Testing FAST TCP over Abilene

Stanislav Shalunov

Internet2 Fall Member Meeting, Indianapolis, 2003-10-15
People who made the testing possible

- Cas D’Angelo, SoX GigaPoP
- Steven Low, Caltech
- John Moore, NC-ITEC
- Matt Mathis, PSC
- David Richardson, PNW GigaPoP
Test FAST TCP over a real network

- **Goal:** See how FAST TCP runs with conventional TCP

- 8 machines
  - 4 machines at SoX in Atlanta (1 Reno, 3 FAST)
  - 2 machines at PNW in Seattle (1 Reno, 1 FAST)
  - 1 machine at NC-ITEC in Raleigh (FAST)
  - 1 machine at PSC in Pittsburgh (FAST)
  - All machines have **independent 1 Gb/s paths** to Abilene
Test Topology

- 4 paths (1 Reno, 3 FAST)
  - path1: Reno, from Seattle to Atlanta, RTT = 57.6 ms
  - path2: FAST, from Raleigh to Atlanta, RTT = 23.7 ms
  - path3: FAST, from Seattle to Atlanta, RTT = 57.4 ms
  - path4: FAST, from Pittsburgh to Atlanta, RTT = 26.9 ms
- All machines in Atlanta on different router ports
- Individual path bottlenecks are at the senders
- System bottleneck is the OC-48 interface of Abilene core router in Atlanta going towards SoX
Test Plan

• Builds upon earlier testing (see http://www.internet2.edu/~shalunov/fast-tcp-abilene/)

• Baseline: **15 minutes** on each path

• Concurrent test: each flow lasts **1 hour** and flows start with 15 minute delays in the following order: path1, path2, path3, path4

• Test window: 2am-6am Eastern Time

• Use of 2.5 Gb/s link by background traffic: < 10%

• An OC-48 link (the link from ATLA to SoX) was **saturated**

• Both throughput and round-trip time were measured in 1-second intervals
i2perf—Network Throughput and Delay Tester

- A network tester more or less like iperf
- Developed for these FAST tests to measure RTT
- i2perf, briefly described in terms of main differences from iperf:
  - can measure RTT
  - produces machine-readable output
    - single stream only at this time
    - TCP only at this time
- http://www.internet2.edu/~shalunov/i2perf/ to download
Baseline throughput of path1 (Reno)

Limited by sender CPU speed. Canary in a mine.
Baseline RTT of path1 (Reno)

Stable.

Baseline delay of path1 (Reno)

path1 baseline rtt, min = 57.626ms

Round-trip time, ms

Time since start of experiment, s
Respectable, but I've seen 1Gb/s flat lines.
Baseline RTT of path2 (FAST)

Not sure what could cause the small spikes.
Baseline throughput of path3 (FAST)

Similar to path2 baseline.

Baseline throughput of path3 (FAST)

path3 baseline xput, avg = 511.642 Mb/s

Goodput, Mb/s

0 200 400 600 800 1000

0 100 200 300 400 500 600 700 800 900

Time since start of experiment, s
Baseline RTT of path3 (FAST)
Baseline throughput of path4 (FAST)

Less oscillation, hiccup in the beginning, note the slope.
Baseline RTT of path4 (FAST)

Rapid internal queue buildup and slow draining in the beginning.
Main Test Throughput (Concurrent Run)
Main Test RTT (Concurrent Run)
Observations

- Reno almost retained throughput even during congestion (170 Mb/s vs. 180 Mb/s)
- Reno had nearly constant RTT even during congestion (57 ms vs 62 ms, or about 2 ms per FAST stream)
- Reno suffered no single loss even during congestion
- While congesting network, FAST streams appear to equalize RTT
  - But, since Reno sharing bottleneck does not experience any significant increase of queueing delay, FAST queues have to be internal to the hosts
- Presence of several competing FAST flows actually seems to stabilize their behavior
Questions to Answer with Future Testing

• Do streams stall often like they did approximately before second 4500?

• Would interactive users notice the presence of FAST streams making full use of the link?

• Are late-comers really at a disadvantage? (See the spikiness of path4 even after other paths stop sending.)

• More dynamic testing to see FAST TCP behavior under a wide variety of network conditions.