November 2021
Geoff Huston

IETF 112

Virtual meetings continue in the IETF, and the latest one was IETF 112 in November. Here's my notes from some selected working group meetings that caught my attention. These cover some of the topics that are not directly associated with the DNS, as I've separately commented on the status of IETF work on the DNS. Here I'll look at routing security, IPv6 and transport topics.

BGP

Security - SIDROPS

There are a few implications from using X.509 certificates as the base of a PKI, and one is an inclination to continue to use the ASN.1 data description formulation. Which is fine except that interpreting ASN.1 is a dark art, and without resorting to a set of odd and entirely unmemorable incantations to invoke the OpenSSL package to perform the decryption of ASN.1 object then the choice is pretty limited. I haven't yet tried the incantation "openssl -eye_of_newt -toe_of_frog" but it wouldn't surprise me if it worked! Ben Maddison has developed a Python tool that performs ASN.1 decoder and data dumper for ASN.1 encoded objects. It's a useful addition to the RPKI tool set. The code for this tool is at https://github.com/benmaddison/rpkimancer. Good one Ben!

The session also heard a report from the IETF hackathon, working on an implementation based on the ASPA drafts (AS Provider Authorization) as an aid to detect some forms of AS Path manipulation and mitigate some forms of route leaks (yes, that not exactly a ringing endorsement, but it's a complex area and even some levels of risk reduction is a Good Thing, so it's actually better than this somewhat tempered description would suggest!). It's hard to predict whether ASPA will gather deployment momentum at this stage. It's my opinion that BGPSEC is never going to be deployed, so if we want even a rudimentary form of AS Path checking then the ASPA work is all that's around. But the very slow progress with ASPA calls into question whether we are really interested in AS Path checking in the first place. All these technologies struggle with balancing increased operational complexity and fragility against the quantification of promised benefit, and while ASPA appears to have aimed at a less ambitious set of goals and supposedly done so with a simpler framework than BGPSEC, it's still not clear that ASPA has managed to get over this critical level of deployment resistance.

BGP over QUIC

There was an interesting proposal in the Inter-Domain Routing Working Group (IDR) to use QUIC as the transport protocol for BGP. The motivation is not necessarily to protect the BGP session from eavesdroppers, although QUIC can do that, nor to protect the BGP session from potential hijacker and disruptors, although QUIIC can do that as well. QUIC can perform mutual endpoint identity verification as a part of its TLS functionality, although once more this is not the primary motivation for using QUIC. In this case the main feature of QUIC is the ability to multiplex multiple streams into one session and not encounter the TCP-related issues of head of line blocking and the ability to control individual logical streams without the coarse response of a full session reset.

Using QUIC each AFI/SAFI becomes its own QUIC bidirectional stream, with its own OPEN, update stream and shutdown control, using the BGP MultiStream capability signal on BGP session start. All other aspects of the BGP protocol are unaltered by this change of transport.

The details are in draft-chen-idr-bgp-over-quic-00.html. This looks like a Fine Thing to do for BGP. So let's do it.

Dynamic Capability for BGP

Yes, the IETF can take its time to get some work through, but sometimes the process is so slow that it could be measured in the rings on tree trunks! This draft, draft-ietf-idr-dynamic-cap, was first submitted as a working group draft in August 2001. Yes, that's twenty years and three months ago. Yes, it was in hibernation from 2011 to 2021, but in terms of content nothing much changed over that protracted hiatus.

The concept is simple. These days we are all loathe to tear down a BGP session as it causes unacceptable service interruptions. But capability negotiation only occurs in the OPEN message exchange, so it you want to change the session capabilities you need to tear down and restart the BGP session. The work defines a new BGP capability termed "Dynamic Capability", which would allow the dynamic setting of capabilities over an established BGP session by using a Capabilities Message. This capability would facilitate non-disruptive capability changes by BGP speakers.

This seems like another Fine Thing to do. I have no idea why it's taken two decades to bubble to the surface, as it's just one more of those "Why weren't we doing this from the start?" kind of ideas.

IPv6

There is continued interest in tweaking various parts of the IPv6 design. As well as source routing, at this IETF meeting we heard about proposals to add alternate marking and augmenting the Extension Header set to carry a VPN identification field. Neither of these proposals have gained much traction and engagement in the IPv6 working groups as far as I can tell, and perhaps this is a broader reflection on the lack of a common desire to make further changes to IPv6 at the moment. Further ornamentation of the IPv6 protocol and its packet header does not appear to address novel and otherwise unaddressed vital use cases, but to simply discuss at some considerable length alternate forms of packet ornamentation for their own sake.

It seems that the protocol is being torn between the conflicting desires of one part of this industry that wants to increase the scope and value of the network layer and another part that wants to commoditise and essentially freeze the network layer and focus on the application space as the vanguard of evolutionary change and value creation. The network is now at a size where it has accumulated considerable stasis at the lower layers, and not even SDN has been able to inject much in the way of flexibility back into the network. Most of these activities look to me to have little likely impact for many years to come, if any. But that has not stopped the IETF from continued IPv6 tweaking.

Source Routing

Some topics just never go away, and various forms of source routing is one of these perennial topics in networking research. In a hop-by-hop destination address based forwarding paradigm only the destination address is relevant to the network, and a case can be made that the role of the source of address in the IP packet header is a purely cosmetic one! Any form of reverse signalling from the network is inherently untrusted and with internal forms of packet encapsulation, including MPLS, the entire concept of what is being signalled and to whom when such a reverse signal occurs is called into question. There was a reason why ECN signalling is a forward signal!

Despite this, there have been persistent calls for some form of a source-address routing scheme where the intended path taken by the pack through the network is specified in part or fully by the packet at the point of entry into the network. I presume this is intended to address a perceived need for policy-based network segmentation for policy, traffic engineering or service quality reasons where the packet sender assumes some control over the network's handling of the packet, or the network itself makes different forwarding decisions based on some assumption about the identity and authority that has been given to the packet's sender.

The IPv4 packet options, including time stamping and loose and strict source routing, never found much traction outside of highly specialised environments, and they certainly play no role whatsoever in the public IPv4 Internet. The requirements for raw switching speed over all other considerations mandated a very simple packet processing flow where IPv4 packets loaded with such options were either quietly ignored or dropped without further trace.

The IPv6 specification moved these options into the so-called Extension Headers, which in the case of all bar fragmentation controls was a cosmetic relabelling of these IPv4 options. Again, the pragmatic observation is that Extension Headers appear to have a role only in specialised environments, for similar reasons including switching speed and security considerations. But such mundane considerations have never really stopped the IETF's ability to ornament technology by adding generally pointless detail and complexity that, at best, has a limited role in highly specialised environments, if at all. Yes, if you are a hardware vendor flogging switching equipment, then packing a unit with all this functionality can certainly add to the supposed value being invested into the network infrastructure and provide some justification for upping the price of what otherwise would just be commodity silicon performing a simple utility switching function, but as far as I can see it's a vapid value proposition in the larger scheme of things!

There is a broader tension at play here as well and it concerns where and how functions are implemented into the networking environment. There is a continuing pressure to add features and functions into the network level. From this perspective the network is the overall platform orchestration component, and the upper-level applications signal requests to the network for specific functions, such as path diversity, explicit path selection and similar. The other side of this tension is for the application to treat the network as a collection of essentially featureless 'dumb' pipes that perform destination based forwarding and nothing else. Functionality is loaded into the application and the application is meant to adapt to the network conditions as the application finds it at the time.

The SPRING effort has been the focus of much attention, particularly from the vendors of network equipment. It's an option question as to whether this attention has resulted in a simpler framework that looks deployable, or whether the addition of more features, behaviours, and service responses adds complexity without much in the way of common benefit in the general use case. SPRING is complex enough that the issues discussed in IETF 112 cannot be readily summarised in a few paragraphs, so I won't even attempt it here, but on the other hand it’s entirely unclear at this stage if SPRING is a useful tool for the larger Internet or a niche behaviour of limited applicability and relevance. Personally, I tend to the latter view!

IPv6 Fragmentation Reconsidered

One proposal was however quite fundamental in nature, and it was one that attracted little in the way of comment at the 6MAN working group session. This is the proposal to augment IPv6 to permit fragment retransmission.

This is a relatively novel concept for the IP architecture. IPv4 permitted routers to fragment a packet in flight, in order to adapt to the limitations of various networks through which an IP packet may have to pass. It was the task of the destination host to reassemble these IP fragments back into the original IP packet and then pass this reassembled packet payload to the upper transport layer. The destination host uses a reassembly timer, and if the timer expires before all the fragments arrive, then the fragments are discarded. The host may or may not respond with an ICMP fragment reassembly time exceeded message when it does so. At this point the sender needs to resend the entire packet, either in response to the ICMP message or upon expire of some sender timer, assuming that it has kept a copy of the packet and its willing to resend it.

This additional behaviour and the inefficiencies it introduced was part of the justification for the view that was solidifying in the late 1980’s that IP fragmentation should be avoided. This view was part of the reason why the IPv6 specification limited IP fragmentation to a source function. Routers should not be gratuitously fragmenting IPv6 packets. This new work (draft-templin-6man-fragrep) proposes to add a couple of additional fields to the IPv6 fragment header, and a new ICMPv6 message, a fragmentation report, that details the received fragments of an IPv6 packet. The result is that a destination can explicitly signal which fragments have not been received so that the sender can resend just those fragments.

I’m struggling for context here when I consider this in the light of transport protocol behaviours. In UDP, where there is no explicit acknowledgement of packet receipt, the sender immediately discards the packet once sent, so the receipt of a lost fragment report would make no difference to a UDP sender. The sender could hang on to sent UDP packets for some period, but without explicit knowledge of the network delay then the sender is left guessing as to how long this local retention period should be, and the additional requirement to retain sent packets may be a performance constraint for high-speed high-volume transactions. In TCP one wonders what the gain here might be as compared to the combination of path MTU discovery and selective acknowledgement (SACK). The existing TCP mechanisms appear to offer the necessary functionality without the need for new mechanisms associated with fragmented segments.

Extension Headers

The debate over IPv6 Extension Headers continues. There is an undeniable tension between network equipment that wants to perform a basic forwarding function in a minimum number of hardware cycles and proposals to allow IP packets to specify a more complex network response to the packet. The experience with IP options in IPv4 has enjoyed no lasting success in the public Internet. Why should IPv6 be any different?

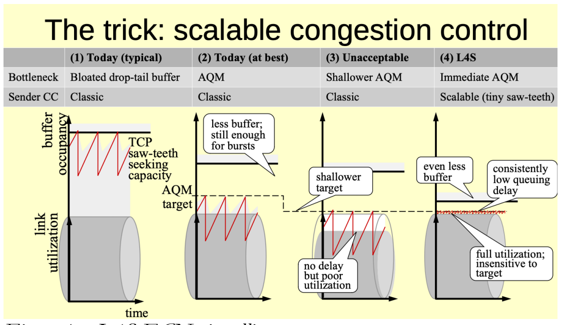
We've gone through a number of iterations in this process. Firstly, there have been efforts to deprecate the Fragmentation Extension Header, which founded. Then there was the notion that we should just avoid using all Extension Headers as they are not well supported on the public Internet. The latest conversation on this topic is narrowing in on the Hop-by-Hop extension headers and advocating deprecating just these Extension Headers. After all, these are the options that expose the network's infrastructure to potential resource exhaustion attacks. Deprecating these headers on the basis that there seems to be little of value in such HBH options and much that could be used to attack the network appears to make sense. Right? Well, no! Because there is always someone who is using these options and wants to continue to do so. It might not be the public Internet, but the protocol specification is meant to be general purpose, not just for one realm of use.

I must admit here that erring on the side of extreme simplicity and highly limited option set (even to the extent of no options at all!) appeals to my sense of a robust minimal design architecture. Yes, there are then things that the IP layer would find it hard to perform without explicit IP options and signals, but that's why upper layers exist. Hanging on to cruft in specifications just in case someone might want to use it someday seems to me to be a pretty retrograde position that results in more complex and less robust service outcomes for our networks.

Transport

L4S

The work on standardising Low Latency Low Loss Scalable throughput (L4S) services (https://riteproject.eu/dctth/) continues in the Transport Services Working Group and three documents have been through Working Group Last Call. Part of the effort is to perform some re-definition of the Explicit Congestion Notification signal (RFC3168) so that the ECN signal is generated at the start of queue formation rather than at the point where congestion loss is imminent (Figure 1). It's a slow standardization path for a high-speed architecture, and the working documents have now attained a five-year maturity level in the working group. Unlike BBR, which is a sender-side only approach the L4S approach is attempting to change sender and receiver behaviour as well as change ECN-aware network behaviour. It's a much more ambitious agenda when everybody needs to make changes. Our track record of making such widespread changes is not exactly impressive in recent years, and I suspect that this effort is on a similar path of deployment failure, irrespective of the obvious merits of a faster network.



The trick: scalable congestion control

	(1) Today (typical)	(2) Today (at best)	(3) Unacceptable	(4) L4S
Bottleneck	Bloated drop-tail buffer	AQM	Shallower AQM	Immediate AQM
Sender CC	Classic	Classic	Classic	Scalable (tiny saw-teeth)

buffer occupancy
link utilization
time
TCP saw-teeth seeking capacity
AQM target
less buffer; still enough for bursts
shallower target
no delay but poor utilization
even less buffer
consistently low queuing delay
full utilization; insensitive to target

Figure 1 – L4S ECN signalling

MP-DCCP using BBR

DCCP was a hybrid approach of attempting to combine a TCP-like congestion control algorithm with a UDP-based data flow. The anticipated benefit was that volume based USP flows could coexist in a shared network with concurrent TCP flows without collateral damage to either class of traffic. This work, presented in the Internet Congestion Control Research Group (ICCRG) session attempts to take Multi-Path DCCP and use the BBR flow control algorithm as the congestion control algorithm.

It's an interesting approach and the early results have been promising with lower latency and good traffic distribution across multiple paths according to the individual path capacities. Compared to TCP BBR, the phase of BBR where the sender probes the path capacity and then adjusts its sending rate is not well coupled in DCCP and the probe phase results in a delayed low throughput interval. Changing the post-probe recovery rate for DCCP appeared to address the issue and restore some parity between TCP BBR and MP-DCCP BBR performance.

Of course the obvious question here is: why not use a variant of QUIC without sender retransmission?

The Game Theory of Congestion Control

I was impressed by the performance of BBR when I looked at this control algorithm some years ago, and I wasn't the only one. Evidently many services now have switched to use BBR, and the current metric is that close to 18% of the Alexa top 20,000 websites are now using BBR. With the introduction of CUBIC in the 2000's many sites switched to CUBIC for improved performance, and we are now seeing a similar transition to use BBR. The question posed in this presentation to the ICCRG session was to speculate on the next steps.

There are some starting assumptions here, namely that a small number of BBR flows will often achieve higher than a fair share flow rate when competing against CUBIC flows. However, when there are short CUBIC flows in a larger environment of BBR flows, then CUBIC will perform well because of its propensity to use the network queues to improve its throughput.

To the researchers at the University of Singapore this looked like a situation that could be modelled as a Nash Equilibrium and their conclusion was that we will continue to see a mix of CUBIC and BBR in the Internet. If there is a Nash Equilibrium point then at such a point of maximal individual optimality, then if any current CUBIC flow switched to use BBR then all the BBR flows will perform marginally worse than before, and similarly if any BBR flow switched to use CUBIC than all CUBIC flows will fare worse as a consequence.

BBRv2

Work on BBR continues and BBRv2 is now reported to carry all internal production TCP traffic within Google. BBrv2 incorporates ECN signals, loss, bandwidth estimation and Round Trip Time (RTT) measurements to guide the control adjustments. However, there is a subtle caveat to this observation as some of this traffic is now being carried using a variant, they're calling BBRv2.Swift. This variant includes a slightly different RTT measurement, namely Network_RTT, that excludes receiver-side packet delays. They see this as useful in intra-datacentre environments (which I guess is much of Google's internal environment).

Their external traffic, which has a large YouTube component, is currently transitioning to BBRv2. The A/B experiments that are being conducted point to lower RTTs for BBRv2 as compared to both BBRv1 and CUBIC, and a reduced packet loss rate that is closer to CUBIC's levels. Given the observations relating to Game Theory and Nash Equilibrium it's hard to tell if the dominant factor here is the interaction with other BBR sessions, or the interaction between CUBIC and BBR sessions and the radically different responses of how to cope with network queue contention. I also assume that the ECN behaviours that they assume in BBRv2 is imminent loss signalling as described in RFC3168, and not the queuing onset signalling being defined in the L4S effort. Were the ECN signalling behaviour to change then it looks like BBR would also need to change its interpretation of the signal.

BBRv2 has a different flow profile than the original BBR and rather than a regular pulse and drain behaviour every eighth RTT it performs separate probes for path capacity and RTT, as shown in Figure 2.

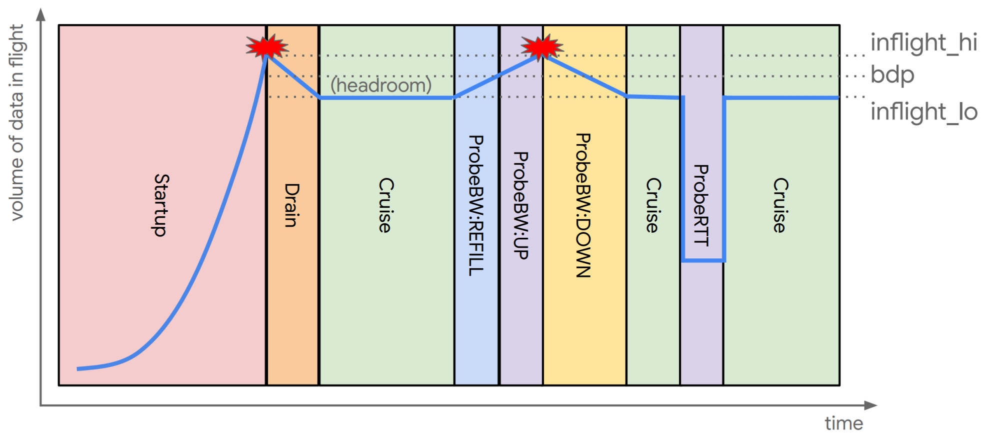


Figure 2 – Example Idealized Data Flow in BBRv2
(from https://datatracker.ietf.org/meeting/112/materials/slides-112-iccrg-bbrv2-update-00)

For those wanting to play along at home, BBRv2 is now available as an alpha release (https://github.com/google/bbr/blob/v2alpha/README.md).

There is also work on tuning BBRv2 as a QUIC control algorithm. It has involved some issues with a low initial estimate of the bandwidth delay product of the path when packet loss occurs in startup due to a limited congestion window value (cwnd) in the QUIC client. The PROBE-UP phase can exit early due to network queue build up. The fix is to maintain the PrOBE_UP state for at least one minimum RTT and to reach the point where the data in flight in this round is 25% greater than the estimated Bandwidth Delay Product (BDP) of the network path. In other words, any data still in flight from the prior BBR state is not counted in the PROBE-UP interval. Finally, there is a fix for intermittent data flows at source that pushes BBR into an idle state. If this occurs during the PROBE_RTT phase the resumption of data would continue in the suspended RPOBE_RTT state. The fix is to exist this reduced transmission state if the elapsed time is long enough even if no further data has been sent.

TCP Congestion Control

Despite any impressions to the contrary that you might have picked up from this report so far, BBR is not the only show in town, and there is much work still taking place in loss-based TCP congestion control algorithms as well. The material presented in the TCP Maintenance and Minor Extensions (TCPM) is a good window on these related activities.

I'll avoid going into detail here, as there is much to consider when you lift the lid on congestion control algorithms, but instead provide an over view of a couple of the topics considered in the TCPM Working Group session.

The first topic is revisiting RFC6937, Proportional Rate Reduction (PRR) an experimental specification that proposed replacing the fast recovery and rate halving algorithm. The intention is that PRR accurately regulates the actual flight size through recovery such that at the end of recovery it will be as close as possible to the slow start threshold (ssthresh) value expressed as a segment count, as determined by the congestion control algorithm. The major intention of this revision is to shift this specification from the Experimental to Proposed Standards track, but there are some other details being addressed at the same time. Fast Recovery is the reference algorithm for the sender adjusting its sending rate in response to a signal of packet loss. Its goal is to recover TCP's self-clocking by relying on returning ACKs during recovery to clock more data into the network. The Strong Packet Conservation Bound on which PRR and both Reduction Bounds are based is patterned after Van Jacobson's Packet Conservation principle: segments delivered to the receiver are signaled through the ACK stream and this signal is interpreted as the clock to trigger sending the same number of new segments into the network as have left the network. (It should be noted that this algorithm uses segment counting and not byte counting. Yes, that's an important distinction!) As much as possible, PRR and the

Reduction Bound algorithms rely on this self-clocking process and are only slightly affected by the accuracy of other estimators, such as pipe [RFC6675] and congestion window management (cwnd). This is what gives the algorithm its precision in the presence of events that cause uncertainty in other estimators.

CUBIC is a standard TCP congestion control algorithm that uses a cubic function instead of the linear window increase function on the sender side to improve scalability and stability over fast and long- distance networks. It has been in use for more than a decade and has been adopted as the default TCP congestion control algorithm in a number of platforms including Apple, Linux and Windows systems.

CUBIC is designed to behave in a manner similar to TCP Reno in smaller networks (networks with a small Bandwidth-Delay product and networks with a small RTT). CUBIC uses a more aggressive window inflation function for fast and long-distance networks, addressing a critical weakness in the Reno algorithm. The recent changes in this work include the recommendation to use HyStart++ to avoid large scale overshoot in slow start, the possible use of Proportional Rate Reduction in response to packet loss, and the use of the number of segments in flight (FlightSize) instead of the sender's congestion window (cwnd) to recover after a congestion event, unless the segments in flight has been limited by the application.

IETF 112 Materials

More information on the agendas of the working group sessions, the presentation materials, working group minutes and pointers to recordings of the session can be found at the IETF meeting agenda site: https://datatracker.ietf.org/meeting/112/agenda. Obviously, there is much I have not touched upon in this brief summary of some of the sessions that were held during the meeting.

IETF 113 is scheduled for 19 – 25 March 2022. The latest update on whether it will be fully online meeting or a mixed mode meeting is at https://mailarchive.ietf.org/arch/msg/ietf-announce/jcVD9xgDAj8FhX5u1FjnGhdn1mQ/. It seems that the situation is still somewhat uncertain in the light of further waves of COVID-19 infections that are sweeping the world.

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