The ISP Column
A column on various things Internet

June 2025
Geoff Huston

Ossification and the Internet

Networks are typically built to provide certain services at an expected scale. The rationale for this focussed objective is entirely reasonable: to overachieve would be inefficient and costly. So, we build service infrastructure to a level of sufficient capability to meet expectations and no more. In ideal conditions this leads to a widely deployed and highly efficient infrastructure that is capable of supporting a single service profile. At the same time such service platforms are often highly resistant to being altered to support additional service profiles. Each element of the network incurs a marginal cost in altering the service capabilities of the element, and the sum of these costs must not exceed the net incremental benefit that can be accrued from supporting such a new service profile. As the network increases in scale the number of elements increases, and the sum of these marginal costs also increases. The net benefit threshold of feasibility of a new service also increases, as it has to exceed this sum of per-element marginal cost. This makes the network increasingly resistant to change as the network grows in size. In other words, the network ossifies.

The telephone network was a highly successful network that was specifically built around the human voice, and as long as human physiology didn't change, the telephone network did not need to change either! As computing devices became both cheaper and more capable, the potential for computer-to-computer communications services grew in importance. There was the promulgation of fax systems in the 1970's to add document transmission to the services provided by the telephone network, quickly followed by the analogue modem to allow more generic computer-to-computer communications. The challenge was that the core of the telephone network remained a synchronous network that used time-switching in 64kbps increments. While human speech demanded synchronicity and a constant bit rate services computers did not impose such stringent conditions on the network. However, for the first couple of decades of the evolution of the Internet we constructed the Internet on top of the existing telephone infrastructure simply because the Internet service market was just not big enough or valuable enough to justify reengineering the telephone network.

However, the telephone network was incapable of supporting the long-term scale of demand from computer communications, and this provided the opportunity to define what a communications network would look like without the overheads of supporting a telephone service. The new networks dispensed with time division multiplexing and synchronous time switching and instead pursued a simpler network service model of stateless packet switching. By stripping functionality from the network, namely time synchronicity and resource management, it was feasible to construct larger networks for lower cost, bypassing the telephone network model completely. This has been summarised as replacing a network model of "smart network, dumb devices", with a model of "dumb network, smart devices".

The hope was that we did not have to go through this disruptive comprehensive replacement process in the future. A dumb common substrate packet-switched network could be able to support a huge variety of digital service profiles, as the service profile was defined in the connecting devices at the edge, and not in the switching equipment in the middle of the network. The network model was intentionally so sparse that it was incapable of becoming ossified!

However, it has not played out as easily as we might have hoped. Ossification is still a major issue in today's networking environment, and while it's not a theme in the architecture of the transmission platform, we see it in the Internet Protocol itself, in our transport protocols, in our routing protocols, and in various applications. The rich soup of ideas in the 1980s in the early days of computer networking has been filtered into a small set of particular choices that have now been deployed across tens of billions of devices at the edge. And this environment is now extremely resistant to change. Much of the Internet is now hardening up and ossifying!

Let's look at some examples.

IP

Perhaps the most prominent example is the Internet Protocol itself. At the time of its inception in the 1970s, in a world of a small connection of hideously expensive mainframe computers and where "smart" watches were confined to comic books, most computer protocols used 8-bit device addresses (Yes, it's necessary to address every device if only to avoid confusion as to where to deliver every packet!).

The choice of a fixed size 32-bit address field was considered by many at the time to be needlessly extravagant, yet by 1990 it was evident that the computer industry had a much larger vision of the future than a 32-bit address field could accommodate. The chosen "solution" to change almost nothing except the address field size to 128 bits seemed at the time to be an extremely conservative approach. No, this new IPv6 protocol was not backward compatible with IPv4, but the Internet at the time was tiny, so the concept of an ossified protocol base was a completely alien concept. When the first specification of IPv6, RFC 1883, was published in December 1995 the Internet had some 30,000 address prefixes, 1,200 autonomous networks, and a rough estimate of 4.8 million connected devices. By today's metrics these were hardly big numbers, so asking these networks and all devices to support both IPv4 and IPv6 did not seem to be an impossible request.

Yet nothing of any significance happened at the time. Why were we so tied to an IPv4 Internet then?

I suspect that the explosive entrance of the Internet into the consumer market world-wide in the early 1990's made extremely difficult demands on the supply-side of this market. Vendors of consumer equipment were concentrating on adding advanced signal processing capabilities to modem to get kilobits per second out of a 1200 baud baseband carrier, and operating system vendors had spent the last decade working on proprietary networking protocols, so the industry was already under significant pressure to deliver IPv4-capable devices into the consumer market at a volume that matched demand. There was apparently no spare developmental or operational support capacity to also deliver on IPv6.

Now, some thirty years later, in the mid-2020's the picture has changed. The network now has some 1M address prefixes, 77,000 autonomous networks and some 30 billion connected devices (or more!). Yet only 40% users on today's internet have IPv6 access. Progress along this part of adoption of IPv6 has not come to a halt, and it is managing to exceed to growth rates of the Internet, but the remaining work to do still exceeds the cumulative outcome of the effort so far.

Are we on a path to remain in this hybrid mid-transition IP world indefinitely? Those networks and services that support a dual protocol environment are not in a position to retire support for IPv4 until a significant proportion of the IPv4-only networks and services embark on their dual dual-stack transition. At the same time, not every network is experiencing growth pressures, and those IPv4-networks with a relatively stable platform of users and usage appear to feel no urgent pressure to add IPv6 into the service portfolio. It not easy at this stage to identify how this impasse can be overcome.

Are we ossified in this transition, unable to make a clean break with IPv4 and unable to contemplate any successor protocol after IPv6? It certainly appears to be the case!

Fragmentation and Packet Sizes

The IPv4 protocol is relatively unique in defining the ability to adapt to varying packet sizes on the individual hops encountered by a packet when it is passed through the network. When a router attempts to forward a packet where the packet is larger than the size defined by the local interface to the next hop network, then the router is able to split the IP packet into a sequence of fragments, where each fragment fits within the allowed packet parameter size. Each packet contains the common IP packet header, and fragmentation control information that permits the destination host to reassemble the original IP packet. In this way the IPv4 protocol supports the theoretical transmission of individual IP packets of up to 65,535 bytes in size.

In practice this is infeasible on the public Internet. Packet fragments present a range of unacceptable security vulnerabilities. The result is that most IP hosts have a far lower upper bound on the maximum size of an IP packet that they are prepared to accept, and security firewalls typically discard all packet fragments. IPv6 took this one step further and disallowed routers from performing packet fragmentation on the fly. Some protocols, such as QUIC and DTLS simply disallow any form of IP packet fragmentation, whether its by the source or on the fly.

So without the ability to use packet fragmentation what's the largest acceptable IP packet size and why?

In the IPv4 network the general answer is 1,500 octets. This number is derived from the design of the 10Mbps CSMA/CD Ethernet standard, where a maximum Ethernet frame size of either 1,024 bytes or 2,048 bytes would've resulted in an acceptable level of latency to allow other devices on the same common bus to have access to the network without what was considered at the time to an unacceptable delay. The compromise of 1,500 bytes was adopted as a median value between the two. Fifty years later this value has been fixed into computer networks. We no longer use 10Mbps common bus local networks, and with vastly greater transmission speeds the considerations of unacceptable latency when attempting to pass a packet into the network due to large packets are simply not relevant. But we have so much deployed equipment that has the value of 1,500 bytes locked in as a maximum packet size there is no great desire to exert the effort and lift this value, and no visible consensus as to what a new value would be even if we were prepared to try and change this!

Oddly enough, IPv6 took this one step further, and for no compelling reason at all declared that all IPv6 networks should support a packet size of 1,280 bytes without fragmentation! The result is that as a lowest common denominator, the Internet is now locking in 1,280 bytes as the universally supported maximum packet size.

Transport

The Internet Protocol suite was defined with two major end-to-end protocols, UDP and TCP.

UDP, the user datagram protocol, is a simple abstraction of the unreliable datagram delivery model of IP itself, adding port fields to the protocol header to allow the attachment of specific services to a UDP socket on a device.

TCP, the Transmission Control Protocol, is an example of a conventional sliding window positive acknowledgement data transfer protocol that transforms the unreliable datagram model of IP into a flow-controlled reliable stream protocol. Part of the reason for TCP's longevity and broad adoption is TCP's incredible flexibility. The protocol can support a diverse variety of uses, from micro-exchanges to gigabyte data movement, transmission speeds that vary from tens of bits per second to tens and possibly hundreds of gigabits per second. TCP is undoubtedly the workhorse of the Internet. But even so, there is always room for refinement. TCP is put to many different uses, and the design of TCP represents a set of trade-offs that attempt to be a reasonable fit for many purposes but not necessarily a truly ideal fit for any particular purpose.

TCP has its problems, particularly with web-based services. These days most web pages are not simple monolithic objects, but contain many separate components, including images, scripts, customized frames, style sheets and similar. Each of these is a separate web object and if you're using a browser that is equipped the original implementation of HTTP each object will be loaded in a new TCP session, even if they are served from the same server. The overheads of setting up both a new TCP session and a new Transport Layer Security (TLS) session for each distinct web object within a compound web resource can become quite significant, and the temptations to re-use an already established TLS session for multiple fetches from the same server are close to overwhelming. But this approach of multiplexing a number of data streams within a single TCP session also has its attendant issues. Multiplexing multiple logical data flows across a single session can generate unwanted inter-dependencies between the flow processors and may lead to head of line blocking situations, where a stall in the transfer of one stream blocks all other fetch streams. It appears that while it makes some logical sense to share a single end-to-end security association and a single rate-controlled data flow state across a network across multiple logical data flows, TCP represents a rather poor way of achieving this outcome. The conclusion is that if we want to improve the efficiency of such compound transactions by introducing parallel behaviours into the protocol, we need to look beyond the existing control profile of TCP. But is it possible to deploy a new transport protocol that can do a better job for this application?

Here is where we see ossification in the protocol stack within the Internet. One of the ways we've managed to steer clear of the address crunch in IPv4 is to introduce transport-layer multiplexing through the widespread use of NATs (*network address translators*). We've segmented the networks into clients and servers and then determined that clients do not need to have a permanently assigned public IPv4 address. We can share a public address across multiple client devices by using a front-end piece of network middleware, a NAT, and allow the packet streams to be distinguished by using different port addresses for each communications stream as they are passed across the NAT's private/public network boundary. NATs are the mainstay of today's IPv4 network. The side effect is that IP packets that indicate that they are using UDP or TCP transport protocols can be processed by a NAT. Otherwise the packet is dropped by the NAT.

What if we want to define a new transport? Perhaps a UDP streaming protocol that while it doesn't offer TCP's reliable delivery, can perform a congestion control function that wouild allow a UDP streamer to fairly share the network's resources with concurrent TCP sessions? Or a multiplexed TCP state that shares a single authentication and end-to-end encryption state across multiple data streams? We're not short of ideas about novel approaches to transport protocols that can provide a better match to particular service characteristics. But if we venture outside of the existing transport header formats used by UDP and TCP then we can be pretty confident that NAT behaviours would ensure that such protocols would not see the daylight of widespread deployment in the public Internet.

For this reason, we see that QUIC, a recent addition to the suite of transport protocols in IP is implemented as a UDP application, rather than taking the conventional path for transport protocols in being implemented directly on IP.

We're somewhat wedged here in our inability to introduce new transport protocols into the IP protocol suite, which is about a good a practical definition of ossification as you can get.

Routing

Routing protocols have been an important component of the Internet, and BGP is one of the oldest still-used protocols, and the environment of inter-domain routing is the exclusive domain of the BGP routing protocol.

Some aspects of the original design of BGP appear to be ill-suited to today's environment, including the general approach of session restart when unexpected events occur, but this is merely a minor quibble. The major outcome of this protocol has been its inherent scalability. BGP is a protocol designed in the late 1980's, using a routing technology described in the mid 1950's, and first deployed when the Internet that it was used to route had less than 500 component networks (Autonomous Systems) and less than 10,000 address prefixes to carry. Today BGP supports a network which now has a million prefixes and heading towards 100,000 ASNs. There were a number of factors behind this scalability, including the choice of a reliable stream transport in TCP, instead of inventing its own message transport protocol, the distance vector's use of hop-by-hop information flow allowing various forms of partial adoption of new capabilities without needing all-of-network flag days and a protocol model which suited the business model of the way that networks interconnected. These days BGP also enjoys a position of entrenched incumbent which itself is a major impediment to change in this area, and the protocol's behaviour now determines the business models of network interaction rather than the reverse.

This is despite the obvious weakness in BGP today, including aspects of insecurity and the resultant issue of route hijacks and route leakage, selective instability and the bloating effects of advertisement of more specific address prefixes.

Various efforts over the part thirty years of BGP's lifetime to address these issues have been ineffectual. In each of these instances we have entertained design changes to the protocol to mitigate or even eliminate these weaknesses, but the consequent changes to the underlying cost allocation model or the business model or the protocol's performance are such that change is resisted. Even the exhortation for BGP speakers to apply route filters to prevent source address spoofing in outbound packets, known as BCP 38, is now twenty years old, and is ignored by the collection of network operators to much the same extent that is was ignored twenty years ago, despite the massive damage inflicted by a continuous stream of UDP denial of service attacks that leverage source address spoofing.

The efforts to secure the protocol are almost as old as the protocol itself, and all have failed in practical terms. Adding cryptographic extensions to BGP speakers and the protocol in order to support verifiable attestations that the data contained in BGP protocol packets is in some sense "authentic" rather than synthetic impose a level of additional cost to BGP that network operators appear to be unwilling to bear. The issues of security itself, where it can only add credentials to "good" information, imply that universal adoption is required if we want to assume that everything that is not "good" is necessarily "bad" only adds the formidable barriers of universal adoption and the accompanying requirement of lowest bearable cost, as every BGP speaker must be in a position or accept these additional costs. Admittedly we have made some progress in the use of authentication of route origination in recent years, but in the absence of robust mechanisms of BGP path authentication these mechanisms are little more than a route leak prevention measure and offer little in the way of protection against a capable and determined routing attacker.

We have not seen the end of proposals to improve the properties of BGP, both in the area of security and in areas such as route pruning, update damping, convergence tuning, traffic engineering and such. Even without knowledge of the specific protocol mechanisms proposed in each case, it appears they such proposals are doomed to the same fate as their predecessors. In this common routing space cost and benefit are badly aligned, and network operators appear to have little in the way of true incentive to address these issues in the BGP space. The economics of routing is a harsh task master, and it exercises complete control over the protocols of routing. In this sense BGP is best regarded as an ossified protocol.

DNS

In the DNS there are three distinct classes of actors, the *authoritative servers* who publish information relating to a DNS zone, the *stub resolvers* in end host systems who pose queries and receive responses, and the intermediaries who perform DNS resolution, the *recursive resolvers.*

The introduction of new DNS query types into the end user application environment (stub resolvers) is extremely uncommon. Stub resolvers generally ask for A records (IPv4 address), AAAA records (IPv6 address) and little else. It is a common security measure to withhold DNS responses for other DNS query types from stub resolvers. The theory is that stub resolvers don't ask for any other query types, so the use of these unknown query types in DNS responses that are being directed to stub resolvers is an obvious case of an attack, and refusing to pass on the DNS response is thought to be an effective countermeasure.

However, new query types are being introduced to the DNS at a steady rate. Many of these query types reflect the actions of recursive resolvers, but some specifically are intended to be used by end devices. The most recent of these is the introduction of the SVCB query type, and the related HTTPS query type. When the Safari browser took the step of introducing the HTTPS query type as a means of determining if a web server supported the QUIC transport protocol, then this DNS response filtering behaviour of was bought into prominence. In today's public network around one half of Safari users act as if they do not receive any response to their HTTPS queries. This is a significant failure rate, and calls into question the viability of an approach to use the DNS as a means of improving the current inefficient (and slow) process of establishing a connection between client and server.

Ossification

I have often heard the Internet environment as one that is characterized as enabling *permissionless innovation*. By this it's intended to describe a space that is not only accepting of change, but one that embraces change. The standards that underpin the common technology base that defines a single coherent network are generally minimal in nature, allowing a broad scope for experimentation in diverse behaviours on the network. If any such changes generate positive incentives for broader adoption, then they can be standardized and integrated into the mainstream technology base.

But the rigor required by extremely large-scale deployments with multiple suppliers, multiple service operators, diverse users and diverse uses all demand a far more restrictive operational framework. This framework is antithetical to one that allows a broad scope for experimentation in diverse behaviours on the network. The large-scale operational environment is looking for a stable set of parameters that define a common service platform that operates within known parameters and is willing to define all other operational behaviours as aberrant!

I suspect that this is the inevitable price of a technology's success in the market. With an increasing deployment base, the technology becomes more fixed in its modes of operation and is resistant to supporting different services and modes of use. Over time this imposed stasis creates a pressure for change that is embodied in a new generation of technology, which, if it succeeds, then goes through the same cycle once more.

So yes, it's often frustrating to observe the glacial transition to IPv6, or the woeful state of web security, or the inability to introduce new query types into the DNS. It's tempting to decry this situation as the untimely ossification of key Internet technologies. However, I would observe that this situation appears to be an inevitable price of success. The aspects of the Internet that were of great importance when the Internet was challenging the hegemony of the telephone network and its operators are of no use to the Internet now that it has assumed the central dominant position in the world's communications system. The rules for incumbents are totally different to the rules for challengers!

Disclaimer

Author

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

www.potaroo.net