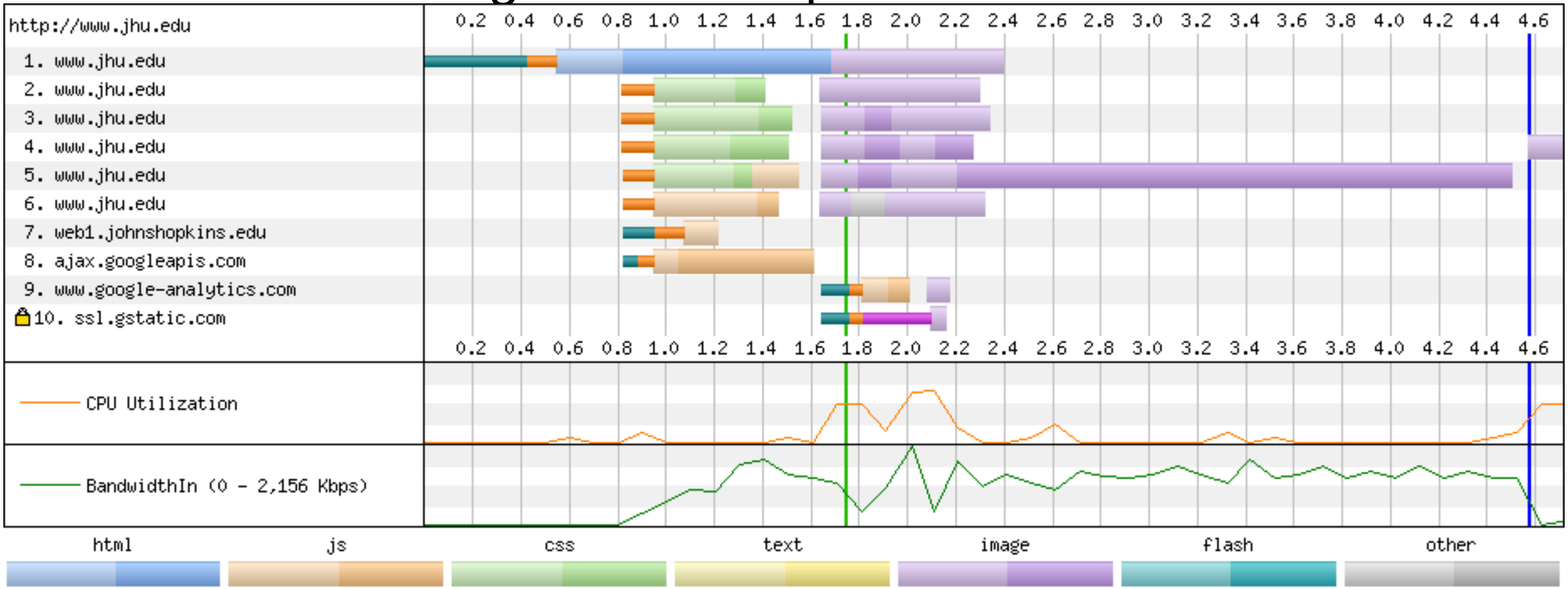
Client-aided Congestion Management for TCP

Yuchung Cheng

Andreas Terzis, Matt Mathis, Nandita Dukkipati

Motivation: TCP throttles app performance

- Apps today use lots connections
  - Even with intelligent ADF multiplexing, e.g., SPDY
  - Persistent connection is common practice
- Every new connection has to (re)discover the network
  - ~90% Google HTTP responses delivered in initial



A smart (or not-so-dumb) transport should ...

- Share network states among connections
  - Past and current active ones
  - Save or amortize reprobing time, e.g., slowstart

- A congestion manager (CM) on top of connections, *not* inside connections
  - New connection starts fast (as if it's never been disconnected)
  - Should recover fast and avoid timeout at all cost
  - N connections are as good as 1
    - Disincentivize parallel connections

Sender-side CM approach

- Theory and practice
  - RFC 2140 - TCP Ctrl Block Sharing by J. Touch '97
  - Congestion Manager by H. Balakrishnan, SIGCOMM '98
  - Ensemble-TCP by L. Eggert, CCR '00
  - SCTP, '00
  - Structured Stream Transport by B. Ford, SIGCOMM '07
  - Multi-path TCP

- Pros
  - Easy: sender traditionally holds all CC states
  - Fast deployment: maybe one side change only

- Cons
  - Scale: connections to same dst must hit same (physical) host
    - Difficult with large server farm load-balancing
    - Need big cache for the ever-growing Internet
  - Fragile: many devices/paths behind one client IP due to NAT

Can the client help?

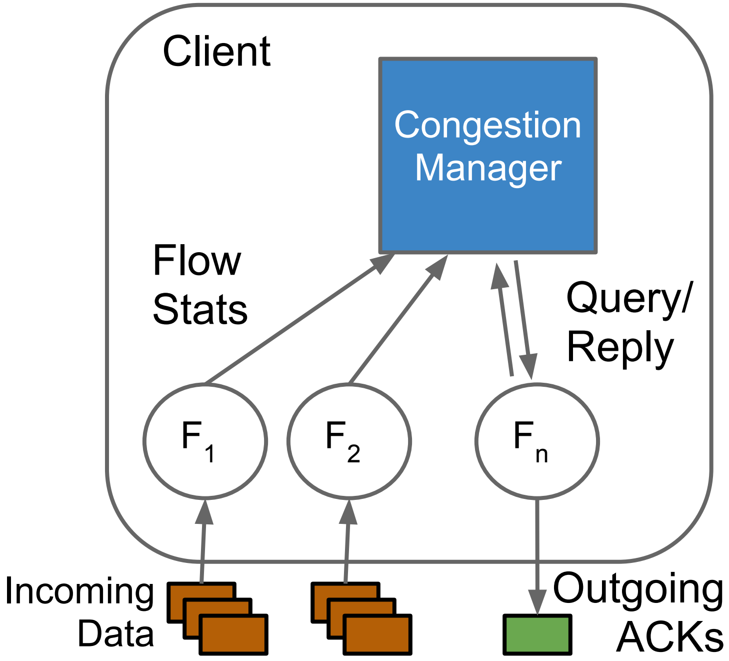
- The client is the hub of *its* connections
    - Naturally the place for caching and sharing
    - Scales well
    - NAT is not a problem

- The client often knows better about the bottleneck: last-hop
    - Link properties: wired, wifi, or cellular
    - Link rate: edge vs 3G
    - Link failure and recovery
    - Dormant or active

- E.g., why RTO backoff then slow-start when a client can hint the sender the broken cellular link has recovered

Great Snipe: a client-based congestion management

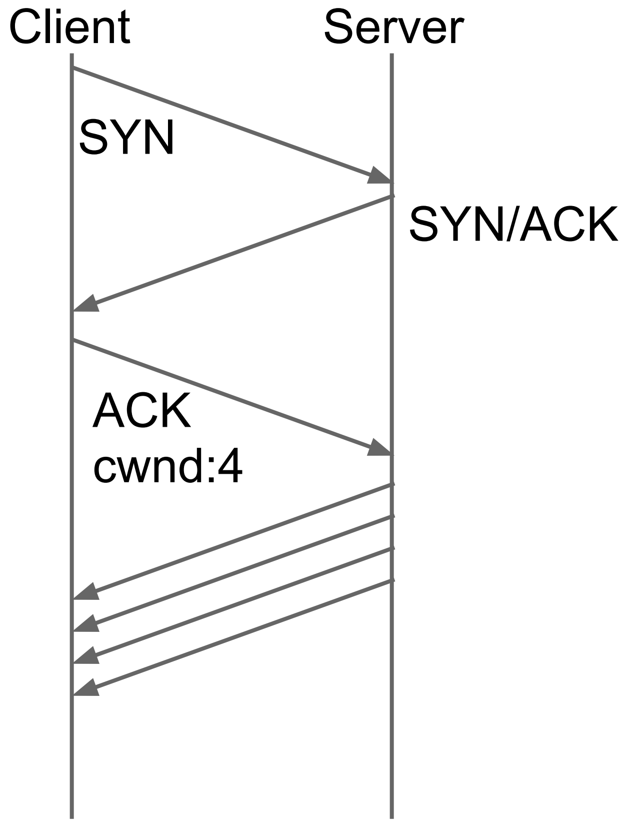
- A new TCP CC **framework**
  - *Not* a new congestion control algorithm
- Move congestion control to the client
  - Connections on the same path share one cwnd
    - also RTT, loss rate, reordering, etc
  - Network properties cached at the client
  - Use options for signalling
- Server handles e2e reliability
  - Detect and recover losses



Client-based congestion control

- Client as data receiver
  - Client maintains size of congestion window (cwnd)
  - Client passes cwnd to sender in ACKs
  - Sender limits # of outstanding packets to cwnd
- Benefit
  - Allows cwnd caching and reuse

- Client as data source
  - Same as TCP today



Implementing standard AIMD

- Connections on the same path share one cwnd (acwnd)

- On startup
  - Slow start with IW10 if no prior history
  - cwnd = acwnd / N otherwise

- On losses
  - Server performs traditional loss recovery and informs client
  - acwnd = ssthresh
    - Reduce once across multiple losses or connections
  - acwnd = 1
    - If nothing received from the same dst for last RTO

- After cwnd reduction
  - acwnd += 1 per RTT

- Upon completion
  - acwnd remains same

Research issues / opportunities

- A pure client-based maybe overkill
    - What if client just guides the server somehow
- Sender announces the backlog to allow better acwnd allocation?
- Track one way delay (OWD)
    - New delay-based congestion control?
- Co-exist with traditional TCP and other protocols
    - E.g., interactive or real-time protocols
- Detect shared bottlenecks among different paths
- Reusing / sharing other states, e.g., loss rates, reorderig, etc
- Energy efficiency

Conclusion

- Congestion control should be on top of individual logic connections

- Server-based congestion manager has practical scale issue
  - Client may offer interesting opportunities to improve CC today!
  - Often knows the network better
  - Naturally the sharing point
  - Scale well

- Great Snipe: move CC to client and on top of indiv. connections
  - Still in early development stage
  - Will release to the public for testing like Laminar

- Feedback & ideas welcome! ycheng@google.com