Challenges of building accurate web speed tests for high-speed access links

Ricky Mok, kc claffy (CAIDA/UCSD)
Adnan Ahmed, Zubair Shafiq (U. Iowa),
and Amogh Dhamdhere (Amazon)

April 16, 2019

Speedtests

- Diagnose the network
- Validating access link capacity
- Measuring the throughput between (video) servers/caches and users

How do they work?

- Flood the network with TCP measurement flow(s)
  - HTTP GET/POST
  - WebSocket
- Download test: HTTP GET
- Upload test: HTTP POST

  Bandwidth = # Bytes transferred / Time
- Vary by measurement parameters and server deployment

Measuring high-speed network

- Are they accurate?
  - [Goga12]
- Can they measure high-speed (1Gbps) access links?
  - Require more data to fill up the link
  - More sensitive to network/system factors
- Can we locate the bottleneck link?

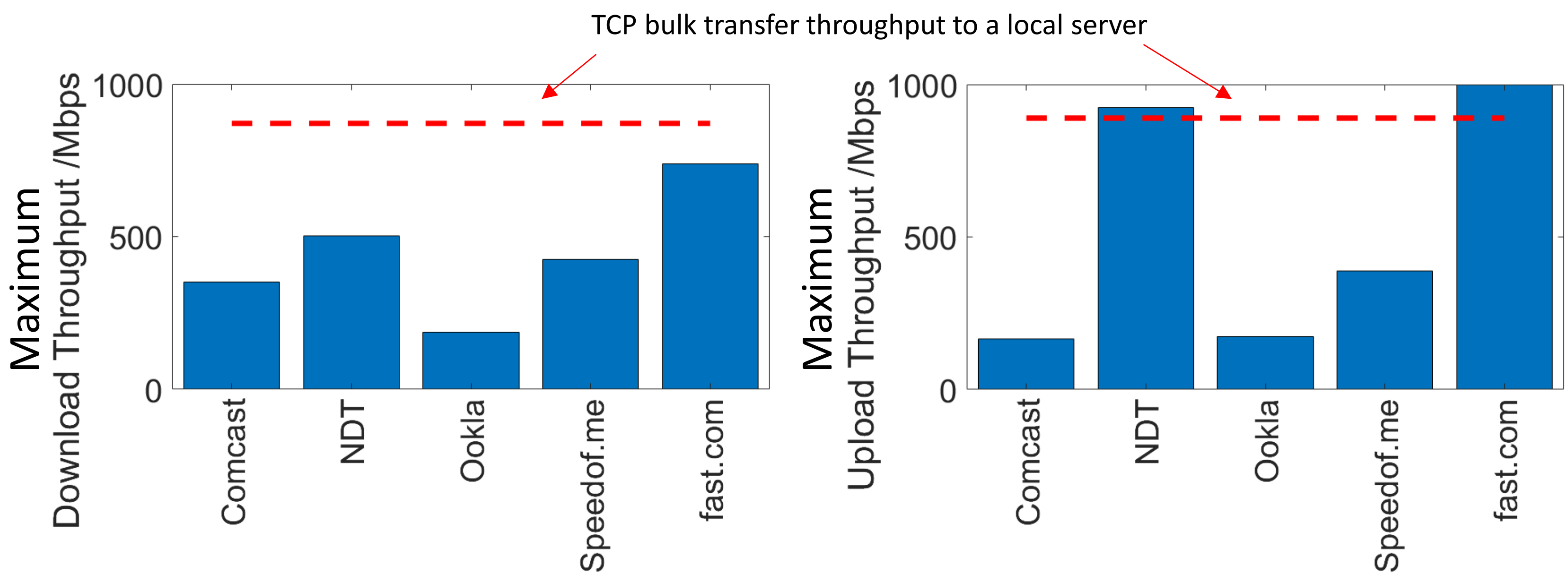
[Goga12] Oana Goga and Renata Teixeira, "Speed Measurements of Residential Internet Access", Proc. PAM, 2012

A lab experiment

- We used a headless chromium browser to consecutively run 5 speed tests for 20 times from a host connected with campus network using Gigabit Ethenet.
  - IPv4
  - “Default” server selection
  - Record browser performance trace
  - tcpdump

Results

- All download tests underestimated the downlink throughput from 15% to 78%.
- Comcast and Ookla tests only reports the uplink throughput as ~170Mbps.



Measurement challenges

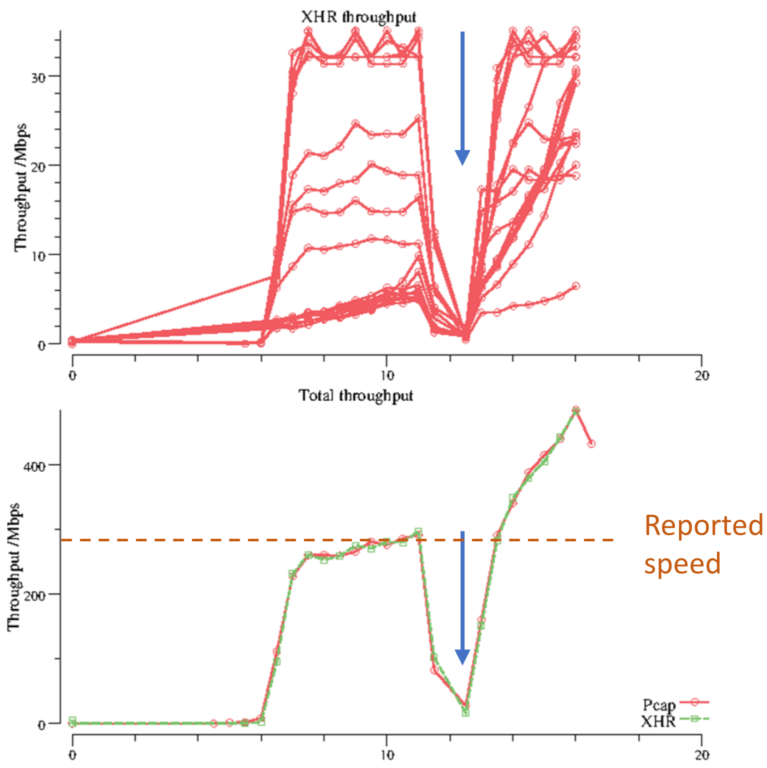
- TCP behaviour
  - TCP Global Synchronization
  - TCP fairness
- Browser
  - Cross-origin resource sharing (CORS) policy
  - Internal overheads
- Selection of measurement server

Measurement challenges

- TCP behaviour
  - TCP Global Synchronization
  - TCP fairness
- Browser
  - Cross-origin resource sharing (CORS) policy
  - Internal overheads
- Selection of measurement server

TCP Global Synchronization

- Concurrent TCP flows can saturate the bottleneck link, but they can be synchronized by bursty packet loss. [Zhang91]
- The test can easily underestimate the throughput.



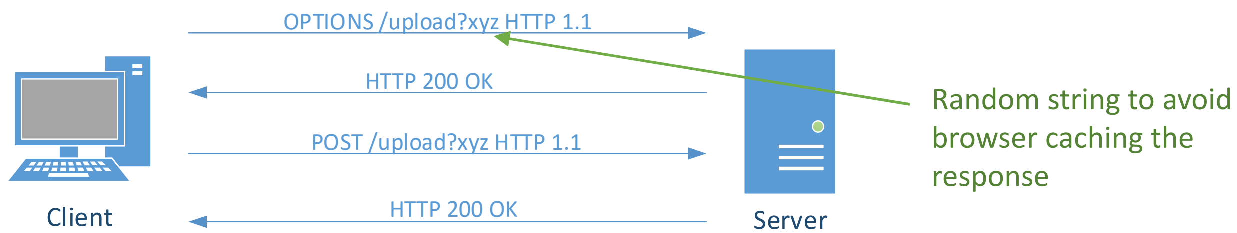
[Zhang91] L. Zhang, S. Shenker, and D. D. Clark. Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic. In *Proc. SIGCOMM*, 1991.

Detect and react to TCP behavior

- Detect abnormal changes in throughput
- Adaptively change the measurement parameters
  - test duration
  - number of flows

Brower CORS policy

- Browser elicits *preflighted requests*+ for the HTTP headers before cross-origin HTTP POST requests



- Extremely high overhead when the upload test is implemented using 0-content length HTTP POST
  - An upload test can send few hundreds of POST requests
  - Preflighted requests are invisible to JavaScript
- The upload test of Xfinity speed test and Ookla speedtest suffered from this problem.

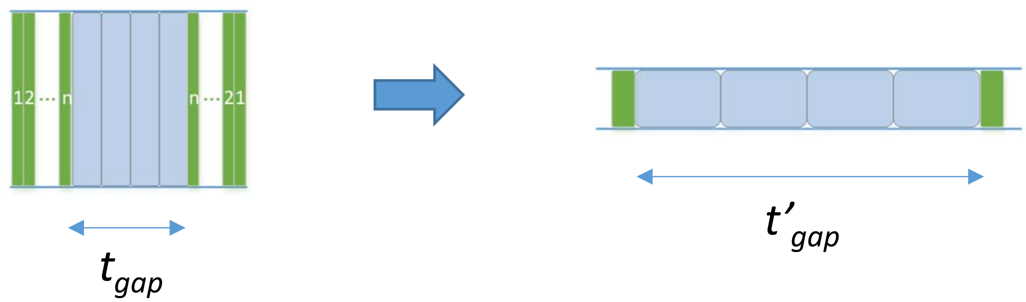
+https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#Preflighted_requests

Avoiding preflighted requests

- "Simple requests"
  - No custom HTTP header
  - Few content type
- Re-use the exactly same HTTP requests
  - Use Cache-control: no-cache header to prevent the browser cache the responses
- Cache the results of the first preflighted request
  - Access-Control-Max-Age

Locating bottleneck

- Not only measure the throughput, but also locate the bottleneck router
- Pathneck. Recursive Packet Train (RPT) [Hu04]

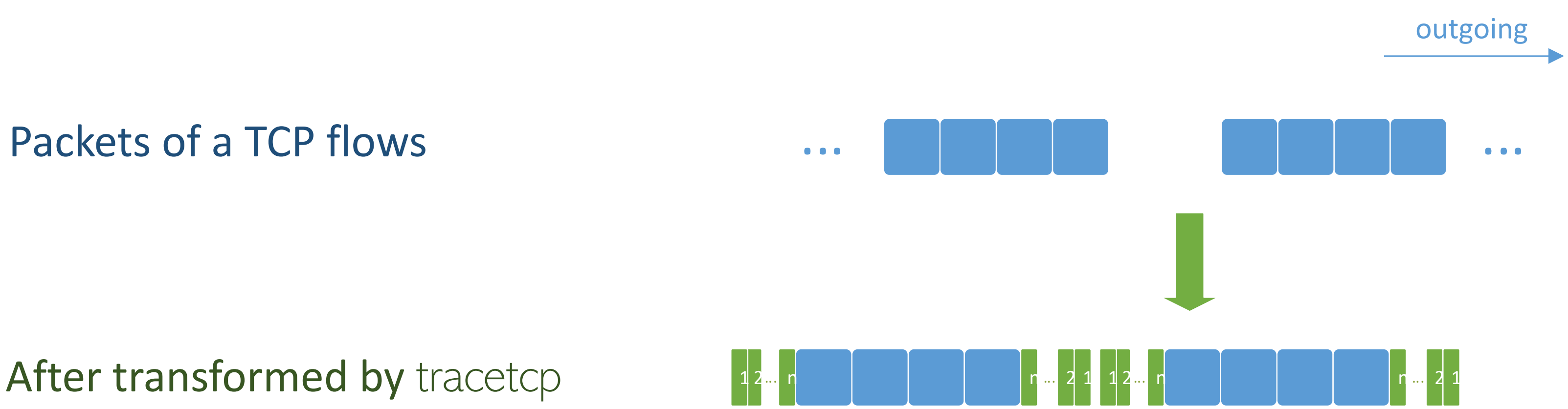


- Use the response time of the TTL-limited probes to capture the packet train dispersion (t'_{gap}) after traversing each hop

[Hu04] Ningning Hu, Li Erran Li, Zhuoqing Morley Mao, Peter Steenkiste, Jia Wang. "Locating Internet Bottlenecks: Algorithms, Measurements, and Implications." Proc. ACM SIGCOMM, 2004.

In-flow measurement

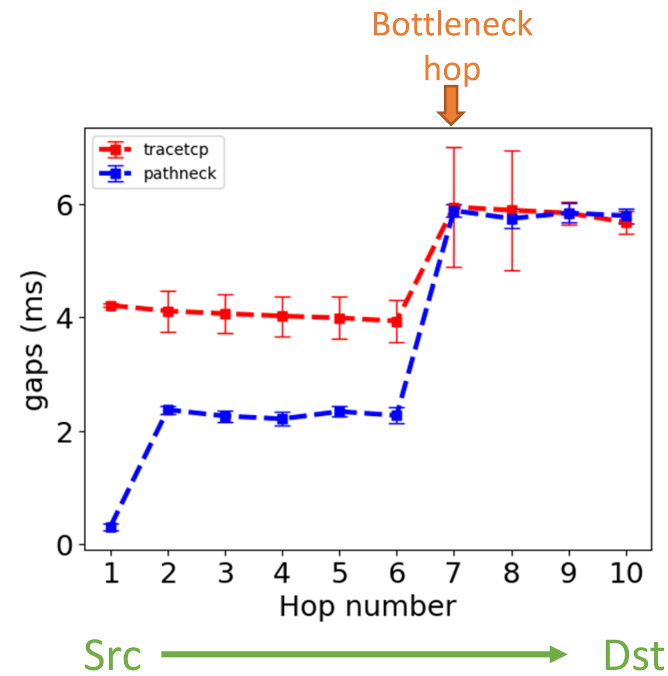
- We developed tracetcp* to
  - manipulate an existing TCP flow to construct RPTs
    - Not require to inject additional data packets
  - reveal the performance experienced by the TCP flow



* Joint work with Adnan Ahmed, Zubair Shafiq (U. Iowa), and Amogh Dhamdhere (Amazon)

Emulab experiment

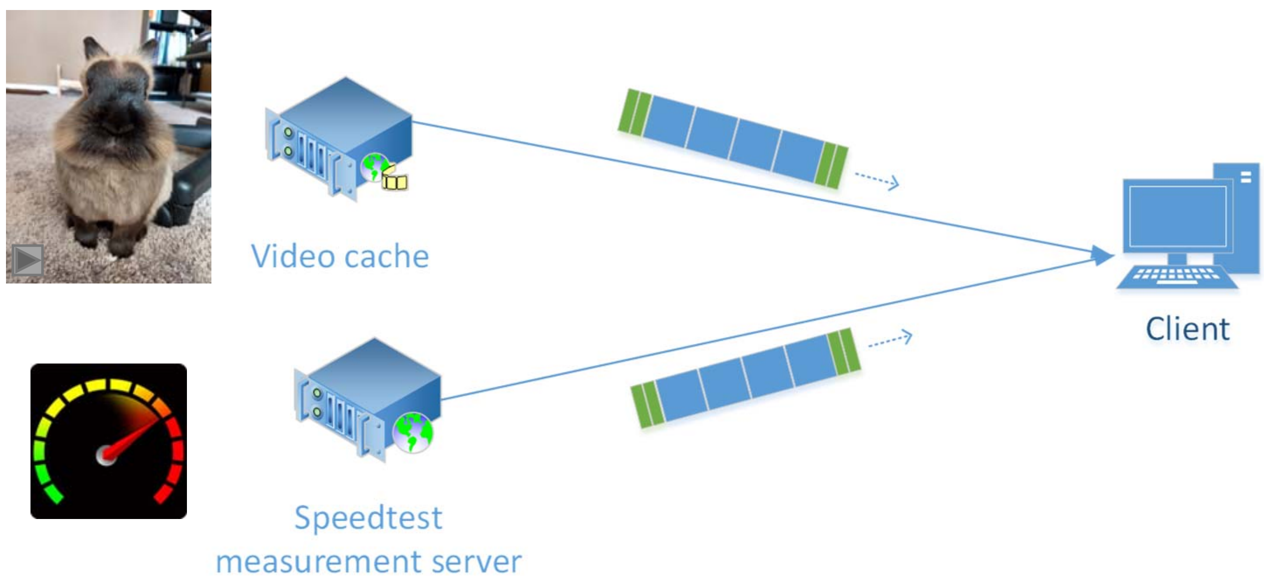
- Set up a 10-hop testbed in Emulab #
- tracetcp exploits an iperf TCP flow to locate the bottleneck router
- Packet train dispersion significantly inflated at the 20-Mbps bottleneck



#https://www.emulab.net

Deployment scenarios

- We can deploy tracetcp in the server-side
  - Downstream link bottleneck



Conclusion

- Current speedtest platforms have room to improve
  - TCP
  - Browser
- We developed tracetcp, which
  - Employs *in-flow* paradigm
  - Locates bottleneck
  - Enables server-side measurement

Thank you

cskpmok@caida.org