Evaluating the Performance of TCP Stacks for High-Speed Networks

Doug Leith
Hamilton Institute
Ireland

Thanks: Yee Ting Lee, Baruch Even, Robert Shorten.

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.

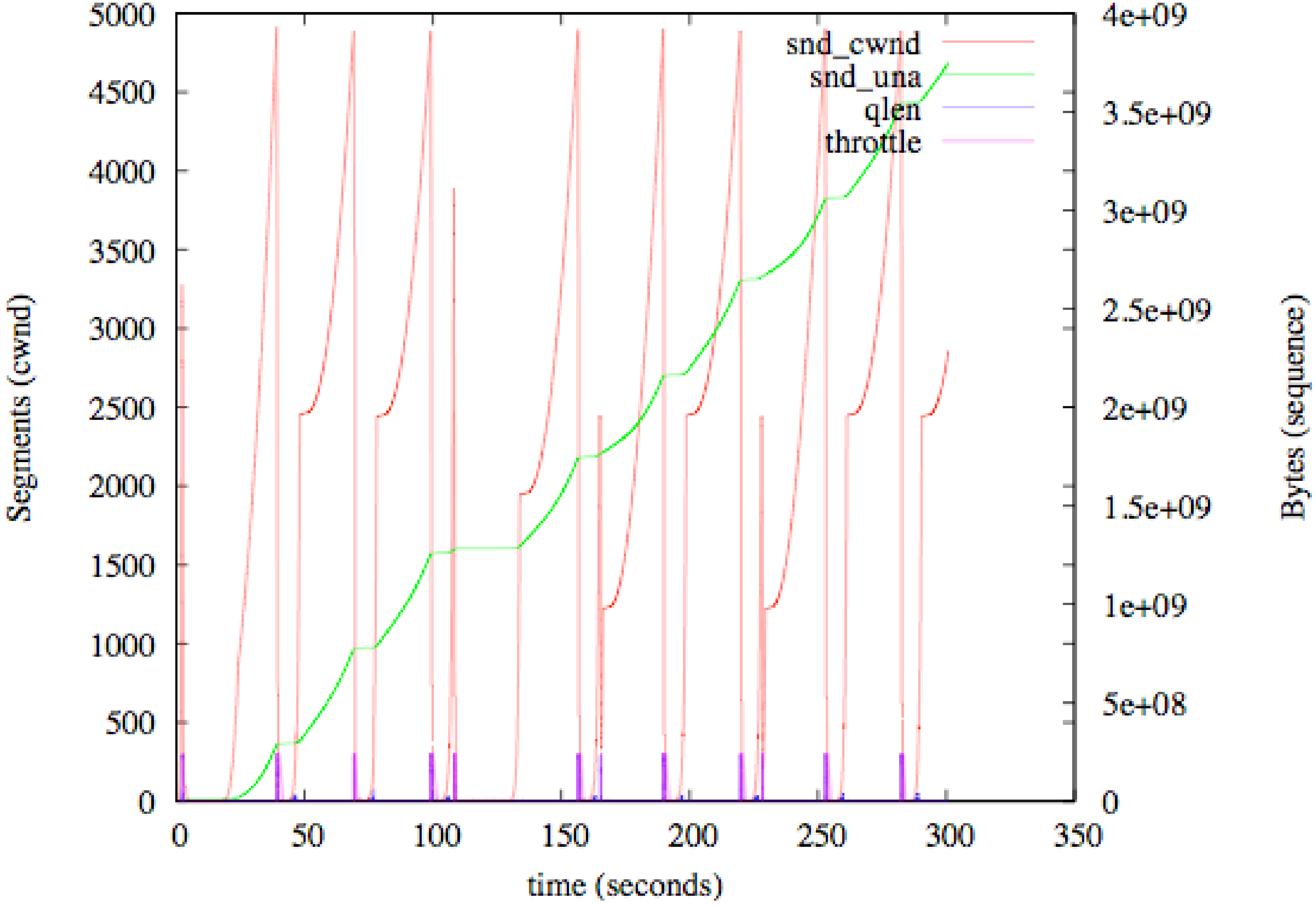
Three practical issues

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

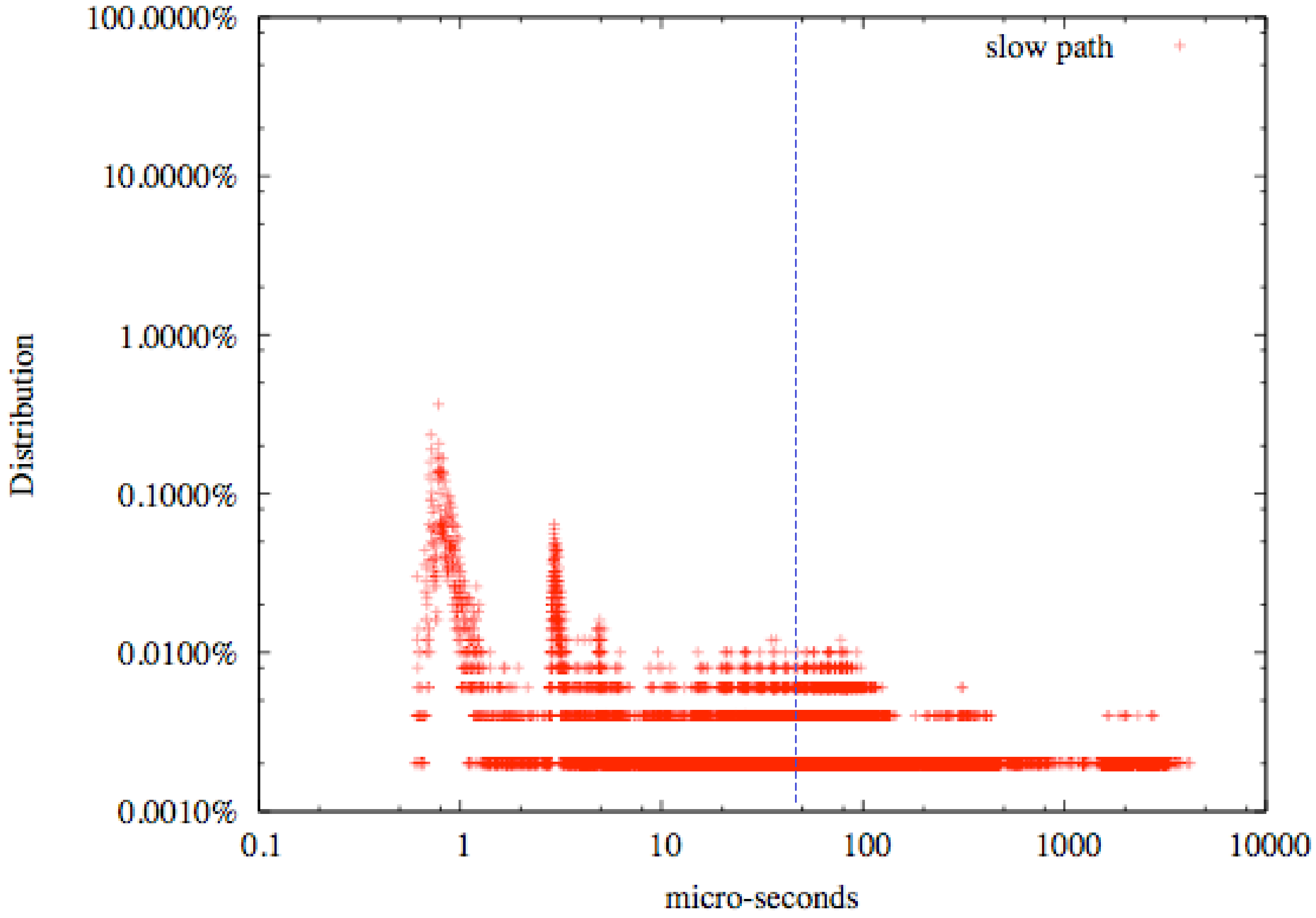
.

Network stack implementation

Linux 2.6.6, 250Mb/s, 200 ms RTT



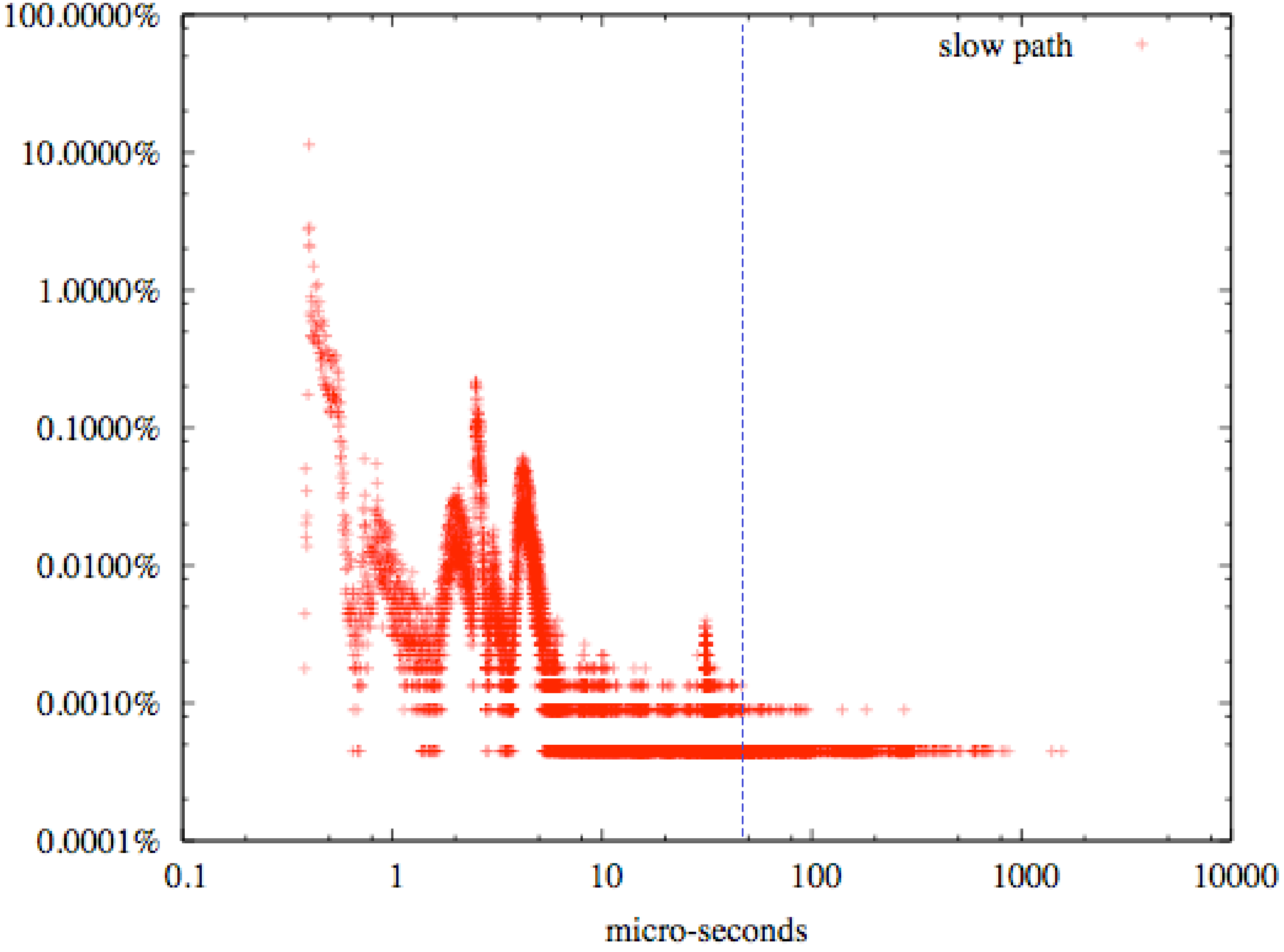
Network stack implementation



Network stack implementation

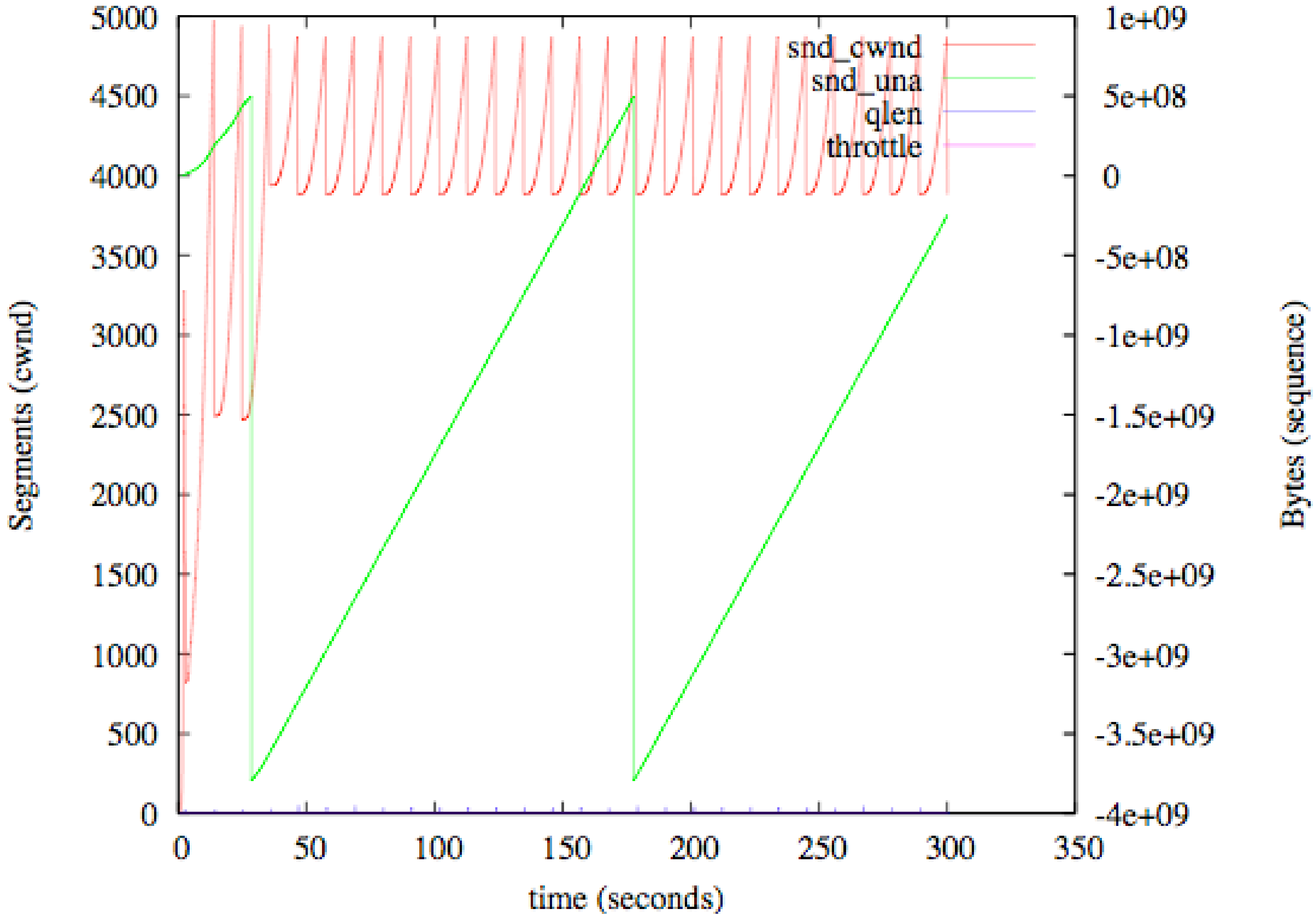
.

Linux 2.6.15, 250Mb/s, 200 ms RTT



Network stack implementation

.



Network stack implementation

- We have validated performance of recent changes up to 1Gb/s-200ms for a single flow on Xeon hardware.
- Higher-speed operation still to be checked.
- Multiple flows/machine operation still to be checked

Buggy congestion control implementations

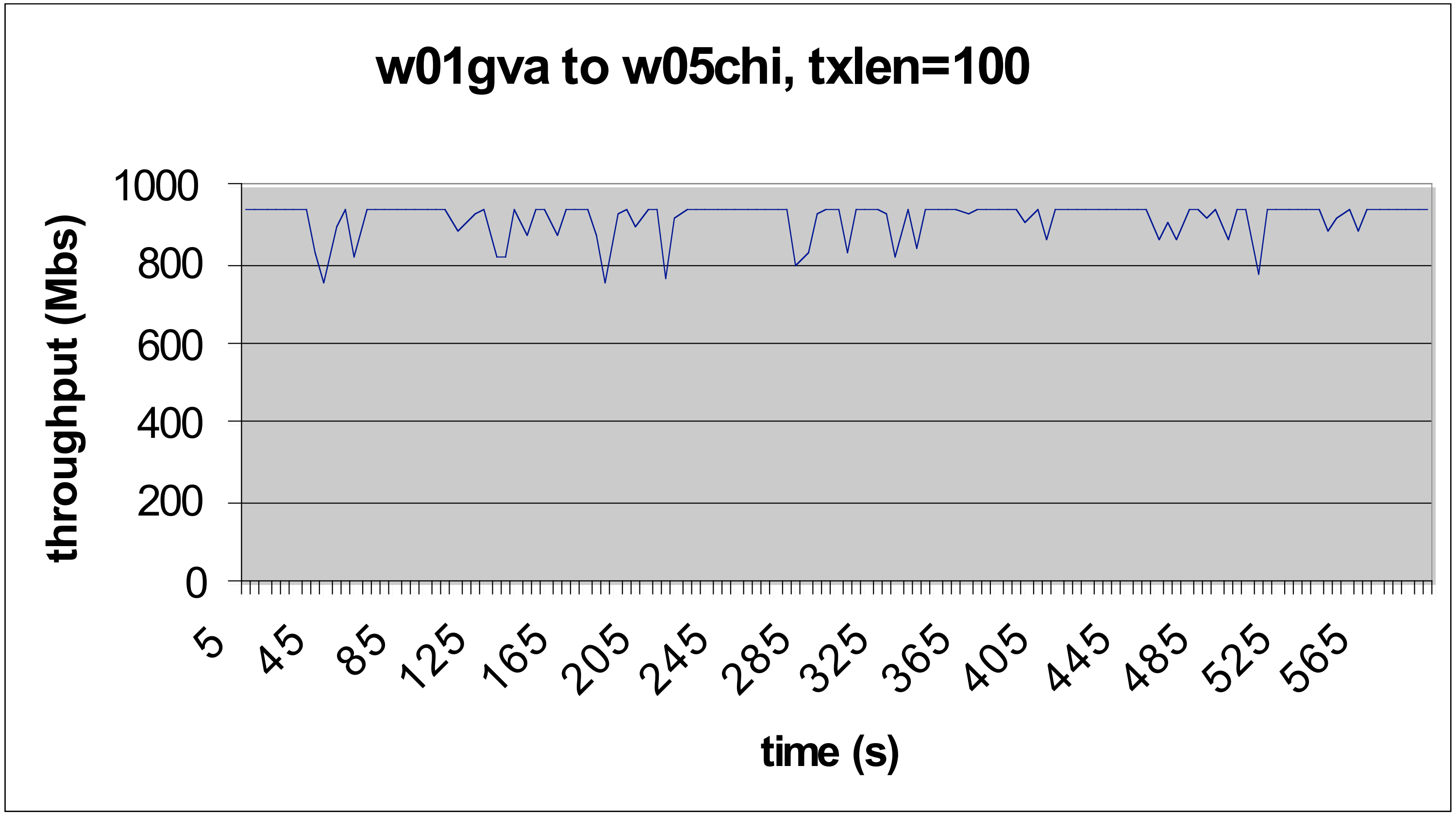
•Linux BIC bug. Detailed at www.hamilton.ie/net. Fixed in 2.6.7 and later.

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

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.



Putative 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. Focus on long-lived flows initially - this 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)

.

Putative Performance Measures (cont)

Important not to focus on a single network condition.

-We know that current TCP behaviour depends on **bandwidth**, **RTT**, **queue size**, **number of users** etc. We 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 10Mb/s, 100 Mb/s and 250Mb/s
-two-way propagation delays of 16ms - 324ms
-range of queue sizes from 5% - 100% BDP.

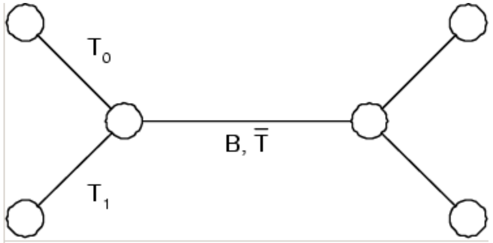
Putative Performance Measures (cont)

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

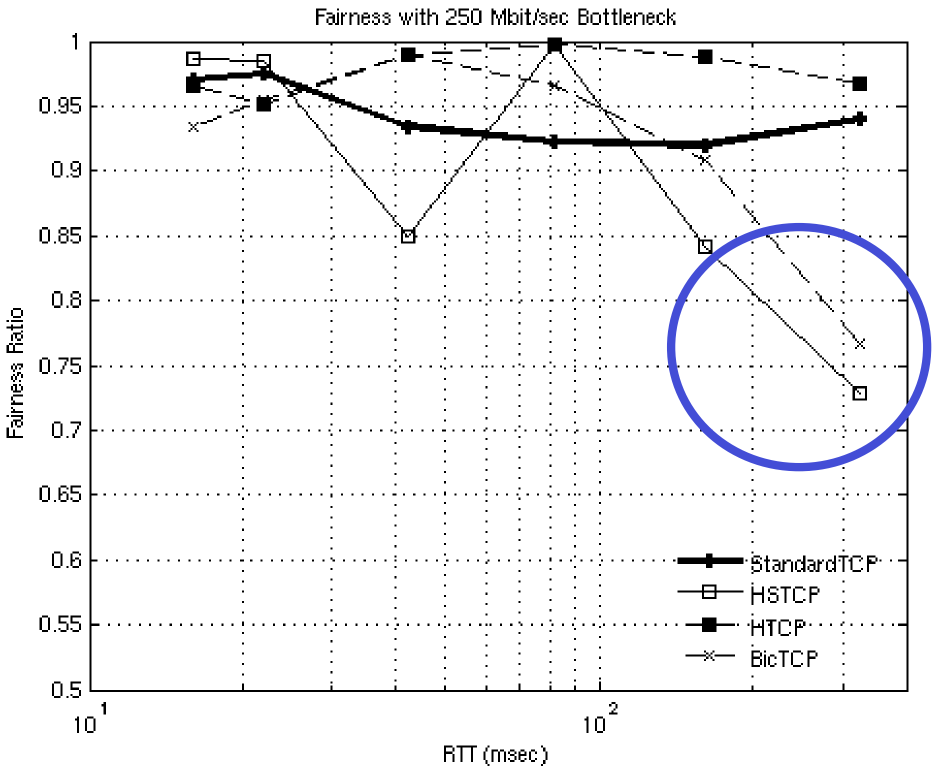
⇒

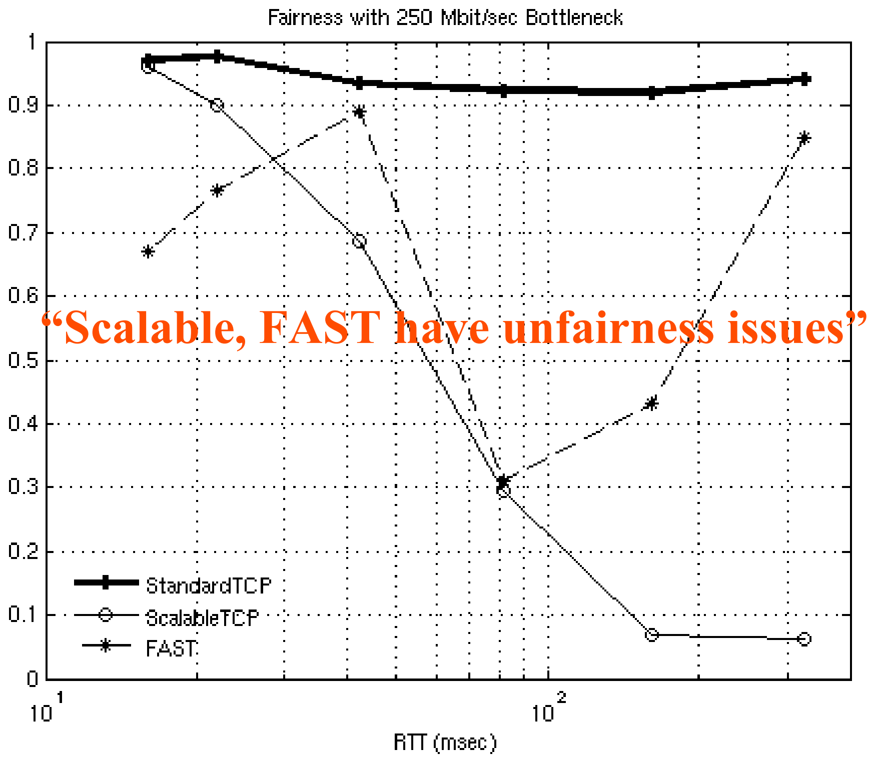
1. Symmetric conditions – flows use same congestion control algorithm, have same RTT, share common network bottleneck.

   - 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

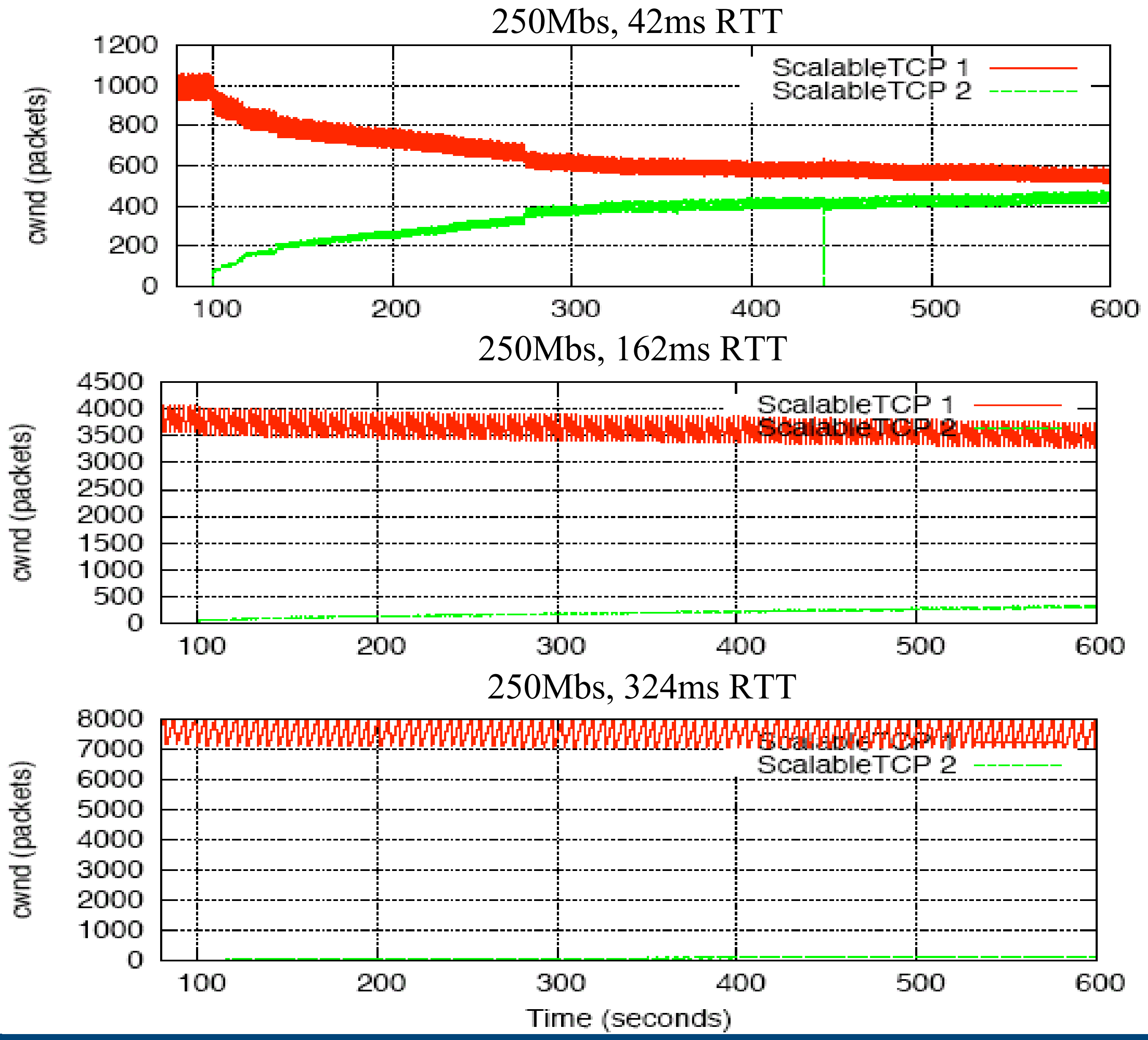




“Scalable, FAST have unfairness issues”

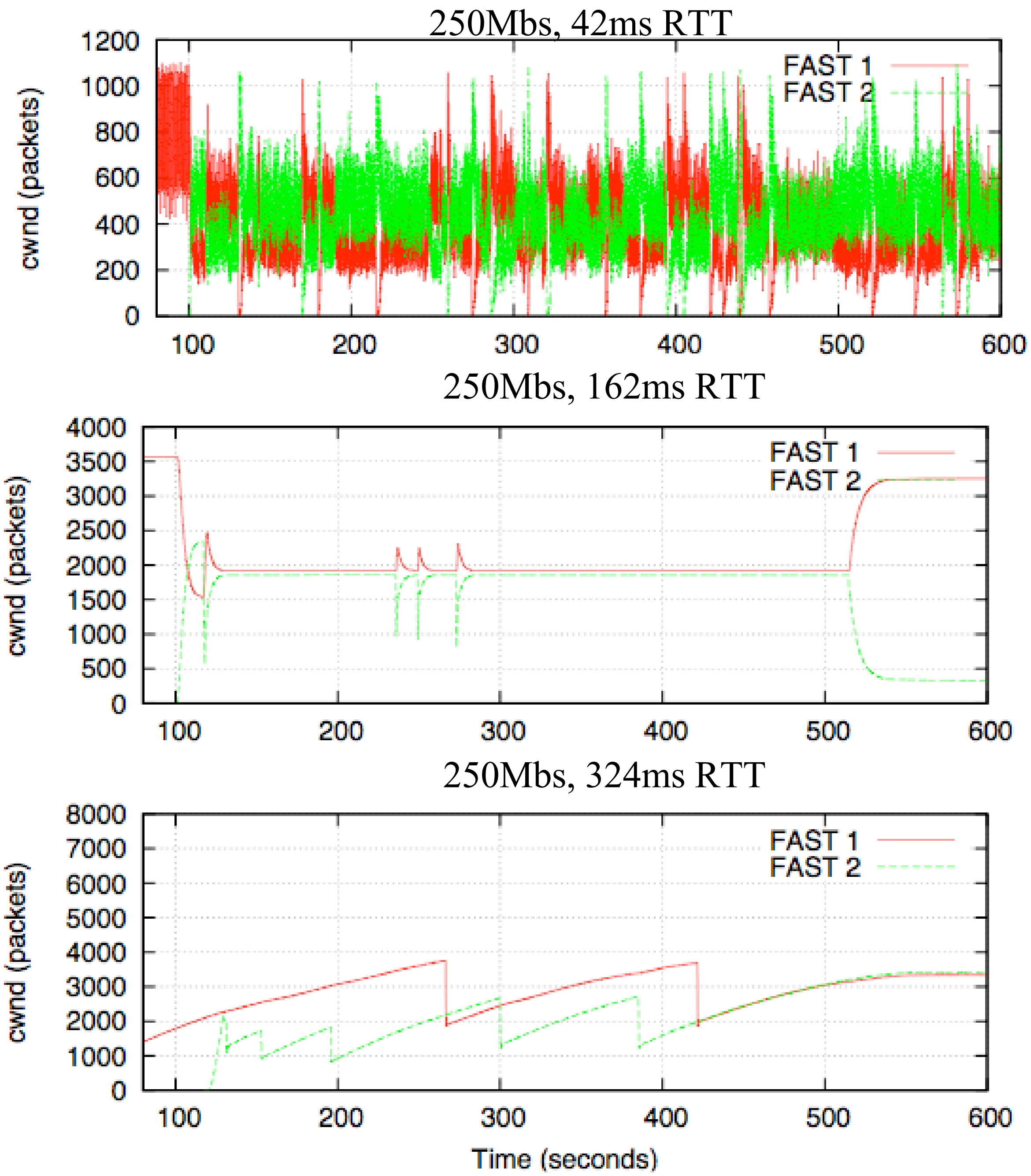
- Common network stack implementation used
- Averages over 5 tests runs
- Queue 20% DBP

Symmetric conditions (2 flows): Fairness

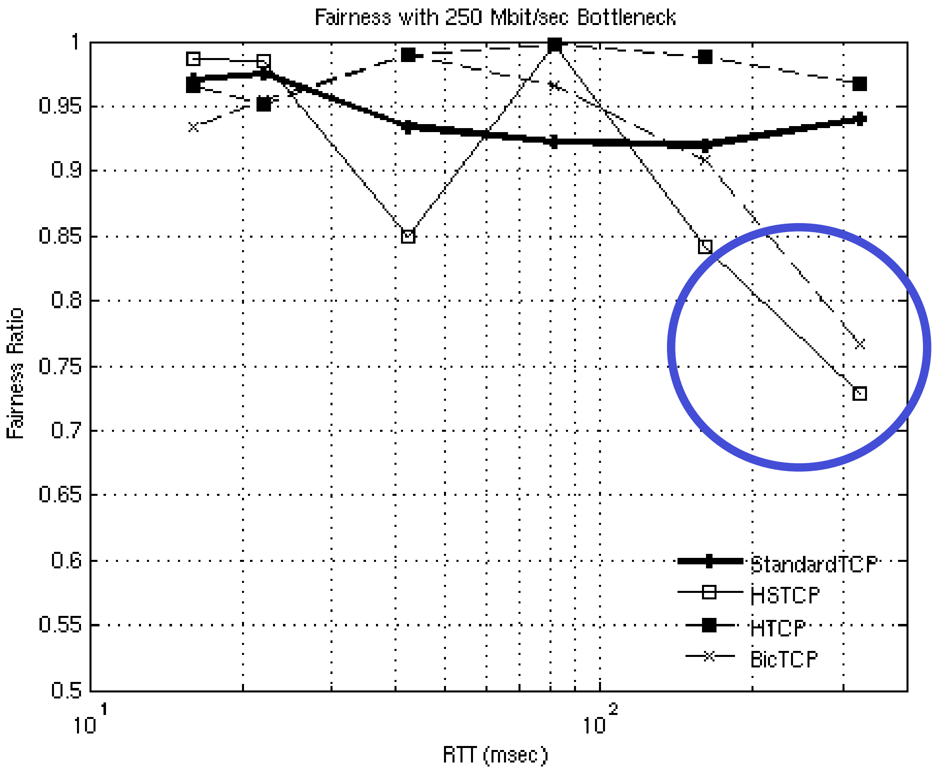

250Mbs, 42ms RTT

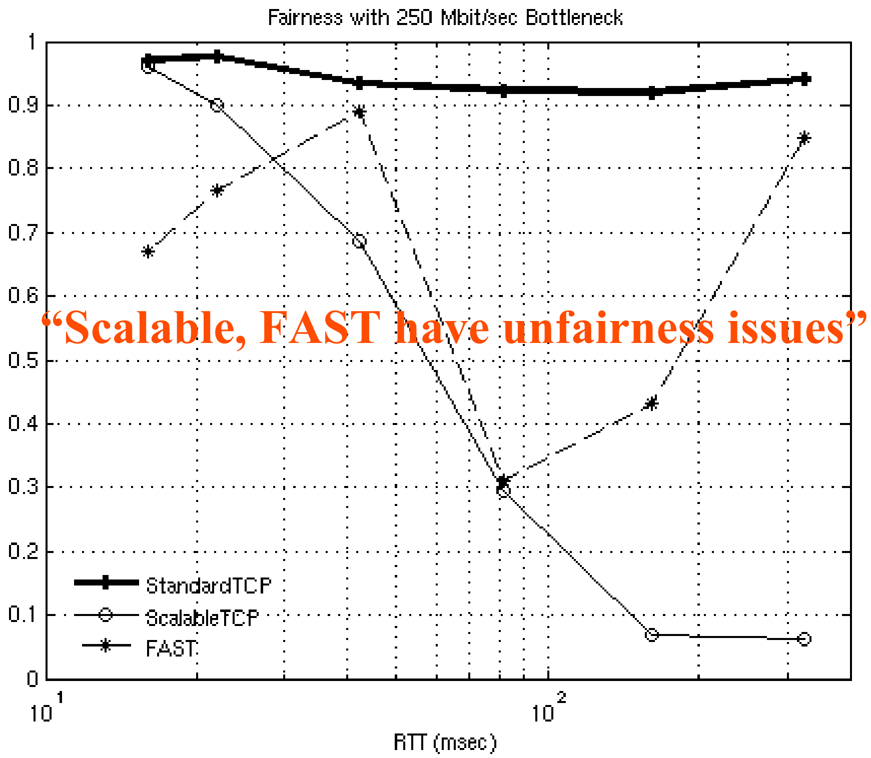
250Mbs, 162ms RTT

250Mbs, 324ms RTT


250Mbs, 42ms RTT
cwnd (packets)
0
200
400
600
800
1000
1200
100
200
300
400
500
600
FAST 1
FAST 2
250Mbs, 162ms RTT
cwnd (packets)
0
500
1000
1500
2000
2500
3000
3500
4000
100
200
300
400
500
600
FAST 1
FAST 2
250Mbs, 324ms RTT
cwnd (packets)
0
1000
2000
3000
4000
5000
6000
7000
8000
100
200
300
400
500
600
FAST 1
FAST 2
Time (seconds)

Symmetric conditions (2 flows): Fairness



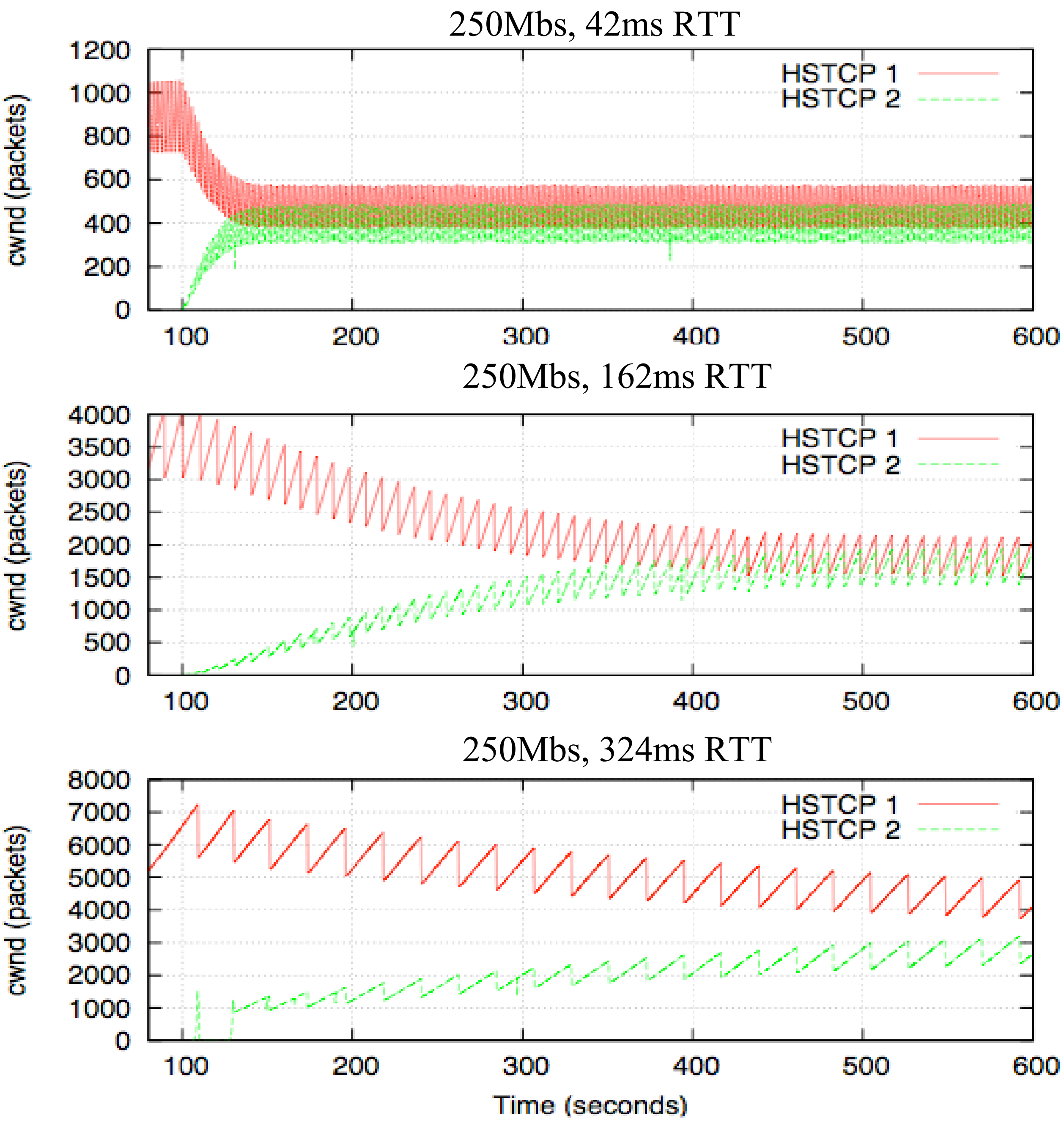


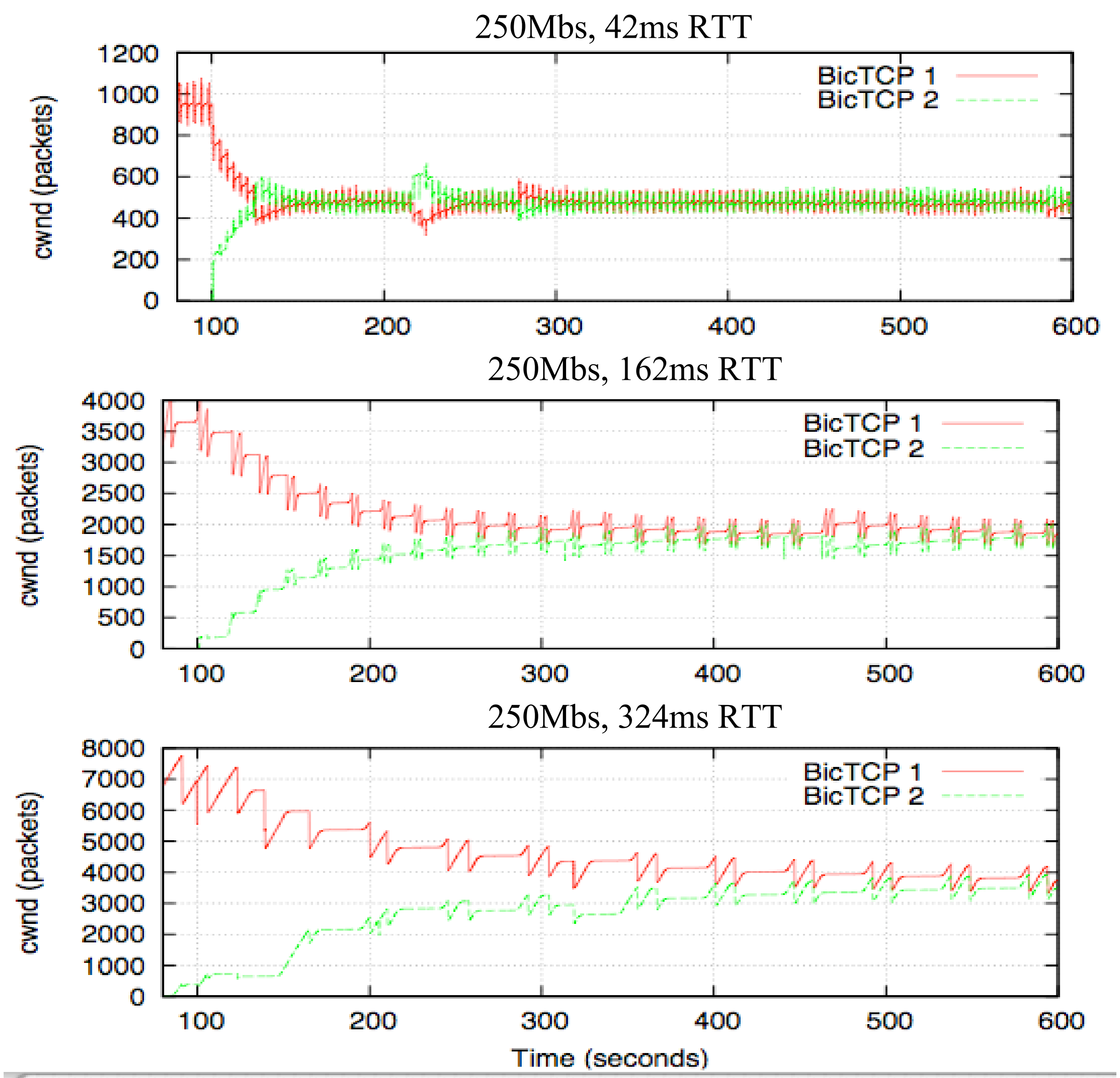
"Scalable, FAST have unfairness issues"

•Common network stack implementation used

•Averages over 5 tests runs

•Queue 20% DBP


250Mbs, 42ms RTT
cwnd (packets)
1200
1000
800
600
400
200
0
100
200
300
400
500
600
HSTCP 1
HSTCP 2
250Mbs, 162ms RTT
cwnd (packets)
4000
3500
3000
2500
2000
1500
1000
500
0
100
200
300
400
500
600
HSTCP 1
HSTCP 2
250Mbs, 324ms RTT
cwnd (packets)
8000
7000
6000
5000
4000
3000
2000
1000
0
100
200
300
400
500
600
HSTCP 1
HSTCP 2
Time (seconds)



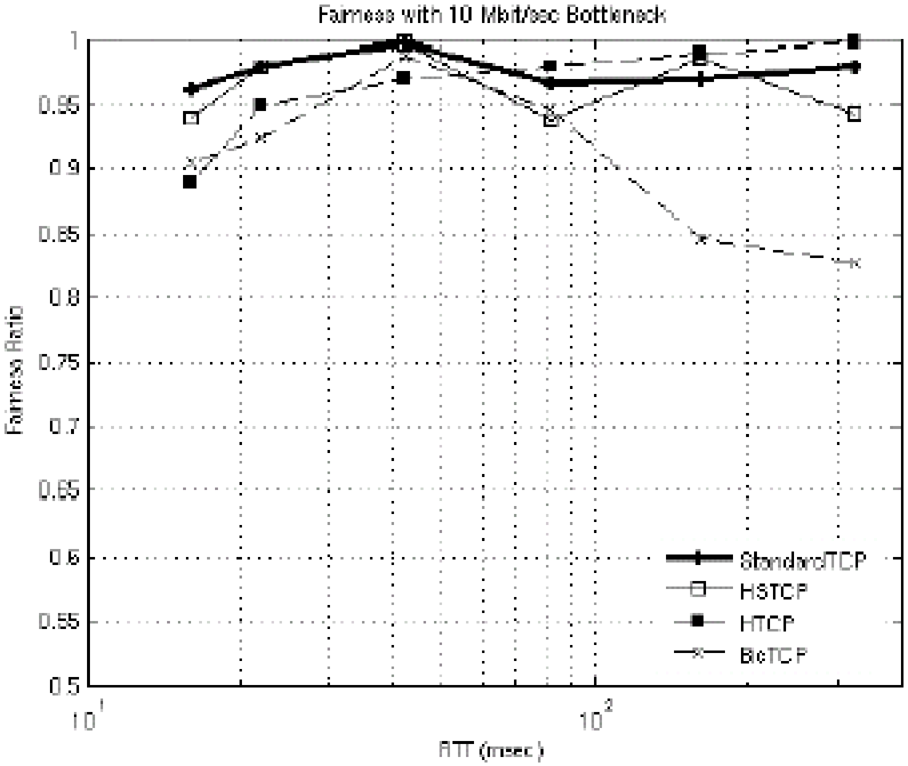
250Mbs, 42ms RTT

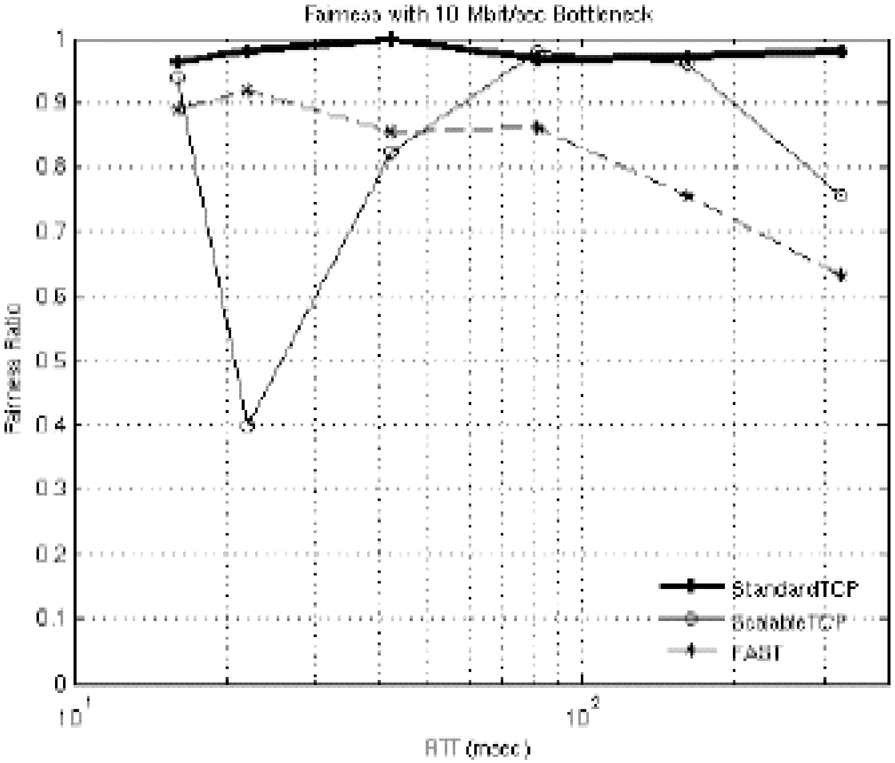
250Mbs, 162ms RTT

250Mbs, 324ms RTT

Symmetric conditions (2 flows): Fairness

Impact of BDP …

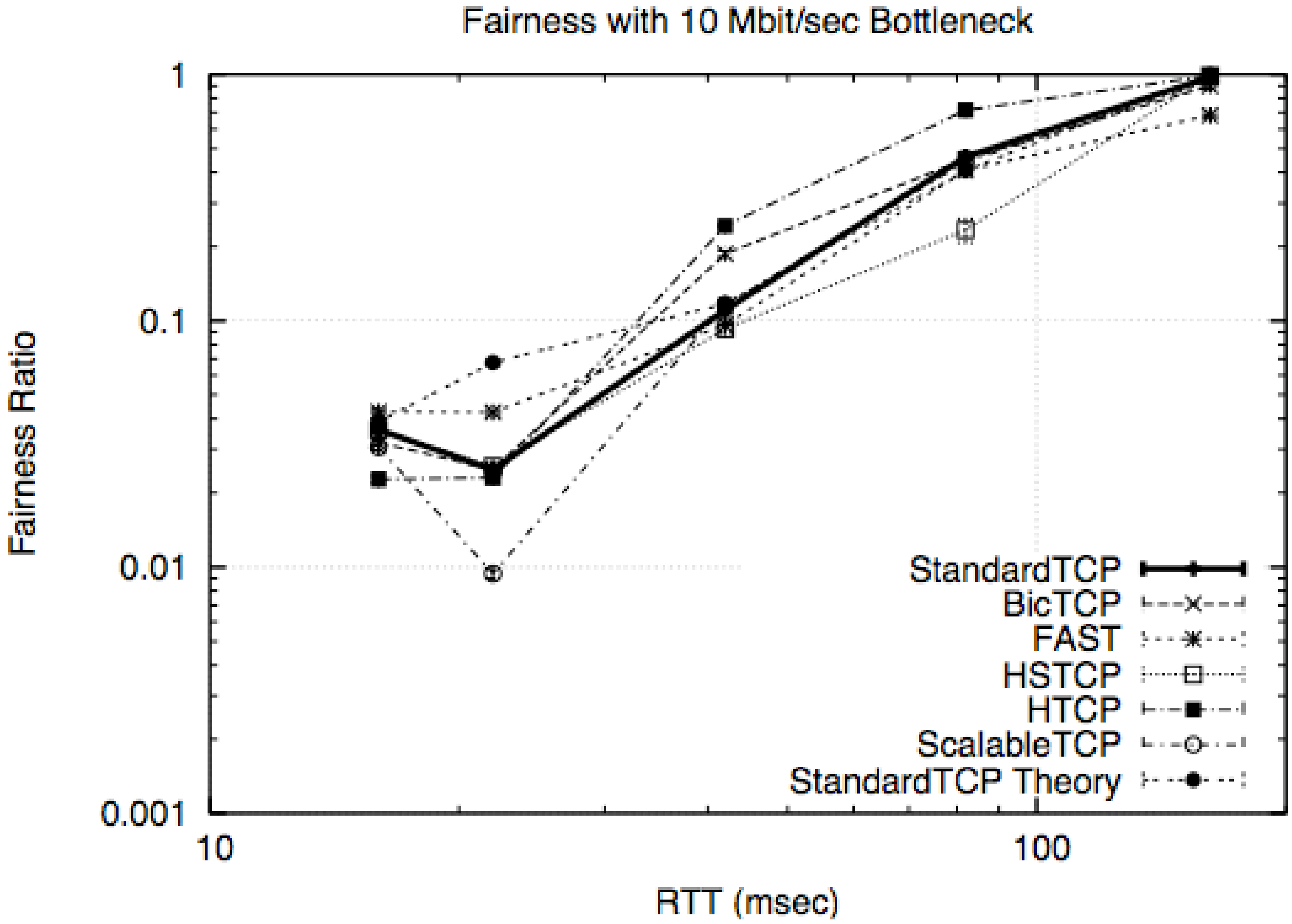




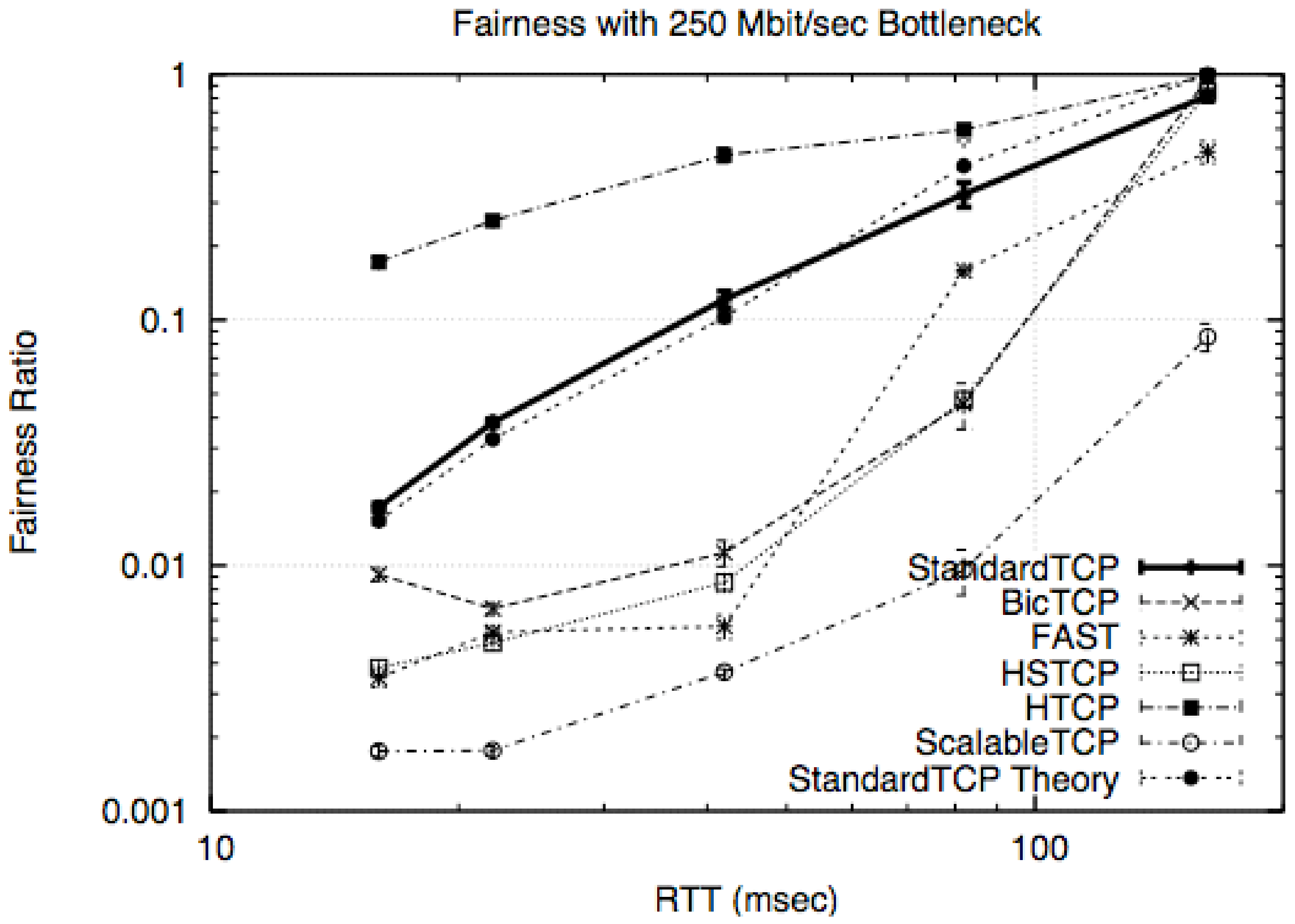
RTT Unfairness

- 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).

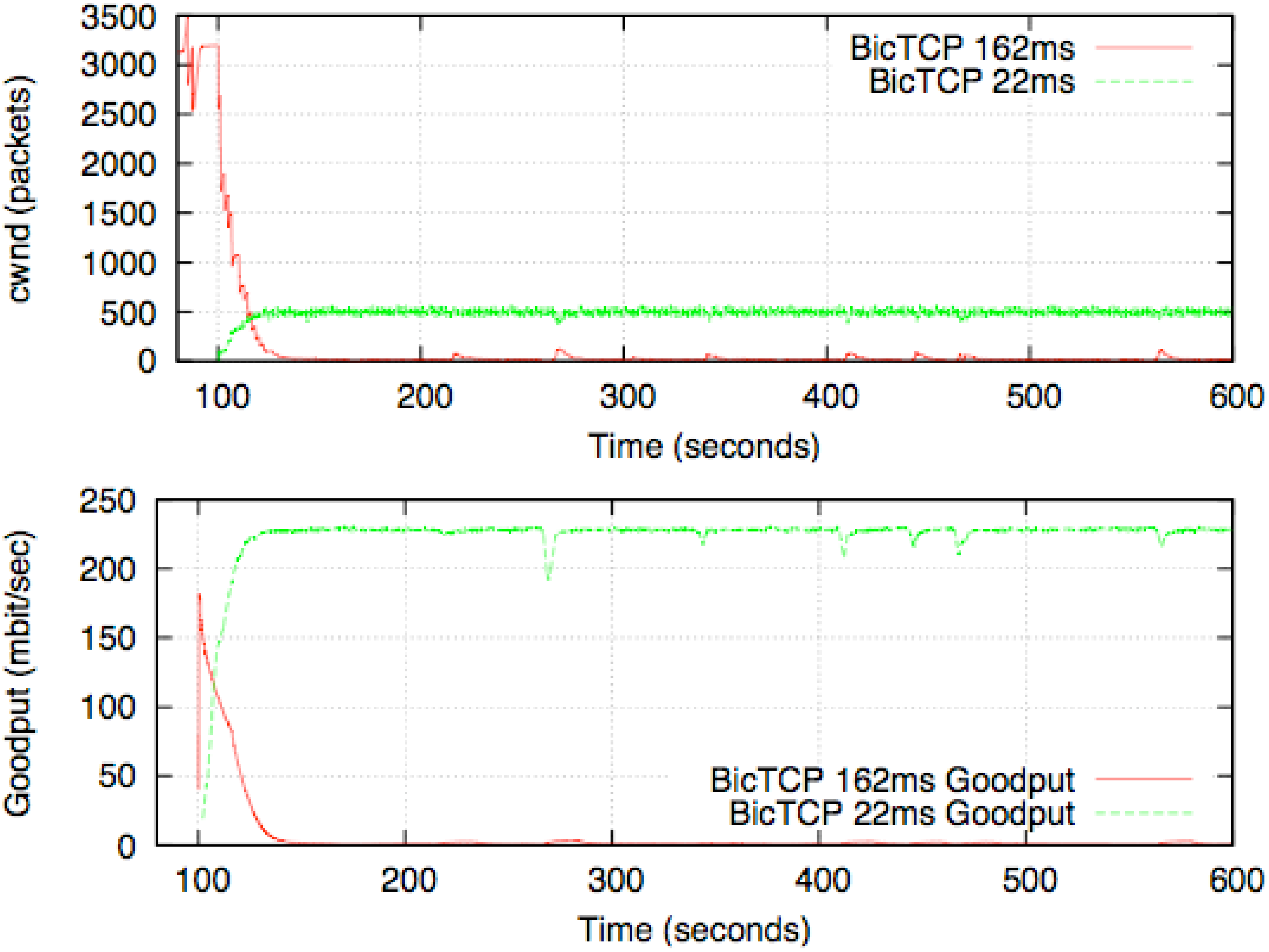
RTT Unfairness



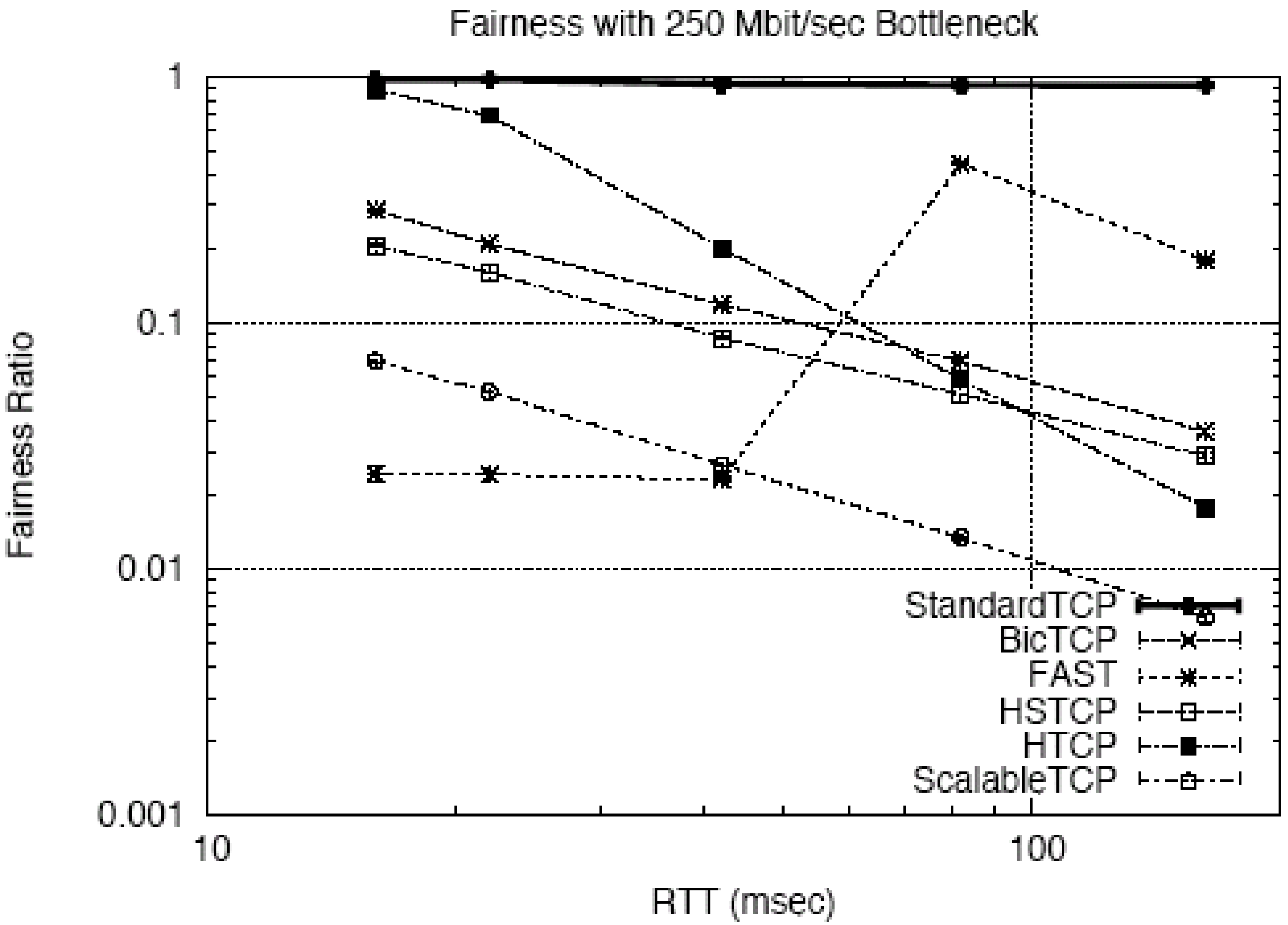
RTT Unfairness



RTT Unfairness



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



Summary

•Demonstrate that even simple tests can be surprisingly revealing. Suggests that some screening is indeed worthwhile.

•Do **not** claim that these results are exhaustive, only that they are a useful starting point.

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

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

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

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