Some thoughts from a disarrayed mind on the evolution of transport abstractions

Jana Iyengar

What’s in our purview?

- Resource Pooling, Mobility (MPTCP)
- Better congestion control
- Lower network latency
- Channel encryption (TCP-Crypt)

What's in our purview?

Are these new *services* … or *bug fixes*?

- things we've always wanted to do
- Internet plumbing

These are important … and we're really good at them!

From an app developer’s point of view however, we’re working on improving implementations, not building new abstractions.

Do apps need new abstractions?

- Apps (things that humans interact with) today
  - traffick in units of streams, messages
  - build to semantics available at design-time
  - care about reliability, latency, throughput
  - don’t care about the shape of the bits on the wire

- Transport *services* should map to app needs
  - .. but our transport services don’t. There is a gap.

What’s in the gap?

- Common design patterns
  - message dependencies, framing, interleaving …
  - mobility?
- Performance optimizations
  - zero-RTT transactions
  - “slow-start avoidance”?

- New transports lie in this gap.

App developers fill these gaps with new transports using *conventional* transports as low-level building blocks …

- for deployability on existing hosts
- for deployability through middleboxes
- eg: RTMP, RTMFP, QUIC, Minion

Unfortunately, our transports don’t make great building blocks.

- SCTP is not the answer if all an app wants is reliable message semantics.
- Improvements to congestion control seem hidden under TCP’s bytestream API.

Thoughts on the evolution of transport abstractions

- We ought to fill this gap to the application
- We should consider encapsulating our mechanisms as building blocks
  - LEDBAT is an example
- We should consider enabling compositions of mechanisms at design time