The ISP Column
A monthly column on things Internet

May 2008

Geoff Huston

The End of End to End?

One of the major principles of the architecture of the Internet was encapsulated in a paper by Saltzer, Reed and Clark, "End-to-End Arguments in System Design". This paper, originally published in 1981, encapsulated vary clearly the looming tension between the network and the application:

> *"The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system. Therefore, providing that questioned function as a feature of the communication system itself is not possible."*

At the time this end-to-end argument was akin to networking heresy!

The predominant approach to digital network architecture at the time was that data was sacrosanct, and the network had an obligation to do everything possible to ensure that it did not drop user data. This led to a genre of network protocols in the 70's and 80's that used reliable point-to-point protocols as the base transaction across individual data links. In other words, in a network path of switching elements A, B, and C, element A will store a local copy of the data and await a positive acknowledgement from B that it has received the data before assuming that the packet has been sent and discarding the local copy. If the data is not appropriately acknowledged then A will stop and resend the missing data. Similarly, when B passes this data onto C, B will await a positive acknowledgement from C, and so on. The intended outcome was a digital equivalent of a synchronous analogue circuit: the packets passed into the digital switching system at one end were delivered at the other end in precisely the same order, without any loss and without any data corruption.

This style of point-to-point reliability lead to an general approach to the architecture of digital networking of the form of a "smart" network that could drive "dumb" terminals. The end points did not have any necessary role in reliability, or even flow control of the data stream; their role was to assume a network model of reliable delivery and operate on the assumption that whatever was passed into the network would be correctly and faithfully delivered.

The end-to-end argument was a product of the architecture of the Internet, although its antecedents in terms of designing reliable networking systems on top of cheap and potentially unreliable components dates back to Paul Baran's work at RAND in the early 1960's. The basic approach to networking architecture in terms of the Internet can be informally described as "every packet is an adventure!"

In fashioning the network service as a datagram delivery system, rather than a reliable circuit delivery system, a number of critical differences emerge at the network level: in a datagram delivery system packets can be discarded mid-flight, packets can be reordered, packets may be corrupted in flight. In the end-to-end argument the response to this potential for casual packet damage is quite benign. The end-to-end argument observes that data integrity, data order and reliable data flow is best known to the communicating instances of the applications at either end of the communication, and its best left to these end application to detect and repair data delivery problems, as appropriate.

In the context of the architecture of the Internet the end-to-end application can be seen in part as the Transmission Control Protocol (TCP), an end-to-end protocol that provides to the upper level application a virtual reliable end-to-end data delivery stream. Its at the TCP level that the unreliable IP datagram service is transformed into a reliable data bit stream. Its at the TCP level of the protocol stack that a data sender awaits positive acknowledgement from the receiver that data has been successfully delivered to the receiver, or retransmitting data as necessary, and at the TCP level that the receiver reassembles the received data stream into the order in which it was sent.

So what was the leverage of the shift of roles as espoused by the end-to-end argument?

The impact was in terms of efficiency of networks. A datagram network requires far less of its switching elements. There are no local data buffers in a datagram switch that are required to store a copy of the sent data until the next element in the network path has acknowledged it. There is a vastly reduced amount of processor time required to manage each packet, simpler internal queue structures, and simpler network management protocols that reflect the topology of the switching elements rather than the level of individual link conversations, nor is there a required overhead of virtual circuit creation and removal. Unreliable datagram networks can be constructed using far simpler, and far cheaper switching elements. These forms of simple networks can drive the transmission capacity into much higher levels of use. A reliable circuit network operates very inefficiently when any element within the network reaches full capacity, as any form of buffer overflow creates a "stop and repair" condition within the network that has the potential to ripple across the entire network in much the same way that gridlock occurs in vehicle traffic. With datagram networks the excess traffic is simply dropped at the point of saturation. Its deliberately left as an exercise for the end-to-end application to detect and repair the data loss. This creates a far more flexible network that is capable of having parts of the network hit saturation without have this saturation condition cause back pressure across the rest of the network.

So the true leverage of the end-to-end argument was one of an architecture of a "dumb" network and "smart" applications, quite the opposite of the prevailing common wisdom of network architecture of the time.

This allowed Internet networks to demand less of the switching elements within the network, allowing network switches to process a far higher packet rate within a fixed amount of memory and processing capacity, and allowing the network itself to realize efficiency gains through making the entire transmission capacity available to the applications. In other words, the Internet could be radically cheaper and radically faster.

On the other hand the Internet was also radically worse rather than better in terms of delivered services to the end points of an IP conversation. But this was where the end-to-end application stepped in, and "repaired" any damage via an end-to-end protocol exchange. Once you assumed that capable data processing technology was ubiquitous rather than an exclusive property of the network, then the network could outsource the most expensive and most difficult functions, that of reliability, quality and even resource management, out of the network and pass the entire function over to the end systems. The result is what we see today, in a truly massive communications system that operates at cost efficiency levels and transmission speeds undreamt of even when this end-to-end paper was written in 1981.

But I was intending to look at whether end-to-end has ended, rather than extolling its virtues.

So the question is: Have we gone past end-to-end? Are we heading back to a world of bewilderingly complex and expensive networks?

The model of a clear and simple network where end hosts can simply send packets across a transparent network is largely an historical notion. These days we sit behind a dazzling array of so-called "middleware", including Network Address Translators, Firewalls, Web caches, DNS interceptors, TCP performance shapers, and load balancers, to name but a few.

For many networks middleware, in the form of firewalls and NATs, are a critical component of their network security framework and middleware has become an integral part of the network design. For others middleware offers a way to deliver scalable services without creating critical points of vulnerability or persistent black holes of congestion overload. For others its the only possible way make scarce public IP addresses stretch over a far larger pool of end hosts.

For others middleware is seen as something akin to network heresy. Not only does middleware often break the semantics of the internet protocol, it is also in direct contravention to the end-to-end architecture of the Internet. Middleware breaks the operation of certain applications.

Emotions have run high in the middleware debate, and middleware has been portrayed as being everything from absolutely essential to the operation of the Internet as we know it through to being immoral and possibly illegal. Strong stuff indeed for an engineering community.

The common theme of these issues is that there are a set of inconsistent assumptions at play here. One the one hand, the assumption of an end-to-end architecture leads an application designer to assume that an IP session opened with a remote peer will indeed be with that remote peer, and not with some intercepting network-level proxy agent attempting to mimic the behaviour of that remote peer. On the other hand is the assumption that as long as transactions adhere to a consistent and predictable protocol, the transactions may be intercepted and manipulated by middleware as long as the resultant interaction behaves according to the defined protocol.

Middleware cuts across the end-to-end model by inserting directly into the network functionality which alters packets on the fly, or, as with a transparent cache, intercepts traffic, interprets the upper level service request associated with the traffic and generates responses by acting as a proxy for the intended recipient. With middleware present in an internet network, sending a packet to an addressed destination and receiving a response with a source address of that destination is no guarantee that you have actually communicated with the address remote device. You may instead be communicating with a middleware box, or have had the middleware box alter your traffic in various ways.

The result we have today in the internet is that its not just the end applications which define an Internet service. Middleware also is becoming part of the service. To change the behaviour of a service which has middleware deployed requires the network's middleware be changed as well. A new service may not be deployed until the network's middleware is altered to permit its deployment. Any application requiring actual end-to-end communications may have to have additional functionality to detect if there is network middleware deployed along the path, and then explicitly negotiate with this encountered middleware to ensure that its actual communication will not be intercepted and proxied.

But its probably too late now to consider middleware and end-to-end as alternative destinies. So it appears that the Internet has somehow evolved into a middleware system rather than a coherent and simple end-to-end system. Middleware appears to be here to stay, and now its up to applications to work around middleware. And applications have certainly responded to the challenge in various ways.

There's a class of applications that have gone client-server in order to work consistently in the face of pervasive NAT deployment. This has gone beyond the simple interactions of a web browser with a web server to extend into models where the client is then loaded with additional capabilities that permit it to share a more complex state with the server and perform local processing. In other words once the client-server state is established, the application then moves to a shared state, exploiting the open circuit to create a shared state.

There's a class of applications that perform NAT discovery and multi-party rendezvous. This approach is currently prevalent in SIP applications, where the end client makes contact with a SIP server, establishes the characteristics of any NATs on the path, and then uses the SIP server to advertise the presence and connectivity capabilities of the SIP client. A similar approach has been used in Teredo, the IPv6-in-IPv4 tunnelling protocol. This form of application architecture is a step away from the traditional two party communications and introduces the concepts of agents, servers and multi-party rendezvous mechanisms, all to compensate for the limitations in the communication model that have been enforced by pervasive middleware. And there's a class of applications typified by Skype, that initialize by performing a form of local topology and context discovery, and then connect into a dedicated overlay network to complete the rendezvous operation, and treat the entire underlying network platform simply as a datagram network.

There's a class of applications that have observed that the *https* protocol appears to be the only remaining protocol that works across most firewalls, filters and NATs, and layering the real application within the *https* payload restores connectivity. It also allows the inner content payload to be encrypted, defeating the efforts of deep packet inspection middleware. The manner in which this has been exploited by applications includes simple *https* content distribution systems through to running IP over *https*, creating in effect virtual circuits that are intended bore through intervening middleware and re-establish an end-to-end as a virtual concept, implemented as a tunnelled IP level path across the underlying network obstruction.

And there's a class of applications that are essentially "victims" of middleware. Its now close to impossible to field a new transport protocol simply because the deployed mass of firewalls, NATs, and edge devices are hardwired to recognise UDP, TCP, and ICMP, all over IPv4, and nothing more. So if you are looking to deploy SCTP, or DCCP, for example, then the prospects are not good. Even IPv6 has this problem, in that a large number of deployed firewalls and related security devices are simply ignorant of IPv6.

While the proliferation of middleware is the most significant disruption to a coherent end-to-end network architecture, the temptation to construct complexity out of simplicity has always been present on the part of those network providers that entered the Internet from the background of a telco legacy that was well versed in the network architecture of a capable and functional network. The consequent tensions between the models of a simple datagram network that outsourced a set of essential control functions out of the network and onto the end points and a richly functional network that enforced a set of imposed control attributes on payload traffic has played out in various ways. We've seen activities that have ranged from efforts to add QoS into the network, and efforts to re-invent the virtual circuit model using MPLS, through to efforts to combine an unlikely combination of apparently implausible network-centric technologies into a framework optimistically termed "Next Generation Networks", or NGNs. We've seen attempts to make networks application-aware through the IMS framework that attempts to broker all applications across a SIP model. We've seen efforts to equip networks with triggers to detect and remove "bad" packets.

While the track record of these efforts of network ornamentation have been particularly underwhelming so far, there is still this continuing effort to add various functional elements back into the Internet network that go beyond conventional middleware functions and impose various control regimes on traffic. One suspects that behind these continuing efforts there is a persistent school of thought over there in telcoland is that if only they could find the "right" model of network ornamentation, with the "right" amount of application awareness built into the network, with the "right" amount of filtering and control functions built into the network, then they could transcend their rather limited role in the Internet world as an undistinguished utility provider of commodity IP switching services and move up the value chain to re-establish their previous role as a "full service provider." But, in spite of these expectations that there is some approach out there, somewhere, somehow, that will re-establish the value of a richly ornamented network, the track record so far for these efforts would have to be considered disappointing at best, and more realistically be labelled as abject failures. If there is such a magic solution that will richly reward the operator of a heavily ornamented, feature-laden, fully converged, application-aware Internet network platform, then its been downright difficult to find it any time in the last twenty years. At some point it might be sensible to stop spending all this money looking for something that apparently does not exist. At the network level it appears that, at least for the time being, we'll continue to use relatively simple datagram networks that just pummel as many packets as possible through as quickly and as cheaply as possible.

This implies that if end-to-end is changing its not at the level of the basic transmission and switching functions of the network, but at the level of middleware that is deployed at the edges of the network, or, *edgeware*.

So where are we with end-to-end? Is this proliferation of edgeware simply throttling end-to-end to the point that end-to-end has indeed ended, or is end-to-end still alive in some form or fashion?

I suspect that the path that took us to the proliferation of this edgeware is going to be hard to deconstruct anytime soon.

But maybe this actually reinforces the end-to-end argument rather than weakens it. Not only do end-to-end applications need to take into account that the network will treat packets with a certain cavalier attitude, but also need to take into account that parts of the packet may be selectively rewritten, that entire classes of packets may be deliberately discarded or quietly redirected by this edgeware. What we have as a result is actually a greater level of capability being loaded into end-to-end applications. We've extended the end-to-end model from a simple two-party connection to a multi-party rendezvous process, and added additional capabilities into the application to detect other forms edgeware behaviour above and beyond network level behaviours of simple packet discard, reordering and re-timing. And, oddly enough, the more we see edgeware attempt to impose further constraints or conditions on the communication, the more we see applications reacting by equipping themselves with additional capabilities to detect and react to such edgeware behaviours.

So it looks like end-to-end is still a thriving and vibrant activity, but it too has changed over time. Its no longer a model of "dumb" applications making a simple connection over TCP and treating TCP as a reliable wire. The end-to-end argument is no longer simply encapsulated in an architecture that promotes TCP as the universal adaptor that allows "dumb" applications to operate across "dumb" networks. Over the years, as we've loaded more and more functions onto the connection edge between the local network and the public internet, we've had to raise the concept of end-to-end to another level and equip the application itself with greater levels of capability to adapt and thrive in an edgeware-rich network.

Yes, its still end-to-end, but its no longer a model that uses just TCP as the universal adaptor between applications and networks. These days the applications themselves are evolving as well to cope with more complex network behaviours that have resulted from the proliferation of edgeware in today's Internet.

Further Reading

There are three papers that form the core of the contributions to the end-to-end argument:

End-to-End Arguments in System Design
J.H. Saltzer, D. P. Reed and D. D. Clark, ACM Transactions in Computer Systems, November 1984
http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf

Rethinking the design of the Internet: The end to end arguments vs the brave new world
M.S. Blumethal, D. D. Clark, ACM Transactions on Internet Technology, August 2001.
http://portal.acm.org/citation.cfm?doid=383034.383037

Tussle in Cyberspace: Defining Tomorrow's internet
D. D. Clark, K. R. Sollins, J. Wroclawski, R. Braden, SIGCOMM '02, August 2002.
http://www.sigcomm.org/sigcomm2002/papers/tussle.pdf

Disclaimer

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

About the Author

GEOFF HUSTON is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. he graduated from the Australian National University with a B.Sc, and M.Sc. in Computer Science. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001.

http://www.potaroo.net