Reinterpreting the Transport Protocol Stack to Embrace Ossification – Position Paper

Stephen McQuistin
University of Glasgow, UK
s.mcquistin.1@research.gla.ac.uk

Colin Perkins
University of Glasgow, UK
csp@csperkins.org

***Abstract*—Ubiquitous deployment of middleboxes has resulted in ossification of the transport layer, with TCP and UDP becoming part of the narrow waist of the Internet. This is a necessary stage in the evolution of the network, caused by its progression from research, to production, to increasingly critical infrastructure. New transport layer protocols will be needed in future, but since we are working with essential infrastructure, we cannot expect to have scope to make wholesale rapid changes. Future development must be done using the existing protocols as substrates, always maintaining on-the-wire compatibility. To advance, we must embrace the ossification of the network, and learn to reinterpret and extend the existing protocols.**

I. Introduction

The Internet architecture assumes a common internetwork layer is sufficient substrate for building a global network, considering communication on a host-to-host basis [1]. Time has shown this to be only part of the story, however. Not only does the network need to know the hosts that are party to a communication, it has become clear that knowledge of higher layers is also required. This is visible in the plethora of middleboxes that have become part of the network, inspecting both transport layer headers and higher layer protocol data to 'help' endpoints and perform various policy enforcement functions. The tussle space [2] between application and service provider needs is not effectively realised by the IP layer of the Internet. Deep packet inspection and other tools are used to extract the information needed to enforce provider goals.

This has resulted in widespread ossification of the network. Knowledge of the transport layer, and higher layer protocols, is encoded in middleboxes throughout the Internet. It is steadily more difficult to change the core network protocols, because doing so requires changes to increasingly many middleboxes and policies, in addition to the endpoints of the communication.

This is a good thing.

The Internet is no longer a research network. Indeed, in many ways it has moved beyond being a production service, and is now critical infrastructure in many countries. It *should* be difficult to change the infrastructure that a country relies on to run its emergency services and healthcare systems, manage physical utilities and industry, coordinate distribution of goods, and run banking and financial systems. The bar to change ought to be high. Strong backwards compatibility *has* to be required.

This does not mean the network should never evolve. There will be new requirements, and better ways of solving old problems. However, effective evolution of the network can only occur within the constraints of the existing infrastructure.

We believe future protocol development has to come by embracing ossification and reinterpreting the stack. Rather than force change where we should be conservative, we must accept protocol durability. In doing so, however, we need not abide by layer boundaries or other semantic limitations that are not reflected on the wire. We re-imagine and reinterpret the stack for future needs.

The network of the future will be built on the network of today. It will have an IP layer; eventually, it may have an IPv6 layer. TCP and UDP will still exist. However, where the layer boundaries may be drawn, what those layers will be called, and how their semantics might evolve, are open questions.

II. Reinterpreting Transport Protocols

We are used to thinking of the Internet as having a common network layer and two widely deployed transport protocols (TCP, UDP), with other transports, such as SCTP or DCCP, struggling to find broad deployment. In this model, TCP provides reliable, in-order, and congestion controlled byte streams, while UDP provides unreliable, connectionless, and uncontrolled datagrams.

Perhaps, though, we should view the protocols and their boundaries differently. We might consider deprecating UDP as a user-visible protocol, and reusing protocol number 17 to indicate that a transport identification header sits between the IP layer and the transport. This header would have the same wire format as a UDP header, subsuming the port numbers to become dynamic identifiers for the transport protocol that are signalled out-of-band, and providing a header checksum that can optionally be extended to cover the transport layer. The actual transport layer headers would fit into what is currently the UDP payload, and could be implemented in the kernel, or as a set of user-space libraries that offer a more flexible API than the traditional sockets layer. Encumbrance by legacy behaviour is comparatively limited, since UDP imposes few constraints on what data can be sent, provided transport protocol identifiers are negotiated to avoid well known UDP port numbers subject to middleboxes (e.g., don't negotiate transport protocol 53).

By reinterpreting UDP headers as a transport identification headers, we gain an additional protocol demultiplexing point. This gives scope to signal new transport protocols, and to experiment with new designs. Protocols such as DNS, RTP, DCCP, and SCTP can all be easily deployed in this manner, interpreted as native transports, rather than as second class

citizens that must tunnel within native UDP packets. In many ways, of course, this is merely a semantic game, and nothing has really changed. That doesn't make it less important: opening a standard extension point will encourage innovation in a way that tunnelled deployment will not, by legitimising development. Perception that an extension point is available matters.

Not everything can run over such a transport identification layer shim. There are middleboxes that only allow TCP, and some applications want semantics that are similar to those of TCP. We must retain a base layer of transport features that provide a shell of TCP compatibility for these environments. This base has to include those aspects of the TCP state machine that are visible to the network, including the three-way handshake, sequence and acknowledgement numbers, retransmission of lost sequence numbers, and a sliding window. Furthermore, the header format and checksum must be retained.

This leaves much that can be reinterpreted to support novel transport services. Header sequence numbers provide a building block that can be used to order datagrams or provide reliable delivery, but there is no requirement that data be presented to higher layers in order, or without gaps, and an unordered stream protocol, avoiding head-of-line blocking, can trivially be built, exposing a new API to deliver portions of the stream as data arrives. Framing can also be provided as a higher layer with very low overhead [3]. The header sequence number space must be completed by retransmissions matching gaps arising from packet loss, but retransmitted data need not match the original ([4] shows the network will deliver *some* version of the data covering that sequence space), allowing relaxed reliability to be provided. Finally, defining new flow and congestion control algorithms has a long history, since the algorithm for managing the sliding window is a local choice. It is clear that reliability, framing, and ordering semantics, how fast or slow data is sent, and how flow control is managed *can* be changed within the current transport framework and middlebox environment, *provided* we accept new views on existing protocol semantics.

III. Directions for Transport Evolution

Tunnelling protocols such as DCCP in UDP [5] point towards reinterpreting UDP as a transport identification layer, with dynamic negotiation of the encapsulation port identifier. The WebRTC media channel shows further scope for such development, with five protocols (STUN, DTLS, RTP, RTCP, and SCTP) currently running on a single UDP port that could run on an explicit demultiplexing layer (e.g., a variant of [6]).

Multipath TCP [7] shows how a transport can be extended to make use of multiple flows. We rather consider single flow semantics: relaxing reliability, ordering, timeliness, and congestion control, and treating TCP as a protocol for reliable delivery of sequencing data, that provides building blocks reliability and ordering, allows flexibility in the higher layers.

Exploration of the design space of transport protocols has largely been conducted through the development of clean-slate solutions. By modularising existing transport protocols into transport services, clean-slate designs become easier to deploy.

IV. Example: Unordered, Time-lined, Transport

As an example of the ideas expressed in Section II, we sketch an outline for a new unordered, time-lined, transport protocol, designed to improve performance for latency-sensitive real-time applications, as a TCP variant [8]. This provides three key features: partial reliability, time lines, and dependencies.

First, we layer on the base shell of features mandated by TCP to give an unordered, partially reliable, datagram delivery service. This reuses the sequence numbers to detect loss, but not provide ordering; the connection and 3-way handshake; and unmodified TCP congestion control. We then add a time-aware layer above the shell of TCP, allowing applications to specify an expiry time for each datagram [9]. Once a data datagram expires, it will no longer be retransmitted, and a suitable replacement will be selected from live datagrams. Finally, we allow dependencies between datagrams to be expressed; if a datagram expires, then all of its dependencies also expire. These changes provide two benefits. By removing the guarantee of in-order delivery, we remove head-of-line blocking and reduce latency, improving real-time performance. Then, by relaxing the reliability guarantee, we enhance good-put by increasing the probability that datagrams entering the network will be useful to the receiver. Both are beneficial to our target applications.

It is important to characterise the protocol behaviour expected by middleboxes, and delineate this from the semantics that are imposed only by endpoints. The design of this unordered, time-lined, transport illustrates how components of existing transport protocols (e.g., TCP's sequencing component) can be combined with application-layer knowledge to produce a deployable *domain specific protocol* within these constraints.

V. Conclusions

The lower layers of the Internet protocol stack are critical infrastructure, that rightly resist change. New transport services are only deployable if they respect the syntactic and semantic constraints imposed by the existing network. As we have shown, there is scope to change transport within these constraints. However, the standards community has held inviolable aspects of the transport semantics that are not broadly interpreted by the network, limiting development. To move forward, we must understand what semantics are constrained only by tradition, and allow them to change to support new services.

References

[1] D. D. Clark, "The design philosophy of the DARPA Internet protocols," in *Proc. SIGCOMM*. Stanford, CA: ACM, Aug. 1988.

[2] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden, "Tussle in cyberspace: Defining tomorrow's Internet," in *Proc. SIGCOMM*. Pittsburgh, PA: ACM, Aug. 2002.

[3] M. F. Nowlan, N. Tiwari, J. Iyengar, S. O. Amin, and B. Ford, "Fitting square pegs through round pipes: Unordered delivery wire-compatible with TCP and TLS," in *Proc. NSDI*. San Jose, CA: USENIX, Apr. 2012.

[4] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend TCP?" in *Proc. IMC*. Berlin, Germany: ACM, Nov. 2011.

[5] T. Phelan, G. Fairhurst, and C. S. Perkins, "A DCCP UDP encapsulation for NAT traversal," IETF, 2012, RFC 6773.

[6] M. Westerlund and C. S. Perkins, "Multiple RTP sessions on a single lower-layer transport," IETF, Oct. 2013, work in progress.

[7] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," IETF, 2013, RFC 6824.

[8] S. McQuistin, "Transport-layer support for multimedia applications," MSci Thesis, School of Computing Science, University of Glasgow, Apr. 2014.

[9] B. Mukherjee and T. Brecht, "Time-lined TCP for TCP-friendly delivery of streaming media," in *Proc. ICNP*. Osaka, Japan: IEEE, Nov. 2000.