Evolving TCP. How Hard Can It Be?

Zubair Nabi, Toby Moncaster, Anil Madhavapeddy, Steven Hand, Jon Crowcroft
Computer Laboratory
University of Cambridge
firstname.lastname@cl.cam.ac.uk

ABSTRACT

Over the last decade TCP has become the de facto "narrow waist" of the Internet — a one-size-fits-all transport that is poorly suited to the needs of modern applications. As middleboxes have become ubiquitous, it has become nigh impossible for alternative transports to exist and so application developers have come to view opening a TCP socket as the only reliable way to connect to a server. Some recent proposals circumvent this problem by camouflaging new transports so that they appear like TCP to middleboxes. We draw the key lessons from this approach and show how this could lead to a true one-size-fits-all transport: "Polyversal TCP".

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – Network communications

Keywords

Transports, API, multipath

1. INTRODUCTION

Application developers have two choices for sending data over the Internet — TCP, with its reliable, ordered, congestion-controlled byte stream model or UDP with its unordered, unreliable datagram model. Middleboxes such as firewalls and intrusion detection systems have effectively hard-wired TCP into the Internet and have made it increasingly hard for novel transport protocols to be deployed [4].

TCP and UDP support a remarkable variety of applications over a huge range of connection speeds and latencies, but are struggling to meet the demands of today's high-bandwidth, low latency applications. This has led developers to use the underlying transport as a substrate over which to run application layer transports. Examples of this are Minion [5] that uses TCP as its substrate but allows the application to trade reliability in favour of reduced latency, and uTP [8], which provides TCP-like reliability on top of UDP for BitTorrent and uses a custom approach to congestion control.

Multipath TCP (MPTCP) takes a different approach [6]. The transport is designed to work alongside TCP, and clever design choices result in a new protocol that looks like TCP on the wire, but which is able to make far better use of the available bandwidth resource pool across multiple interfaces.

MPTCP points to a new approach for evolving transport protocols. Rather than expecting a new protocol to survive in an Internet dominated by middleboxes, we suggest that it should adopt a form of camouflage. Raiciu et al. [6] identify three design goals that are applicable to any new transport: to be able to work with unmodified receivers and APIs, to work in all cases where TCP currently works and to offer performance at least as good as TCP in any circumstances. In this paper we take these goals and generalise them to show how this leads to the design of a new Polyversal TCP that adapts to suit any underlying network: from WANs with RTTs measured in seconds, to NUMA multicore interconnects in the nanosecond range, from low speed sensor networks with bandwidths of *kbps*, to high-speed datacenter networks with bandwidths of *10s of Gbps*.

2. TRANSPORT DESIGN GUIDELINES

Middleboxes have reduced the choice of underlying transport protocols to two: UDP or TCP. This is symptomatic of the ossification that has been evident for some years [3]. However, we believe there is a simple solution — if a new transport looks like TCP on the wire, then it survives the first hurdle to adoption in the wider Internet. But in and of itself this isn't enough, there are additional guidelines that should be followed if it is to be successful:

1. Connections should be established using a TCP handshake and the behaviour of standard TCP control bits should remain intact.
2. The protocol should fail gracefully in the presence of aggressive middleboxes, coping with transparent erasure of TCP options and falling back to vanilla TCP.
3. The new transport should offer real deployment benefits — the history of the IETF is littered with new transports that have never got traction because there was no realistic deployment model.
4. The new protocol should exhibit stability and resilience in the face of adverse network conditions. In particular the protocol must be aware of the risk of fighting with itself in cases where it causes self-congestion.

CoNEXT Student'12, December 10, 2012, Nice, France.

There is also an important non-goal that has hampered the adoption of many new proposals: TCP or flow-rate fairness. This is the flawed notion that at any bottleneck every flow should receive an equal share of the resources. There are many objections to this idea [2], but among the most critical is that it fails to take account of applications that simply open multiple flows in order to get a greater share of the available bandwidth [1].

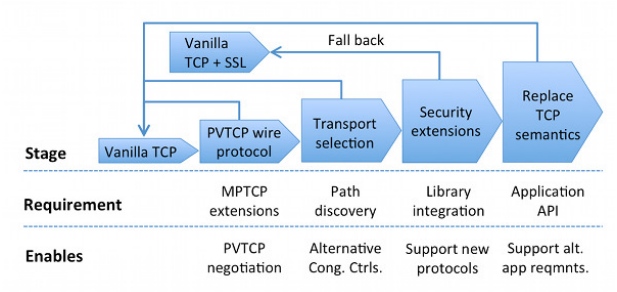

Figure 1: The evolution of PVTCP. At every stage we offer enhanced performance over TCP and provide sensible fallback strategies.

3. FROM UNIVERSAL TO POLYVERSAL

The universality of TCP means it is the jack of all trades and master of none. We believe that by applying the guidelines above we can change that, creating a transport protocol that can offer any feature the application designer wants while still retaining the ability to fall back to vanilla TCP. In short, we envisage a Polyversal TCP.

PVTCP builds upon the MPTCP subflow mechanism by allowing the application to customize each subflow independently. These subflows can exhibit different *characteristics* (congestion control, reliability, ordering, security etc.) depending on application requirements and the underlying network (including middleboxes). During setup, PVTCP performs path characterisation using mechanisms similar to path-MTU discovery. Using this information, PVTCP can either transparently choose the transport semantics for a particular subflow or if the application wants fine-grained control, then it can use `setsockopt()` to explicitly customize each subflow via a per-subflow socket. PVTCP maintains backwards compatibility with the traditional socket API by keeping the `socket()`, `bind()`, and `listen()` socket calls intact. If problems are found at any point during the lifetime of a connection, it can simply fall back to standard TCP for that connection or use alternatives such as MPTCP or SSL over MPTCP as shown in Figure 1.

3.1 Motivating Applications

We present two motivating examples exploring opposite ends of the spectrum: datacenter networks, where the entire network is under a single control domain (which alleviates the middlebox issues) and mobile networks, which are at the mercy of middleboxes, link layer loss and other entities.

Datacenter Networks.

Datacenter networks present an extreme case in which virtual hosts maintain multiple communication channels within and across physical machines. The underlying subnetworks of these channels can vary from on-chip multicore interconnects to inter-host Ethernet, optical or Infiniband links. They exhibit an order of magnitude difference in performance depending on the transport regime and the layout of the underlying network [7]. In such situations, applications can customize each subflow directly through the PVTCP socket API or allow PVTCP to do so on its behalf. For instance, PVTCP can choose the transport based on the size of the transfer and the location of the destination. In addition, for virtualized hosts, PVTCP can ensure robust live migration by temporarily switching to standard TCP to allow shared memory channels to be replaced.

Mobile Networks.

MPTCP can already stripe the same connection over multiple interfaces in multihomed mobile devices for robustness and high throughput [6]. Under PVTCP the congestion control, reliability, and ordering of each subflow can be modified based on the link layer technology (WiFi, 3G, etc.) to enhance performance and suit application needs. In addition, PVTCP can provide resilience by allowing connection rejuvenation in case of signal drop-out or unreliable handover.

4. CONCLUSIONS AND FUTURE WORK

The well-documented increase in middleboxes has resulted in TCP becoming almost universal. TCP has proved remarkably adaptable, but it is not ideal for many modern applications. This leaves protocol designers two choices: either write an application layer transport on top of TCP or UDP, or disguise the new transport as TCP. The second approach is more flexible and this paper has given guidelines for how to do this safely and effectively. To show the strength of this approach we briefly described how Polyversal TCP can offer a much more powerful and flexible alternative to standard TCP in datacenter or mobile scenarios, but we think that PVTCP has far wider application than this. We also believe that PVTCP can help with network deossification by decoupling the evolutionary fate of the network and transport layer, allowing both to evolve independently. We are now in the process of developing this protocol and the related API enhancements to allow it to be integrated with existing standard libraries thus enabling transport protocols to evolve to reflect the needs of future applications.

5. REFERENCES

[1] M. Belshe and R. Peon. SPDY Protocol. 2012. IETF draft: draft-mbelshe-httpbis-spdy-00.

[2] B. Briscoe. Flow rate fairness: Dismantling a religion. *ACM SIGCOMM CCR*, 37(2), 2007.

[3] M. Handley. Why the Internet only just works. *BT Technology Journal*, 24(3), 2006.

[4] M. Honda et al. Is it still possible to extend TCP? IMC'11.

[5] M. Nowlan et al. Fitting square pegs through round pipes. NSDI'12.

[6] C. Raiciu et al. How hard can it be? Designing and implementing a deployable multipath TCP. NSDI'12.

[7] S. Smith et al. The case for reconfigurable I/O channels. RESoLVE workshop at ASPLOS'12.

[8] L. Strigeus et al. uTorrent Transport Protocol, Jun 2009. `http://www.bittorrent.org/beps/bep_0029.html`.