June 2020
Geoff Huston

Measuring IPv6

It's now the season of virtual workshops, and without the overhead of time spent travelling to these events it's been possible to participate in a number of these events all over the Internet in the space of a few days.

This week I participated in a workshop on measurement of IPv6, organised by the US Naval Postgraduate School's Centre for Measurement and Analysis of Network Data (CMAND) and the folk at UC San Diego's Center for Applied Internet Data Analysis (CAIDA) [https://www.cmand.org/workshops/202006-v6/].

IPv6 is now a 25-year old technology, and the approaches to measurement of this new network protocol have changed over the years. The first efforts were to try and see any use at all in production networks, which was a challenge given that almost all of the early efforts to get IPv6 out of the lab on into public networks were based on tunnels and overlays. We then progressed to measuring the performance of these tunnels and overlays and the experience was completely disheartening. There is a general rule that each generation of network engineers evidently needs to learn first-hand: tunnels just don't work! Then we moved on to the early production deployments, looking a decade ago at the early deployment efforts of IPv6 by ISPs in Japan, Belgium, Slovenia, and Romania. Then we observed uptake from larger consumer and enterprise ISPs in the United States and Europe, and also within the various content data networks. It's by no means the end of the story, and while there is a lot of IPv6 around, it's not distributed uniformly. Around a quarter of the Internet's user population that can use IPv6 are likely to be located in India, the United States, China, Brazil or Japan. These five economies host 72% of the entire IPv6-capable user population in today's Internet!

The IPv6 measurement story is so much more than just counting users and mapping deployments. How similar are the IPv4 and IPv6 networks in terms of internal topology? Are the two protocols handled identically within service provider networks, with the same load balancing and traffic management systems? Or are they treated differently? Do the two protocols offer similar performance? How is the dual-stack network performing? The Happy Eyeballs approach of both preferring the use of IPv6 while offering a very fast failover to IPv4 is a critical component of the entire transition process. Does Happy Eyeballs work as intended? And what happens at the end of this protracted transition, and when do we know we have reached such a point? Are IPv6-only networks viable yet, or is it still only a prospect in a distant future?

A three-hour online workshop was nowhere near enough time to explore these topics in any detail, but it certainly was very useful in terms of understanding what IPv6 measurement activities are being undertaken these days and what areas might benefit from further attention.

Load Balancers in IPv6

Load balancing in IPv4 networks is very commonplace. Spreading traffic across a number of discrete packet paths can immediately solve the practical problem of having too much traffic to squeeze comfortably into a single path without chronic congestion loss. When performed with care and attention load balancing can increase the overall network capacity and carriage efficiency.

We can detect load balancing through careful analysis of traceroute reports. By using a collection of traceroute servers and tracing to a large collection of destination addresses it is possible to detect points where the packets are passed through load balancers. Even more careful probing can detect the parameters used by these load balancers to perform traffic segmentation.

In an experiment conducted by researchers at UFMG, Italo Cunha reported to the workshop that three quarters of the paths they examined in a large-scale study of the IPv4 Internet had one or more load balancers and there were a visible number of paths with more than 10 load balancers! Applying the same discovery technique to the IPv6 Internet showed that IPv6 does not have the same density of load balancing deployment. Just over one half of the IPv6 paths had one of more load balancers and 80% of these IPv6 paths had 3 or fewer load balancers. Does fewer load balancers imply relatively less IPv6 traffic in a Service Provider (SP) network?

There are a number of ways to divide traffic across multiple paths. There is a strong preference not to place packets that are within a single TCP session across multiple paths, as this would increase the likelihood of packet re-ordering within the TCP flow and out-sequence packets can be disastrous for TCP protocol performance. One way to preserve packet order is to use the 5-tuple of source and destination IP address and port number, plus the protocol value and use this as the hash key for load balancing. This way all packets in the same TCP flow get passed along the same path. A simpler approach, but potentially coarser method, is just to use the destination IP address as the hash key. The load balancer can also infer an application from the port numbers and load balance on that information. And, of course, the IPv6 protocol header specification has reserved 20 bits for the flow label which was meant to provide an externally visible indicator for flows, allowing the network to perform load balancing yet preserve flow integrity.

Do IPv6 networks use the IPv6 flow label for load balancing? This study shows that the answer is "Not really". Figure 1 shows a slide from this presentation that characterises routes that contain load balancing based on the type of balancing detected. IPv6 load balancing make more use of the 5-tuple approach than IPv4, and a scant 3% of IPv6 load balancing uses the flow label. IPv6 load balancing is simpler and less prevalent according to the results of this study.

Classes of Load Balancing

	IPv4			IPv6		
	UDP	**TCP**	**ICMP**	**UDP**	**TCP**	**ICMP**
Per-flow	69.6%	69.8%	1.5%	77.6%	78.5%	0.3%
Per-dest	24.4%	24.1%	94.2%	13.3%	13.5%	90.1%
Per-packet	0.1%	0.1%	0.1%	0.1%	0.1%	0.3%
Per-app	1.8%	2.1%	0.0%	0.9%	0.8%	0.0%
v6 flow label	-	-	-	3.1%	2.7%	3.2%
Other	2.3%	2.6%	2.7%	3.2%	2.8%	3.9%

Figure 1 - Classes of Load Balancing, from Italo Cunha, UFMG, "IPv4 vs IPv6 load balancing in Internet routes"
https://www.cmand.org/workshops/202006-v6/slides/cunha.pdf
https://homepages.dcc.ufmg.br/~cunha/papers/almeida20infocom-mca.pdf

I can't help but observe that it seems as if every recent proposal to add additional signalling into the IPv6 packet header attempts to re-purpose the 20-bit flow label field. The flow label is not being used as an externally visible flow identifier so there are 20 bits that are up for grabs! This re-purposing of the 20-bit IPv6 flow label so common that I suspect it's a mandatory attribute of every such IPv6 proposal these days!

Understanding the Part of IPv6 that does not Ping

Tobias Fiebig of TU Delft reported on a program that used surveys and questionnaires to administrators of dual stack hosts where a port scan of the host revealed inconsistent host responses between IPv4 and IPv6 probes. The response rate to the follow-up questions and survey has not been that high, but it does illustrate that IPv6 does not enjoy the same level of deep understanding about host configuration and security that has been gathered with IPv4.

Customer IPv6 Prefixes

Once upon a time we used to use dial-up modems to access the internet. When the modem connected it was assigned an IPv4 address and held it for the session. When the modem disconnected the call, the address was returned to the pool. In these days of permanent connectivity for the end user network it is still common for an ISP to assign an IP address for a short period and then the address is returned, and a new address is assigned.

Why do ISP behave in this way? Why create this instability for the customer?

The answer, at least for me, lies in the commercial model of the ISP. They want to discourage the practice of using a consumer grade (and consumer priced) connection for a service activity. It's likely to be an opportunity cost, and commercial services cost more because the commercial customer will pay more. But I'm sure if you ask the service provider you will get a concocted story about how the tariff of a consumer service does not encompass the anticipated higher traffic profile that would be generated by the use of online services. It might even be the case that the service provider's spin is different for IPv6. Periodically changing the public IPv6 prefix somehow protects the user by ensuring that no device in the consumer's internal network is consistently addressed with a public address over long periods of time. This supposedly prevents the device from being victimised if it has poor security.

The use of NATs in IPv4 means that most of the potential renumbering implications of this practice is hidden from the end user. When the Service Provider-side IPv4 address is changed the NAT binding table then uses addresses from a different public IP address. Obviously, this will break existing sessions, so most ISPs perform renumbering in the early hours of the morning to minimize this disruption, but otherwise the internal NATted hosts are not impacted.

In IPv6 the change of the delegated prefix requires a forced renumbering of all hosts on the internal network. Of course, this is all meant to be seamless and automatics, but it's likely it fails in many cases. Most users would not notice this failure because, like many other instances of broken behaviour in IPv6 networks, dual stack fixes it and the network applications establish IPv4 connections and restore connectivity.

I find this push to ensure that services are not located behind consumer grade services somewhat ironic given the way the dial-up Internet service providers gamed the telephone system tariffs and effectively forced the telephone companies to make unplanned investments to upgrade the customer access network due to long held calls and different call patterns. It could be argued that some telephone companies entered the dial-up ISP market only to stem the revenue loss from this

> massive change in call usage patterns with dial-up services. The phone companies were, on the whole, prevented from disrupting long-held calls by disconnecting after a short time (although it was said that Indonesian telcos enforced a three-minute call by a forced disconnection after this time!) The Service Provider industry is under no such constraint from disrupting their consumer customers and thereby extorting a "business premium" for providing stable IP addresses to a customer. It is perhaps unsurprising that ISPs have now managed to alienate their customer base in the same way that telco's did some decades ago.

There are no NATs in IPv6, or so we say to ourselves, and certainly for consumer services the consumer network is provided with a prefix drawn from a public IPv6 address pool and needs to distribute that prefix within the internal network. The external IPv6 renumbering event, if it happens now ripples through the internal network and requires all internal devices to also renumber.

It was claimed in IPv6 that prefix renumbering was meant to be "easy." I'm not sure we really lived up to that claim and it is possible that this practice causes IPv6 connectivity issues in internal networks, but of course like many other IPv6 operational problems these days the Dual Stack environment saves our bacon and IPv4 (and IPv4 NATs) seamlessly papers over the problem and the user notices nothing amiss at all!

Ramakrishna Padmanabhan from CAIDA reported on a collaboration between researchers at CIADA, the RIPE NCC and Akamai to study IPv6 prefix renumbering events. What is the holding time of a V6 prefix? When a prefix changes, what is the relationship between new and old prefix values? The study used RIPE Atlas and an hourly retrieval of HTTP fetch of http://ip-echo.ripe.net. This URL returns the IPv6 address used to access the server, and a data set consisting of the time and its IPv6 address can be assembled.

> RIPE Atlas probes do not use IPv6 privacy addresses, so there is no need to record the probe ID - the EUI-64 part of the V6 address can be used to track the probe over time. This experimental technique leverages precisely the same behaviour that so alarmed the privacy folk some years back when they observed the implications of device tracking in IPv6, hence the use of V6 privacy addresses in most consumer products. The RIPE Atlas probes are now an anomaly in their continued use of the EUI-64 field in their IPv6 address.

An earlier study of IPv4 address tenure found that most services providers in the locale's where Atlas probes are deployed performed periodic re-assignment and use IPv4 address assignments that last for weeks. In IPv6 the tenure period was observed to be longer (typically months) and the number of SPs performing this reassigned was far lower. Internal address architectures vary with each Service Provider as illustrated by the changing prefix itself. Some service providers use a common /44 pool for customer prefixes, while others have been observed to use prefix lengths across the range of a /40 to a /56.

A similar study has been undertaken at NPS, and Eric Rye reported on a study that used the yarrp tool to trace to a random prefix within a collection of 45M /64 prefixes. This study concentrated on one provider who changes the customer assigned IPv6 prefix on a daily basis. What they found was a reassignment using a cycling bit shift, where the customer prefix was drawn from a common /46. They then probed each /64 inside this /46 every 30 minutes and found that the prefix reassignment was performed between 1am and 6am local time. The Service Provider was observed to use a staggered renumbering, where some customers were shifted out of a prefix block and then subsequently other customers were shifted into this newly vacated prefix block.

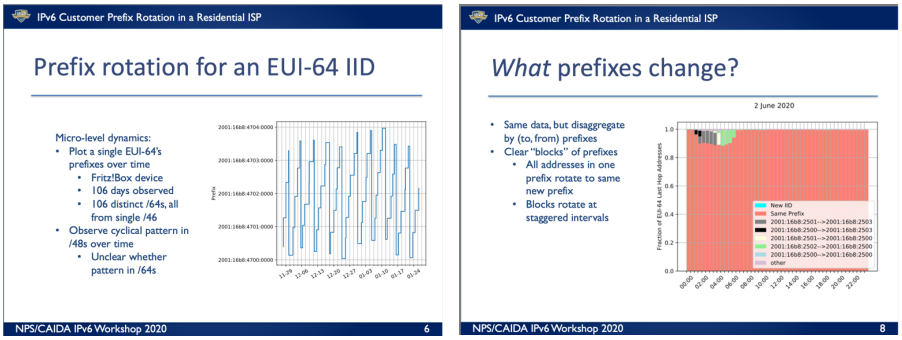

Figure 2 – IPv6 Prefix Rotation From Kirstin Thordarson and Erik Rye, NPS, "IPv6 Customer Prefix Rotation in a Residential ISP", https://www.cmand.org/workshops/202006-v6/slides/rye.pdf

While the previous study exploited the behaviour that RIPE Atlas probes use EUI-64 addresses, this study simply used devices that responded with an EUI-64 address. They ask the question why some CPE devices use EUI-64 addresses on the networking facing side of the unit. I for one do not have an answer!

V6 Address Stalking

Akami's Dave Plonka has used a fascinating twist on the yarrp tool, exploiting one-off-use addresses, as presented at the recent TMA conference (https://tma.ifip.org/2020/main-conference/).

> It's certainly the case that wherever we look for evidence of stalking in the Internet we find it in astounding quantities, and it can be observed that the economics of the entire internet is nothing more or less than the economics of surveillance and stalking. Everything else in this digital realm is a distraction from this core business!

> The evolution of the IPv6 address architecture is not a pretty story. It's probably a classic case of why committees should never design any technology, which in itself is a lesson that the IETF, and many others, simply refuse to learn.
>
> Back in the 90's when we thought it was important, we devised a two part IPv6 address, The front half was the "routing part" that was used to pass a packet through the network from source to destination and the back half was a permanent 64 bit identifier that was used within the local network to direct the packet to its ultimate destination. This was touted as "location and identity disambiguation, or "8+8". It was meant to assist multi-homing and renumbering in networks.
>
> It was also a total failure. The issues with multi-homing and multi-addressing have resisted the subsequent 25 years of IETF tinkering to try and make it work. But worse than that is the horrendous privacy leak where a device betrays its constant identity irrespective of its network location (a behaviour exploited in the previous two presentations in this workshop).
>
> It wasn't long before we dispensed with these 64 bits and in privacy addressing (which almost every consumer device uses, these) the low order 64 bits are both random and temporal. Why are IPv6 addresses 128 long and not, say 64 bits? Why do routers need to make their high-speed address lookup paths 128 bits wide, at significantly higher cost? Why does every IPv6 packet carry a total of 128 bits of what I can only characterise as ballast? Is it because the extra bit weight keeps the packet upright? Or is it just because we just stuffed up this fundamental part of the IPv6 protocol design?

What if the interface identifier part of the IPv6 address, which is commonly set to a random value as a privacy measure and held steady for a few hours or days, was taken one step further and a platform was set up that each IPv6 packet it emits uses an entirely unique random 64 bit pattern? Now this extreme

form of privacy addressing would be challenging even for QUIC to support but can be exploited in traceroute or yarrp. Now 2 raised to the power 64 is a big number, so that is an awful lot of packets that can be sent out to the Internet with one-off addresses.

The second part of this exercise is to listen for these addresses to come back. Being traceroute (or yarrp) we should expect one packet using this address in a destination as an ICMP TTL exceeded notice, or as a destination in the echo packet if it reached the destination hosts. But 1 packet is all we should expect, or maybe 2 or 3 in those deranged parts of the net that duplicate packets (yes, it happens).

The study took a collection of some 15M IPv6 end host addresses and used yarrp and these unique source IPv6 addresses to seed the network with single use addresses. They seeded using UDP port 443 (the same port used by QUIC) as both a source and a destination port address, and in ICMPv6 Echo probes. A total of some 45M yarrp traces were performed using these one-off IPv6 addresses.

And then they listened. They listened on reverse DNS queries, the listened on a packet capture at the addressed host and they listened using a passive DNS feed. They saw these once-only addresses being used more than 200,000 times, related to original use in 268 networks. These addresses were seen most often in reverse DNS, but around 20% to 25% of the time the addresses were used in a reverse probe back to the experiment host.

The reverse DNS queries originated most often from the popular open DNS providers, indicating a probable explanation that the query is being made through conventional local stub resolvers sitting behind these recursive resolvers. It's also likely that this process is part of an automated log processing routine, where unknown addresses captured in logs are passed to the reverse DNS for possible resolution.

It's an interesting question to ascribe likely motivations for this form of address processing.

While the dark arts of state espionage may be at play here, it's probably due to a far more banal reason.

It's likely that this experiment is observing automated systems that perform traffic surveillance as part of a larger process of data analytics performed by specialised enterprises engaged by service providers. After all, in a consumer-focussed business such as the internet, knowing your customer and what your customer does, any knowing it better than the customer knows themselves is a key part of providing services that each customer both uses and values. Advertising platforms now excel at this level of private sector surveillance and network operators also now understand the potential value of network-level profile data as well, both as part of network forecasting, but perhaps far more importantly as a commercial product. These days we see specialised enterprises such as Arbor, Nokia Deepfield and many others, that have good network tools that perform such detailed traffic analysis. Maybe that's what is being observed in this setup.

It's perhaps appropriate these days to think of the Internet as a massive distributed surveillance and recording system operated by private enterprise outside of most forms of regulated control or even common awareness of its existence.

Header Extensions in IPv6

> It can be observed that the IPv6 design effort took out the bits of the IPv4 header that were not used, namely the IPv4 options and packet identifier and header checksum, and then added some truly deranged features that have proved to be a problem ever since!
>
> The use of variable length extension headers is a classic blunder in protocol design, as is the broken treatment of packet fragmentation, and the brokenness of the lower 64 bits of the IPv6 address plan, as well as the discarding of ARP and replacing it with multicast and then replacing DHCP with Router Advertisements and subsequently discovering that is was such a bad idea that DHCPv6 was quickly introduced! IPv6 deployments can be observed to work only if they steer an extraordinarily careful path through these icebergs of protocol lunacy!
>
> It's hard to say what is the most inspired piece of IPv6 protocol design madness might be, but my prize has to go to IPv6 Extension Headers.
>
> Variable-length fields in packet headers were considered in the 1980's in the design work for IPv4 and the concept was discarded at the time because it was felt that the additional processing time to extract the fields in the packet impacted packet processing performance and resulted in unnecessarily slower networks. You'd think we would remember that, but no. Variable length fields are back again with IPv6. Silicon is faster these days, so is there still a problem? Yes!
>
> To see the problem, let's look at an IPv6 TCP ACK packet. It's 60 bytes in size, and on a 400Gbps wire, the wire can deliver some 833 million such packets per second, or one packet every 1.2ns. Now the cycle time of DDR2 SDRAM memory is 3.75ns. If a packet processor wants to unravel a header extension chain it will need to perform a number of memory cycles, taking at least 25ns in total, and probably a lot longer in conventional switching products there are being offered in today's market. But to keep pace with a high-speed wire it needs to complete the entire chain unravelling within 1.2 ns to avoid spewing short packets on the floor. Variable length header fields make the memory speed problem a limiting performance factor in high-speed IPv6 systems and even massive parallelism and pipelining is not enough to get around the problem here.

It seems that IPv6 Extension Headers are a bad idea and we might want to avoid them. But just how bad an idea are they? If we go back to the earlier presentation on load balancers, it appears that one half of the observed paths through the IPv6 network has one or more load balancers in the path. The predominate form of load balancing in IPv6 is the 5-tuple. The port addresses used to complete the 5-tuple are located in the transport protocol and can normally be found at byte offsets 40 and 42 from the start of the IPv6 packet for both TCP and UDP. If Extension Headers are present, they come between the packet header and the transport protocol header, and to find them it needs to traverse the Extension Header chain. This necessarily means that the load balancer has to thrash through memory to traverse the sequence of next header offsets. Which means it will take time to get to the port values when Extension Headers are in the packet. A network operator would only install load balancers when there is a need to spread the traffic across paths, and this need normally only arises where there are situations of high traffic loads. This implies that a network would benefit from load balancers when the traffic load is intense, but in such scenarios of intense traffic load the time budget to process each packet is very tight and there is not enough time to unravel the Extension Header chain.

What is a designer of an IPv6 load balancer to do? One very tempting approach is to avoid the heartache of trying to force memory to perform faster than the clock speed would permit and just discard all IPv6 packets with Extension Headers. Another approach is to place every IPv6 packet with Extension Headers on a queue for "special processing" through the CPU path, and discard packets when this internal queue

overflows. It's not clear whether the discard everything model or the special purpose queue model is more prevalent in today's network, but the result appears to be much the same in any case; the IPv6 Extension Header packet discard rate is truly awesome in the public Internet.

The IETF published RFC 7872 on June 2016, that reported on sending crafted IPv6 packets to various popular services on the Internet and included IPv6 Extension Headers in these server requests. The results pointed to a drop rate of between 30% to 55%.

At APNIC Labs we conducted the reverse of this experiment, sending IPv6 packets with extension headers to thousands of DNS recursive resolvers and millions of client hosts from a small set of test platform servers. A first pass in 2017, where DNS response packets used the IPv6 Fragmentation Extension Header, we saw a drop rate of 38%. In other words, some 38% of IPv6 end users where behind a recursive resolver that was unable to receive a fragmented IPv6 packet. A repeat of this same DNS experiment in 2020 saw the packet drop rate affect 41% of users.

We used the same measurement platform to add a fragmentation header to end-to-end HTTPS sessions. In this context the observed fragmentation header drop rate affected some 22% of end users.

We plan to repeat these tests using hop-by-hop and destination extension headers, and also to see if there is any discernible difference between UDP and TCP payloads.

> It is entirely unclear if this problem can be fixed. If the root cause of this packet drop lies in the design choice of variable length packet headers in IPv6 and the basic incompatibility of variable sized header fields with high speed packet processing, then there is no fix. It may be that all IPv6 Extension Headers, including packet fragmentation of any form is just unusable.
>
> "Fixing" the extension header problem is likely to be a case of declaring IPv6 Extension Headers unusable.
>
> I can't help but conclude from this rather frustrating exercise that the intrinsic design flaws in the IPv6 protocol are the result of taking IPv4 and replacing the useless bits with unusable bits and replacing the bad bits with even worse bits!

IPv6 Anycast

Marcel Flores of Verizon reporting on work with IPv6 anycast. The basic problem that anycast attempts to address is service scaling. How do you increase the capacity of a service? One approach is to keep augmenting a single delivery platform with processing and network capacity to match the load. Many large-scale enterprises have tried this approach and found that a popular service ultimately reaches a level of load that cannot be delivered from a single platform located at a single point in the network. At that point it makes sense to divide the clients and serve each client set with a dedicated instance of the service that is closer to each set. Anycast uses the same IP service address for each server instance and relies on the Internet's routing system to divide the client base into service sets for each anycast instance.

Today the Internet relies on anycast as we use anycast to support the root service in the DNS. But it's not just the DNS. Many content providers use anycast to provide a distributed service platform that can be augmented seamlessly as required.

The question asked in this study is does the IPv6 routing system provide a comparable anycast service platform to that of IPv4? The question is relevant given that not only is the IPv6 network smaller than IPv4, the transit and peering paths in the routing fabric differ. Given that anycast implicitly divides up the network based on BGP routing, does a smaller, and potentially sparser interconnection fabric create inferior anycast service outcomes?

Using dual stack RIPE Atlas probes, they observed that 80% of the dual stack Atlas probes will map to the same anycast site in IPv4 and IPv6. Some 70% of sites saw a decrease in performance for IPv6, and of these some 20% saw an RTT variance where IPv6 was more than 20ms slower. When the test was constrained to a single continental region or even constrained to the same network the relative performance differences reduced. This is perhaps a result that was expected. Anycast will create outcomes that are more accurate when the interconnection mesh is denser, and when comparing IPv4 and IPv6 network topologies clearly IPv4 is a far denser network.

There was also an analysis of IPv4 / IPv6 traffic in this study. There are still some major IPv4 traffic sources that are single stack. In IPv6 much of the traffic comes from a small set of large sources that show dual stack behaviour with a visible preference for IPv6 (Figure 3).

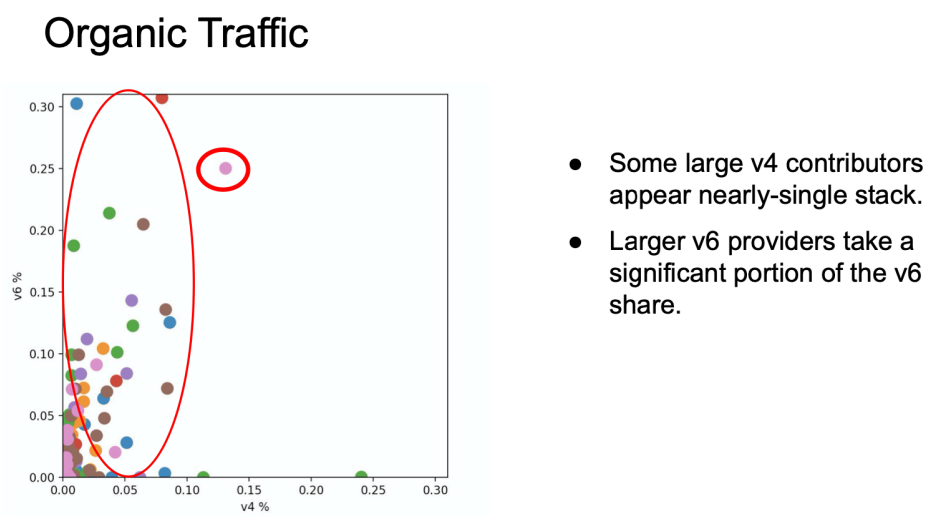

Figure 3 – Traffic Sources by Protocol, from Marcel Flores, Verizon, Scooting along to IPv6 Anycast, https://www.cmand.org/workshops/202006-v6/slides/flores.pdf

Dual Stack Performance

Casey Deccio of BYU presented an interesting angle on dual stack performance. These days it is rare to see a web resource that consists of a single object. The web page may include CSS style sheets, which in turn may include more images. There are also images and scripts, and it's quite common to see these scripts to be sourced from different service points This leads to some interesting dependencies. A dual stack web page may reference a CSS style sheet that is served from an IPv4-only service site, that in turn references an image served from a dual stack site. This will not present any issues for dual stack clients, or even IPv4 only clients. What about IPv6-only clients (assuming that any exist)? Any dual stack resources referenced through IPv4-only resources are as inaccessible as the IPv4-opnly resource itself.

In looking through the Alexa most popular web sites this form of "orphaning" of dual stack resources behind IPv4 only is common. As long as we are all prepared to believe that this dual stack environment will be an enduring and permanent feature of the Internet, then this is not a major problem. If we ever want to contemplate the end of this protracted transition and look toward an IPv6-only Internet (whatever that means!) then a lot of existing arrangements need to be fixed, including this class of protocol-specific dependencies.

Client-Side Measurements

I presented on our APNIC Labs work to measure all-of-internet aspects of IPv6 deployment and performance using online advertisement networks to enrol clients to perform the measurement tests. The scripts passed across the advertisement network that are ultimately delivered to end hosts to execute are extremely simple. Necessarily so, as advertisers are acutely aware of the inherent risks of being co-opted into becoming a gigantic malware distribution network! This means that the ad script is limited to

simply fetching http objects from a small set of service points. But that's exactly what users do most of the time these days. They fetch network objects, commonly using the web. Our ad is doing what every other user does in resolving DNS names and fetch HTML objects. We go to some lengths to make the DNS names absolutely unique in order to bypass conventional forms of caching, so that the measurement system's server platforms see all the activity caused by the ad. Particular behaviours can be measured by using DNS names essentially as microcode, where the name itself causes a particular behaviour from the server. This allows a common service platform to deliver web objects using only IPv6, to use responses in IPv6 that have IPv6 Extension Headers added, to use IPv6 only transport for DNS queries and similar.

This measurement platform has been measuring aspects of IPv6 performance for over a decade and has tracked the adoption of IPv6 from the extremely small-scale efforts in 2010 to the Internet-wide 25% adoption level we see today. Not only can the platform measure global adoption levels but it can also provide clear views on a per region and per-economy levels and can also drill down to each individual service provider network. On today's network it seems that online ad delivery is truly ubiquitous.

The measurement system can distinguish between IPv6 capability and dual stack preference. It is unusual that despite the efforts to bias the Happy Eyeballs dual stack preference in favour of IPv6, we consistently see a slight preference to use IPv4 across the daily set of measurements. The system can also look at relative performance of IPv6 and IPv4 in dual stack hosts, and also look at the IPv6 TCP handshake tp determine IPv6 connection completion rates. Recent work has also looked at IPv6 in the DNS and the issues with IPv6 packet fragmentation extension header drops.

> I've reported the outcomes of various IPv6 measurements extensively on my blog (https://www.potaroo.net), so I won't repeat that material here.

My Takeaways

The workshop was a very dense and very stimulating few hours.

IPv6 is by no means the ideal IP protocol and experience with deployment has shown that there are many aspects of protocol behaviour that we really need to avoid.

Some might be tempted to say that this is just too hard and maybe we should continue along the path of more intense IPv4 address sharing and ultimately head to a network that uses a completely fragmented IPv4 address scheme. While this course of network architecture seems to be the default choice for many service providers over the past few years, such a fragmented approach to the Internet seems to be intrinsically wrong to me. Such an approach creates barriers and costs to communication and relies on intermediaries to relay the communication between various address realms. We've already played with such a fragmented heterogenous network model in the 1980's and our experience was so painful that a single uniform homogenous Internet looked unbelievably attractive in contrast.

Despite all the misgivings with IPv6 as a protocol and all the problems with aspects of its design, it's still the only protocol on the table if we want to preserve a single coherent network. I really don't think we have a choice here.

It appears from the material presented at this workshop and similar ones in the recent past that it would be really valuable to understand a whole lot more about the ways the IPv6 network actually works and the precise ways that it fails. With this understanding we could perhaps understand more about what a viable path forward that brings online the other 75% of users into a dual stack networked realm would look like. From that point we hope that it is a far shorter step to shift to an IPv6-only Internet.

The workshop material is online at https://www.cmand.org/workshops/202006-v6/agenda.php.

My sincere thanks to Rob Beverly and kc claffey for organising the workshop and inviting me to participate.

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