November 2023
Geoff Huston

Measurement and Analysis of Protocols at IETF 118

The IETF met in Prague in the first week of November 2023. I attended the meeting of the Measurement and Analysis of Protocols Research Group, and here are my impressions from that meeting.

QUIC Topics

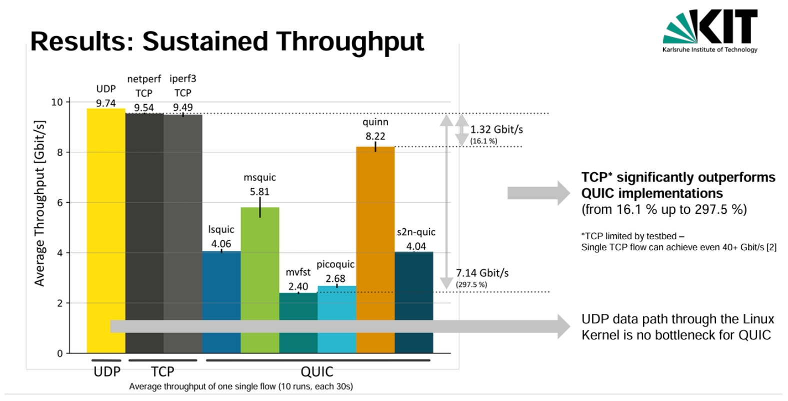
QUIC is essentially a variant of TCP, but at the same time it also represents a different approach to a transport protocol. To create greater levels of flexibility at the transport layer, such as removal of head of line blocking, reliable remote procedure calls and in-session address agility, the approach used by QUIC is to use UDP as the base protocol and implement a modified form of TCP into the user level application space. The reason for this multi-layering of the transport protocol over UDP is that the Internet is now quite ossified, and if you want to deploy a new general purpose end-to-end protocol it has to work seamlessly with existing CPE devices, NATs, DNS infrastructure and various packet filters. If there was ever a window that would allow the deployment of a new transport protocol on top of IP that window is now firmly shut. QUIC uses UDP as a minimal abstraction of the underlying IP datagrams service.

QUIC Performance

There are two transport congestion control algorithms generally supported in QUIC implementations, namely CUBIC and BBR. In a benchtop experiment, reported by Karlsruhe Institute of Technology's Roland Bless the researchers tested the Linux implementation of CUBIC against a collection of QUIC implantations. The QUIC implementations used in this test include Microsoft's *msquic*, Litespeed's *lsquic*, Meta's *mvfst*, Amazon's *s2n-quic*, *picoquic* and *quinn*, and comparing their performance against UDP and the CUBIC kernel implementation in *iperf3* and *netperf*. The testing was a simple benchtop configuration of server and client using a 10G connection.

Why perform this comparative test? CUBIC is CUBIC. Right?

Well, no! None of the tested QUIC implementations match the performance of conventional TCP with CUBIC in this setup, as can one seen in Figure 1.



Results: Sustained Throughput

TCP* significantly outperforms QUIC implementations (from 16.1 % up to 297.5 %)

*TCP limited by testbed – Single TCP flow can achieve even 40+ Gbit/s [2]

UDP data path through the Linux Kernel is no bottleneck for QUIC

Figure 1 - Comparison of TCP and QUIC (from https://datatracker.ietf.org/meeting/118/materials/slides-118-maprg-sustained-throughput-performance-of-quic-implementations

The presentation explored likely reasons behind these performance differences, which differ between the various QUIC implementations. One obvious difference is that QUIC uses TLS, and the overheads of this additional data processing may negatively impact QUIC performance. They tested the two fastest QUIC implementations comparing the performance with crypto to a non-crypto version. In the case of *msquic* the performance increased from 5.81Gbps to 6.92Gbps and for *quinn* it lifted performance from 8.22Gbps to 8.82Gbps when they used a null crypto algorithm. The conclusion was that the additional overheads of cryptography and the inefficient use of CPU resources by QUIC implementations imposed a performance penalty in many cases. However, the outstanding performance of *quinn* in this test appears to indicate that the performance differences in other QUIC implementations are likely to be based on implementation issues rather than some intrinsic set of issues with the QUIC protocol.

This result of a slower outcome for QUIC, as compared to TCP, does not appear to correspond to the actions of the major content streamers, including Google, Meta and Cloudflare, where there has been a major shift in recent years to use QUIC for their service delivery needs. However, the picture in the Internet is nowhere near as controlled as the results in a lab environment. Also, QUIC servers can select their own flow control protocol and in many cases for these high volume servers they appear to prefer to use BBR due to its ability to exert a consistent flow pressure in shared networks and not experience flow collapse in the event of consistent packet loss.

QUIC SPIN

One of the changes that the IETF QUIC Working Group made to Google's offering was the exposure of the "spin bit".

When Google offered the IETF the opportunity to take the work on QUIC and produce an open standard that could be used by all, it excited a debate within the IETF as to how much transport information should be deliberately occluded from the network. The general principle used by QUIC appears to be to expose as little as possible, and in the short form QUIC header what's left is basically a connection identifier and a packet number. The IETF decided to add a further single bit to this open part of the QUIC header.

This bit, the "spin bit" is intended to be used by passive observers on the network path to expose the round-trip time of the connection. The management of the bit's value is simple: the server simply echoes the last seen value of the bit in all packets sent in this connection. The client echoes the complement of the last seen bit when sending packets to the server. The result is that when there is a continuous sequence of packets in each direction this spin bit is flipped between 0 and 1 in time intervals of one Round Trip Time (RTT). Not only is this RTT time signature visible at each end, but it is visible to any on-path observer as well.

Is exposing this spin bit "safe"?

Some claim that exposing this bit, and the associated ability by onlookers on the traffic path to derive the connection RTT, is reasonable and the information exposed here does not represent any particular harm to the user.

Others take the view that the gratuitous exposure of any information beyond the IP header and the base essentials of a UDP header is both unnecessary and unsafe. There is no compelling protocol reason for this spin bit to be exposed, and there is an unknown factor in how this bit may be used were it to be added to the protocol.

Is the SPIN bit actually being used in the wild? A presentation at MAPRG suggests that the use of the spin bit is limited to around 10% of domains. Notably Google and Cloudflare-hosted servers appear to eschew the spin bit. It also notes that the RTT estimated based on the spin bit can be quite inaccurate, overestimating the RTT by up to 4x the actual RTT.

So the SPIN could hardly be judged as a successful innovation on the part of the IETF.

QUIC ECN

A similar story exists for Explicit Convention Notification (ECN) where there appears to be a marked reluctance to support ECN in QUIC implementations. ECN can have a profound impact on the performance of loss-based congestion control algorithms, such as CUBIC, providing an early indication of network buffer formation well before the loss-control that occurs with the buffer overflows. The path to a more efficient network that does not rely on transport sessions pushing the network infrastructure into buffer overload and packet loss relies on early onset signalling in the form of ECN. If the end-to-end transport protocol ignores this signal, then there is little point for network routers to support ECN signalling.

The observed picture for ECN use in QUIC parallels the picture of the use of the Spin bit, in that uptake is somewhat sporadic at best.

More DNS Badness

Many of the security issues for the DNS environment happen at the edge of the network. One of the biggest issues a decade or so ago was the proliferation of "open" DNS resolvers, which would provide a DNS resolution service to anyone.

These open resolvers were edge-based resolvers that were configured to respond to queries from both the "inside" and the "outside" of the edge network that they served. They were extensively abused to launch distributed denial of service attacks, and number of projects were spun up to track the proliferation of such open DNS resolvers.

A variant of the open resolver is the so-called "transparent forwarder." This device takes received DNS requests and forward them to a recursive resolver, but leaves the source address of the forwarded query as the source of the original querier (Figure 2).

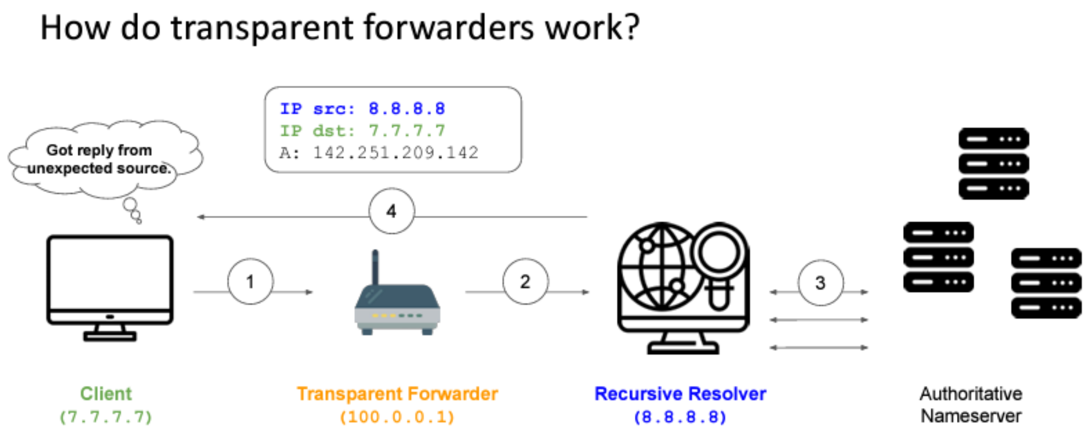


Figure 2 – Transparent Forwarders (https://datatracker.ietf.org/meeting/118/materials/slides-118-maprg-transparent-forwarders-an-unnoticed-component-of-the-open-dns-infrastructure)

It appears that this is a degenerate behaviour mode of an edge device (CPE) that combines a NAT with a DNS forwarder. If a DNS query is received from the "inside" of the CPE the device will rewrite the source address with a NAT binding address and send the packet on to a recessive resolver. But if the same device accepts a DNS query from the "outside" of the CPE it will not perform an address transform of the source address and simply forward the query on to the recursive resolver. Such behaviour can be readily exploited in DNS amplification attacks. A recent scan detected some 2M such open transparent forwarders in the Internet.

The CPE environment is a classic example of a highly cost-constrained market sector that is generously populated with low quality products, to be point of being unsafe to use. When such products are used in high volumes in the deployment of retail services it quickly becomes a significant problem for everyone.

More information on such open transparent DNS forwarders can be found at odns.secnow.net.

In another DNS census study it was found that some misconfigured DNS resolvers that apparently operate promiscuously as open DNS resolvers lie in some of their responses! It's hard to be surprised by this result.

Measuring Dropping of ROV-Invalid Prefixes

There are some inherent issues in attempting to measure the extent to which networks are dropping routing updates for prefixes that fail Route Origination Validation.

The problem is that if you use a single vantage point for a ROV-based measurement, or even a small set of vantage points, then if any network on the path between the measurement target and the vantage point is performing such ROV-Invalid filtering then it looks as if the target is performing the filtering.

If we want to identify individual networks that are performing ROV-Invalid filtering, then we have to use a more concentrated form of measurement. The ideal measurement situation is to place an instance of an anycast measurement sentinel point collection just one AS hop away from every AS, and enrol test clients inside every one of these ASes. If the test client cannot send a packet to this sentinel, then its reasonable to infer that it has no route to this address, and the likely cause is that the test network is dropping ROV-Invalid routes (Figure 3).

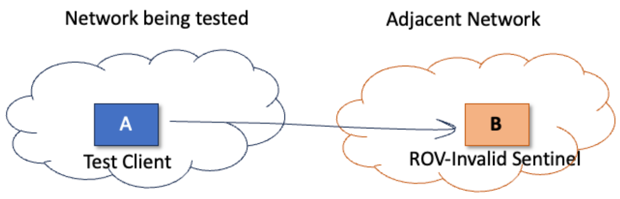


Figure 3 – ROV-Invalid test

The problem is that setting up some this population of AS-adjacent test and sentinel devices across all 75,000 ASes to test is a daunting challenge. Many of the measurement frameworks that look at ROV-Invalid drop (including the RPKI measurement platform operated at APNIC) try to mitigate this situation by using a large pool of vantage points, and enrol a very large collection of clients to perform the measurement, but in such a measurement scenario there are still a large number of multi-AS transit paths and the results are not always able to clearly identify the ROV-Invalid filtering status of every network.

Are there other ways to perform this measurement? At its core this measurement is performed as a simple reachability test: Can Host A pass an IP packet to Host B?

Like all simplifications, this elides one important aspect, namely that the route used by the network that houses Host A must use an explicit route for the network that houses Host B. If the tested network uses the default route to reach Host B, then we are none the wiser.

There are a number of other ways of performing this form of reachability measurement. One approach, used by ROVISTA, is to conduct this reachability measurement using a remote observer. The approach leverages those hosts who implement the IPv4 identifier in every sent packet as a simple counter. The measurement observer sends a TCP SYN packet to a sentinel that uses IP identifiers. The observer will receive a SYN/ACK response with an IP identifier field. The observer then sends a SYN packet to the test client host, using the spoofed source address of the sentinel. The test client host will generate a SYN/ACK response, but it will send this packet to the sentinel. As the sentinel never sent the SYN packet, it will respond to this SYN/ACK with a RST, incrementing its sending IP identifier value when sending this RST packet. If the observer then sends another SYN to the sentinel, then the IP identifier in the SYN/ACK response will tell the client whether the target can reach the reflector or no (Figure 4).

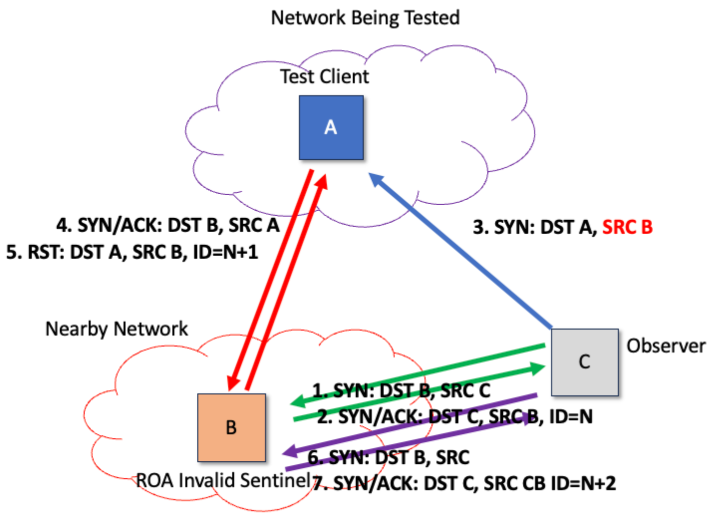


Figure 4 – ROVISTA ROV-Invalid test

This is certainly a novel use of the IPv4 identifier field, but I'm not sure it gets around the basic problem of needing to set up a large set of paired target and reflector hosts if you want to distinguish between those networks that perform ROV-Drop Invalid directly (ROV Deployed) and those that sit behind transit networks that perform this drop function (ROV Protected). The observer still needs to be able to send packets to a ROV-invalid destination address, and also the observer's host network needs to accept packets with spoofed source addresses. The routing relationship between the test network and the sentinel network is also not apparent in this measurement methodology, so the use of default route can confuse this signal.

However, this uncertainly calls into question the aim of the measurement. Are we trying to measure the extent to which ROV-invalid routes are blocked from **propagating** across the routing space? Or are we measuring the population of individual networks that will not learn a ROV-invalid route?

MAPRG Session Materials

The MAPRG session was recorded and can be viewed here. Meeting materials are also available on the IETF 118 web site.

Disclaimer

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