The ISP Column
A column on various things Internet

August 2021
Geoff Huston

Some not-DNS Topics at IETF 111

I've already scribed my thoughts on the DNS-related topics that were discussed at the July 2021 IETF 111 meeting. It may be surprising to the DNSphiles out there but there really are other topics that are discussed at IETF meetings not directly related to the DNS! These are some notes I took on the topic of current activities in some of the active IETF areas that are not DNS topics.

Transport and L4S

Transport has been undergoing a revival of study in recent years. The original Additive Increase Multiplicative Decrease (AIMD) flow control models were revised because of the emergence of very high capacity extended length network paths (so-called "LFN" or Long Fat Network paths), and there was a set of proposals in the area of high speed TCP that looked at how to tune the TCP flow control protocol to quickly adjust to very high speeds, while still remaining responsive to the so-called "fairness" considerations of sharing a common transmission resource with concurrent conventional AIMD TCP sessions (https://www.potaroo.net/ispcol/2005-06/faster.html). The CUBIC flow control protocol gained popular acceptance, probably based on its use as the default TCP control protocol in a number of widely used Linux distributions.

In this family of AIMD control protocols there is a direct relationship between the performance of the protocol and the size of the buffers in the active switching elements of the networks. If the buffers are too small the protocols fail to increase their speed to take advantage of available path capacity, while buffers that are too large simply act as a delay element. The situation has been exacerbated in recent years when network equipment has been deployed with, at times, quite extravagant buffers, and the combination of large buffers and AIMD-based control flows can induce a condition known as "buffer bloat".

There have been two somewhat different forms of response to this issue. One response is to change the flow control protocol. The BBR flow control protocol (https://www.potaroo.net/ispcol/2017-05/bbr.html) is a good example of a flow control protocol that is not reliant on getting the network buffer provisioning to be exactly right. The BBR protocol makes use of continuing measurements of round-trip delay to inform a sender as to the onset of network queuing, as distinct from the AIMD-related signal of packet drop, which is a signal that network queues have not only formed but have become saturated. The other form of response is to enlist the network elements to inform the end host of the network's state, and in particular the onset of congested-related queue formation. A good example here is the Explicit Congestion Notification (ECN) experimental protocol, described in an experimental specification, RFC3168. Network switching elements marked ECN-aware IP packets that were processed when the switching element was experiencing a congestion load. In TCP this ECN marking is passed back to the sender in the TCP ACK packets, which then treats the received ECN congestion experienced signal in a manner similar to conventional packet drop. Ongoing experience with an increasing level of adoption of ECN has shown that ECN is most effective in conjunction with Active Queue Management (AQM). This approach that combines ECN with AQM has led to the so-called "L4S" effort, intended to deliver Low Latency, Low Loss, Scalable Throughput service (https://riteproject.eu/dctth/). And it's this L4S effort that was reviewed in the Transport Area Working Group session at IETF 111.

Things are moving along and following the publication of RFC8311, which lifts some of the earlier restrictions on the use of the ECN mechanism, there are four L4S working drafts nearing completion. There is an architecture and problem statement (draft-ietf-tsvwg-l4s-arch), an L4S identifier (draft-ietf-tsvwg-ecn-l4s-id), a

Diffserv Code Point (draft-ietf-tsvwg-nqb), a Dual Queue AQM framework (draft-ietf-tsvwg-aqm-dualq-coupled), and operational guidance to coexistence with "classic" ECN (draft-white-tsvwg-l4sops-00).

There are some issues with this effort to re-interpret the meaning of ECN signalling as these days there are already hosts and network devices using ECN marking along what could be called "classic" lines, and the introduction of L4S has issues with L4S hosts misinterpreting ECN signals as L4S marking, and L4S-aware network marking being mis-interpreted as ECN signals. Any effort to make such changes on an in-use system is always going to have these issues, and L4S is no exception. Of course, these days transport has some further complexities, including consideration of multi path transports (MPTCP) distributed congestion control transports (DCCP) and various forms of QoS to contend with. There is also the issue of both inadvertent and hostile signal manipulation, replay attacks and resequencing attacks, all of which requires some robust response for an L4S system.

As the buffer bloat exercise has pointed out, queue delay is now a major impediment to service performance, and it seems somewhat crazy to spend considerable effort to reduce the user experienced delays in areas such as DNS performance, TLS session setup and content delivery when the entire effort is negated by seconds-long network queue delays. We must move beyond a relatively crude common flow control mechanism based on congestion collapse and packet loss to control traffic flows and efforts such as L4S appears to offer some major promise. However, I cannot help being cynical about the prospects for this L4S network signalling approach in the context of the public Internet, and for the further prospects of ECN for that matter. It requires deployment across all networks and clients to perform marking and deployment across clients to echo ECN markings for the actions by the server to use this marketing to control the end-to-end flow. It's a replay of a now quite old network QoS story that its only truly effective it all network elements and all hosts perform the combination of marking, recognition and response. The more the deployment is piecemeal then the more the service outcomes cannot be assured. In other words, such coupled combinations of network and host actions work as intended if everyone plays from the same book. It's possible to orchestrate such collective actions in closed environments, but not so easy in the large and loosely coupled environment that we see in the public internet. When you compare this approach to BBR, which is a unilaterally applied server-side protocol that does not require any matching operation on the part of networks or even remote clients, then BBR looks like a far more viable approach is you want your TCP flows to avoid encountering the delays associated with the formation of standing queues in the network.

IPv6 Maintenance – 6Man

The IPv6 effort split into two streams in the IETF. The first was to consider operational issues associated with the deployment of IPv6, the V6Ops Working Group and the other was to coordinate minor tweaks to the V6 protocol, the 6Man Working Group.

One of the topics considered at the IETF 6Man Working Group was the perennial topic of Path MTU management. As the working draft of a new MTU option (draft-ietf-6man-mtu-option) points out, "The current state of Path MTU Discovery on the Internet is problematic. The mechanisms defined in RFC8201 are known to not work well in all environments." I would claim that as an understatement! This proposal is intended to replace reliance on the ICMPv6 Packet Too Big messages and replace them with a signalling protocol that is intended to discover the Path MTU. Presumably this applies to TCP and TCP-like protocols and would not be relevant for UDP transactions. Its operation is simple: when a router processes an IPv6 packet with this Hop-by-Hop Extension header it compares the MTU of the next hop interface with the minimum MTU value contained in the packet header. If the local value is smaller, then it rewrites the option value with this local value. Upon receipt at the intended destination the receiver is meant to pass this value back to the sender in a return signal.

This approach appears to have exactly the same issues of host and network orchestration noted in the issues associated with ECN and L4S deployment. IPv6 routers need to recognise this hop-by-hop extension header and perform MTU processing and potential rewrite the IPv6 packet extension header, and likewise hosts need to recognise this header within a TCP or TCP-like flow and return the signal back to the sender. Perhaps if the original design process of IPv6 had spent some time considering the issues with ICMPv6 blocking in networks they might have come up with this Extension Header approach. Or not. In the timeline of a deployed

technology the deployed environment exercises a considerable inertial force, and the greater the deployment the greater the resistance to incremental changes. This was a reasonable idea to have in 1993 when it might have been incorporated into the IPv6 design specification at the time, but while the concept is still an interesting use of IPv6 Extension Headers today, its chances of adoption at a useful scale in today's public IPv6 Internet look to be remote. Hence the draft's caveat that this mechanism is intended to be used in closed environments such as within Data Centres.

The whole issue of the viability of IPv6 Extension Headers in the public Internet has been discussed for a number of years in the light of periodically published measurement data showing considerable levels of packet drop when the packet contains an Extension Header. The draft document draft-hinden-6man-hbh-processing attempts to address some aspects of this issue by further defining expectations of node behaviour in processing these extension headers. The observation here is that while Hop-By-Hop (HBH) processing in first IPv6 specification was required for all nodes, there were obvious issues with the ability to process such packets at wire speed in hardware. The common design approach was to push packets with HBH options to the general processor engine (usually referred to as the "Slow Path") as a means of avoiding degraded router performance for all packets. The implication was that packets with HBH extension headers experienced higher latency and higher loss probability. Packets with multiple HBH options exacerbated this problem.

The current IPv6 Specification (RFC8200) has changed this HBH processing requirement to only require such processing if the router is configured to do so. This essentially documented current operational behaviour but did not improve the situation of the viability of using HBH Extension Headers in the public IPv6 Internet, and possibly was one further step toward the complete elimination of such Extension Headers in the IPv6 Internet. The goal of this HBH-processing draft is to try and reverse this situation and specify ways to make HBH processing viable. It advocates that where HBH headers are present then nodes must process the first of such HBH headers and may process more, if permitted by the local configuration. The expressed hope was such such a specification would allow a HBH header to be processed in the conventional processing path on a node and not shunt the packet to the Slow Path. It is unclear that this is viable advice to designers of packet switching hardware and the inexorable demands of faster and cheaper equipment would appear to preclude even this limited use of HBH headers as a common practice. The most viable current advice for using HBH headers in IPv6 packets in the public Internet is simple: don't!

In a similar vein, the proposal draft-herbert-6man-eh-limits advocates imposing limits on the number of extension headers, and limits on the number of options and padding in Destination Options and HBH options. The issue here is that such limites are probably contextual, in that practical limits for the public Internet would be different from what is supportable in more restricted local contexts. I'm not sure that this approach changes the overall poor prognosis for Extension Headers in IPv6 any more than the one and only one processed HBH Extension Header approach advocated in the other draft proposal.

Configuring a new host in IPv6 quickly became a confusing story. While IPv4 quickly adopted the DHCP approach where a server loaded the host with configuration parameters during the boot process, IPv6 was keen to adopt a serverless approach. So IPv6 adopted an approach using ICMPv6 Router Advertisements (RAs), but to provide other configuration parameters it also adopted both stateful DHCPv6 (RFC3315) and stateless DHCPv6 (RFC3736). And then the IPv6 working groups spent a large amount of time debating all kinds of proposed new options for these configuration methods. Part of the reason behind this debate is that any adopted change to these configuration options requires updates to both IPv6 hosts and the network's host management regime. A proposal is being considered for a universal configuration option (draft-troan-6man-universal-ra-option-05) where the options are contained in a self-describing JSON object, modelled on CDDL and encoded using CBOR. The RA or DHCPv6 functions contain an opaque configuration payload, whose configuration object contents are defined in an IANA registry. This seems like a useful step forward, although the discussion of this proposal was more along the lines of a "path not followed" discussion. Unless we are willing to replace RA and DHCP options with this then all we would be doing is adding a third configuration mechanism to already somewhat confusing melange of choices. In any case, I was particularly struck by the Security Consideration section in this universal configuration draft which asserts that: "Unless there is a security relationship between the host and the router (e.g. SEND), and even then, the consumer of configuration information can put no trust in the information received.". True, but like many parts of the Internet nothing

would work at all unless we behave with more than a little credulity in believing in good faith the random noise we pick up from the network!

BGP Operations – GROW

A problem shared is a problem solved? Yes? Probably not in the general case, but that doesn't act as a deterrent for many folk to present their problem and its proposed solution in the hope that everyone else will sign up to the solution. One of those local problems was presented at the GROW Working Group as a proposal to add local AS Looking Glass endpoint to the capability negotiation in an eBGP peer session (draft-jaufeerally-bgp-lg-cap-00). The idea is that a prefix advertiser would learn from the local BGP session a means to see over to the view from the "other side" of the BGP session and confirm that the advertisement had been accepted by the peer. While there is a common need to determine how a prefix is propagated in the inter-domain routing space, and there is useful work that could be done here, I'm afraid that this particular proposal doesn't appear to me to be part of any generally useful solution. The real objectives appear to firstly, be able to track and trace the propagation of a prefix advertisement in the inter-domain routing space, secondly, to detect what forms of alternation were made to the route object is it has been propagated and, thirdly, some hints as to why the remote BGP speaker made a local decision to further propagate or drop the route object. This kind of diagnostic information could, in part, be made available using BMP BGP traces, but the larger issue of exposing local routing policy decisions relating to the local handling of route objects may not have any direct solution.

BGP update messages contain a structured field called a "community" to express some form of policy preference or directive to be applied to ab update. Some are "well-known" communities which are supposedly uniformly supported such as "No-Export". Others are locally defined by a particular AS. The initial definition of a BGP community was a 4-octet field, which was large enough to encode a 16-bit AS number and a 16-bit value that is associated with some route policy or action by that AS. This was extended in the 8-octet extended community construct that used a 2-octet type field and a 6-octet value. With the introduction of 4-octet AS numbers the extended community construct was too small, and RFC8092 extended this with Large communities to use a 12-octet value. How widely are these more recent (Extended and Large) community types supported on the Internet? How widely are they used? How many AS's are seen to propagate route objects that have attached community attributed? The study undertaken by NIST and presented at the GROW session was interesting. Using the snapshots provided by RouteViews and RIS they observed some 3,000 ASes that propagate regular communities, but the number drops to under 500 for both Extended and Large communities. These latter two community types appear to be used to peer signalling and not broader policy attribute announcements. If we wanted to use Extended and Large communities as a transitive attribute, then what additional work and recommendations would be needed to facilitate this?

Oblivious HTTP BOF

The privacy program in the IETF continues, and there has been recent work in the concept of "Oblivious" services. The overall intent is to encapsulate service transactions in multiple wrappers such that no remote party, even the ultimate service delivery point has knowledge of the requestor and the request. IN the case of Oblivious DNS the stub resolver encrypts the query using the public key of the recursive resolver, and then encrypts this message using the public key of a proxy agent. A proxy agent, acting in a role similar to that of a conventional forwarder can decode the outer wrapper and pass the inner encrypted payload to the recursive resolver. Oblivious HTTP is proposed to operate in a similar mannar. The Oblivious HTTP proposal (draft-thompson-http-oblivious) describes a method of encapsulation for binary HTTP messages using Hybrid Public Key Encryption. Like Oblivious DNS, this protects the content of both requests and responses and enables a deployment architecture that can separate the identity of a requester from the request, as long as the proxy and the server do not collude in any way.

This is not the first introduction of this work into the IETF and the proposal was considered previously in the SECDISPATCH Working Group. There are a number of questions unique to this proposal that probably had influenced the IESG decision to discuss this further in a BOF rather than progress the work as a working draft in en existing working group, such as in the HTTPBIS effort. When the IETF considers a system for forming untraceable HTTP requests where the information is spread in a way that no single entity can re-join the requestor and the request then this has some deep implications in a number of areas, including the business

models of content provision that often rely on assembling an intimate profile of the end user in order to maximise the indirect advertising revenue, various public security agencies and the gathering of background intelligence data to name just two. In some ways it's not a change to the HTTP protocol per se, but a form of handling of HTTP sessions that obscure aspects of the session to each party involved in the ttansaction and to the network.

The questions raised at the BOF include the consideration of encrypted requests may have implications that require every content server on the Internet to be modified to be equipped with OHTTP handling functionality. On the other hand, there are so few distinct actual server and client platforms out there in this age of CDNs that this claim that "modifying everything would be impossible" is actually not as impossible as it might sound given the relatively small collection of distinct servers and the equally small number of versions of http browser client software. For example, if you modify Chrome's WebKit engine that what's left in the browser world aside from Safari that needs changing on the client side? This leads to a second concern that this is a step that in the name of increased user privacy actually fosters further pressure for centrality in the market, as it raises the barriers for entry for smaller providers. Another comment in the BOF questioned why this is even necessary given that the IETF could achieve a more generic outcome by standardising TOR. There was also the comment that the Masque proposal (draft-schinazi-masque-01) seemed to have been accepted as an IETF work item without much in the way of comment, and if Masque has not raised eyebrows, then why should OHTTP be singled out?

Given that much of today's Internet is the supporting pedestal for today's highly lucrative surveillance economy there is a real question as to the willingness of the various large actors in this space to voluntarily close down these observational windows in today's environment. There is little doubt that we *can* do it, but questions remain as to whether we *should* do it and *who* would be motivated to actually do it at all!

Securing BGP – SIDROPS

In a hierarchical PKI its quite easy to change keys. It's a dance you perform with your parent to inform them in a robust manner of the new key value and then perform the cutover. But when you want to change the apex key in the hierarchy, or the Trust Anchor key then the story is far more challenging. Conceivably, you could get everyone who uses the PKI to load a new key, but that's typically infeasible. The more conventional way is to get the outgoing, but still trusted, key to sign the incoming key. This is what we did with the recent KSK roll in the root zone of the DNS, and it is what is being proposed for the RPKI (draft-ietf-sidrops-signed-tal-07). This procedure is documented in RFC5011 in the context of DNSSEC, but its guidance is general for trust anchor key rolls. One feature is prominent in RFC5011 but missing in this draft, namely the use of times and timers. Without hold-down timers a hostile party who has compromised the trust anchor could use this key to sign a new key under the control of the attacker and then complete the roll process. At this point there is no way back other than the hard path of getting every user of the PKI to reload Trust Anchor keys from scratch and without any in-band assurance. With hold-down timers an attempt to hijack the key sequence can be revoked, assuming that the hijack is noticed within the hold-down period of the introduction of a successor key.

Key rolls are a challenging subject. The procedure works as long as the key is not compromised but can fail in horrible ways if the key is ever compromised. So, if the roll procedure can't necessarily get you out of trouble when you have fallen into a troubled spot, then why bother having a key-roll at all? There are some who might claim that a key sitting in an HSM suffers from bit rot over time! A less fanciful notion is that the more times a key is used the more information about the key value is inadvertently exposed. I have heard arguments in favour of this notion that over time use compromises the integrity of a key and arguments that discount it when the key is used less frequently, for some indeterminant value of "less". If the key is not compromised through use, and the key roll procedure cannot robustly handle the case of a key compromise then what is the goal here? It might just be that this is something you want to exercise very infrequently and do so as a counter to the increasing risks of algorithm compromise. It's not that the challenge to find the private key value given the public key and samples of encrypted output is completely impossible. It is theoretically possible to do so, but the idea is to devise a crypto algorithm that is *computationally infeasible* to reverse. As computers get faster, or change completely, as in quantum computing, then previously infeasible problems may become feasible. Perhaps in a practical sense it's this consideration that drives the need to define a key-roll procedure. You might

use it infrequently, but when it's time to use it, you really need to have a defined procedure at hand that you've prepared earlier!

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

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

www.potaroo.net