Evaluating New TCP Congestion Control Algorithms

Doug Leith
Hamilton Institute

Thanks: Bob Shorten, Yee-Ting Li, Baruch Even, Gavin McCullagh

Two questions …

1. How can we measure the performance of proposed changes to TCP in a systematic and meaningful way (that relates to issues of practical interest, supports fair comparisons) ?

2. Live experimental testing is time consuming, difficult and expensive. Can we screen for known issues and gotchas at an early stage (e.g. via simulation or lab testing) prior to full scale experimental testing ?

Questions are related of course.

Also, no screening or measurements can be exhaustive – we cannot prove the correctness of a protocol – but we can demonstrate incorrectness and tests can improve confidence.

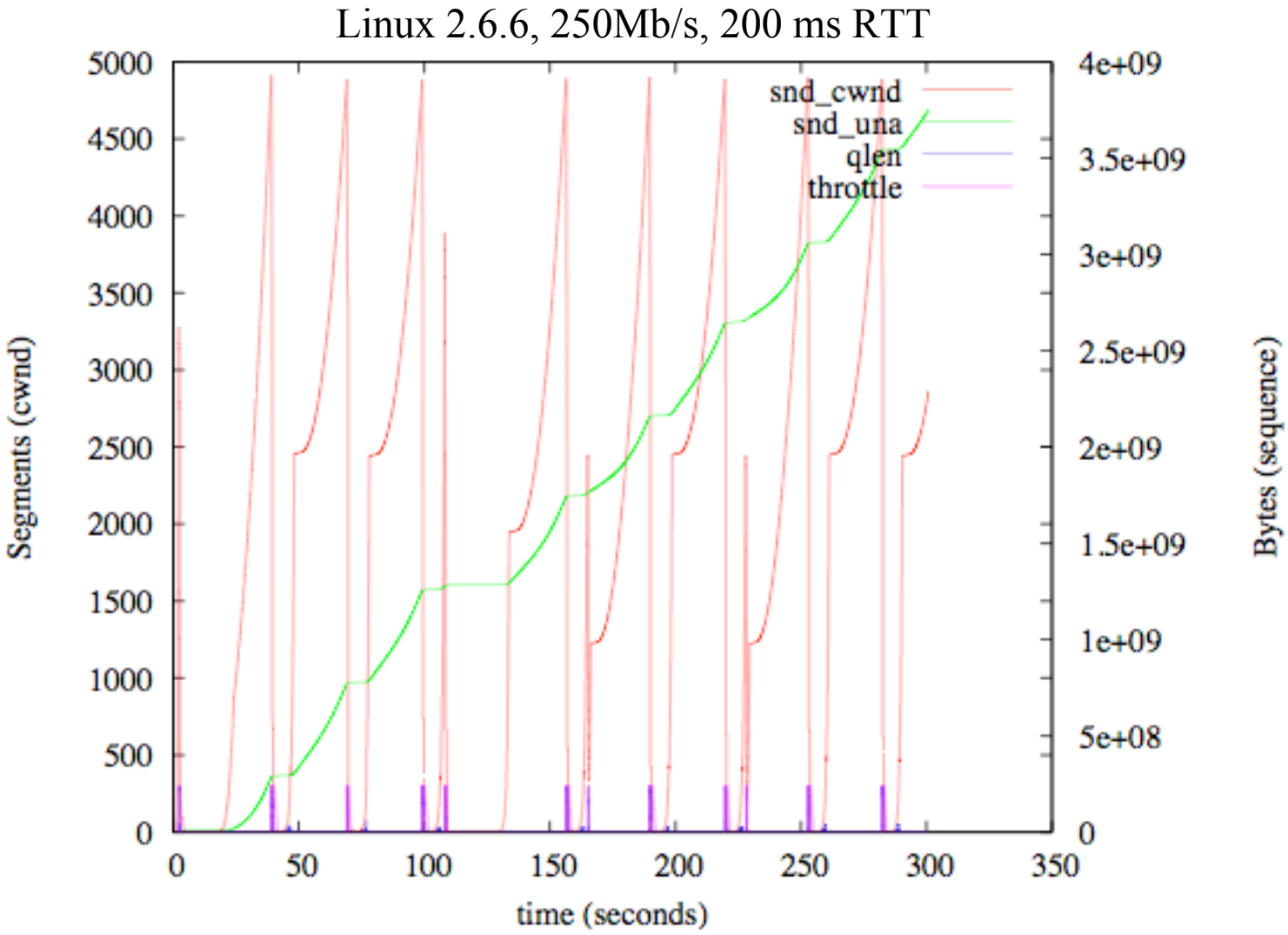
→ Note that because we cannot be exhaustive, insight into sources of observed behaviour are of vital importance -builds confidence in results and fosters a degree of generalisation beyond specific tests.

Three practical issues

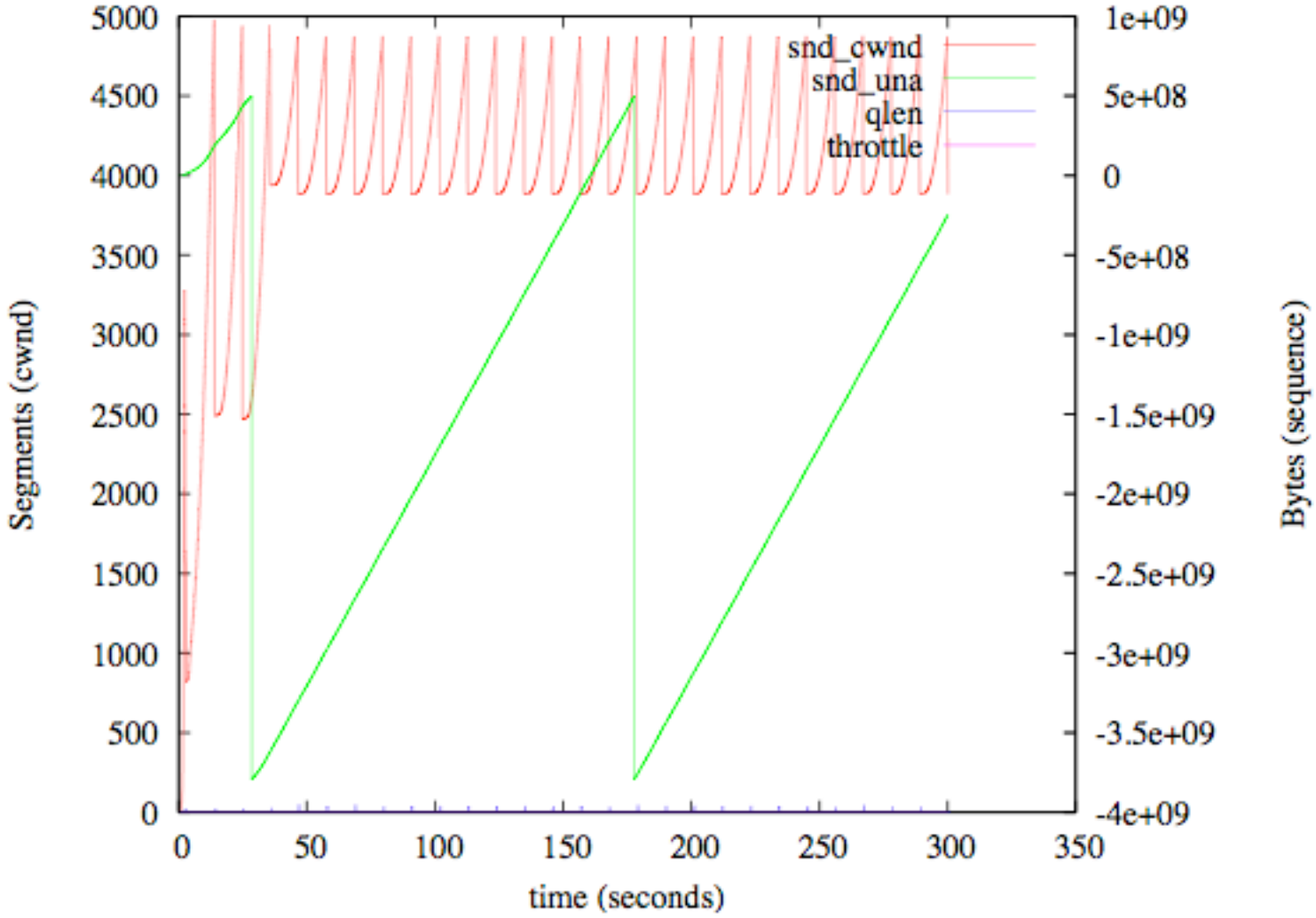
Minor in a sense, but can destroy value of tests if ignored.

- Need to control for different network stack implementations
- Buggy congestion control implementations
- Need to ensure that congestion control action is exercised

.


Linux 2.6.6, 250Mb/s, 200 ms RTT
snd_cwnd
snd_una
qlen
throttle
Segments (cwnd)
Bytes (sequence)
time (seconds)
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
0
50
100
150
200
250
300
350
0
5e+08
1e+09
1.5e+09
2e+09
2.5e+09
3e+09
3.5e+09
4e+09

Network stack implementation



Buggy congestion control implementations

- Linux BIC bug. Detailed at www.hamilton.ie/net. Fixed in 2.6.7 and later.

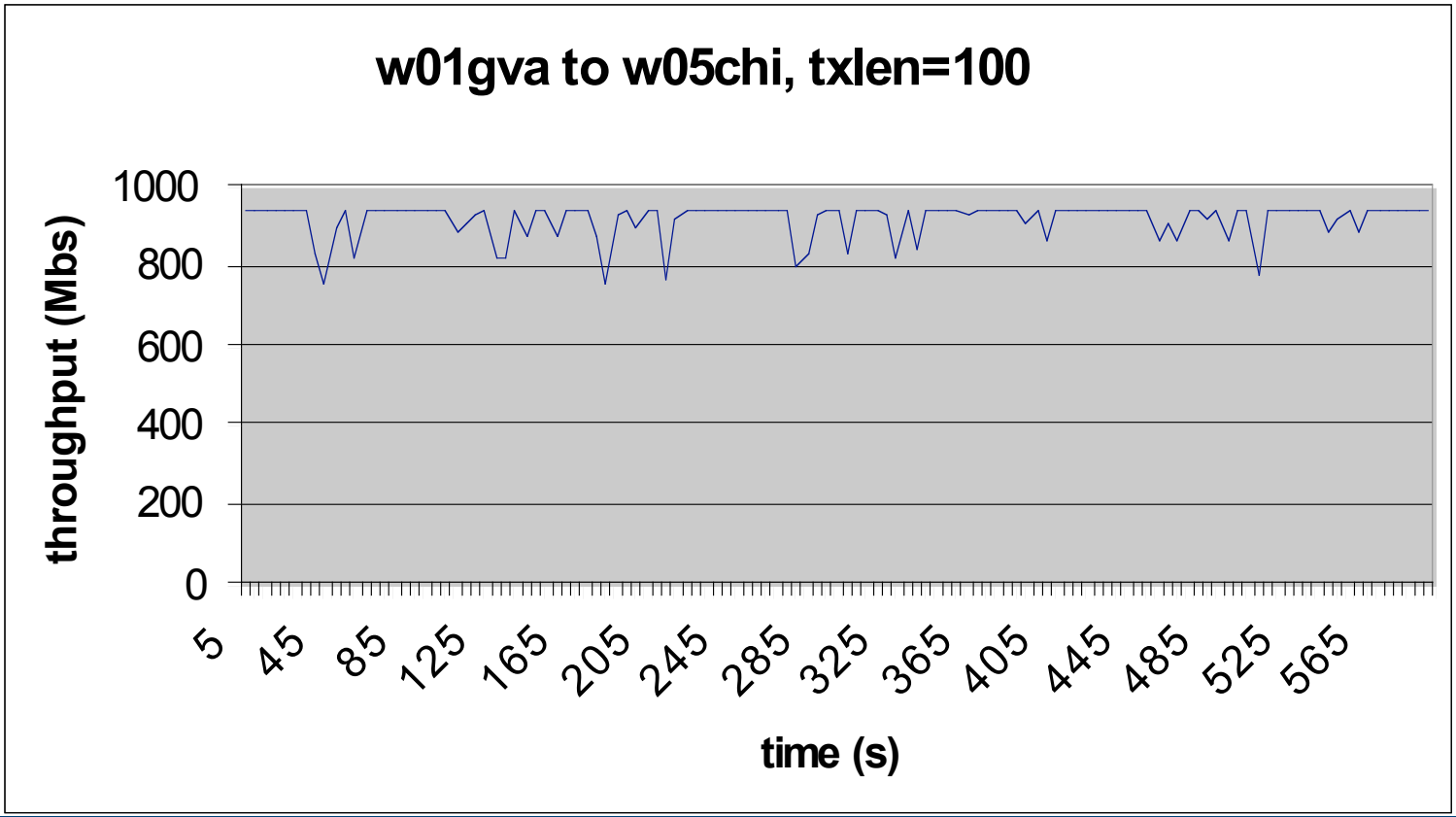
- Linux HTCP bug. Fixed in release 2.6.16 and later but present in all existing releases (2.6.13-2.6.15). Patch available at www.hamilton.ie/net.

- Linux Cubic scaling bug. Found by Hamilton team, fixed in 2.6.19 and later. Fix detailed at http://kernel.org/pub/linux/kernel/v2.6/ChangeLog-2.6.18.2.

Congestion control action not exercised

Initial tests – CERN-Chicago.

Bottleneck in NIC and with web100: throughput max's out regardless of congestion avoidance algorithm used.



Summary …

- Vital to validate experimental setup.
  - Requires detailed instrumentation of network stack
  - Some understanding of expected operation of algorithms, contact with code authors useful
  - Very useful to carry out tests for standard TCP as well as new algorithms - provides a baseline for comparisons and a sanity check on setup

Useful Performance Measures

Most of issues with existing TCP proposals have been associated with the behaviour of competing flows.

Suggest using behaviour of standard TCP as a baseline against which to compare performance of new proposals. Suggests consideration of the following characteristics:

- **Fairness** (between like flows)
- **Friendliness** (with legacy TCP)
- **Efficiency** (use of available network capacity).
- **Responsiveness** (how rapidly does the network respond to changes in network conditions, e.g. flows starting/stopping)

Again, rather than defining a single metric (problematic to say the least), suggest using measurements of standard TCP as baseline against which to make comparisons.

.

Useful Performance Measures (cont)

Important not to focus on a single network condition.

Know that current TCP behaviour depends on **bandwidth**, **RTT**, **queue size**, **number of users** etc. Therefore expect to have to measure performance of proposed changes over a range of conditions also.

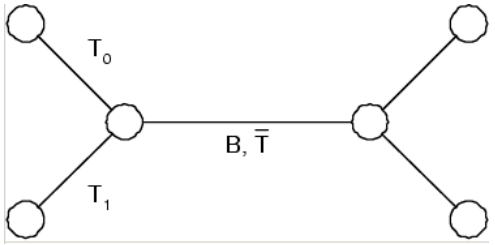
Suggest taking measurements for a grid of data points …

-we consider bandwidths of 1Mb/s - 500Mb/s
-two-way propagation delays of 16ms - 324ms
-range of queue sizes from 2% - 100% BDP.

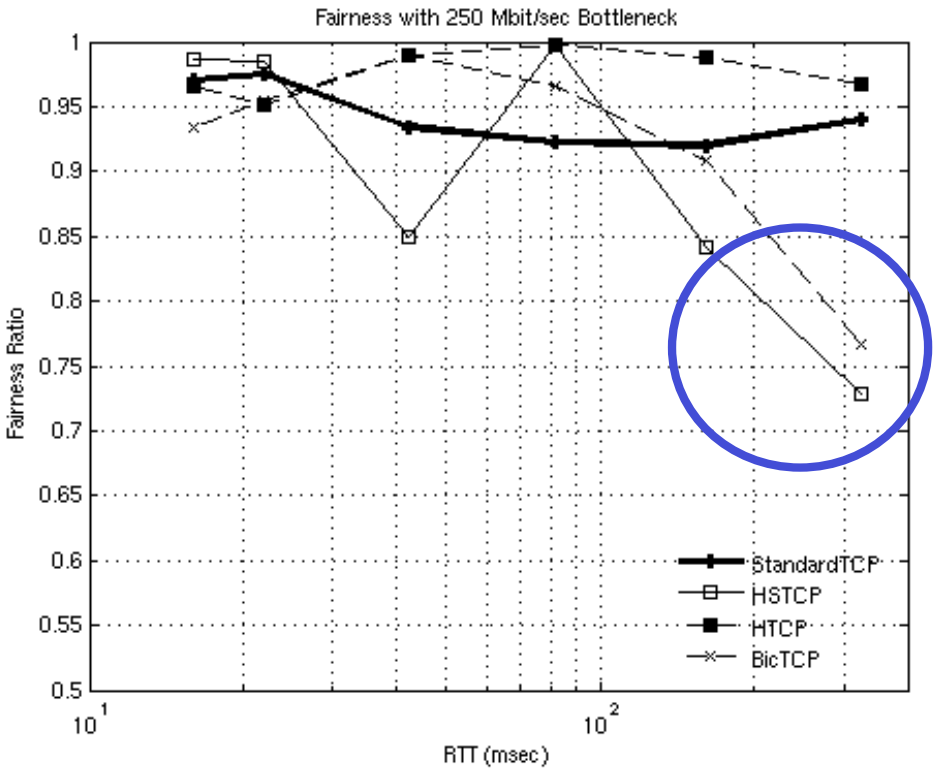
Some Measurement Results

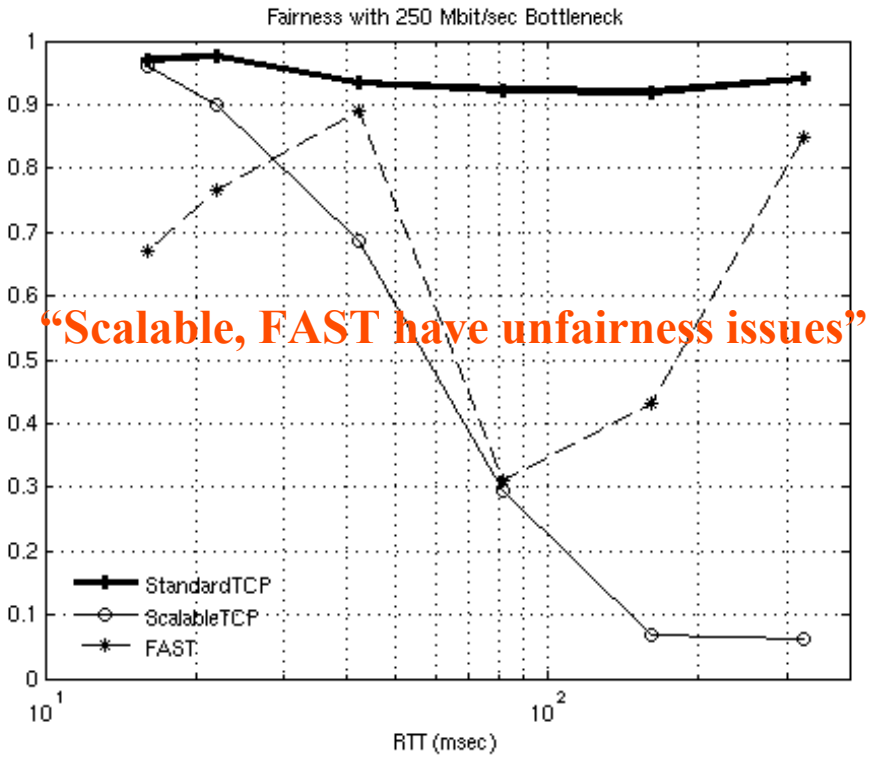
Symmetric conditions – flows use same congestion control algorithm, have same RTT, share common network bottleneck. Expect that:

- Fairness should be largely insensitive to bandwidth, number of users, queue size
- Competing flows with same RTT should have same long-term throughput.



Symmetric conditions (2 flows): Fairness



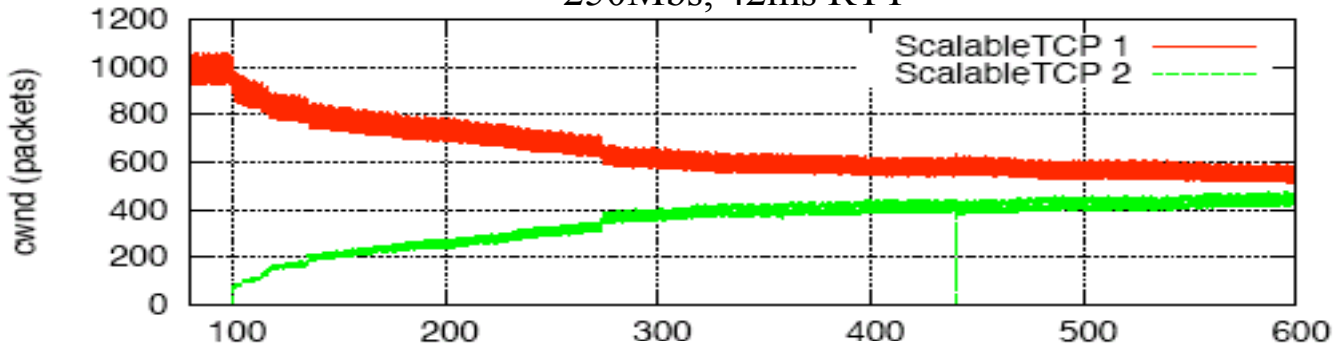


"Scalable, FAST have unfairness issues"

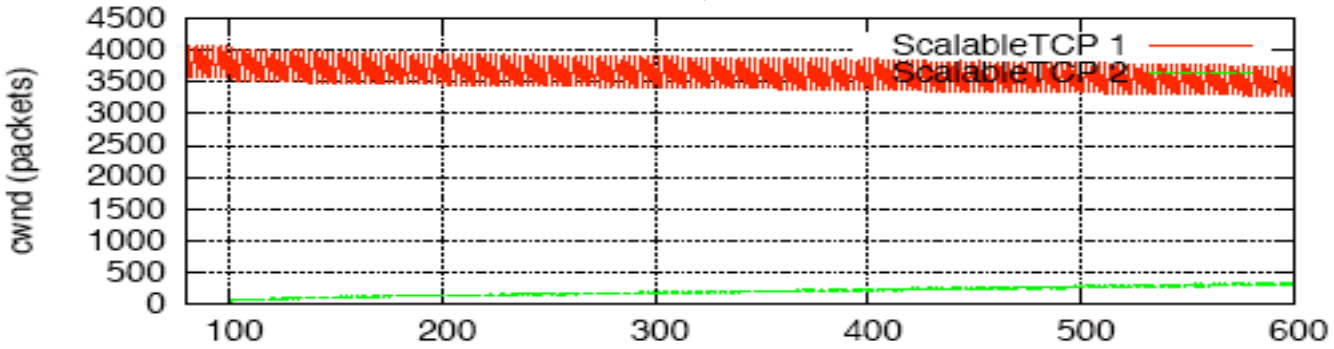
- Common network stack implementation used
- Averages over 5 tests runs
- Queue 20% DBP

Symmetric conditions (2 flows): Fairness

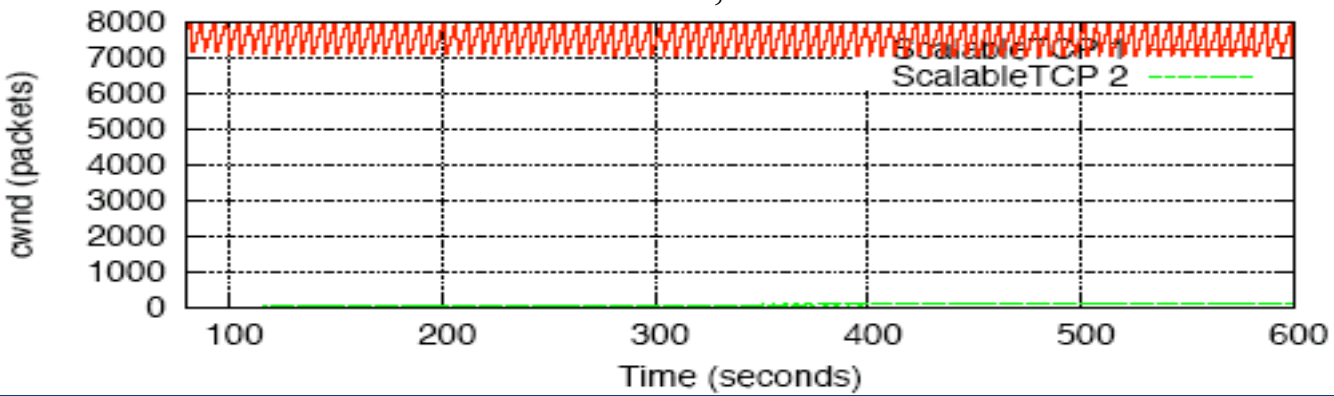
250Mbs, 42ms RTT



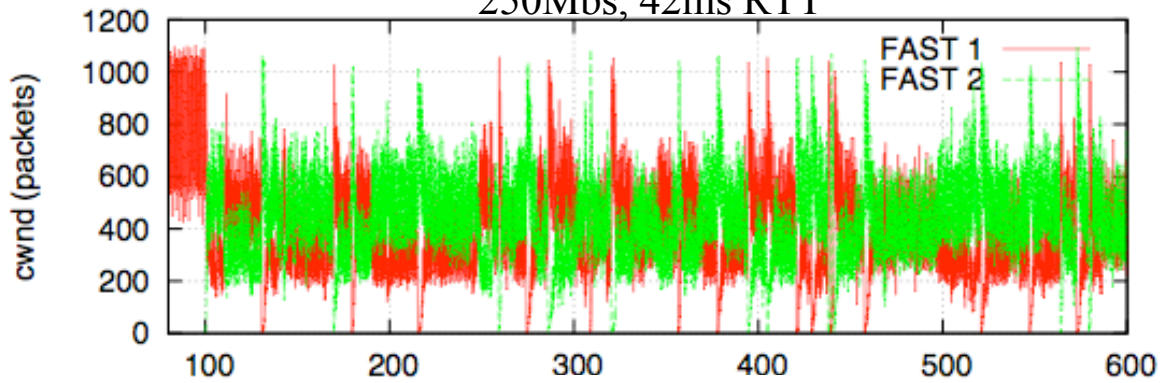
250Mbs, 162ms RTT



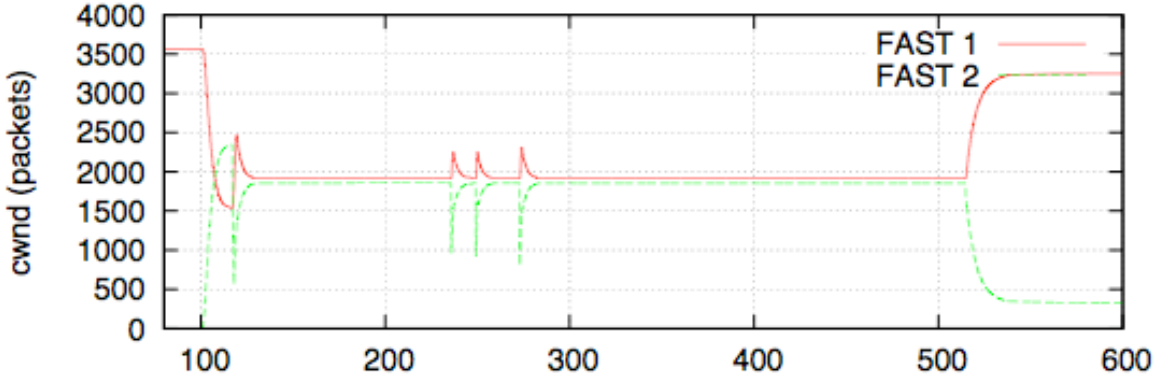
250Mbs, 324ms RTT



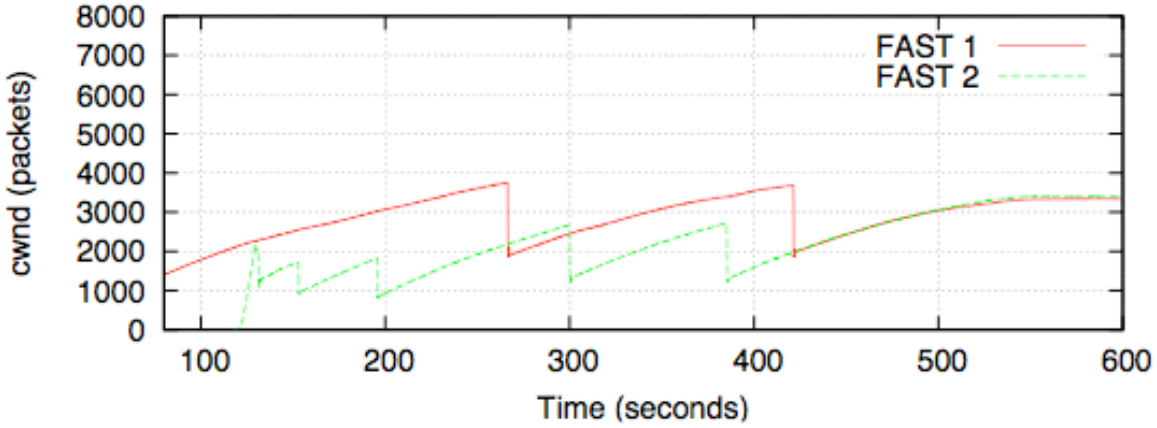
250Mbs, 42ms RTT



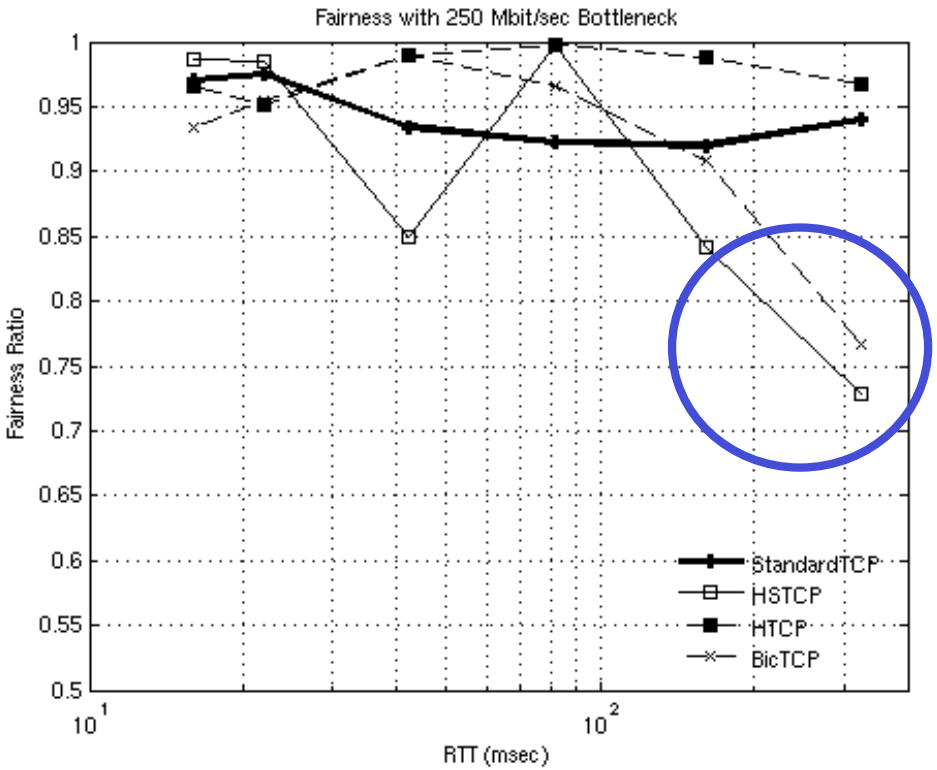
250Mbs, 162ms RTT

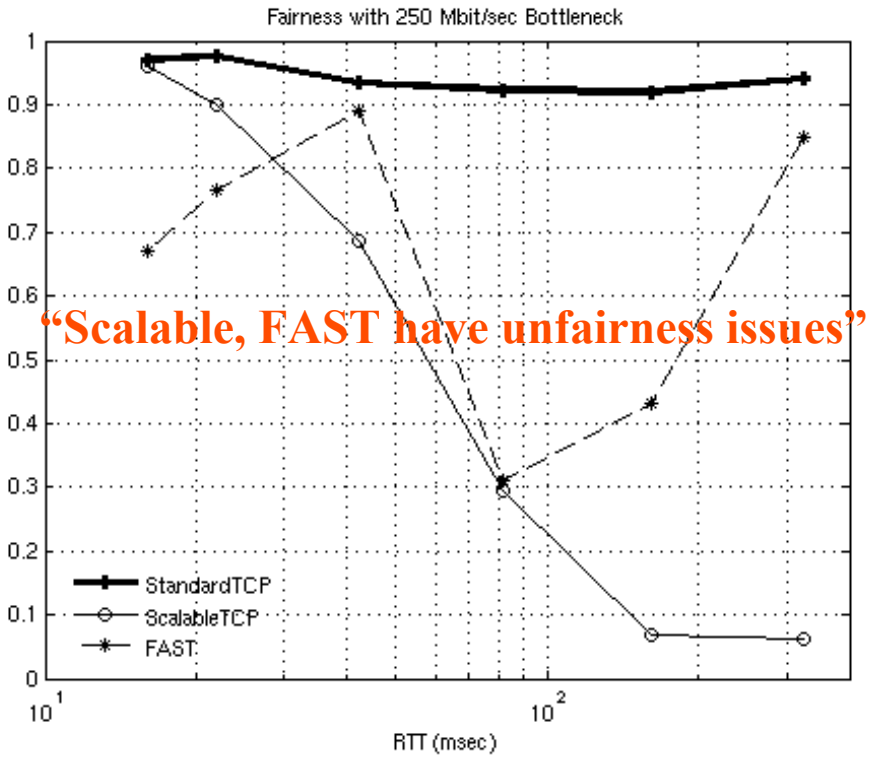


250Mbs, 324ms RTT



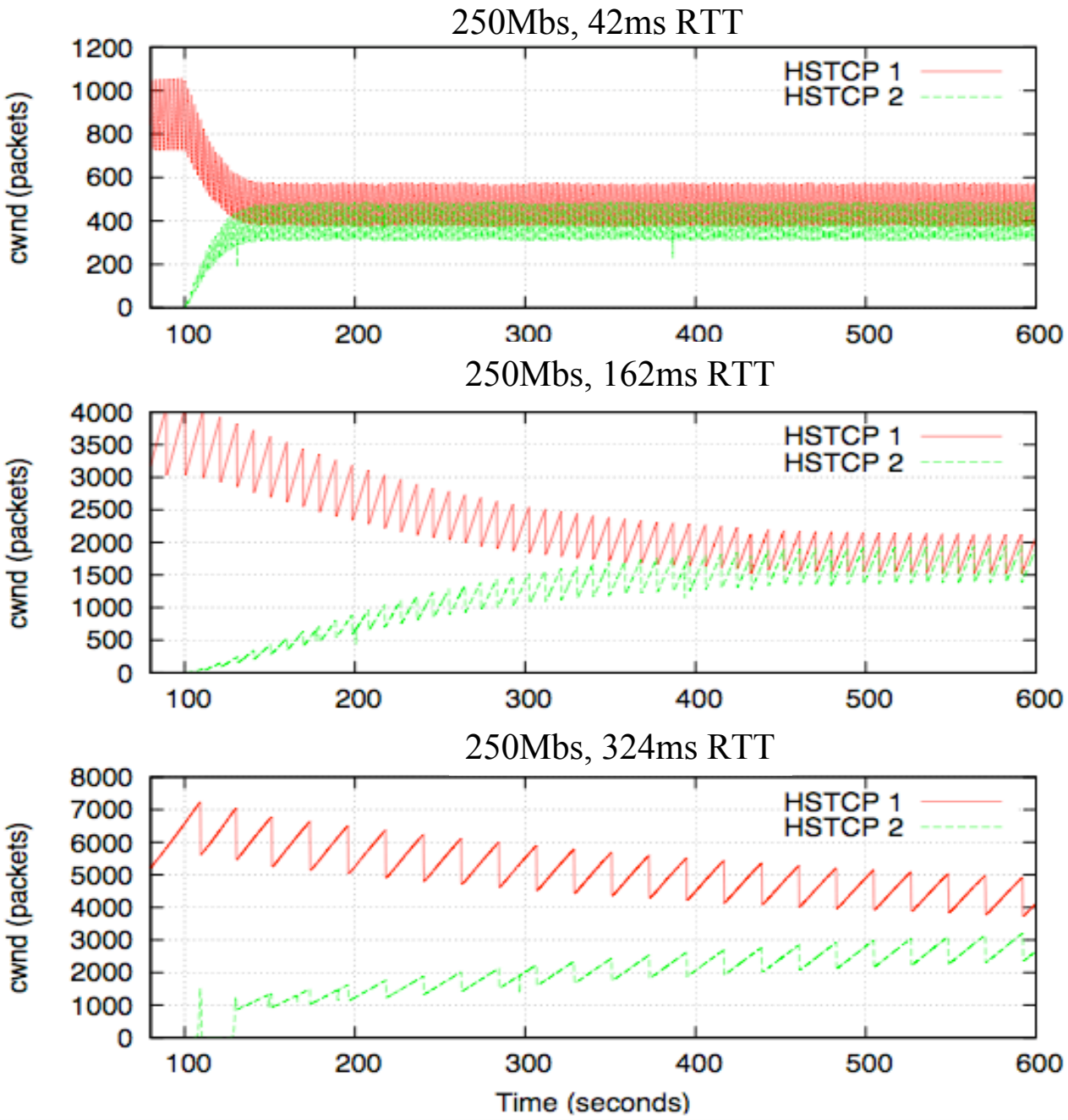
Symmetric conditions (2 flows): Fairness



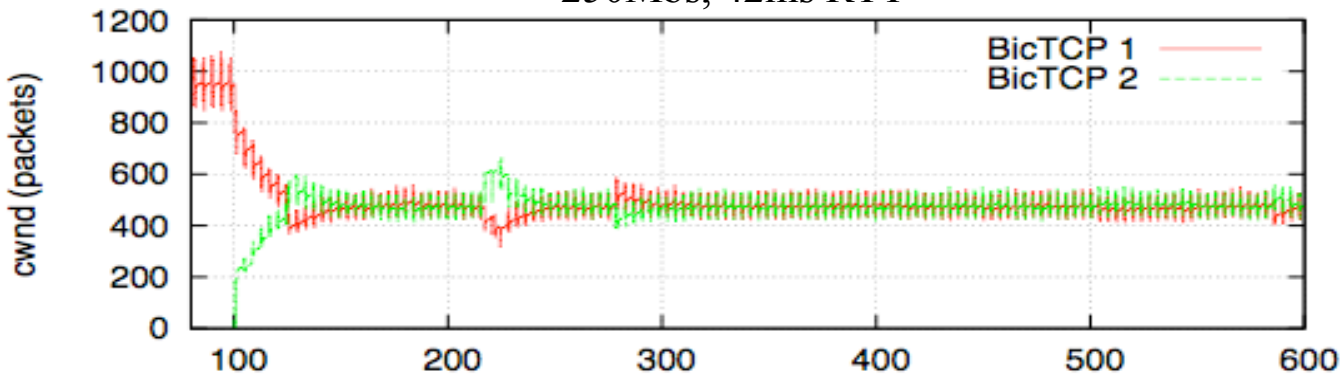


"Scalable, FAST have unfairness issues"

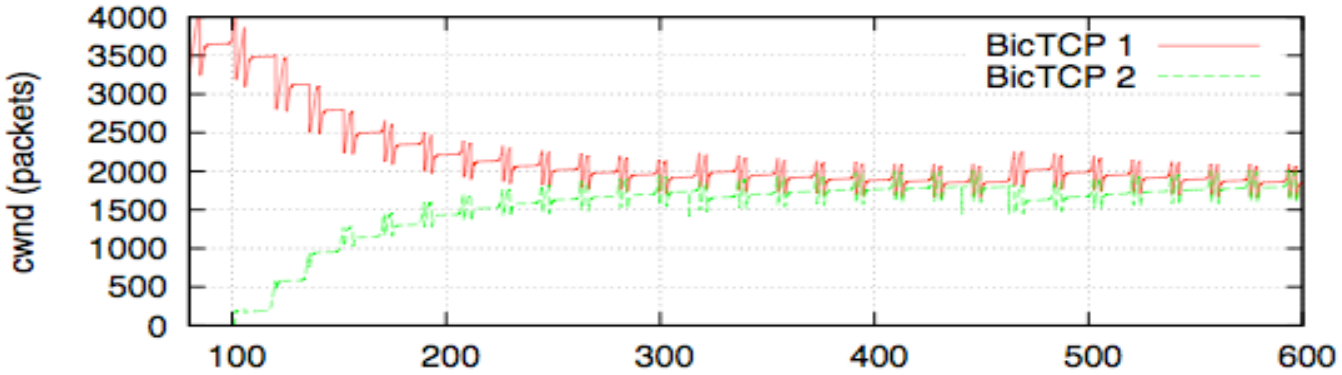
- Common network stack implementation used
- Averages over 5 tests runs
- Queue 20% DBP


250Mbs, 42ms RTT
cwnd (packets)
HSTCP 1
HSTCP 2
250Mbs, 162ms RTT
cwnd (packets)
HSTCP 1
HSTCP 2
250Mbs, 324ms RTT
cwnd (packets)
HSTCP 1
HSTCP 2
Time (seconds)

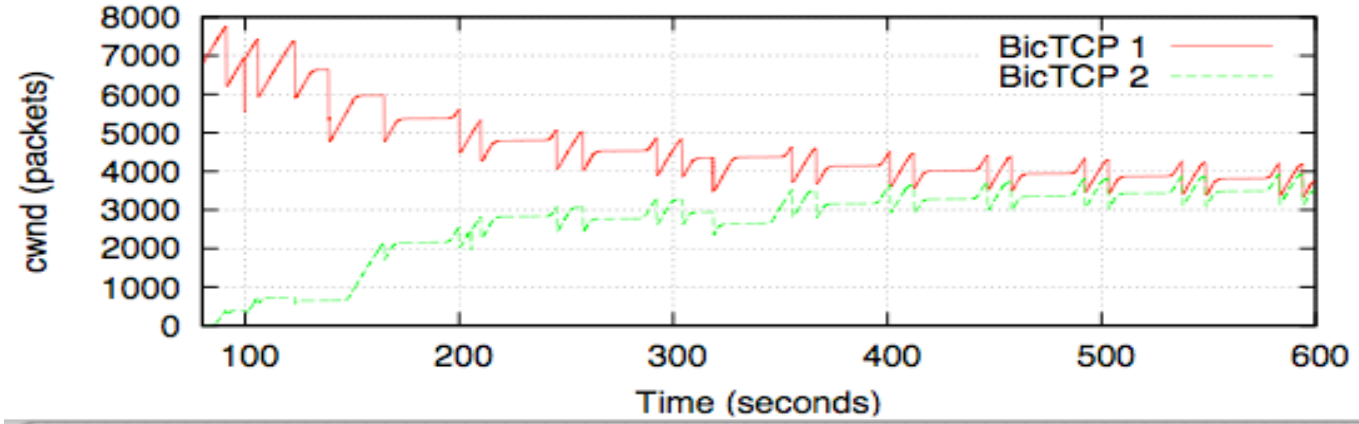
250Mbs, 42ms RTT



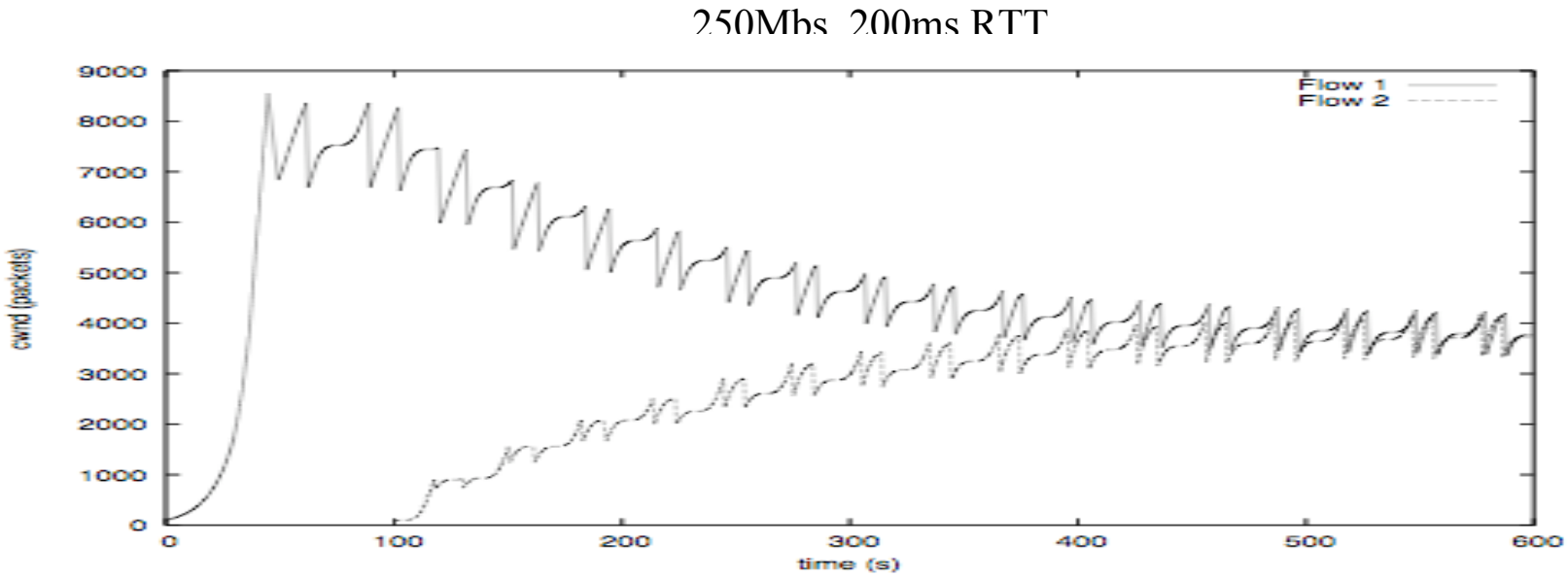
250Mbs, 162ms RTT



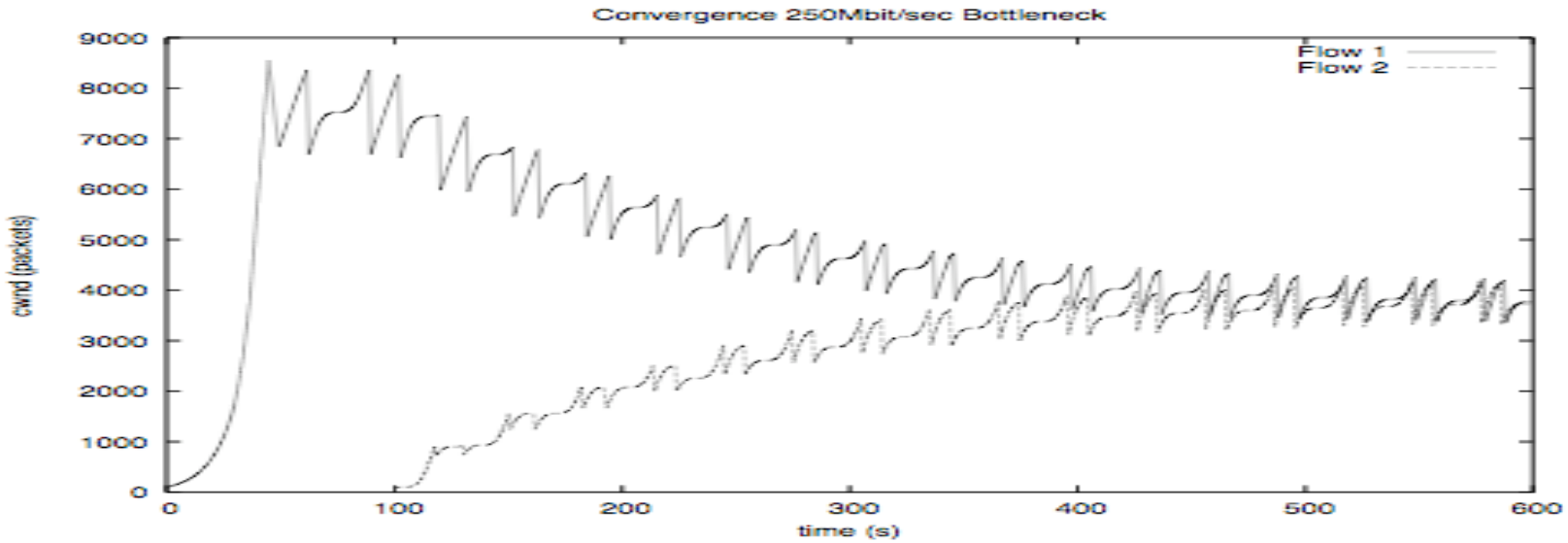
250Mbs, 324ms RTT



Cubic TCP


250Mbs 200ms RTT
Flow 1
Flow 2
cwnd (packets)
time (s)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
0
100
200
300
400
500
600

Source of slow convergence

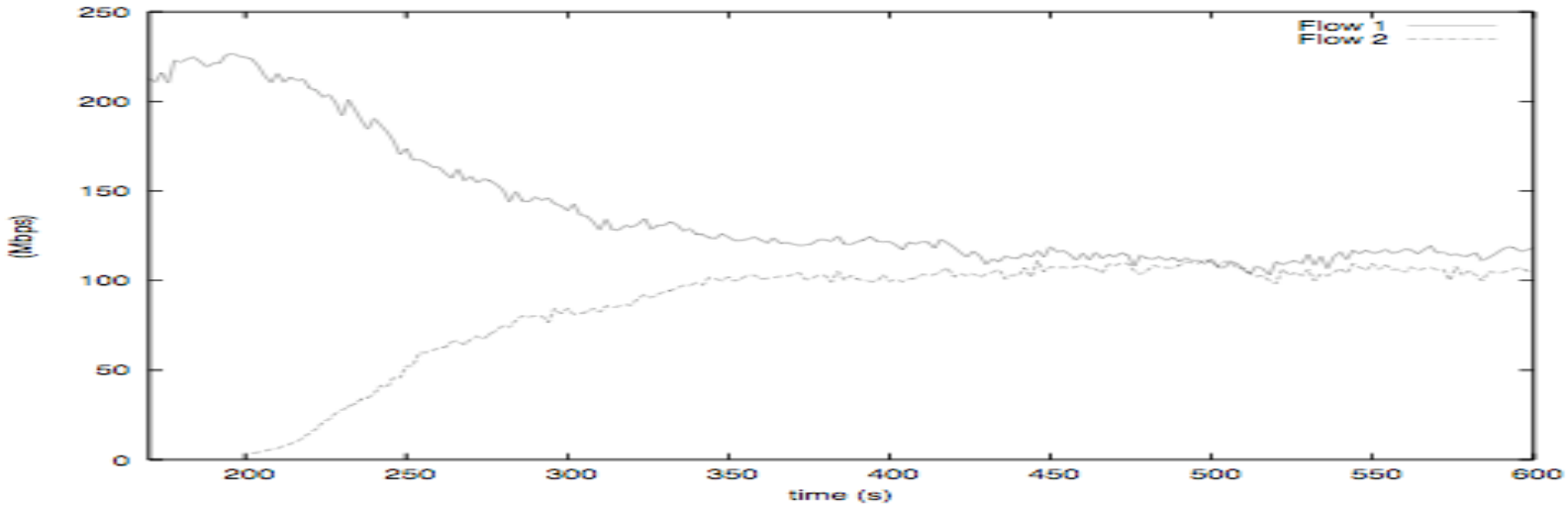


•Flows with low cwnds grab bandwidth less aggressively than flows with large cwnds. Similarly in High-Speed TCP

•Backoff factor of 0.8 (cf standard TCP backoff of 0.5) means that flows release bandwidth more slowly. Similarly in High-Speed TCP

Does slow convergence matter ?

Slow convergence still exhibited when unsynchronised drops

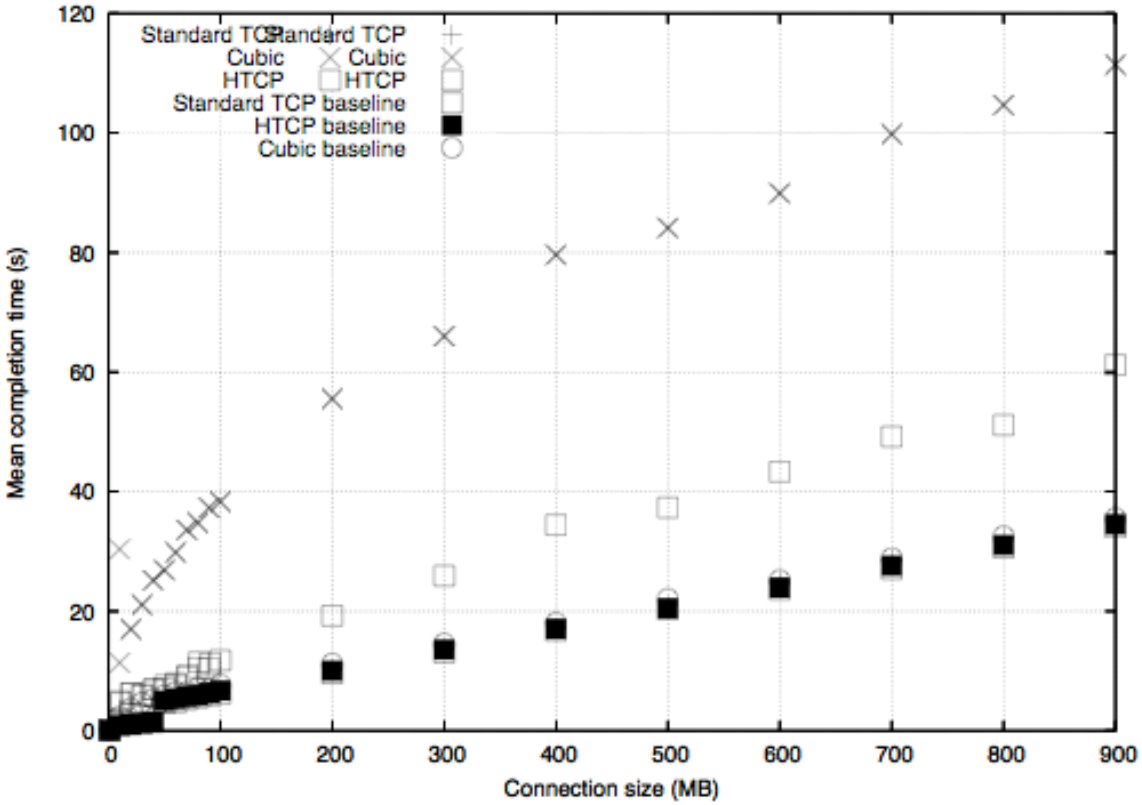


Ensemble throughput averaged over 20 test runs. Cubic TCP, 250Mb/s, RTT 200ms. Link shared with 200 bi-directional web flows.

Does slow convergence matter ?

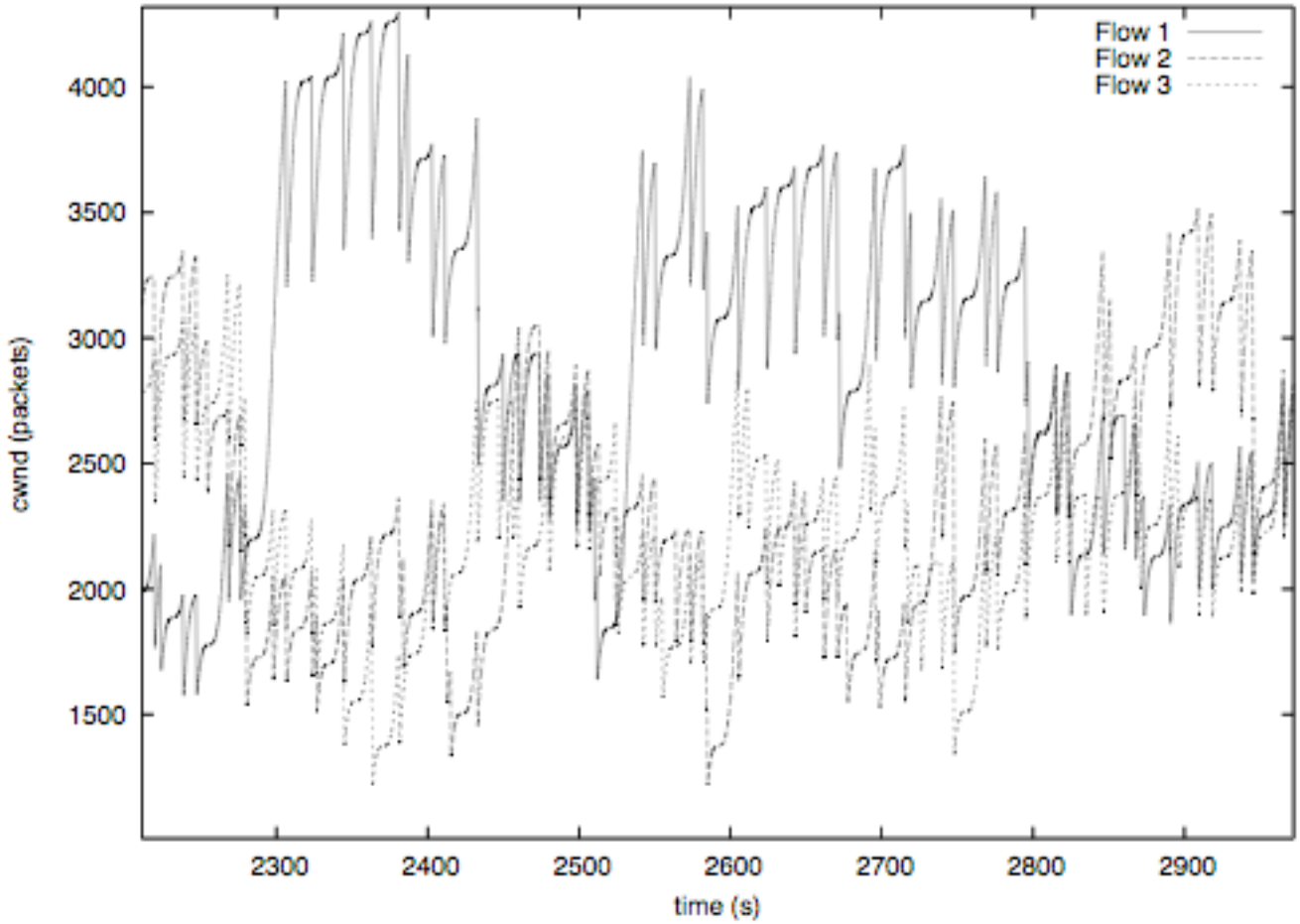
•Implies prolonged unfairness e.g. two identical file transfers may have very different completion times depending on the order in which they are started.

•Long-lived flows can gain a substantial throughput advantage at the expense of shorter-lived flows. Long-lived flow can penalize large number of users, e.g.



Does slow convergence matter ?

•In highly unsynchronised conditions, sustained periods (extending to hundreds of seconds) of unfairness occur.



250Mb/s link, RTT 200ms, 3 long-lived Cubic flows. Link shared with 25 on-off sessions, Pareto connection size mean 100 packets, exponential off periods mean 10s.

Summary

•Careful experiment design is vital e.g. controlling for network stack implementation

•Even simple tests can be surprisingly revealing.

•Propose use of standard TCP as a baseline for evaluating performance

•Argue that it is vital to measure performance over a wide range of bandwidths, RTT's, queue sizes etc and study >1 competing flow.

•Data (full time histories) is all public and available online at www.hamilton.ie/net/

Current Status ?

- Scalable. No longer being actively pursued.
- BIC-TCP. Now replaced by Cubic as Linux default.
- FAST-TCP. Now proprietary.
- HS-TCP. IETF experimental RFC.
- Cubic-TCP. Recently became Linux default.
- H-TCP. IETF Internet draft.
- Compound TCP.

Also LT-TCP, Westwood+ and others.

Delay-based algorithms

Commonly expressed concerns re use of delay as a congestion signal:

•Delay measured by a flow may be weakly correlated with packet loss (sampling issues) i.e. is delay a reliable signal for congestion control ?
•Delay-based algorithms react to reverse path queueing as congestion. One way delay hard (impossible ?) to measure. Loss-based algorithms do not (loss of ACKs vs loss of data packets)
•Variations in baseRTT e.g. wireless links
•Support for incremental rollout unclear.

Also additional issues specific to Vegas and related algorithms:

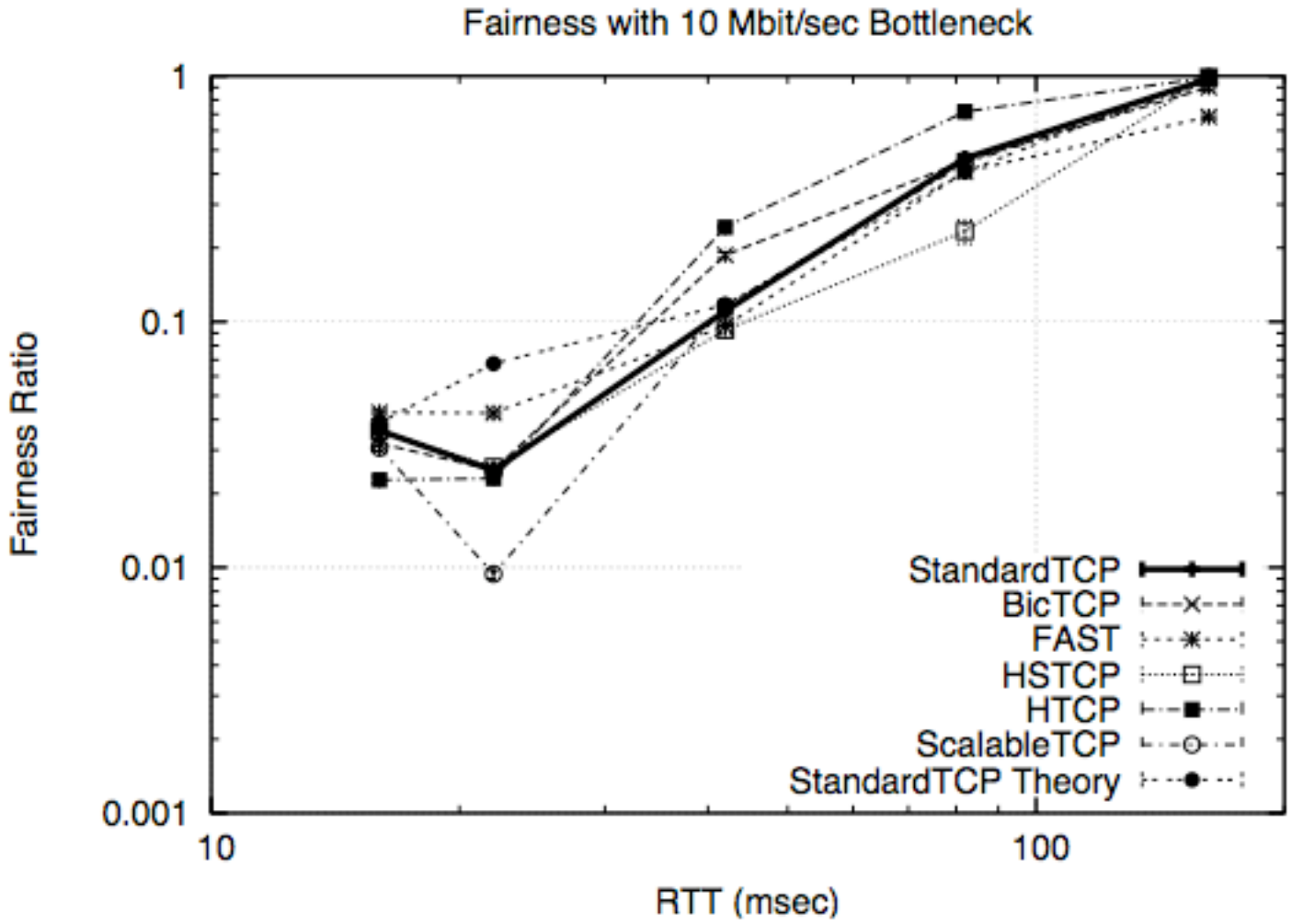
•Delay not regulated to be small -- buffer occupancy scales with number of flows on a link. Overflow inevitable eventually.
•Even under best case conditions, action of algorithm itself makes it difficult to measure delay accurately (esp. baseRTT).

Extra Slides

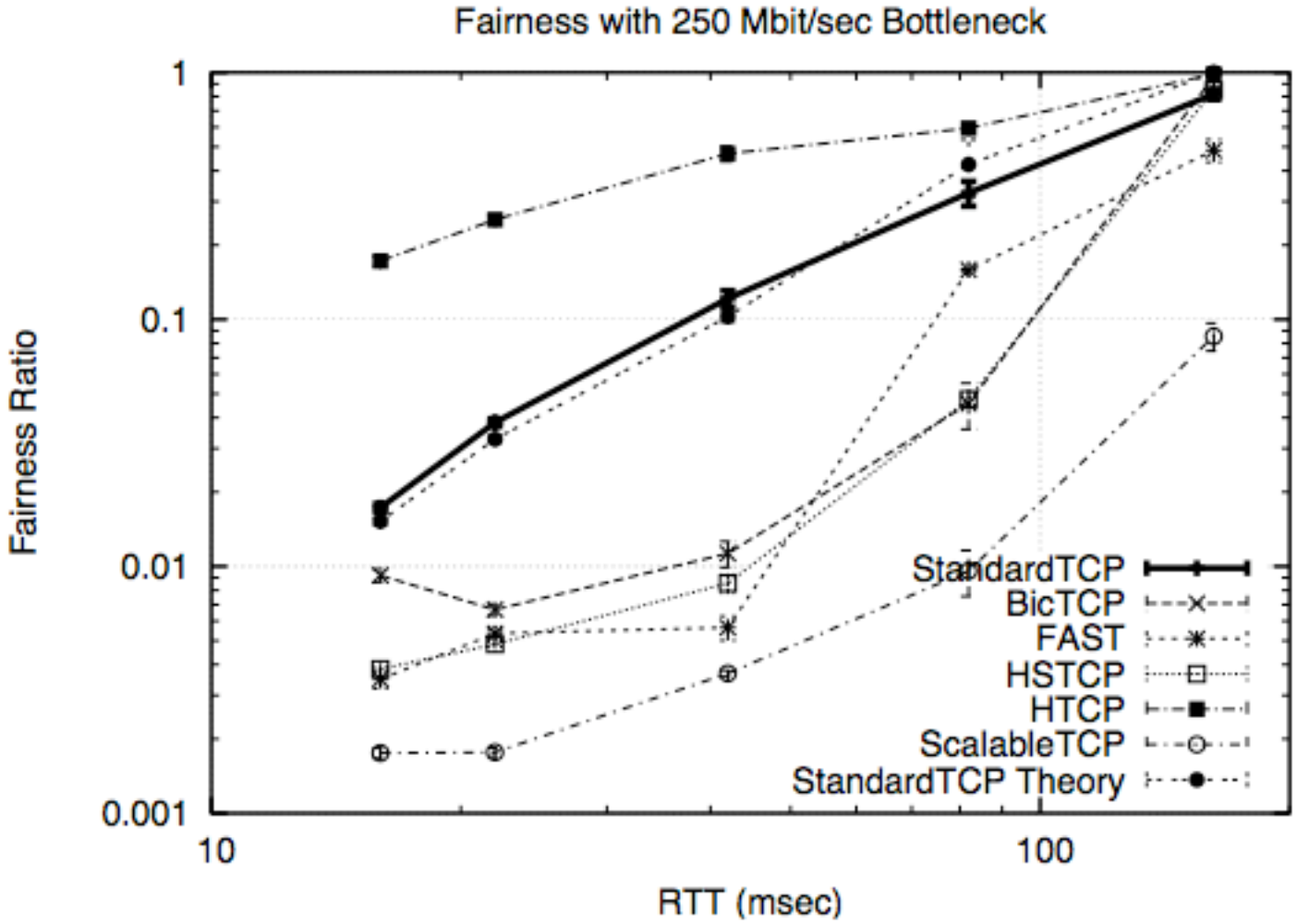
RTT Unfairness

- Competing flows with different RTT's may be unfair;
- Unfairness no worse than throughputs being roughly proportional to $1/RTT^2$ (*cwnd* proportional to 1/RTT).

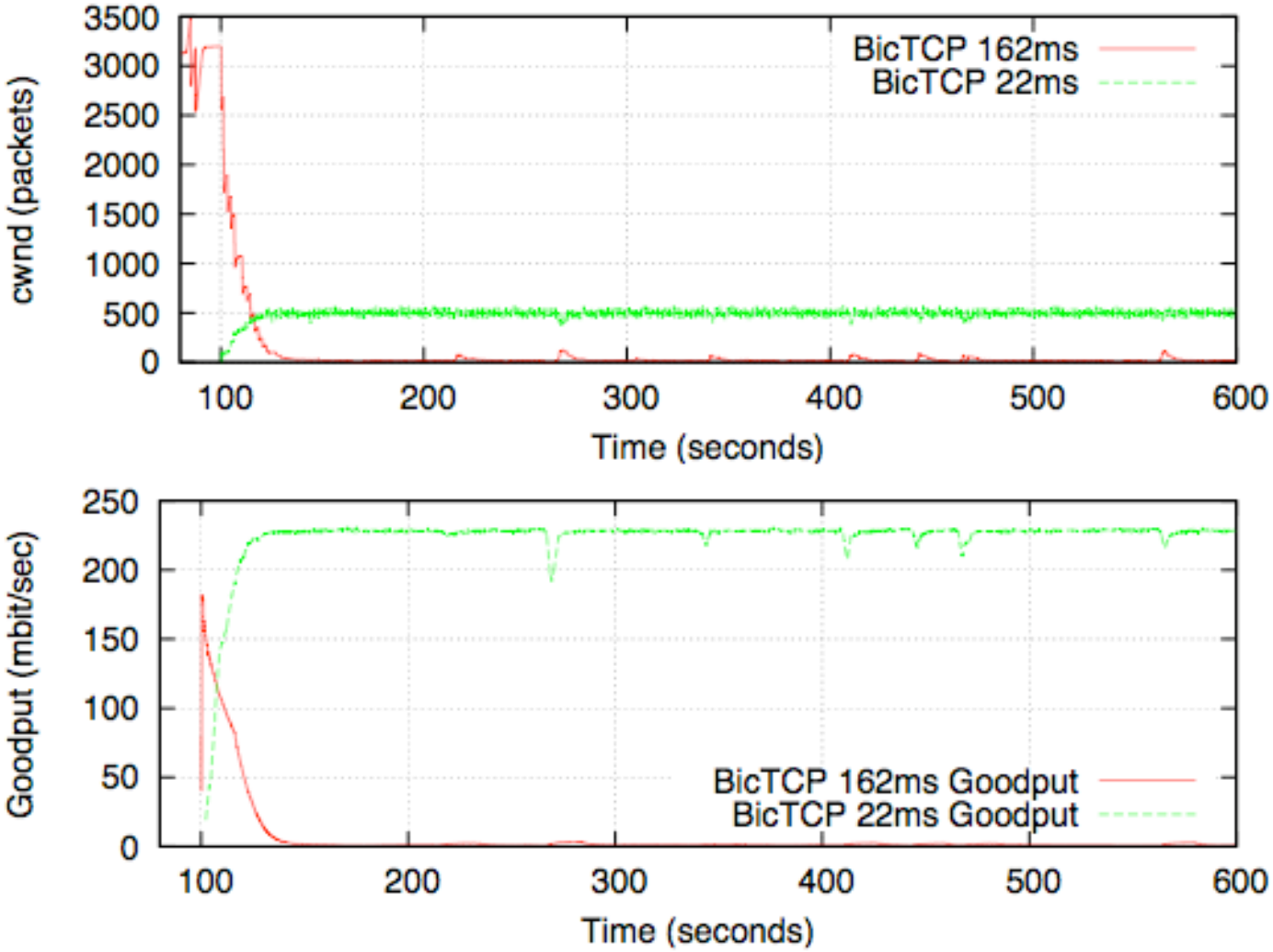
RTT Unfairness



RTT Unfairness



RTT Unfairness



Friendliness (NewTCP flow competing with legacy flow, symmetric conditions)