Evolving TCP. How hard can it be?

Toby Moncaster†, Zubair Nabi††, Anil Madhavapeddy†, Steven Hand†, Jon Crowcroft†

Computer Laboratory, 15 JJ Thomson Avenue, Cambridge, CB3 0FD, UK † first.last@cl.cam.ac.uk †† zubair.nabi@cantab.net

Why care about transport evolution?

The Internet has 2 main transport protocols:

Protocol	Throughput	Reliability	Latency
TCP	High	High	High
UDP	High	Low	Low

But application needs differ. What if you want low latency but high reliability? Or variable reliability?

So why hasn't it happened?

Well, it has! Lots of proposals for alternatives DCCP, RCP, XCP. But none have succeeded...

What makes it so hard?

In a word: middleboxes.

NATs block UDP, change protocol headers

IDS and **firewalls** block unusual traffic

Load Balancers re-segment TCP, change IP addr

Traffic Management boxes block or shape traffic

What can we do about it?

Two clever approaches:

★ **Run on top of UDP/TCP - safe but inflexible**

e.g. ***Minion****, application layer transport which trades reliability for latency using TCP as a substrate.*

★ **Masquerade as UDP/TCP - risky but flexible**

e.g. ***µTP*** *which allows reliable BitTorrent to run over UDP while using a less-than-best-effort congestion controller*

MPTCP *which uses multiple subflows to increase throughput, reliability or connection resilience.*

4 simple rules for evolution

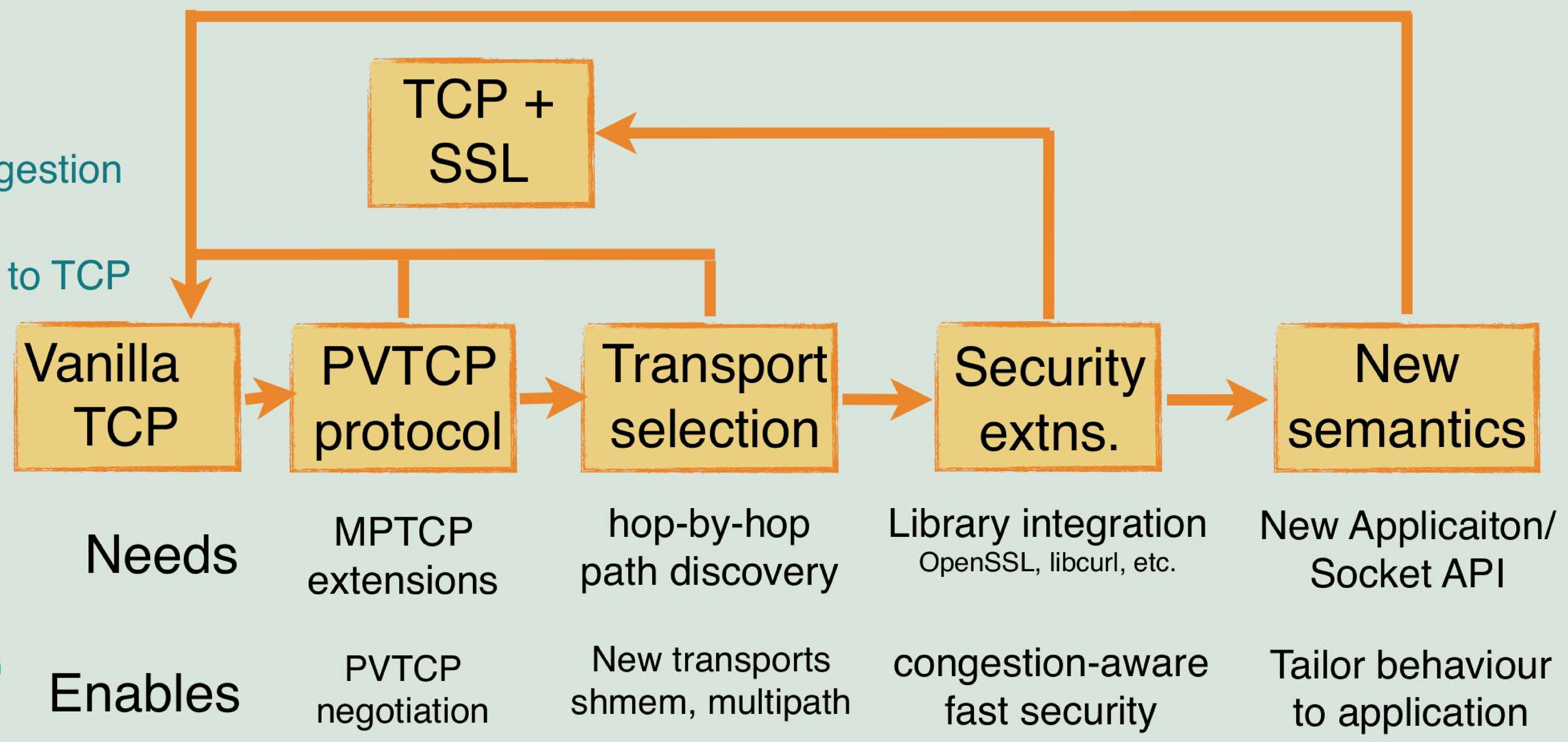
1. Use a TCP handshake to establish each flow and preserve standard TCP control bits.
2. Fail gracefully in the presence of aggressive middleboxes. Have a safe fallback strategy.
3. Offer real deployment benefits with minimal effort by developers and operators.
4. Ensure the protocol is stable and resilient - in particular be mindful of self congestion.

Polyversal TCP (PVTCP)

- ★ A simple extension to MPTCP.
- ★ Subflows can have different wire formats and congestion control across a single session.
- ★ Simple migration strategy which always falls back to TCP for robustness
- ★ Offers optional new semantics for better connection management
- ★ Includes novel transports like shmem

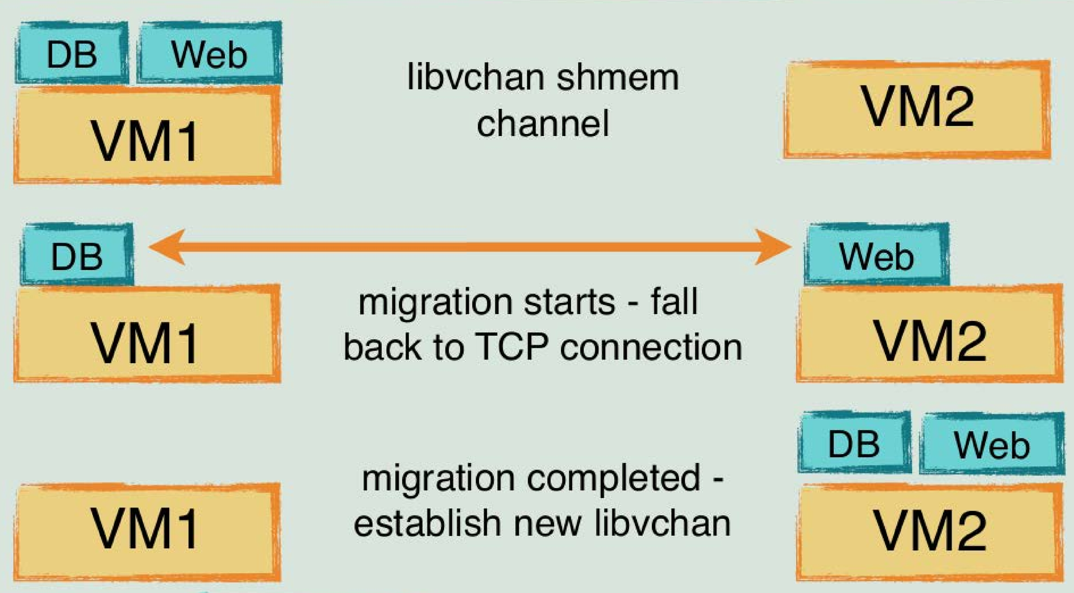
Open questions include:

- ★ how to optimise connection negotiation
- ★ how to use hardware offload (disabled in MPTCP)
- ★ how to establish end-host capabilities



PVTCP example: Datacentre VM migration

- ★ Currently applications use TCP because they don't know where the VMs physically reside.
- ★ If they know they are on same physical die they could use a libvchan shmem channel instead - faster and more stable. But it breaks if the VM is migrated.
- ★ With PVTCP the application can use shmem with a TCP fallback channel. If migration happens the fallback channel maintains connectivity until a new shmem channel is set up.



UNIVERSITY OF CAMBRIDGE

Systems Research Group srg@cambridge:~$

EPSRC Pioneering research and skills

INTERNET INTelligent Energy awaRe NETworks

Wolfson College Cambridge

Approved for public release. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-11-C-0249. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.