HTTP and Transport

Jana Iyengar
jri@google.com

What does TCP do?

- **Loss recovery via retransmissions**
  - Fast retransmit, FACK, early retransmit, tail loss probe, RTO, F-RTO
- **Congestion control**
  - Determine reasonable rate for sending data
  - NewReno, Cubic
- **Bytestream abstraction**
  - SOCK_STREAM

HTTP and TCP

- **HTTP/1.0 and TCP**
  - 1 object per connection
  - multiple connections per page
- **HTTP/1.1 and TCP**
  - Persistence (and pipelining): multiple objects per connection
  - Problem: Loss of parallelism
- **Efforts to fix transport for HTTP use**
  - TCP Session, Congestion Manager, Ensemble-TCP
  - SCTP: Multistreaming, message framing
- **HTTP/2 and TCP**
  - Implements needed services directly above TCP's bytestream

WHAT IF I TOLD YOU
HTTP COULD BE A LOT FASTER

Opportunities

- **Better congestion control**
  - Reno, Cubic, CompoundTCP, are all buffer-filling
- **Faster loss recovery**
  - TLP is great, but not deployed in non-Linux platforms
- **Connection Pooling**
  - Single tail is still better than multiple tails
- **Transport multiplexing**
  - avoids HoL blocking in the transport
- **Faster Connection Setup**
  - TFO? (slowly seeing deployment)
- **Multipath, Mobility**
  - MPTCP?

Opportunities

- **Better integration with applications**
    - because the socket API is not good enough
    - better decisions made based on application intent
    - eg: different congestion controllers
    - eg: sliding-scale reliability
    - eg: bw-intensive vs. latency-sensitive data

Challenges and Trends

- **Deploying TCP changes is hard**
  - SACK took more than 10 years to deploy after standardization
  - TFO started as an Internet-draft in 2011, deployment slow
  - middleboxes stomp on TCP modifications and new IP transports
- **Networking moving out of OS kernels**
  - increases deployment agility, enables customized stacks
  - reduces buffering in the stack
  - packet slinging frameworks such as netmap, Intel DPDK
  - UDP-based transports
- **Success == Ossification**
  - network optimizes for successful protocols
  - hinders evolution of stack

"Layering is an optimization on clarity;
it never improves performance."

Matt Mathis