March 2018
Geoff Huston

Just One Bit

I'm never surprised by the ability of an IETF Working Group to obsess over what to any outside observer would appear to be a completely trivial matter. Even so, I was impressed to see a large-scale discussion emerge over a single bit in a transport protocol being standardized by the IETF. Is this an example of a severe overload of obsessive compulsive behaviour? Or does this single bit represent a major point of design principle, and was the extended discussion about that design principle rather than the use of the bit itself?

The transport protocol under consideration here is QUIC. QUIC was originally developed by Google and is in use by their Chrome browser and by various Google servers. Given the extensive use of Chrome in the Internet and the extensive use of Google services by Internet users, the obvious corollary is that QUIC is used extensively in the Internet. QUIC is a version of an end-to-end transport control protocol that eschews the traditional use of TCP and uses UDP instead. However, QUIC behaves in a manner that is approximately consistent with TCP behavior, and it is intended to function more predictably and potentially produce better outcomes than TCP in many conditions The way it achieves this feat is to treat the UDP transport layer as identical to IP itself, which is largely the purpose of UDP, and use a TCP-like protocol as an "inner" protocol one level lower than UDP. The devil is in the details here, and in this case the detail is that this UDP payload is encrypted. This means that QUIC's pseudo-TCP session control information is hidden within the encryption veil, and the shared TCP state is a secret that is shared between the two end systems of the communication but occluded from all other parts of the network.

Now if you last saw a description of the Internet architecture back in 1990 and you've been sleeping since then, then this deliberate occlusion of the end-to-end transport protocol would not raise your eyebrows at all. The network's switching systems were only supposed to look at the outer IP packet header, and the inner payload was to be exclusively used by the two end systems. (This explains IP packet fragmentation behaviour, where the IP packet header was replicated across all the fragmented packets, but the inner transport protocol header is only contained within the first fragment.) If networks are not meant to look further into an IP packet header than just the IP header, then why should it matter whether the transport layer protocol is hidden by encryption or not?

However, theory and practice have headed down widely divergent paths over the past thirty years. Since 1990 network operators of all shapes, sizes and roles have become accustomed, or even addicted, to seeing deeply inside the IP packet. The firewalls that are ubiquitously deployed in today's network use the inner transport protocol port numbers to guide the accept or reject decisions. Then there is the NAT function, where the 5-tuple of protocol, source and destination addresses and the source and destination port numbers is used as a lookup vector into a translation table, and both the IP and the inner transport packet headers are altered by the NAT before passing the packet onward. This inspection and potential modification of the transport headers goes further than just NATs. Many network operators use the IP and transport packet headers to perform traffic engineering functions, packet interception and forced proxy caching. Various forms of middleware may reach into the TCP control fields and manipulate these values to modify session flow rates. All of these activities are commonplace, and some network operators see this as an essential part of their service.

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 proposal that was considered at the QUIC Working Group meeting at the recent IETF 101 meeting was simple: 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. The sad history of meddlesome middleware appears to support this cautious view, where what was seen as a helpful intervention to gratuitously leak information becomes a point of potential breakage and even a potential data breach. Their view is that even the deliberate exposure of one purportedly harmless bit takes QUIC down an irreversible path. They point to the advocacy of the use of 2, 3 or even more of these bits that could be used to expose packet loss rates, jitter or more. At some point these bits expose the same level of session control information that was in the now encrypted TCP header and the entire point of QUIC's efforts to elude the meddlesome and ossifying grip of network middleware is lost.

Let's raise the level and think about the larger issues that are exposed by this debate within the IETF QUIC Working Group for a second. Is it even possible for these large disparate groups of people to exercise careful and considered constraint in the area of protocol design? Can a set of highly focused technicians looking at a technical matter of protocol behavior also factor in a broader understanding on the compromises between privacy and exposure in the domain of a public communications realm and reach a common understanding of where and how to balance these factors? It sure looks unlikely these days.

But these issues are by no means recent matters. When we look at the original Internet Protocol design we see evidence of similar balancing of such factors, and the decisions made at the time would probably never be made in today's context. As an example of these changing circumstances, the open mutual trust assumptions that underpinned many of the Internet's protocol developments in years gone by are being turned against us in the forms of hostile crippling attacks. Were these protocols poorly designed at the time? Would we have thought about the protocols in a different light had we factored in the assumption that the environment of deployment would be both truly massive and truly toxically hostile?

Are we asking too much of the IETF in its efforts to undertake a group grope towards some ill-defined concept of "rough consensus"? We should bear in mind that the process of attempting to accommodate widely divergent motives and produce a coherent outcome suffers from the same fundamental flaws as the design process that lead to the extremely odd design choice of the 53-byte ATM cell. Sometimes difficult choices admit to no sensible compromise between them and the process simply has to make a contentious decision one way or the other. While individuals make such decisions all the time, the collective process that enlists a large group of diverse perspectives, interests and motivations finds such decision making extraordinarily challenging. Little wonder that they often fail at the task.

Was it a helpful action by Google to push their implementation of QUIC into the open source domain and request the IETF process to produce a standard open specification? It has certainly triggered a useful conversation in the IETF's transport area about the degree to which applications and end-to-end data flows

should be exposed to the network, or even whether any level of exposure is just too much, but at the same time it is unclear how the IETF can use its processes to make difficult decisions about where to draw the lines here.

What emerges from all this is the observation that if you want to expose your session control state to the network, and have network middleware observe and potentially tamper with that session state, then TCP is a fine choice of end-to-end transport protocol. If you would prefer to keep your end-to-end transport session state as something known only to you and the party with whom you are communicating, then maybe you should use Google's version of QUIC. And where does the IETF version of QUIC sit? It's completely unclear where the IETF is heading in this respect.

A cynical view would see the IETF as being incapable of holding the line that the end-to-end control state should be completely withheld from the network, and this spin bit is just one more step along an inexorable path of compromise that once more ends up gratuitously exposing user's actions to the network. There is probably a less cynical view as well, but I just can't think what it may be!

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