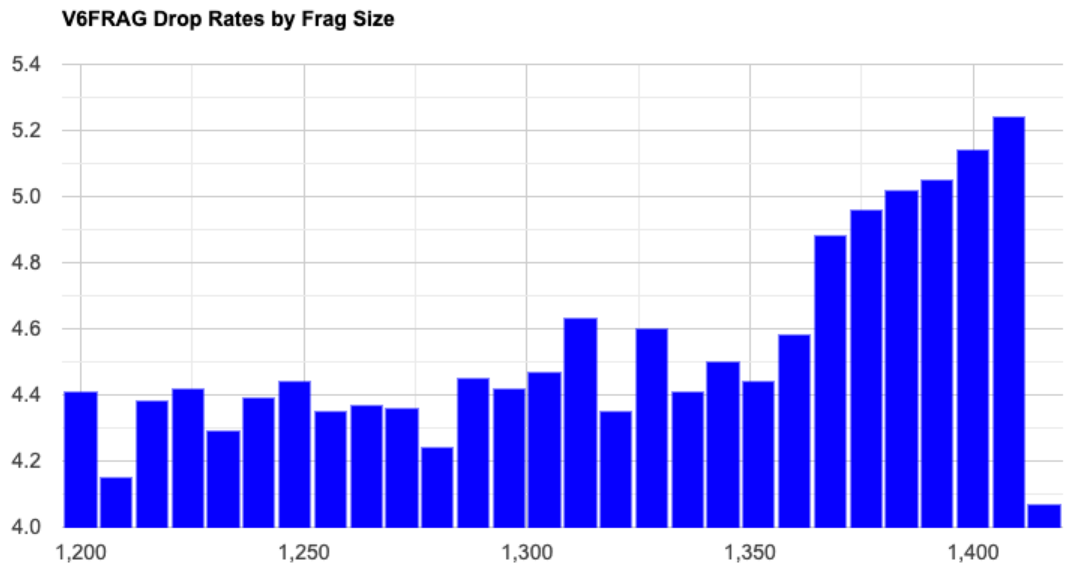


Figure 5 – Fragmented IPv6 Packet Loss Rates vs Packet Size for Asia

In comparison Figure 6 shows the loss distribution for South America, where there is a more uniform loss distribution across the various packet sizes.

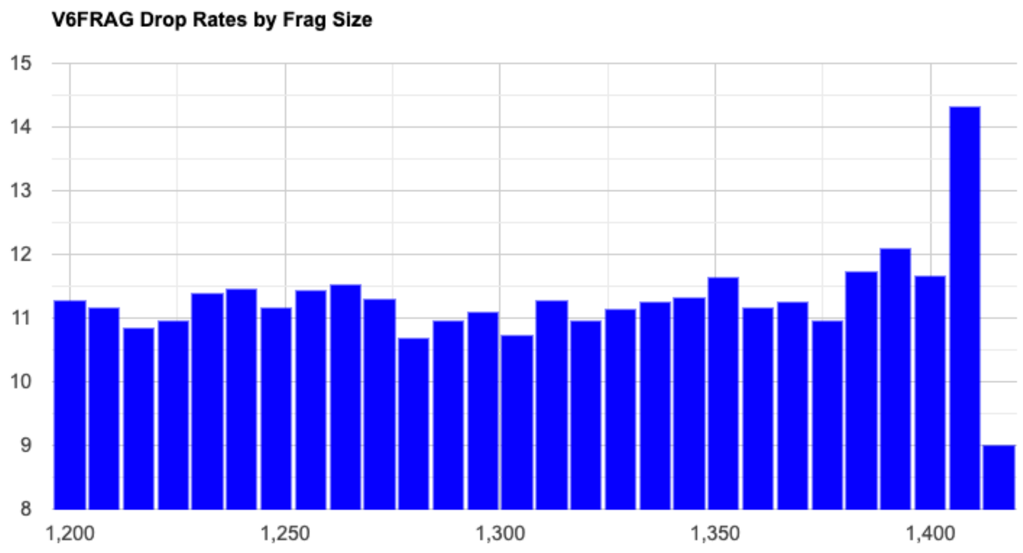


Figure 6 – Fragmented IPv6 Packet Loss Sates vs Packet Size for South America

Why are we seeing these drop rates for fragmented IPv6 packets? There are a number of potential factors at play here:

- Firewalls are often configured to drop fragments. The issue is that the trailing fragments do not contain a transport header, and the conventional port-based filter rules can't be applied to such fragments. Either the firewall has to accept all such fragments (which is a security risk), drop all fragments (which can compromise services) or has to perform packet reassembly itself to apply the filter rules. Firewall drop rules are likely to be size-independent.

- Network equipment may be configured to drop all IPv6 packets that contain Extension Headers. This could potentially be the case in those networks where are have observed fragment drop rates in excess of 90%.

- Network equipment may pass packets with Extension Headers to a "slow path" processing, and this may have an associated queue buildup and cause sporadic loss.

- There may be path MTU issues where larger packets are being dropped. Given that we are seeing this drop rate occur for packets smaller than 1,416 octets in size it is unlikely that this is the outcome of simple IP-in-IP encapsulation

This list is probably not complete and other factors may also be at play here, but the common observation is that within the parameters of this particular experiment, we cannot readily differentiate between them and diagnose the primary causes of packet loss. The widely differing result for different networks tends to suggest that some networks and some client systems have different loss causes.

Online Report Tool

In other APNIC Labs measurement experiments we've found that a comprehensive reporting tool that can report on global, regional, and national numbers and then drill further down to the individual network by its Autonomous System Number has proved to be very useful. Accordingly, a similar report framework has been published at https://stats.labs.apnic.net/v6frag to report on packet loss rates for fragmented IPv6 packets.

The data collection for this measurement commenced at the start of 2021, and we would like to keep this measurement running for some time to see if the overall situation with respect to IPv6 Fragmentation packet loss improves further in the coming months.

Disclaimer

Authors

Geoff Huston AM, M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

Joao Luis Silva Damas is the Senior Researcher at APNIC.

www.potaroo.net