The ISP Column
A monthly column on things Internet

April 2023
Geoff Huston

Notes from IETF 116

The IETF had its 116th meeting in Yokohama, Japan in the last week of March. Here's some notes I made from some of the working group sessions I attended that I found to be of interest.

IEPG

The IEPG is an informal gathering that meets on the Sunday prior to IETF meetings. The intended theme of these meetings is essentially one of operational relevance in some form or fashion.

This time there was a continuation of the saga of the missing IPv6 Extension Headers. In 2016 Fernando Gont and colleagues authored RFC 7872, noting a significant level of packet drop on the Internet whenever a IPv6 packet used an extension header. This would be somewhat inconsequential in general, as IPv6 extension headers are about as useful in the public Internet as IPv4 options, except for IPv6 fragmentation controls, which IPv6 chose to load into an extension header. The observed drop rate was high enough to imply that IPv6 packet fragmentation was all but unusable. This measurement was undertaken some time ago and the situation has been revisited from time to time, with various measurement efforts achieving wildly different outcomes. This seems somewhat odd, in that all these network elements and end systems are intended to conform to a common set of standards. Yet one, admittedly very small scale, experiment reports no packet drop at all while others see significant drop levels for such EH-adorned IPv6 packets.

It appears that there are number of variables in these measurement experiments that can influence the results, such as the selection of the endpoints, the number of endpoints and their role (server, client) and their position (access networks or core), and the nature of the application that is generating the traffic flows where EH packets are being inserted. While it appears to be possible to encounter a scenario where the end points and the path between them pass EH packets cleanly, it's also easily possible to encounter the opposite scenario. My conclusion is that EH packets, including fragmentation extension headers, are sufficiently unreliable to be useless in the public Internet.

Another measurement exercise looked at the delays in the RPKI system, measuring the time between the making a change in the authority objects in a RPKI publication point and making a change in the forwarding behaviour in the network. I must admit that I find this particular measurement to be somewhat disingenuous, in that the original design parameters of the RPKI system did not include any form of low latency between action and outcome. As I recall, a lag of 24 hours was considered not unreasonable in the overall design of the RPKI system. If we wanted extremely low latency performance, then we would not have used an uncoordinated on-demand polling system to perform information flooding. Maybe we would've pushed harder to use a single point of trust authority without local variations. Maybe we would've considered the packaging of signed credentials and their validation chains to enable pre-provisioning of RPKI credentials to clients. We would also have opted for a more effective flooding algorithm including push-based flooding across some forms of spanning tree. So yes, the combination of the RPKI system that we have and the dynamics of the BGP protocol reacts to changes in the order of minutes and not seconds. And as it grows this reaction time is likely to get slower. Considering the overall design of this system such an outcome is totally unsurprising.

One area where high latency is completely unacceptable is in time itself. The Internet runs on an informally organised network of reference time sources, yet the objective of the NTP time synchronisation protocol is to synchronise all client clocks to within a millisecond of these reference sources. The issue here is that the NTP protocol runs in the clear and is susceptible to various form of attack by hostile parties. And if there is one thing that we've all been forced to concede these days is that the Internet is an incredibly hostile environment. The way the time protocol has chosen to defend itself against such on-the-wire attack is to clock the packets in an encrypting wrapper. All well and good, but encryption (and decryption) takes time and computing resources, and it has hard to add packet encryption to NTP without compromising the capacity of existing NTP time servers. One response has been from NETNOD, which has implemented the secure NTP protocol on an NVIDIA engine, offloading the compute load to a specialised processing environment. This has been not just successful, but wildly successful in ways only the Internet can do, and their NTS client population served by these secure time servers has quickly expanded from less than 1 hit per second in February 2022 to some 577 hits per second in the space of 12 months. Some further details of this NETNOD effort can be found at https://www.youtube.com/watch?v=mszu6e4c3Vk

Agenda: https://datatracker.ietf.org/meeting/116/materials/agenda-116-iepg-sessa-00
Video: https://www.meetecho.com/ietf116/recordings#IEPG

IPv6 Operations

The essential part of the design of the IPv4 protocol was the work of two individuals, Vint Cerf and Bob Kahn. The design of IPv6, which was ostensibly a small incremental change to IPv4 to expand the size of the source and destination address fields was the work of a committee spaced over some years. And it shows. One of the challenges with IPv6 deployments is that there are so many different ways to do just about everything.

One of the most extreme examples of this behaviour can be seen in the various dual stack transition mechanisms, where each transition scenario triggered its own bespoke transition mechanism. While it may be a stretch of linking cause and effect, the case can be made that the plethora of transition mechanisms ended up confusing vendors, service operators and network operators to the point that just doing nothing and waiting seemed like a pretty rational choice. This working group session heard about a framework of multi-domain IPv6-only underlay networks and IPv4 as a service, 464XLAT/MAT-T Optimization and IPv4/v6 dual-stack migration for in-house load balancers. I have to wonder and despair just a little. The original transition plan, dating back some thirty years now, was incredibly simple: when an end point is provisioned with both IPv4 and IPv6 connectivity (dual stack) then try to connect to a remote service using IPv6 in preference to IPv4. And that's it. The corner cases we heard in this session seem to me to be rather pointless artifices. For example, what if my DNS recursive resolver only has IPv6 connectivity? The obvious answer is that there are a whole lot of DNS authoritative servers and even resolvers that are not yet dual stack enabled, so you can't reach them. The only time when IPv6-only service platforms make sense is when everyone is already dual stack enabled, and at that point if everyone prefers to use IPv6 then the transition is over. Sigh!

Transition mechanisms are not the only case where a thousand flowers are blooming! There is also the never-ending saga of IPv6 address assignment. Yes, we are talking about SLAC, DHCP, DHCP-PD and version other flavours. Yes, it sounds like a collection of religious schisms in the church of IPv6, and if you had that opinion then you'd probably be pretty close to today's reality. To complicate the picture, we really have never completely agreed on how to treat these 128 bits. Are we really dealing with a 64 bit network address in every case, and allowing SLAC to manage the other 64 bits on each network? If so, then that is incredibly inefficient, and the prospect of IPv6 address exhaustion is not completely out of the question. But if we want to move this network/host address boundary to another value, then what should this value be? Nobody knows.

And then there is "multi-homing". If an IPv6 network at the edge wants to improve the resilience of its network service, then its common practice for the edge network to obtain its own independent network prefix and announce this address prefix to all of its routing transit providers. But this approach does not scale, in that all these small multi-homed edge sites cause the routing table to bloat in size. In IPv4 the answer to multi-homing is far simpler: "If you can't use your own independent address space, then just use a NAT!" This is not a popular approach with the IPv6 folk, as by and large they tend to hold strong opinions of the advisability of using NATs in IPv6. There was the Shim6 effort of some twenty years ago, which embedded the address translation function into the protocol stack on the end hosts. This approach fell over on two hurdles. The first was the need to perform source address routing to pass the outbound packet to the "correct" provider's external interface based on the IPv6 source address prefix in the packet. The second was that a large number of access network providers were averse to having individual end hosts making independent decisions to switch their traffic between providers. Access network operators wanted to control the traffic management across multiple upstream providers at the network level, and definitely not on a host-by-host level. Frankly, I really don't see anything in this latest study that has not been covered numerous times already. There really is nothing new to see here!

It appears to me that the conversations in this IPv6 Operations working group are the conversations that the working group members are most comfortable with, and these are the well-rehearsed conversations that keep on being repeated year after year without any signs of closure or resolution.

Agenda: https://datatracker.ietf.org/meeting/116/materials/agenda-116-v6ops-01
Video: https://www.meetecho.com/ietf116/recordings#V6OPS

6MAN

There are two standing working groups that concentrate on IPv6 in the IETF these days. One is IPv6 Operations, and the other is the maintenance of the IPv6 protocol itself, 6MAN. It's a subtle distinction, and one that has proved to be difficult to maintain, and too often it seems like an ideal opportunity to air the same topic twice!

At this IETF meeting the 6MAN working group considered aspects of SLAAC (Stateless Address Auto-configuration), IPv6 Extension Headers and generalised tunnel mechanisms (presumably as an aid to some forms of transition). These are now quite venerable topics, and the space has been comprehensively traversed over many years. I don't know what others are expecting here but I have no expectation that there is any astounding breakthrough in these particular topics either now or in the future! Which is much the same as my feelings about V6Ops.

Agenda: https://datatracker.ietf.org/meeting/116/materials/agenda-116-6man-03
Video: https://www.meetecho.com/ietf116/recordings#6MAN

TCP Maintenance

While the IPv6 working groups seems to be unable to get past a small set of seemingly endless conversations, the TCP Maintenance Working Group appears to be making some progress, although you probably need to use a pretty liberal interpretation of the term "progress" to reach that conclusion!

One of these areas is the efforts to increase the signal frequency in the explicit congestion notification (ECN) mechanism to allow a signal of the relative level of ECN indicators received in each round-trip time interval. The first draft of this mechanism, Accurate ECN, was pushed out in December 2015 and we are now up to revision 24. The concept is very straightforward, taking the three ECN bits in the TCP header and using them as a three-bit counter to signal the number of received ECN signals received in this RTT interval. This allows some recent ECN-aware TCP protocols, such as DCTCP and L4S controls to respond to the intensity of the congestion signal rather than the earlier binary response used by Reno. A high ECN mark intensity value results in a greater rate reduction by the sender in accurate ECN-aware transports. After almost nine years it whould be good to see this work get through the process and be published as a Proposed Standard. But I'm not holding my breath!

Originally, in the experimental RFC 3168, ECN was defined for use in TCP data packets. This was extended to use ECN in the SYN-ACK handshake in RFC5562. ECN++ proposes to use on all forms of TCP data and control packets. It is intended to work with TCP Fast Open, TCP sessions with large initial window sizes and L4S transports. It is intended to reduce the incidence of packet loss on TCP control packets, which leads to better average TCP connection establishment times in networks which are congestion prone. This is also a Working Group document has been around for seven years, and it also would be good to finally push this out as an RFC!

TCP delayed ACKs date back to a paper in 1982 about the protocol overheads imposed when using TCP for short segments. The delayed ACK mechanism described in RFC1122 allows a received to delay an ACK packet by up to half a second or every second received packet. The problem is that this mechanism is both too much and not enough, depending on the scenario in which TCP is being used. In low-latency high-capacity scenarios this delayed ACK creates idle intervals and inhibits the growth of the congestion window when there is available network capacity. In other scenarios of longer latencies, when the sender is sending bursts of packet trains, it is adequate to send a single ACK for the entire received packet burst. So, there is a proposal to add a TCP option to pass an ACK rate from the sender to the receiver to allow this rate to be tweaked for each TCP session. There is one school of protocol design that takes the position that when the protocol design is unsure of a single "correct" parameter value, then allow the ends to negotiate the value they want to use. Another school of thought takes the stance that simplicity is its own reward, and making a single default choice creates a more robust outcome in general, avoiding implementations trip over in trying to guess the "right" setting for each level and knob in each instance. I think I'm in that latter school of protocol simplicity!

However, in holding that view I'm swimming against the current tide, where, like BGP, there is a distinct preference to negotiation of option settings in the opening TCP handshake. The TCP protocol only allows for 40 bytes of options. These options are used for setting the receiver's maximum segment size, the window scaling factor, selective acknowledgement, and TCP timestamps. There is the Fast Open Cookie request, and the ack rate request value. What if you want more option settings? You might want to alter the TCP Data Offset value to push the data offers further back in the packet using the Extended Data Offers (EDO) Option. Or you could send two SYN packets and carry over the options into the second SYN packet. One proposal is for a "aggregated option" for use in SYN segments, and only in SYN segments, where the On/Off options, such as Fast Open Cookies, could be packed in as single bit flags. Of course, the problem is that there are few such On/Off options, (Fast Open and SACK permitted) and the net savings in option space using this bit vector approach is minimal.

Agenda: https://datatracker.ietf.org/meeting/116/materials/agenda-116-tcpm-03
Video: https://www.meetecho.com/ietf116/recordings#TCPM

Side Meeting on Satellite Networking

As well as Working Groups, IETF meetings have embraced the concept of "side meetings" where folk with a common interest can gather and chat in an open setting. There was an interesting side meeting at IETF 116 on the topic of networking support for Low Earth Orbit satellite constellations, lead by folk from Futurewei and China Telecom

For many years satellites were used as a mirror in the sky (commonly called a "bent pipe"). The satellites were parked in a geostationary orbit (GEO) and the onboard electronics relayed signals received on one receiver transponder to a transmitting transponder, thereby connecting two ground stations with a bitstream circuit. Then came the low earth orbit (LEO) satellite services, positioning the spacecraft in a much lower orbit. At this lower altitude the spacecraft has a far higher relative speed, and in order to provide a continuous service the satellite service operator needs to operate a constellation of such spacecraft to ensure that there is always at least one spacecraft that is visible to the earth-based service point.

This could be a constellation of bent pipe platforms, as is the case with the first generation of Starlink services. In this case the "other end" of the bent pipe needs to be in the same footprint of each satellite, so while the constellation of spacecraft might pass over a major part of the earth's surface, the service area is greatly constrained by the presence of ground stations in the same coverage cell as the clients. Service coverage over oceans are a major issue in such a model.

The next generation of LEO satellites will use some form of communication link between these spacecraft (an Inter-Satellite Link, or ISL). This can expand the range of earth coverage of a LEO constellation, as the signal can be relayed across a number of spacecraft and passed back to earth at some distant point which is not necessarily adjacent to the location of the upload. There is no requirement to persist with a virtual circuit model that links a client to a ground station, and to go further, this ISL function would be a good fit to a packet-switched service model where each satellite is in effect a packet router. If that's the case, then we may look to use a routing system to drive the packet forwarding process through the satellites' ISLs. More generally, is it feasible to construct a LEO-based satellite service using existing standard technologies?

It certainly seems as this could be the case. The 3GPP specifications include studies on the use of access models that permit user devices to communicate directly with satellites, and integrate such services into the 5G architecture. It would appear that for access mechanisms we can use some ready-made approaches.

But once the packet is up there, the question is where and how to bring it back down to earth?

Leo Satellites orbit the earth with a period of around 95 to 100 minutes, which implies a velocity of around 7.7 km/second. Each satellite has a horizon-to-horizon period of visibility of 12 minutes, and they are sufficiently high the sky to be useful from a fixed point on the earth's surface for slightly under half that time, or 5 minutes. So a user device in earth communicating with the satellite service would be switching from satellite to satellite every 5 minutes or so. It is assumed that the ISL links will form a mesh. The assumption here is that these satellites are placed in an inclined orbit, with half of the satellites in a positive inclination, and half in a negative inclination. Each ISL links a satellite to its forward and rear satellite in the same satellite train at this inclination and temporarily connects to adjacent satellites in the opposite inclination as they swing past across this satellite's path. To add to this, it is also a possible to operate a satellite constellation on a number of orbital planes, where the spacecraft at the lower plane has transponders that send and receive from points on earth, and their ISLs pass the packets to a connection of inter-spacecraft switching satellites that operate at a higher orbital plane.

There is also the issue of what the satellite mesh is attempting to achieve in terms of packet forwarding behaviours. An incremental step from the simple bent pipe is to relay the client's packets to the "closest" ground station and then pass it into the terrestrial network. The reverse traffic could be similarly handled by passing it to the same ground station and then into the satellite network. A more ambitious objective is to dynamically route the packets in the satellite network to the ground station that is closest to the packet's destination, leveraging the superior signal propagation performance in space as compared to fibre circuits.

Assuming this is a dynamically routed network that what form of routing technology is suitable? Distance vector approaches might take too long to converge, particularly as the radius of the LEO constellation is so large. A link state algorithm has a large dynamic link state change load but is probably the more viable. However, while the situation is dynamic it's not unpredictable. Indeed, it's the opposite, and the satellite paths and hence the ISL topology can be predicted in advance. Does this mean that we could avoid using a dynamic routing protocol and shift to some form of time varied protocol? It's a fascinating problem space, and one that puts a new spin on our understanding of dynamic networks and routing systems, but I suspect it's getting into the realm of being a bit far-fetched. I suspect that conventional economics dictate that the cheapest path is to drop the packet back to earth-based networks as quickly as possible and treat the ISL services as a way of extending the reach of each ground station.

Material: https://github.com/lh95129/IETF-116-satellite-network-side-meeting

DBOUND2 BOF

In the process of forming a Working Group the IETF needs to establish whether there is sufficient interest from the community to work on the problem space and what the problem space actually is. This is performed by the use of BOF session (Birds of a Feather).

The DNS is a hierarchical name space, but there are points in the name hierarchy where the administrative control of names passes from one party to another. Identifying where these points of change of control, or boundaries, occur in the DNS is unclear. The upper levels of the name hierarchy are generally considered to be "public" names in that parties who are not affiliated with each other nor with the controller of the common suffix domain may register subdomains in this name space, and each delegation represents a boundary of control. Examples of such Public Suffix domains include "org","co.uk", and "edu.au". The limitations that apply to such Public Suffixes are in the actions of Certification Authorities, who should not issue a wildcard certificate for a wildcard name that traverses a control boundary (such as "*.edu.au", for example). Cookies have a similar issue. For example, a web server at "foo.bar.example" sends a cookie with a domain value of "bar.example", which happens to be a public suffix. Without a way of identifying that there is a public suffix boundary, the browser does not know that "bar.example" is a public suffix and sends the cookie in subsequent requests to any other host in "*.bar.example". The browser typically sends these cookies without explicit user consent or action, yet cookies allow sensitive state information, including browsing history and login sessions, to be made known across independent entities. Any such sensitive information in such an over-privileged cookie will be sent to servers that are almost certainly not authorized by the user to view the information, creating significant security and privacy risks.

So how can a browser, or a certification authority, or any other entity know is a given domain is a public suffix or not? And what precisely is a "public suffix" in any case? The answers to these questions are somewhat unsatisfying. We have no clear shared understanding pf what constitutes a "public suffix". And there is no mechanism in the DNS itself to flag such boundaries of control. Some folk, including the popular browsers, make use of a "public suffix list" (yes, a text file). This text file is a community resource operated by volunteers who appear to be loosely coordinated under the auspices of the Mozilla Foundation. The work, laudable as it is, has some clear shortcomings.

Back in 2015 there was some optimism that we could describe this point of change of control in the DNS using some kind of DNS marker, and a Working Group was chartered to work on this problem. However, some two years later the work was abandoned, as the Working Group could not come to an agreement on the problem statement. This topic has come back to the IETF to see if there is any appetite to take up with work once more, perhaps with a more focussed problem statement or a smaller set of potential use cases.

In many ways this does appear to be a case where the self-identification of such administrative control boundaries in the DNS makes a lot of sense. After all, this control boundary is probably best known by the zone administrators themselves. However, the lack of a clear definition of exactly what is an administrative control boundary and the use cases where such a boundary is an important consideration means that we may find a DNS boundary marker being used by many, but with entirely different concepts as to what if can be used for and why!

Agenda: https://datatracker.ietf.org/meeting/116/materials/agenda-116-dbound2-04
Video: https://www.meetecho.com/ietf116/recordings#DBOUND2

DNSOP

Most of the DNS activity in the context of the IETF occurs in the DNS Operations Working Group, and the meetings tend to be very busy. A set of documents are being shepherded through various stages of the publication process, relating to the SVCB resource record, the alt Top Level, fragmentation, glue is not optional, and DNSSEC validator requirements.

Some years ago the IETF recommended the inclusion of a "special use" top level domain *.onion* into the IANA Special Use Domain Name registry (RFC 7686). This was a somewhat contentious action at the time, as it appeared to some to be an end-run around the ICANN processes of opening up the root of the DNS to new generic top level domain names (RFC 8244 documents some of the issues). It was predicted at the time that what works for the TOR project and *.onion* works equally well for a seemingly endless stream of alternative name systems, and the IETF would be once more embroiled in the challenging topic of managing competing bids for top level domain names.

One view is that all these issue of policy around the administration of the DNS name space was passed ICANN decades ago, and the IETF should maintain that stance and leave this to ICANN. But this is not consistent with the IETF's behaviours and there is a continuing knee-jerk reaction to device a "fix" to all problems that sneak into the IETF's tent, irrespective of whatever decisions might have been made in the past. In this case the IETF is working on a generic solution of the special use name registry by inserting a catch-all, and hopefully final entry, *.alt* (https://datatracker.ietf.org/doc/draft-ietf-dnsop-alt-tld/, a draft that is already 9 years in the making). This name will never appear in the DNS, and DNS resolvers should not even attempt to resolve names in that particular namespace. From the IETF perspective a *.alt* name means that this is not a DNS name and it not managed in any by the DNS.

This seems like a workable compromise. Yet, for some reason the topic of alternate name systems still takes meeting time at these DNSOP meetings, and this time it was the turn of the GNU Name System, or GNS.

How should one think about these alternate name systems? Should they be thought of as exercises in innovation in the name system? Often innovation encompasses a deliberate effort to disrupt the status quo and challenge the way the systems currently operate. Sometimes innovation challenges the basic tools and machinery of the current technology. Good examples in this category are DNS over TLS and DNS over HTTP, where the DNS itself and the name space defined by the DNS are unaltered, and only the way in which the DNS resolution mechanisms operate are changed. In other cases, the goal of the effort is to challenge the existing name allocation and registration function, replacing the centralised model used by the DNS with a highly distributed model, such as us seen in the blockchain-based approaches. Or are these efforts little more than deliberate attempts to break down the cohesion of the name system? If the effort uses incremental extensions to the name space by creating an occupied name space from an undelegated domain, then these names are only visible to users who are placed into an alternate name resolution environment. Not every user can resolve every name, and collisions can occur where the same name can be resolved to multiple outcomes using different resolution environments. In this case names lose their deterministic properties and resolution of a name produces variable outcomes where the user or the user's application would find it challenging to navigate. Once names lose their coherency in the networked environment, they cease to be useful. But if we don't have name coherency then what do we have left to bind the Internet together as a consistent single communications realm?

Innovation is the process of challenging the current assumptions about the consensus on how we operate the network and its service environment, pitting new ideas and approaches in direct competition with the established practices. Fragmentation is the process of pulling apart the commonality of a single networked environment, compromising the value of the network by eroding the assumptions about universal reachability and consistency of symbol interpretation and referential integrity. We can't communicate over a network that does not present a coherent name space to all its users.

How should we respond to such pressures? One response is to codify the existing system into a set of rules and rely on regulatory fiat to preserve the consistency of the network. It's unclear how effective such an approach can be, particularly in the longer term. The technology base continues to evolve and if the outcomes cannot be applied into the incumbent system, then the pressures for change will increase to a level that may well shatter the incumbent in a highly disruptive surge of change. Rule-based systems tend to favour incumbents, further increasing the build-up of centrality in the system, resulting in even greater resistance to evolution and change.

Or we could simply let these pressures play out in the market. If innovative ideas capture the attention of users, then they will gather further investment momentum and command the attention of the incumbent operators to come to terms with the innovation one way or another. It's clear there is no "right" response here.

Domain verification is more of an art than a standard. Many service providers need domain name admins to prove that they control a particular domain before granting them some sort of privilege associated with that domain. For example, the ACME protocol has a DNS-based challenge for a user to prove that they control a particular domain, and hence should be issued a cert for it. There is an interesting draft that was presented at this DNSOP meeting that describes the common approaches in demonstrating domain control and the common pitfalls that accompany such approaches. A common approach is to use a value placed in a TXT record at the zone apex, which works moderately will until the number of such records expands, and the size of the query response can head into areas of UDP fragmentation or failover to TCP (just look at the TXT records associated with *bbc.co.uk* or *amazon.,com* for good examples of this form of domain verification). Other verification approaches use dedicated labels in the domain name for the challenge response, such as using the label *_acme-challenge.example.com*, avoiding the bloating of the response that comes from overloading the TXT resource record at the zone apex. Where a domain references a resource that uses an external provider, such as a CDN, then CNAME records are common. While the current version of the verification draft (https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-domain-verification-techniques-01) takes a hard line in the use of CNAMES in domain verification ("It is therefore NOT RECOMMENDED to use CNAMEs for DNS domain verification.") the practical realities of the circumstances of external delegation of zone control means that perhaps a softer position on the use of CNAMES needs to be used. This is a useful effort to catalogue what is used for domain verification and the strengths and potential pitfalls in each approach. However, I suspect that the 01-draft is incomplete and the issues of CNAMES, wildcards, the Public Suffix List and multiple external providers needs to be covered, as do the issues of time to live, description of policy constraints, and even a registry of verification challenge labels still need to be included in this draft.

The diversity of the DNS presents some challenges for posing a single uniform standard for some technologies. For example, the DNSSEC records used to sign a zone include the NSEC record to provide a mechanism for verifiable assertions of non-existence of a label. Generating this record normally requires the zone admin to have a complete copy of the zone file, then generate the NSEC spanning records, sign all the zone records and then serve the signed zone. However, for large zones this may not be feasible, and the authoritative DNS server might use a *front end* signer that signs the DNS records on the fly in response to queries that wish to have DNSSEC signatures attached to the response.

The front-end responder might query the zone database, find the relevant record, attach a DNSSEC signature to the record and push the response to the querier. But that about queries for non-existent domains? Or queries for non-existent types? In both cases the front end be informed by the zone database that the query name is not defined, or the query type is not defined for this name. But this is not sufficient information to assemble a DNSSEC NSEC response, as it is not necessarily aware of the two *closest* labels that span this non-existent label, or not aware of all the defined types for this query name.

One option for a non-existent label is for the front-end dynamic signer to generate a synthetic NSEC record by using a synthetic immediate predecessor label and a synthetic immediate successor label in a NSEC response and use this synthetic information in the NSEC record.

One can go further by observing that a NODATA response is smaller than a signed NXDOMAIN response. In a NODATA response the NSEC record indicates that the queried name exists, but the queried type does not. This signed NODATA response can be generated more efficiently by the front-end signer, and the response is also smaller than a signed NXDOMAIN response

Both of these signed response mechanisms were specified in a now-expired draft (https://datatracker.ietf.org/doc/html/draft-valsorda-dnsop-black-lies). Despite the lack of an IETF standard, this approach has been used by a number of large DNS providers, including Cloudflare, NS1 and Amazon's Route 53.

A residual issue relates to the combination of query name minimisation and the case of ENTs (empty non-terminal labels). One approach is proposed in the draft (https://datatracker.ietf.org/doc/html/draft-huque-dnsop-compact-lies-01), where the NODATA response is use for both non-existent and ENT names, which the NSEC record Type Bit Map having the NXNAME type set for non-existent names.

This illustrates one of the more challenging aspects of the DNS. The DNS is more like an ecosystem than a single software artefact. If you change the way authoritative servers respond to queries, then you need to ensure that the parties that consume these responses understand the nature of the change and its implication. It strikes me that the way these compact responses have been handled is along the lines of: "Well, as far as we are aware, nothing broke when we turned it on!" This kind of approach is a long way from a more careful approach of coordinating the adoption of a new DNS behaviour across servers, resolvers and end client DNS implementations.

It's heartening to see some evidence of learning in the DNS world, as the process of repeating the same old mistakes can get pretty tiresome. We had a problem in sync'ing primary and secondary servers, as the secondary was supposed to continually query the primary to see if anything had changed. It was slow, inefficient and error prone. In its place we used the NOTIFY mechanism (RFC1996) where the primary signals the secondaries that the zone contents have changed and it's time to refresh the local copy of the zone. The same applies to the CDS and CSYNC records, where the parent server is meant to poll the child to see if these records have changed. A lot of wasted polling could be eliminated if the child notified the parent that the record in question has changed. The proposal is at https://datatracker.ietf.org/doc/draft-thomassen-dnsop-generalized-dns-notify/. What took us so long to get here?

Agenda: https://datatracker.ietf.org/meeting/116/materials/agenda-116-dnsop-04
Video: https://www.meetecho.com/ietf116/recordings#DNSOP

SIDROPS

The level of intensity of activity in this group has eased somewhat, although that should not be interpreted as a signal that the work in securing the routing system is largely done. Far from it. Some parts of the work have proceeded at unexpectedly fast pace. More than one half (54%) of the announced IPv6 address space is now described in a ROA object, and a little under one half (39%) of the announced IPv4 space is similarly described in a ROA. But other aspects of the secure framework are not progressing at the same rate. Two thirds of users will still connect to an endpoint when the only path to that endpoint is via a ROV-invalid route. BGPSEC appears to be going nowhere, and instead the AS adjacency mechanism (ASPA) is still in its early days and uptake is still hesitant.

The RPKI infrastructure is reliant on unsynchronised polling to flood credentials to all clients, which leads to uncertainty about the intended responsiveness of the RPKI system, as well as concerns about the scalability of the system if the number of clients and publication points increase (as this is an N^2 scaling issue).

There was one item in the working group that I thought was a useful topic to consider. The RPKI system was not constructed in a vacuum, and there is still widespread use of various Internet routing registries (IRRs). IRRs have a more expressive language, such that they can describe a network's routing policies, enumerate the complete set of prefixes that a network may originate, and even enumerate a network's providers and customers. At present the RPKI describes route origination from the perspective of the prefix holder, and the work in defining an object to list a network's providers is largely complete, but that's it. Which leaves us in an interesting position as to where to go from here. Should we create the signed analogue of all of the various IRR objects as defined objects in the RPKI? Or should we introduce an RPKI-based signature structure into the IRR to allow an IRR client to understand the authority, authenticity, and currency of the IRR data? Or just leave things alone and have the IRRs and RPKI continue to co-exist.

I suspect that if we go down the "everything in RPKI" path, then the challenge will be to describe routing policies in an RPSL-like language that can be also described in ASN.1. And if we take up an old idea to just sign IRR objects there is the long-standing issue of the very partial levels of uptake of describing inter-AS routing policies in the IRR. It may be signed data that clients can validate, but it nobody uses it to publish their routing policies then it's still not a useful tool! It's still unclear to me what the right choice is here, but maintaining operational support for both systems going forward seems to be somewhat inefficient at best, and a source of confusion is the data contained in each system diverges.

Agenda: https://datatracker.ietf.org/meeting/116/materials/agenda-116-sidrops-04
Video: https://www.meetecho.com/ietf116/recordings#SIDROPS

MAPRG

MAPRG is a research group focused on measurement. For me it never fails to have stimulating content that throws new light of protocol behaviour and performance.

The first of these was a study of TLS certificates and QUIC performance. QUIC is attempting to do a lot all at once, and in the opening exchanges QUIC is attempting to emulate the TCP three-way handshake and the TLS handshake in a single exchange. QUIC aims to complete the initial handshake (which incorporates the TLS handshake) within a single round trip time. A QUIC server will respond to an initial Client Hello with the Server Hello and TLS Handshake, which means that the response is potentially far larger than the initial client packet, which, as we've learned from some rather painful experiences, is a potential DDOS vector. QUIC has an objective that the initial server response should be no larger than 3 times the size of the Client's initial message. A scan of the Tranco top 1M servers reveals that these objectives are not being met in general. Server responses tend to be larger than 3 times the initial client message and tend to take longer than 1 RTT. If you are interested in QUIC and its performance it's a good session to review. Smaller certificate chains using ECDSA signature algorithms could help here.

I have already touched upon the LEO routing issue in my comments on the satellite side meeting, and it seems to be a popular topic right now. A presentation in MAPRG focussed on the question of path jitter if the end-to-end path was dynamically constructed across a LEO constellation. This exercise was based on simulations of the Starlink, Kuiper and Telesat systems, using data on the altitude, population and orbit details from the US FCC public filings. Not unsurprisingly, they find a high degree of route churn and path jitter. However. this is likely due to the service model used in the simulation, which retains the packet within the satellite network for as long as possible, and drop it back to earth as close as possible to the packet's addressed destination. I suspect its far cheaper to drop the packet back to the earth networks as quickly as possible, and these shorter paths would exhibit much lower churn and jitter.

There was an interesting commentary about the design of the measurement impacting on the measurement results. In this case the scenario being used is a number of measurement efforts to determine the drop rate of IPv6 packets that carry IPv6 Extension Headers. In this case, the measured behaviour varies greatly between the various measurement efforts and its possible to account for this variability by looking at the location of the measurement points (cloud vs core vs edge), the size of the measurement (number of diverse end points and network paths) and even the applications used to host the experiment (web, mail, DNS, etc). All such measurements are a distillation based on a set of tests conducted using sample points. While it's tempting to generalise from such tests to the general case, such generalisations may not be valid, and could readily lead to a misleading conclusion.

Apple's Private Relay cloud service is an interesting exercise in privacy-preserving services operating over the public Internet. It is a large-scale platform, using Apple platforms and apps. It uses a dual-proxy architecture, with outbound packets passing through an Apple proxy initially and a third-party proxy (operated by Akamai, Cloudflare and Fastly) to interface to the public network. The private-part of the relay service uses the QUIC transport. Their measurements reveal an incremental cost to use the Private Relay service. The Private Relay throughput rates are on average significantly slower than the underlying Internet. Their measurements also saw a major difference between the egress relay providers, but which was faster varied by the locale of the test. The total page load time are slower by between 7% to 60%, which also implies that Private Relay incurs a visible performance hit.

Agenda: https://datatracker.ietf.org/meeting/116/materials/agenda-116-maprg-10
Video: https://www.meetecho.com/ietf116/recordings#MAPRG

IETF 116

Obviously, this is a small snapshot of the sessions that took place across a very busy week at IETF 116. The full schedule of working group meetings, agendas, presentations, and video records can be found online at https://datatracker.ietf.org/meeting/116/agenda.

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