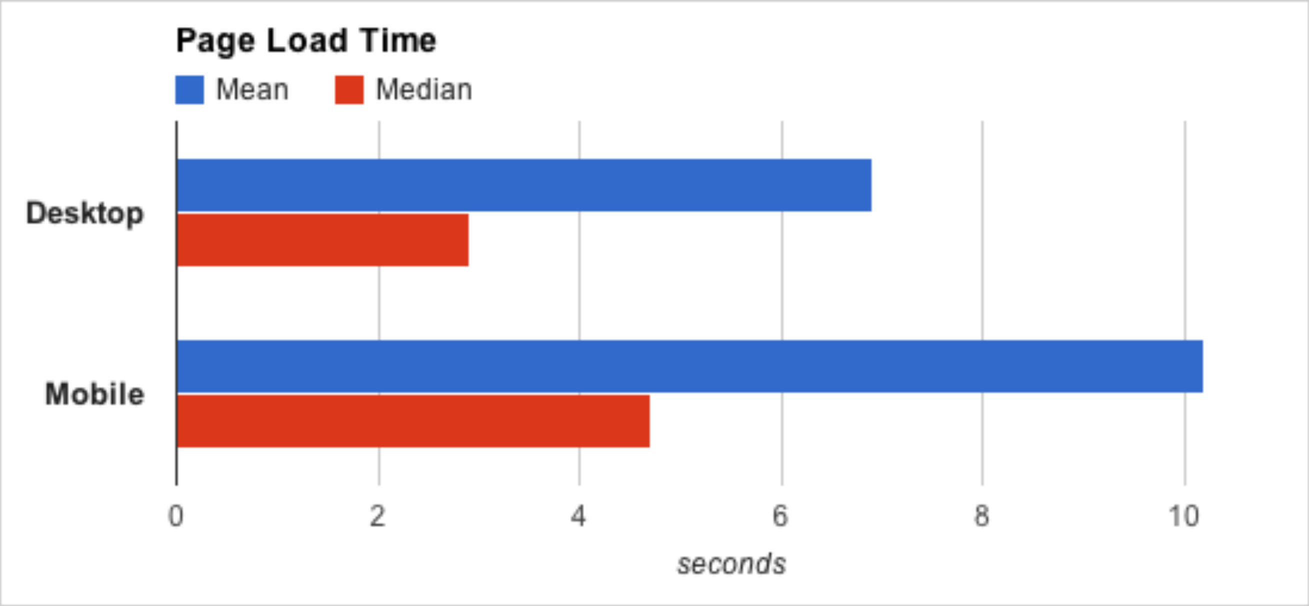
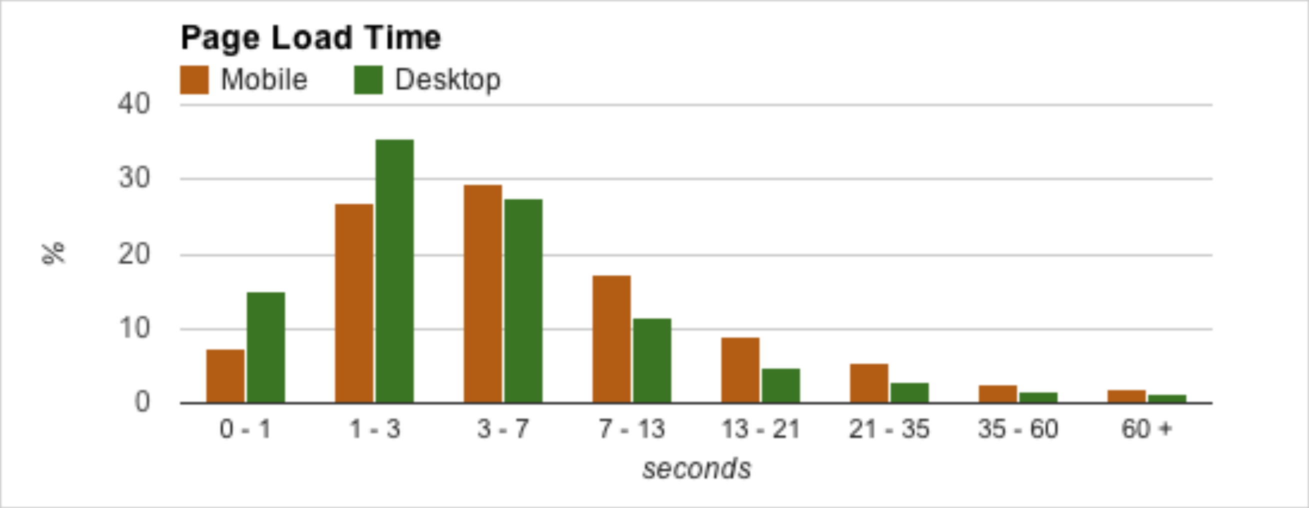
Speed up TCP to make the Web faster

郑又中 **Yuchung Cheng**
Google
ycheng@google.com

http://googlecode.blogspot.com/2012/01/lets-make-tcp-faster.html





Desktop

Median: ~2.7s

Mean: ~6.9s

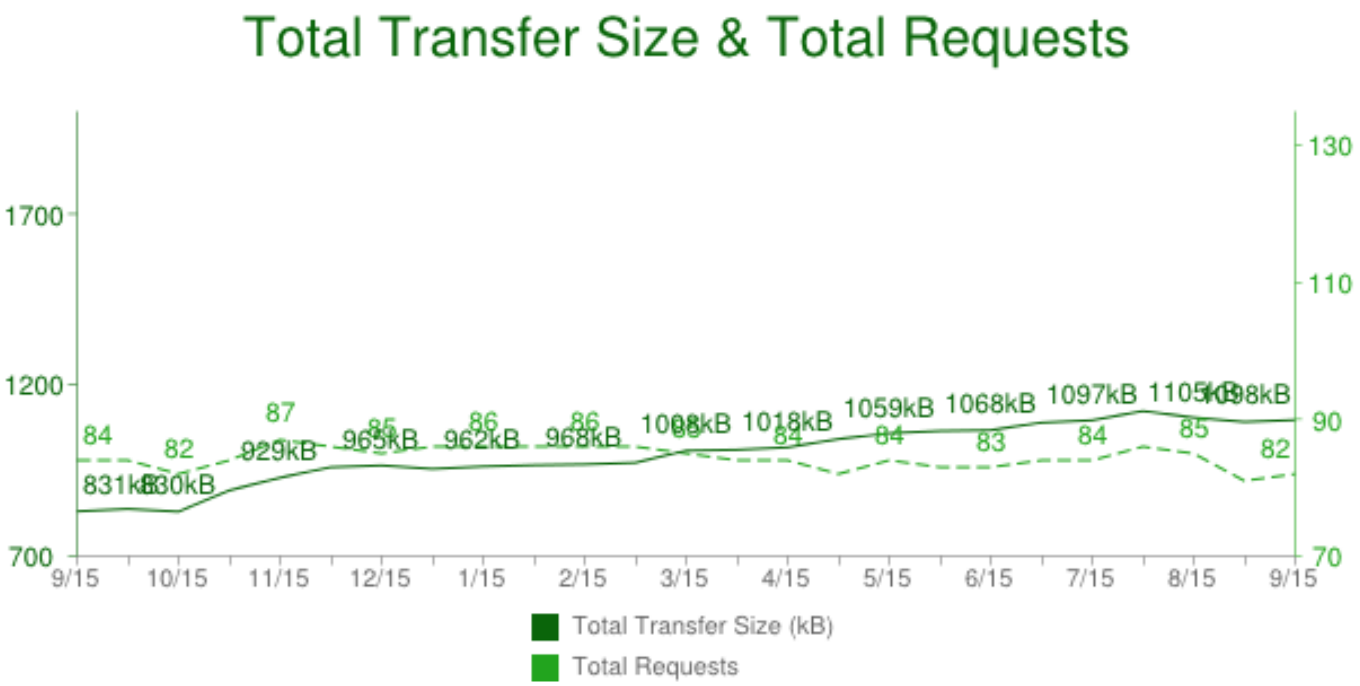
Mobile *

Median: ~4.8s

Mean: ~10.2s

* *optimistic*

[How Fast Are Websites Around The World?]() - Google Analytics Blog (April, 2012)

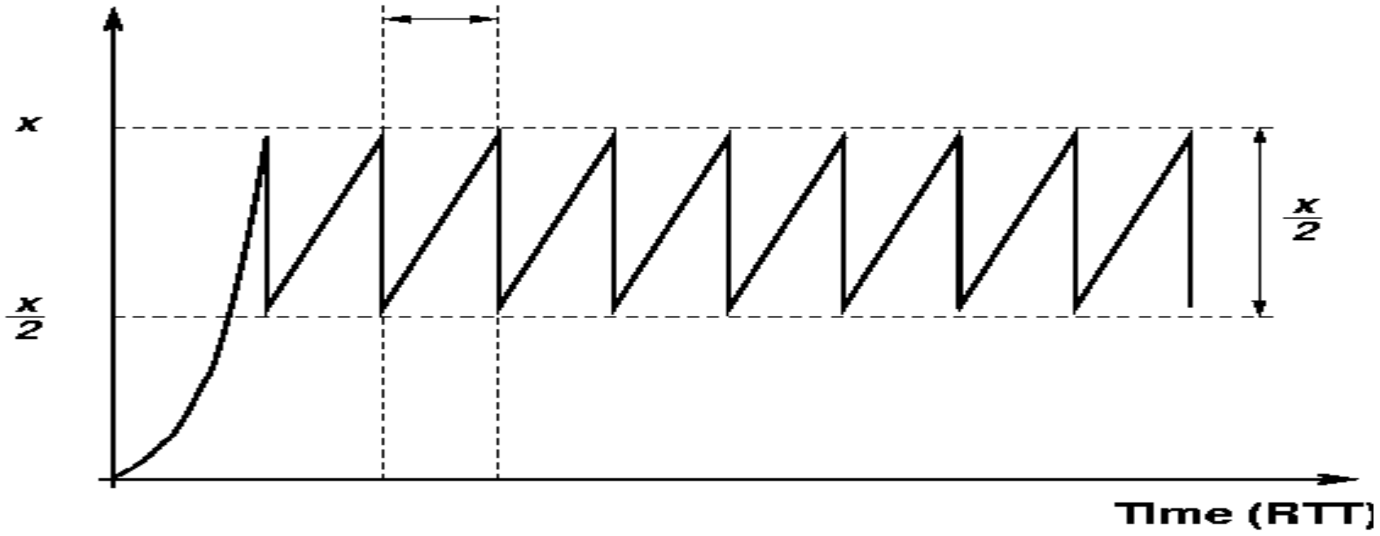


Content Type	Avg # of Requests	Avg size
HTML	8	44 kB
Images	53	635 kB
Javascript	14	189 kB
CSS	5	35 kB

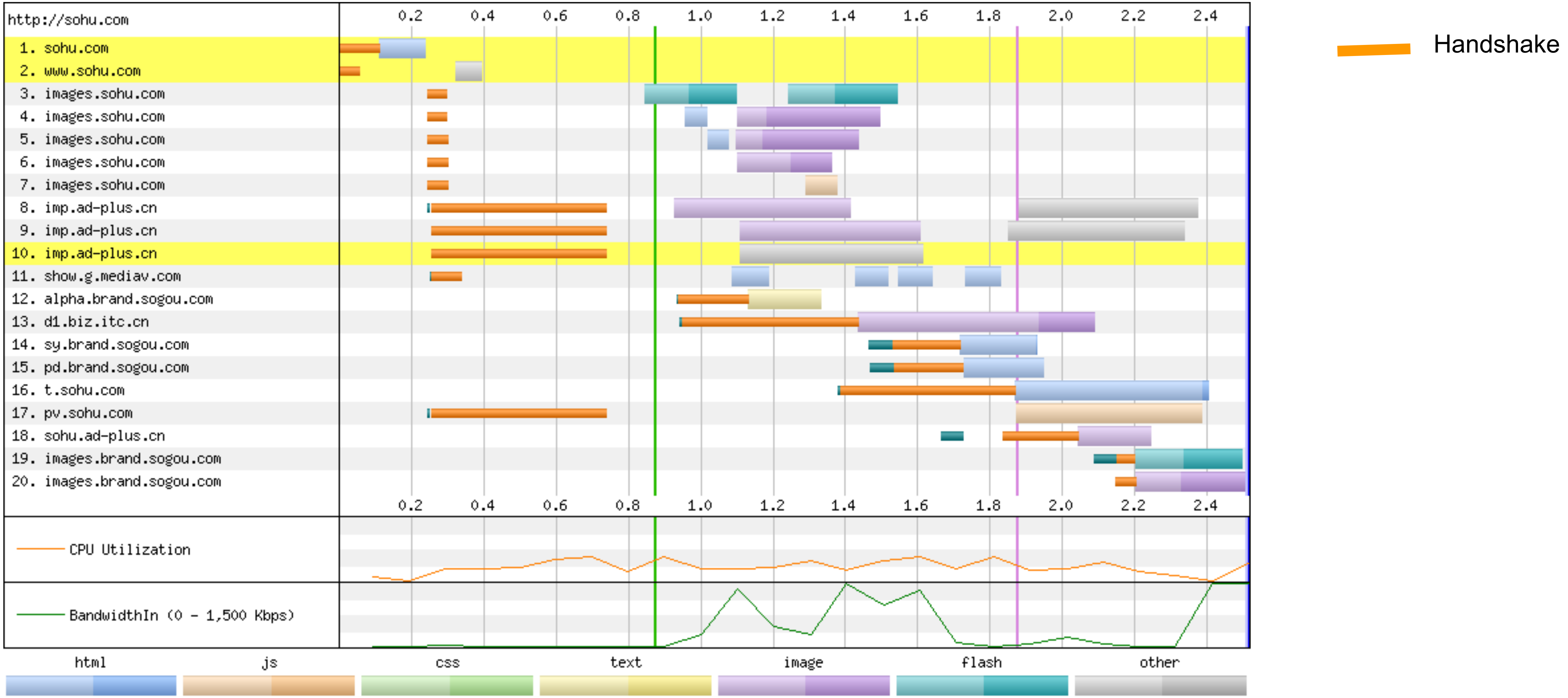
HTTP Archive - Trends (Sept, 2012)

Transport Control Protocol (TCP)

1. Reliable and serialized delivery (RFC 793, 1981 -)
   a. Export a reliable data pipe to apps
   b. Retransmit if packet is not acked
2. Congestion control (RFC 5681, 1988 -)
   a. Adjust sending rate based on ACK rate (ack-clocking)
   b. Slide congestion window to send more data
3. Optimized for bulk transfer (large file)



TCP is slow for the Web



Emulating a Chrome user in Hong Kong with 1.5Mbps DSL accesing sohu.com with a warmed browser cache (http://webpagetest.org)

Existing TCP workarounds

Why it won't work in the long run

Persistent TCP connections

- Open & keep many TCP connections
  - Don't reduce cwnd after idle
- Problems
  - Still slow - TCP handshake for first contact and slow start after idle
  - Don't scale - need to manage many connections
    - Client, server, network
  - Port exhaustion - NAT boxes *silently* drop connections
  - Energy inefficient - TCP/FIN wakes up radios

- Persistent TCP connection is not a long term solution

Let's make TCP fast

Fast startup, loss recovery, and congestion control for mobile

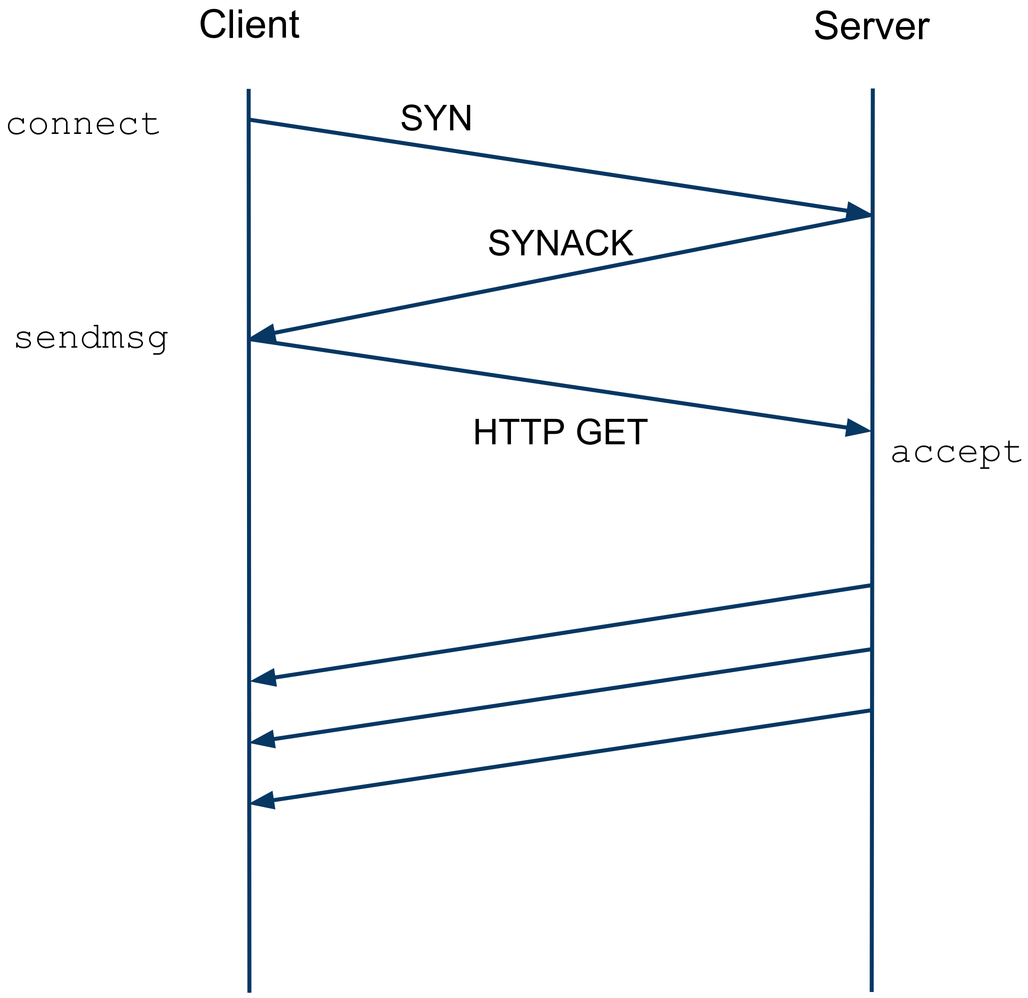
TCP handshake

35% HTTP and 11% SPDY requests wait for handshake

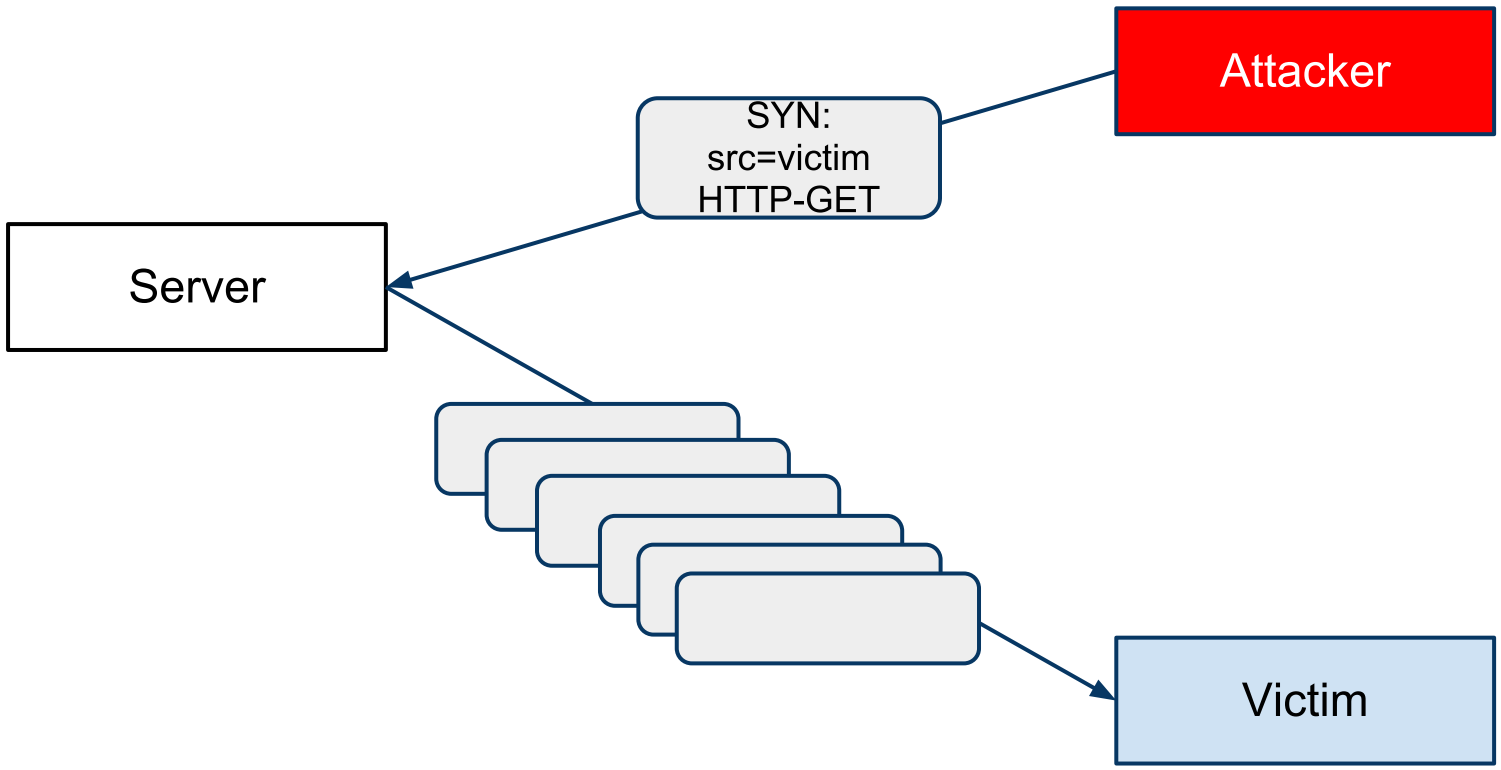
- 1 RTT overhead
- HTTP-GET in SYN

Challenges

- Resource exhaustion attack
- Amplification attack

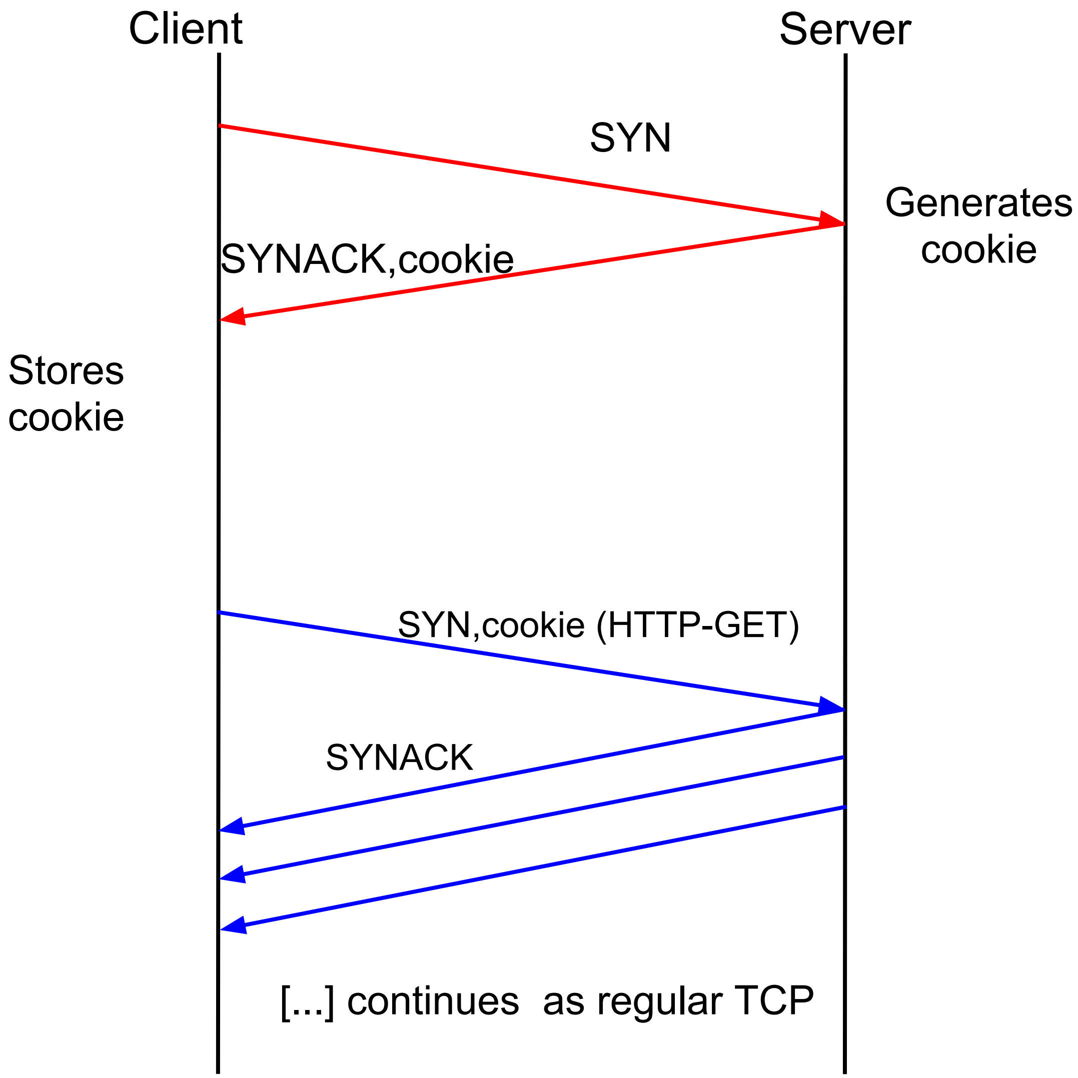


Resource exhaustion & amplified reflection attacks



1 (spoofed) SYN packet for 10 data packets

TCP Fast Open



Server grants a nonce, Fast Open cookie (FOC)

- AES_encrypt(client_ip, secret)
- TCP option (32 - 64bits)

Client sends HTTP-GET in SYN with cookie

Server accepts HTTP-GET in SYN if cookie is valid

Defend simple SYN-data flood attacks

Defending attacks

An attacker can still

- Obtain valid cookie via a mole
- Flood spoofed SYN-data from another bot

Defense

- Periodically rotate server secret
- Disable and use SYN cookie if pressured

Other scenarios

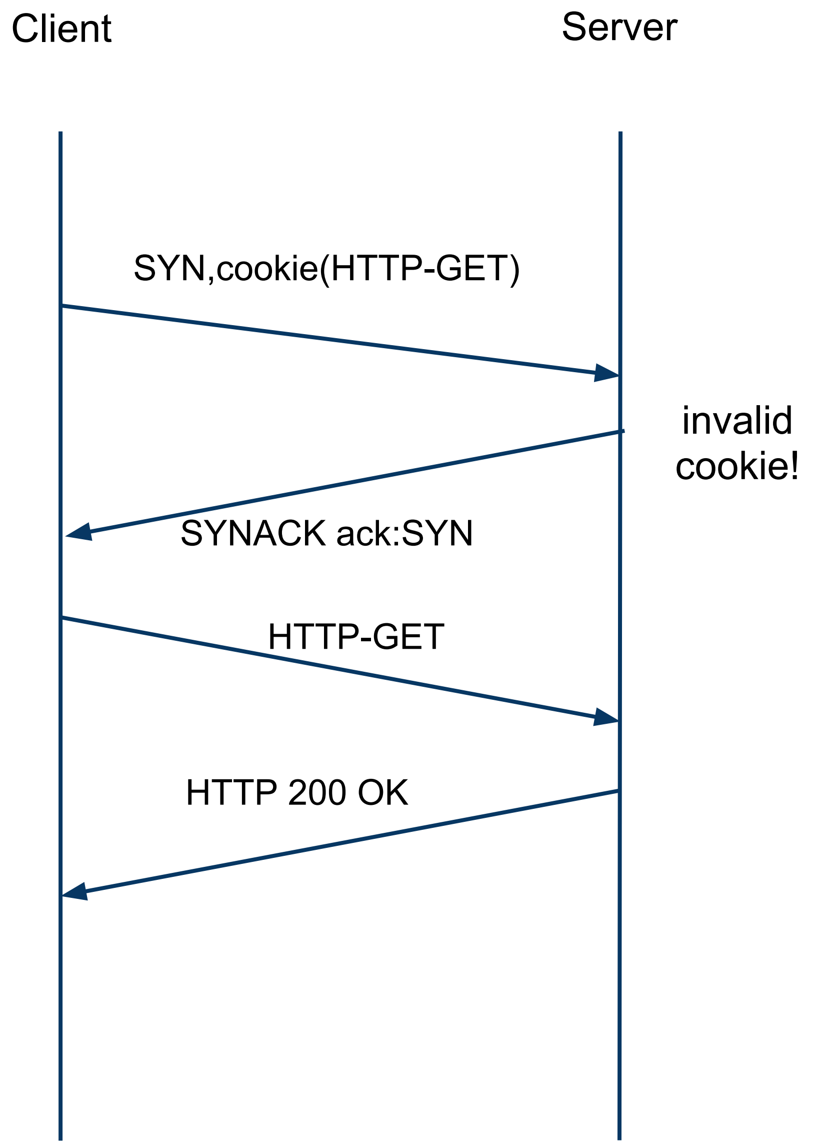
- NAT
- Man-in-the-middle
- Firewalls drop SYN/data or strip cookie option

Graceful fallback

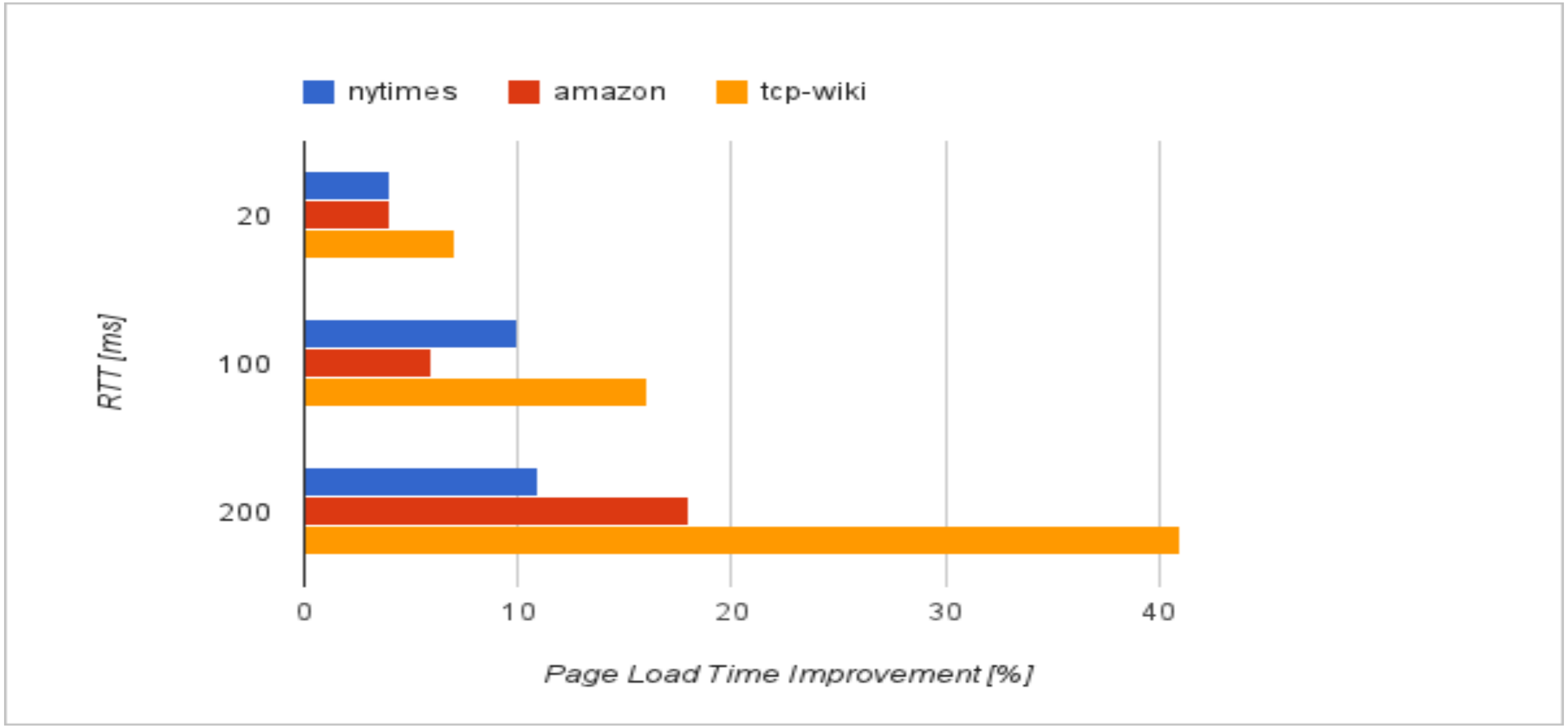
Server can always only acks the initial (SYN) sequence (e.g., SYN flood attack)

Client retries the data after handshake like regular TCP

No performance penalty!



Page load time benchmarks



"TCP Fast Open", SIGCOMM CoNEXT 2011 (best paper nominee)

Using Fast Open for your applications

Client:

- ~~connect() then write()~~
- sendto(data, MSG_FASTOPEN)

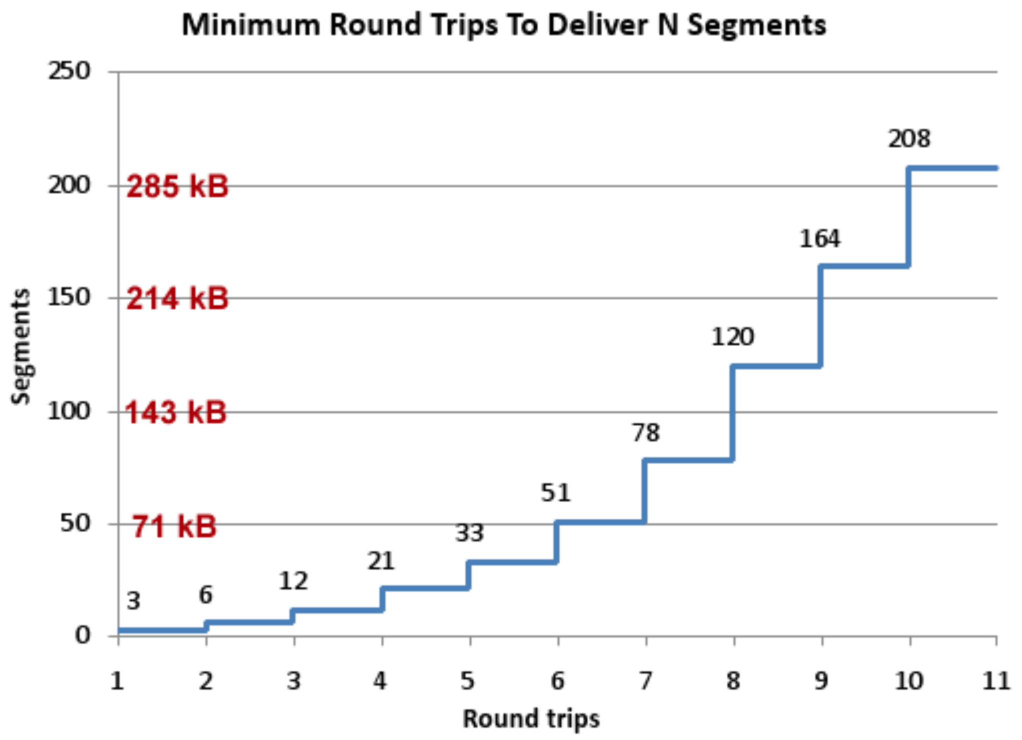
Server:

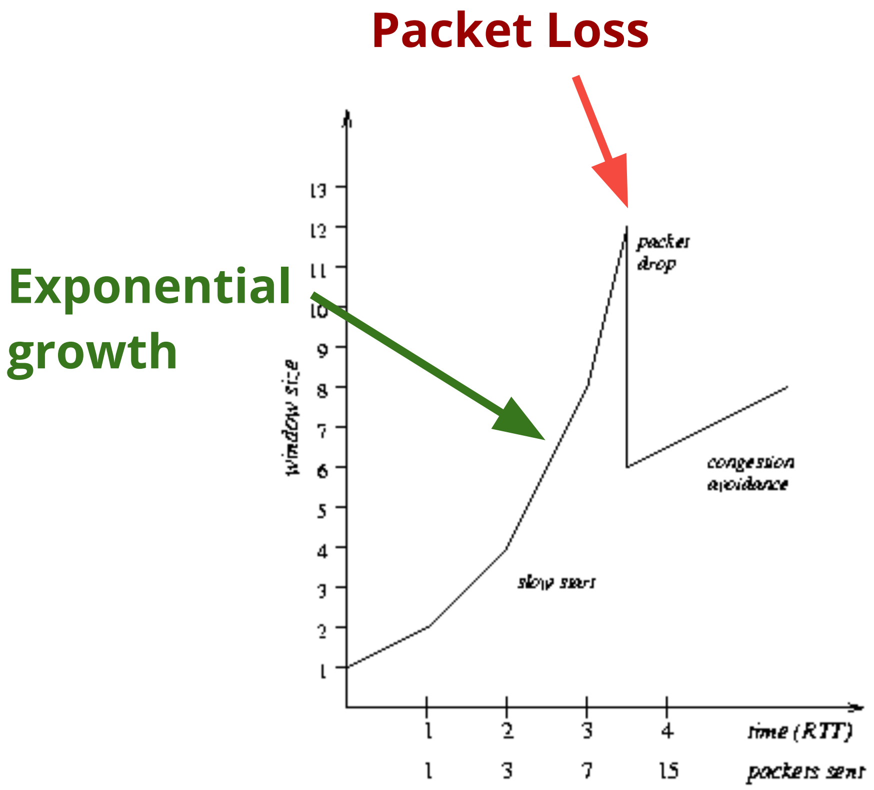
- setsockopt(TCP_FASTOPEN)

Available in Linux 3.7 & being deployed on Google.com

TCP slow start

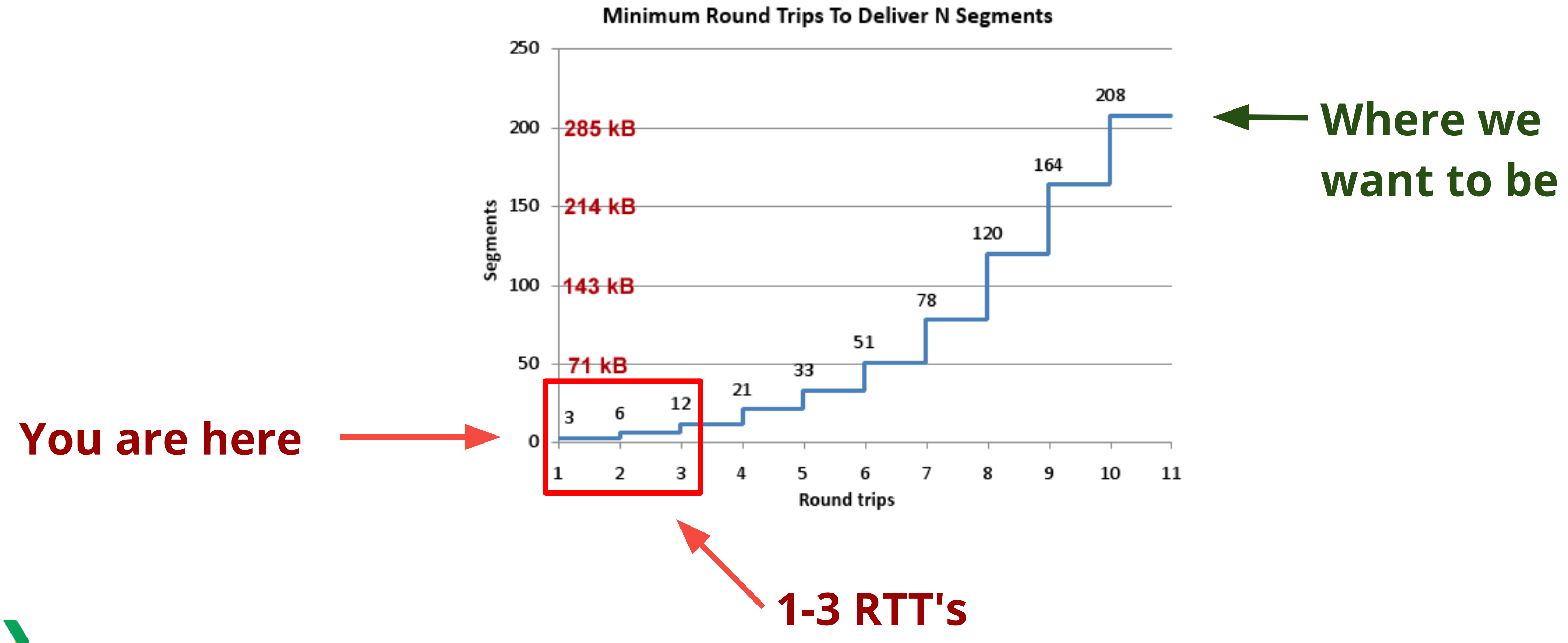
- TCP is designed to probe the network to figure out the available capacity
- **TCP Slow Start** - feature, not a bug





HTTP Archive says...

- 1098kb, 82 requests, ~30 hosts... ~**14kb per request!**
- Most HTTP traffic is composed of small, bursty, TCP flows



An Argument for Increasing TCP's Initial Congestion Window

Nandita Dukkipati Tiziana Refice Yuchung Cheng Jerry Chu Natalia Sutin
Amit Agarwal Tom Herbert Arvind Jain
Google Inc.
{nanditad, tiziana, ycheng, hkchu, nsutin, aagarwal, therbert, arvind}@google.com

ABSTRACT

TCP flows start with an initial congestion window of at most three segments or about 4KB of data. Because most Web transactions are short-lived, the initial congestion window is a cr can dra valu cha

I tion larg ben of network bandwidth, round-trip time (RTT), bandwidth-delay product (BDP), and nature of applications. We show

for standard Ethernet MTUs (approximately 4KB) [5]. The majority of connections on the Web are short-lived and finish before exiting the slow start phase, making TCP's initial congestion window (*init_cwnd*) a crucial parameter in deter- initial short TCP's per a obally width Kbps) appli- oad of Web pages. Popular Web browsers, including IE8 [2], Fire-

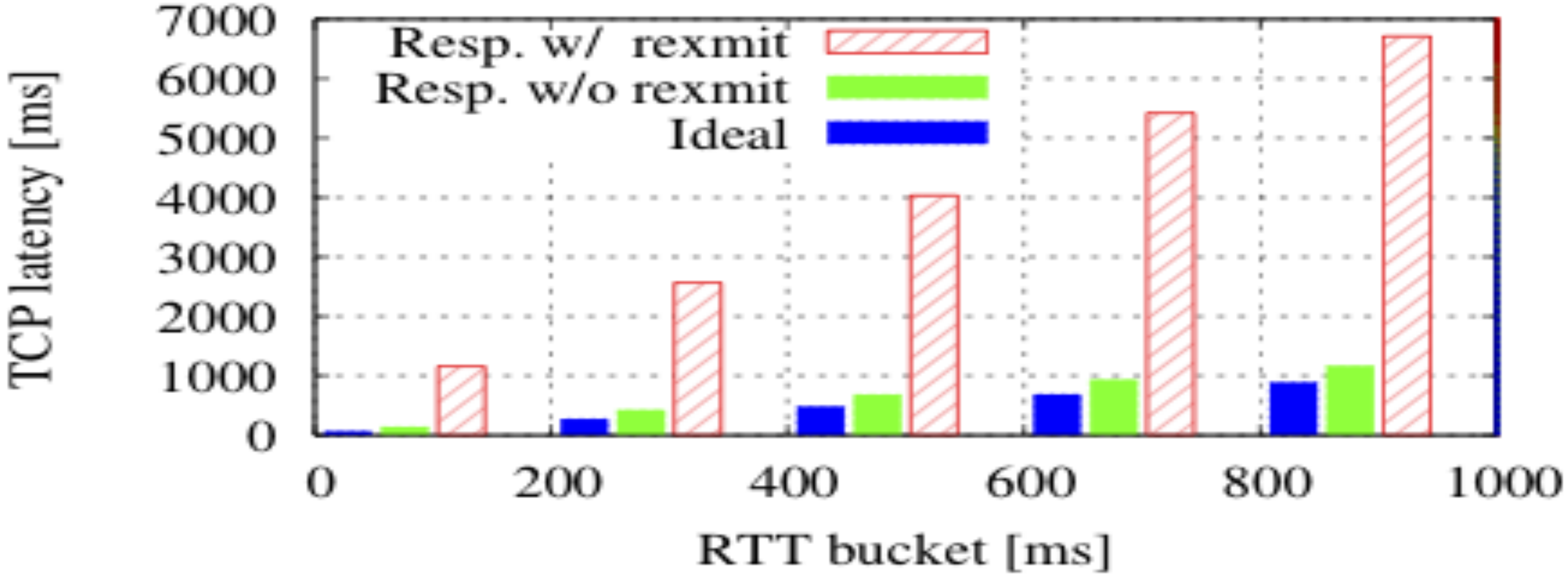
Update CWND from 3 to 10 segments, or ~14960 bytes

Default size on Linux 2.6.33+ - double check yours!

Google servers use it since 2010

[An Argument for Increasing TCP's initial Congestion window]

HTTP/TCP are 5 - 10 times slower on lossy networks



Why is TCP slow on packet losses

TCP recover losses in two ways

- Fast recovery (1 RTT): need dupacks
- Timeout (often 5-10 RTTs)

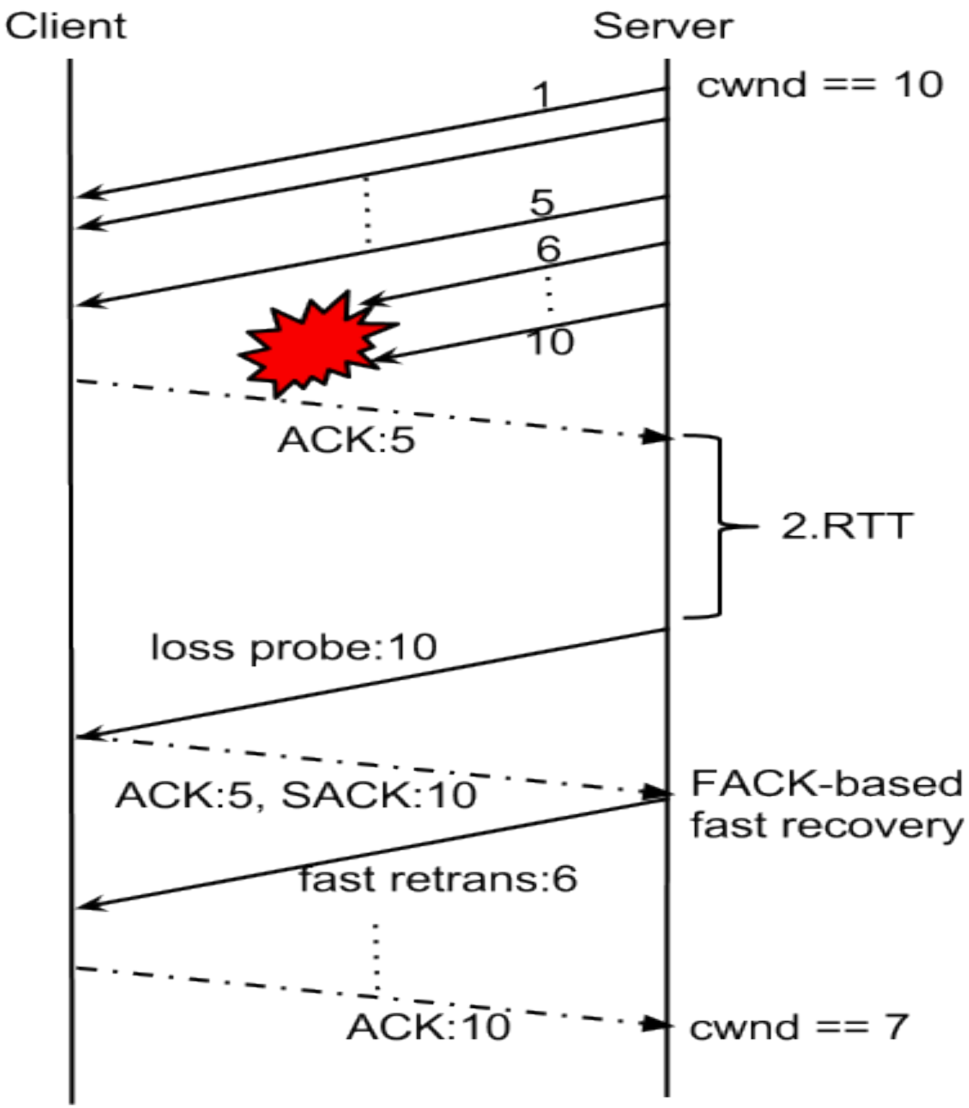
Most losses in HTTP are tail drops (lost last N packets)

- No dupack to trigger fast recovery
- 70% losses on Google.com are recovered by timeout
- Timeout is long on short flows due to few RTT samples

Solution: Tail Loss Probe (TLP)

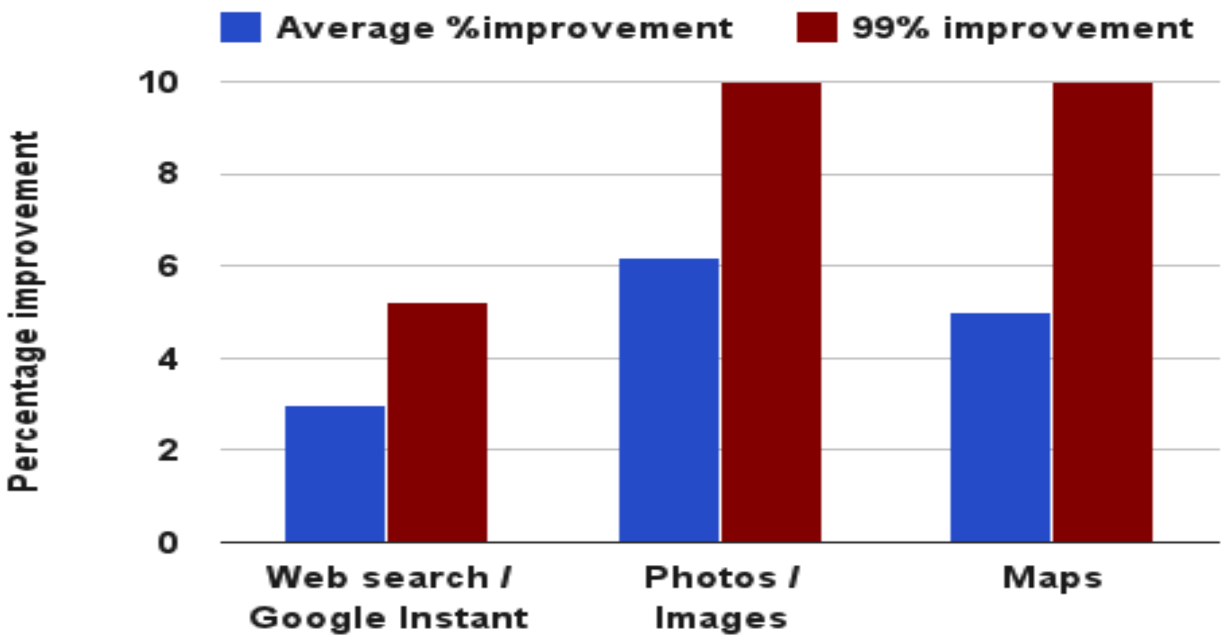
- Retransmit the last packet to trigger fast recovery

Tail Loss Probe

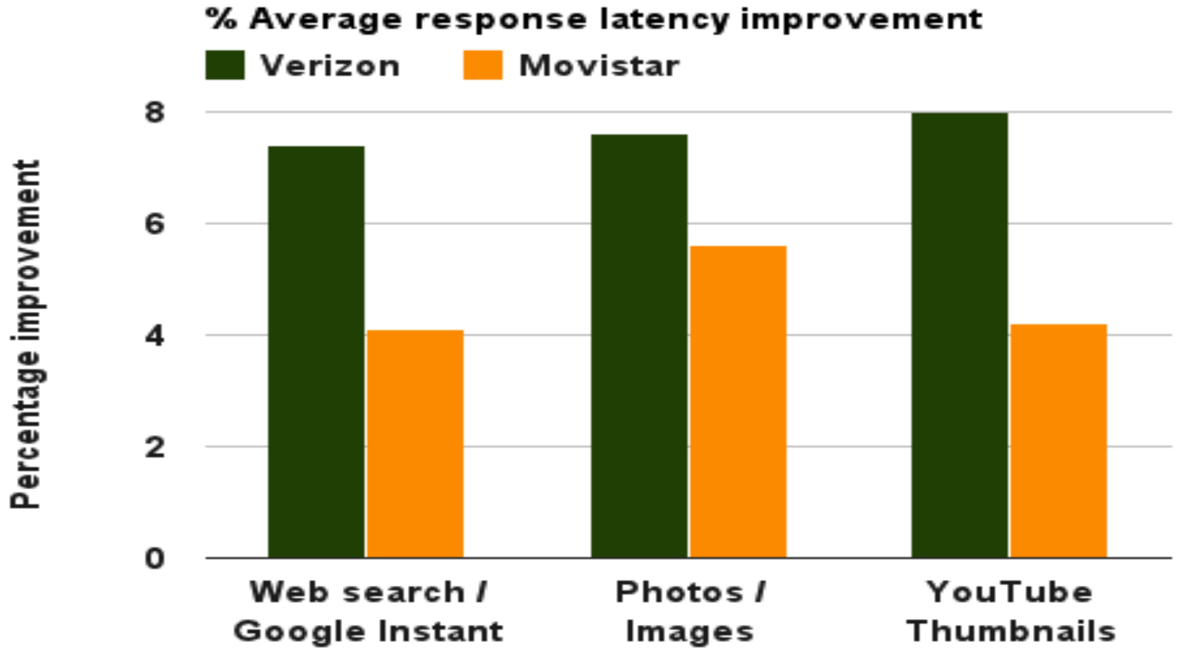


Tail Loss Probe performance

- 6% avg. reduction in HTTP response latency. 10% for 99%ile



Mobile only



But TCP performance on mobile is terrible

We have some ideas ...

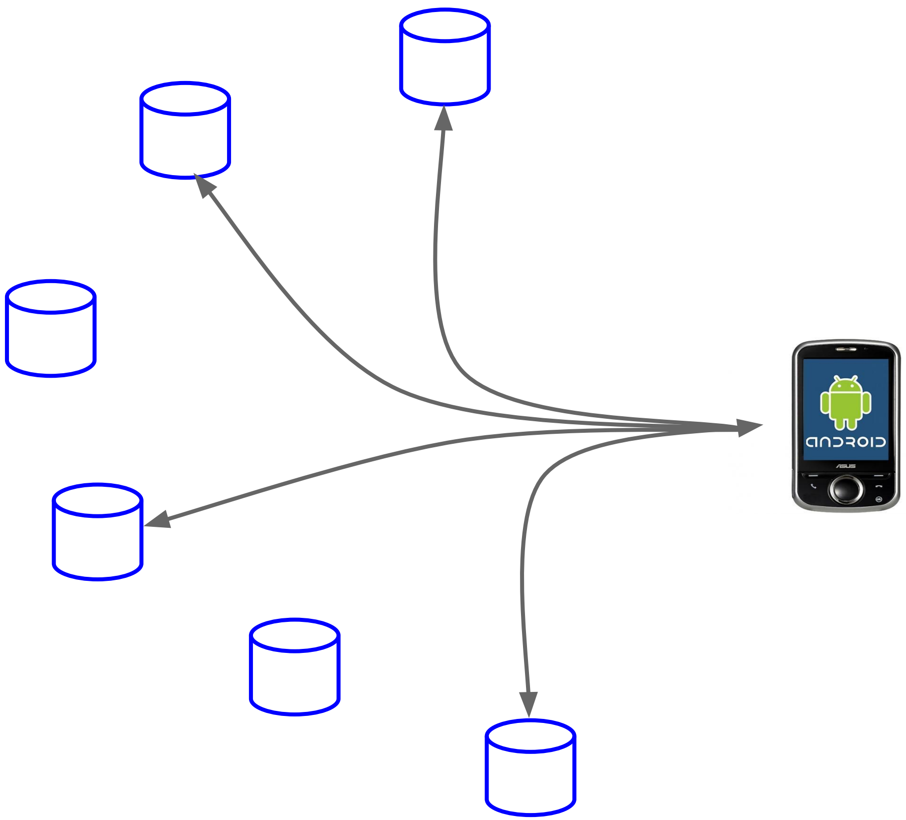
Mobile networks are very different

	Desktop	Mobile
Loss (TCP)	Low	Low (wireless codec / rexmit)
Delay variation	Low (queuing)	High (wireless)
Rate change	Stable (cross traffic)	Fluctuates (wireless)
Cross traffic	Same and other users	Same user
Disconnection	Frequent	Almost never

TCP congestion control is not working on mobile

Current TCP congestion control

- Sender-based
- Slowly probe and react to network rate changes (until loss or delay is too large)
- Per-flow fairness



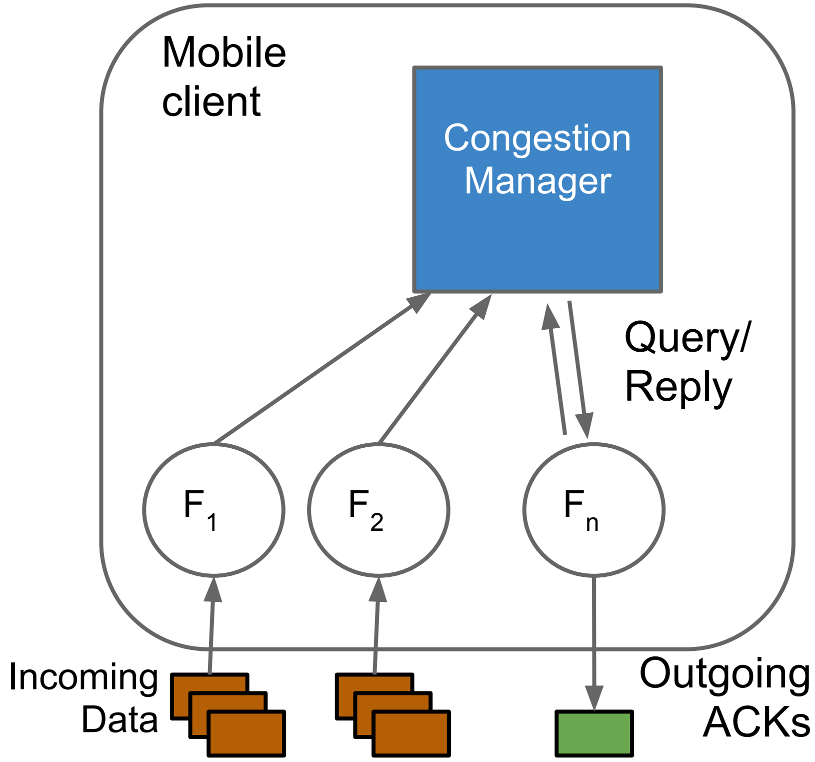
New mobile congestion control

Key feature: client-centric

- Measures the mobile link rate
- Instruments the rate to the sender
- Prioritizes important connections

Version 0.01: SPDY-cwnd-persist frame

- server: SPDY-GO_AWAY (cwnd is 25)
- client: SPDY-SYN (cwnd was 25)



Conclusions

TCP is critical for Web performance but it's not optimized for Web

1. Fast Open - client sends HTTP-GET when connect
   a. Linux 3.7
2. IW10 - server sends 10 packets initially
   a. Linux 2.6.33+
3. Tail Loss Probe - recover losses within 2-3 RTTs
   a. Open source in Q1/2013
4. Congestion control for mobile
   a. Under active research. Will open source in 2013

Google "ietf tcpm google" for our RFC proposals in IETF