Experimental Evaluation of Cubic TCP

Doug Leith, Robert Shorten, Gavin McCullagh

Hamilton Institute

Extended version at: http://www.hamilton.ie/net/pfldnet2007_cubic_final.pdf

Background

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

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)

.

Background (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. Therefore expect to have to measure performance of proposed changes over a range of conditions also.

Take measurements for a grid of data points …

-bandwidths of 1Mb/s, 10Mb/s, 250Mb/s, 500Mb/s
-two-way propagation delays of 16ms - 200ms
-range of queue sizes from 2% - 100% BDP.
-range of background traffic levels - number of sesions 25,50,200, distribution of connection sizes Pareto with mean 1, 10, 100 packets and shape 1.2.

Some minor practical issues

Need to control for different network stack implementations

•Common stack used in tests.

•Measurements always collected for standard TCP to provide baseline/sanity check.

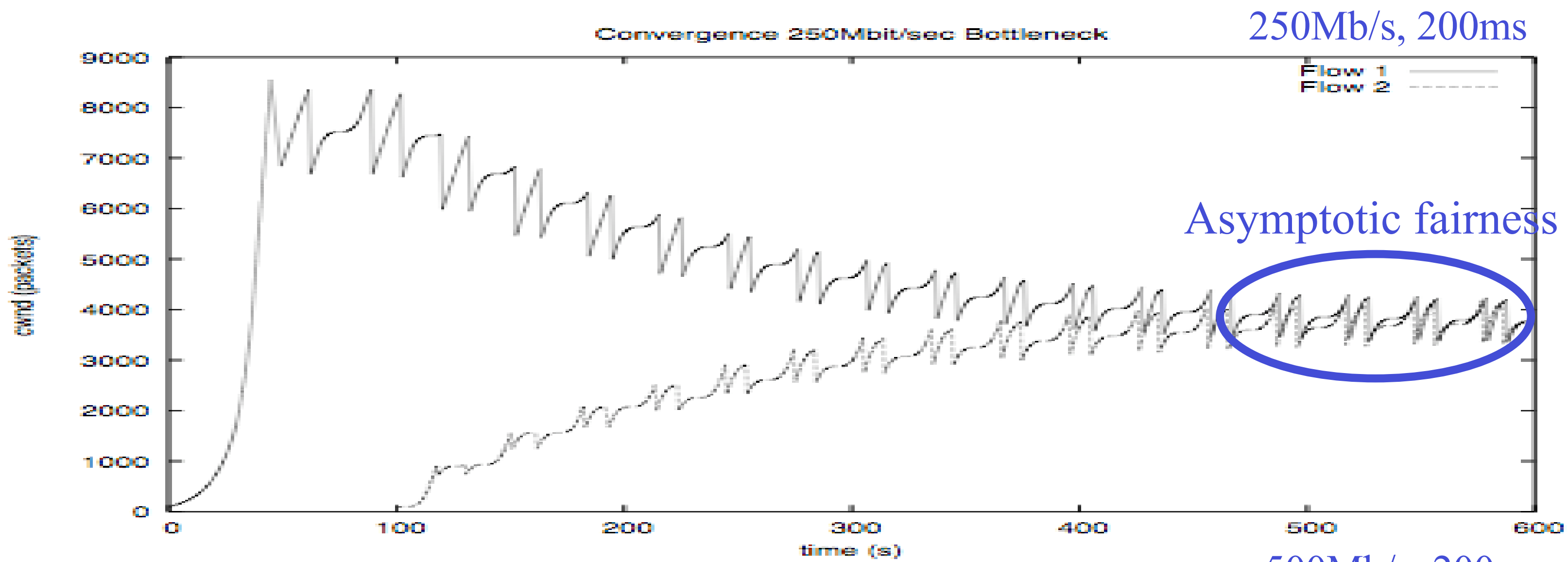
•We have validated performance up to 500Mb/s-200ms under our test conditions on Xeon hardware. Above BDP of about 12000 packets network stack issues appear to arise (again).

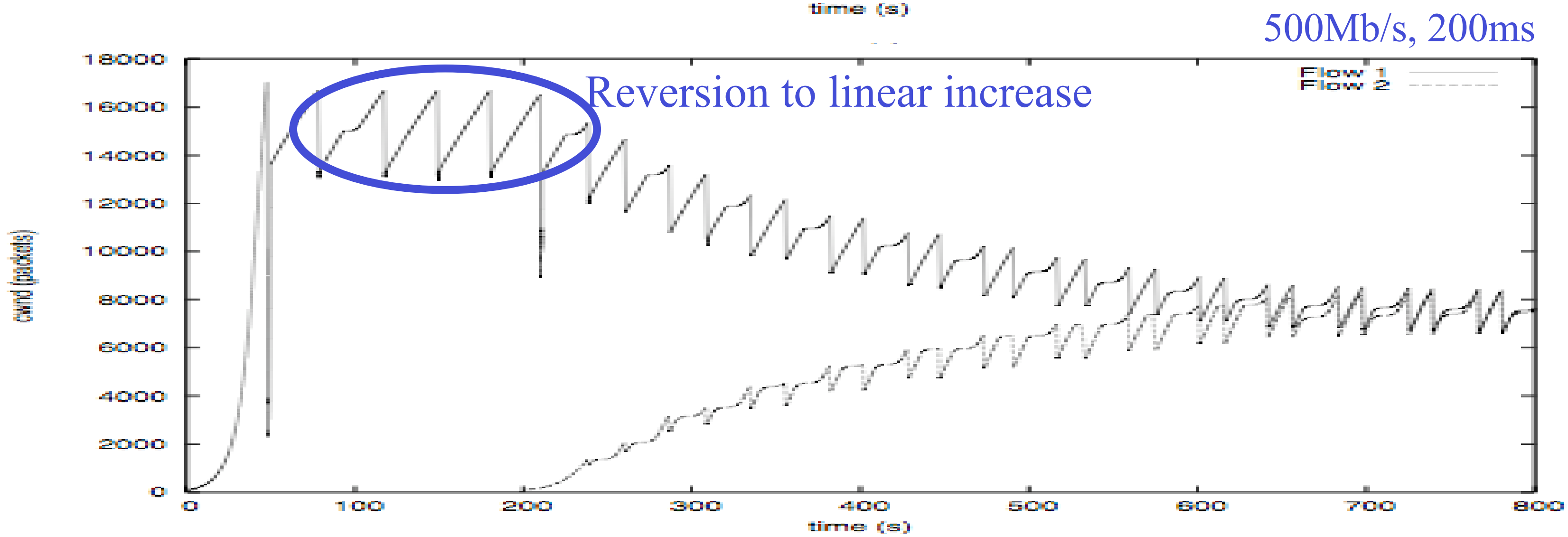
Buggy congestion control implementations

•Cubic TCP scaling bug. Highlighted by our initial testing. Fixed in 2.6.19 but present in earlier versions.

•Also different variants of Cubic algorithm - in original paper, in experimental tests by authors, in original Linux code (buggy), in current Linux code. We use most recent version in our tests.

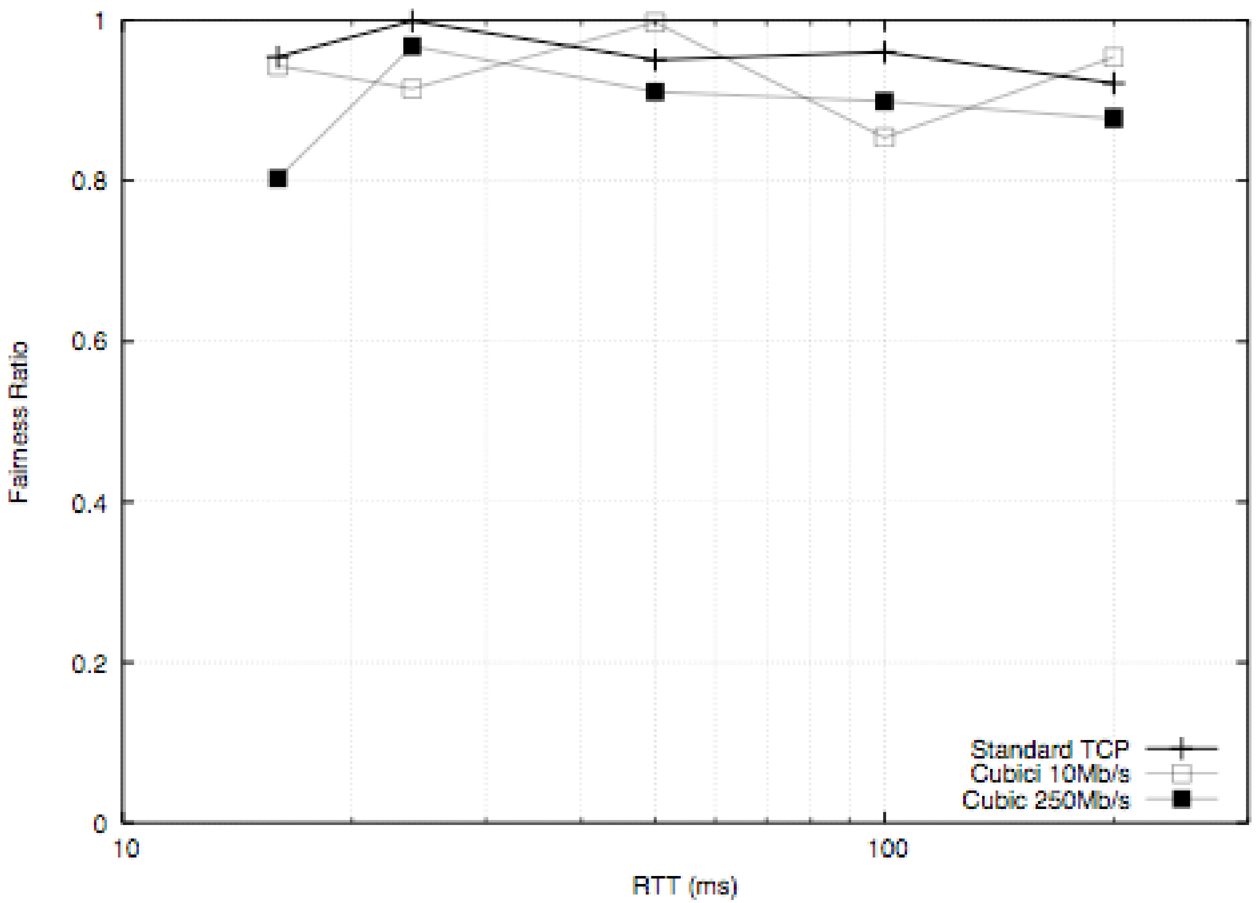
Symmetric conditions (2 flows)





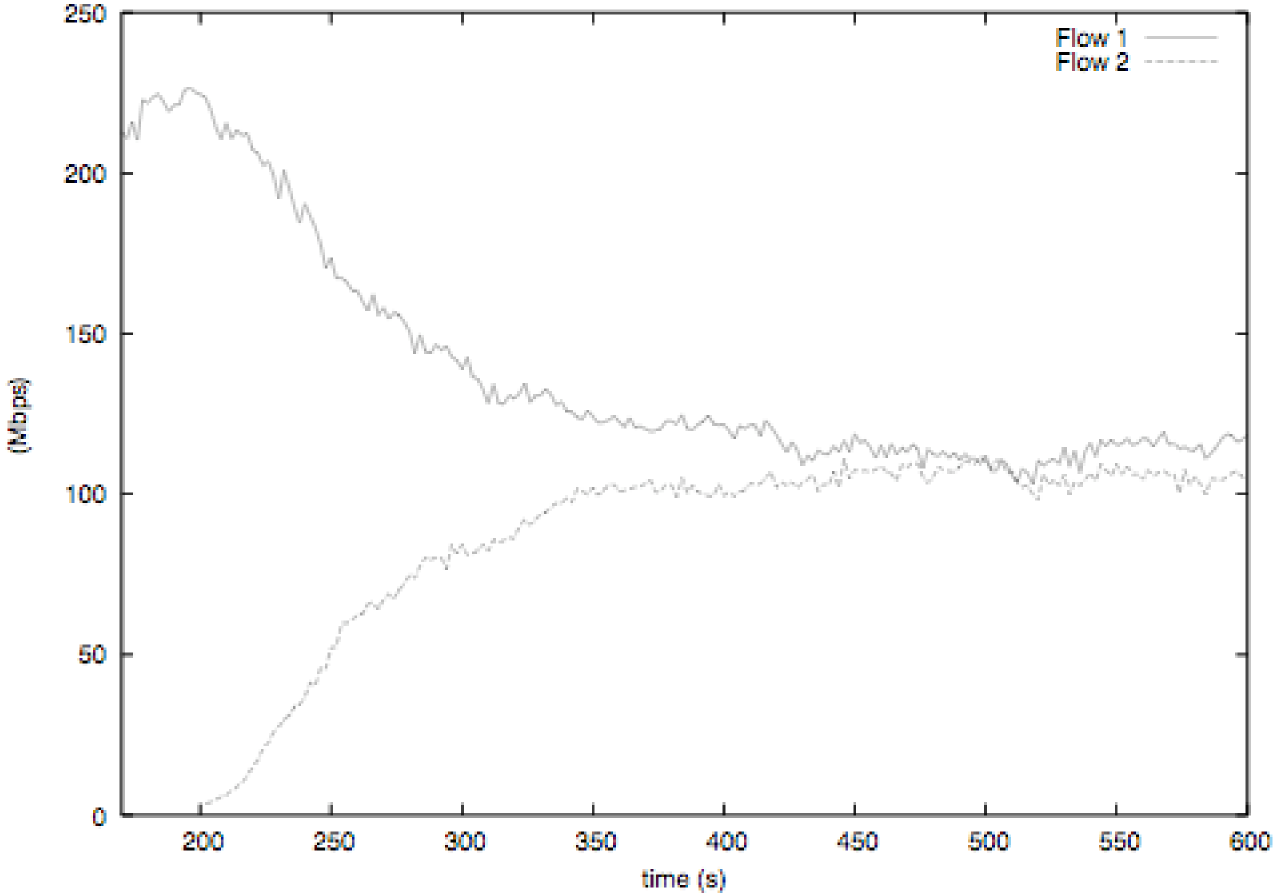
Symmetric conditions (2 flows)

Asymptotic throughput fairness.



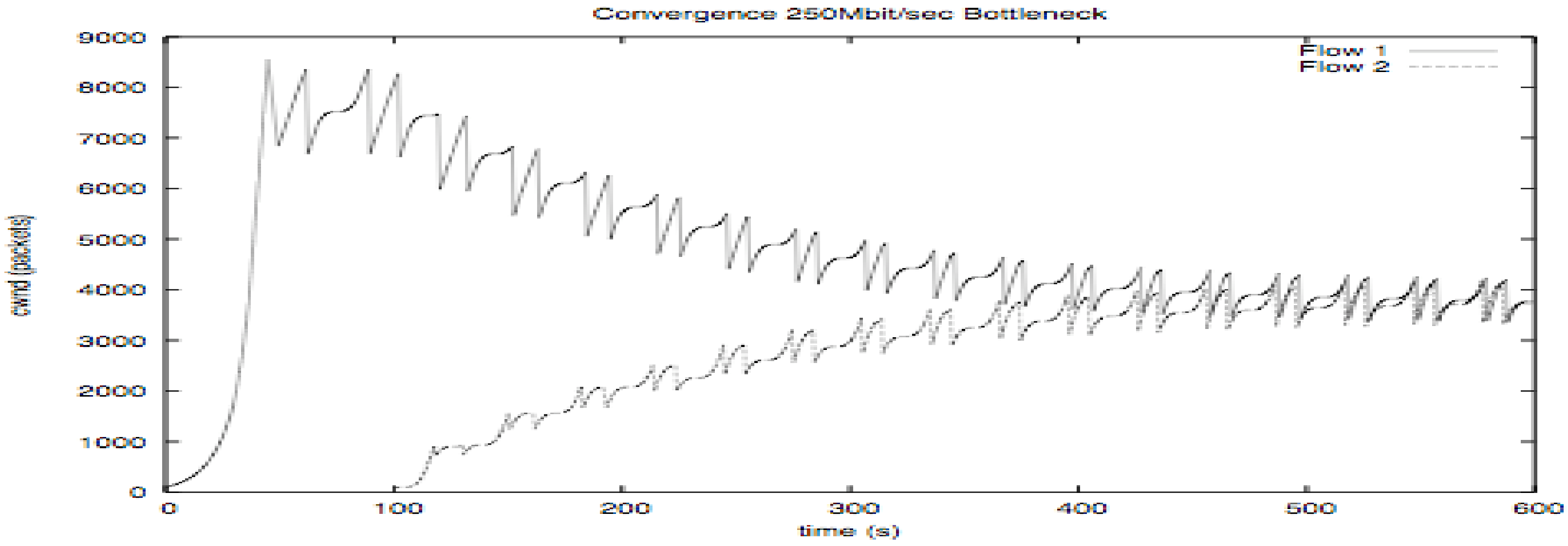
Symmetric conditions (2 flows)

Slow convergence still exhibited when unsynchronised drops



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

Source of slow convergence …



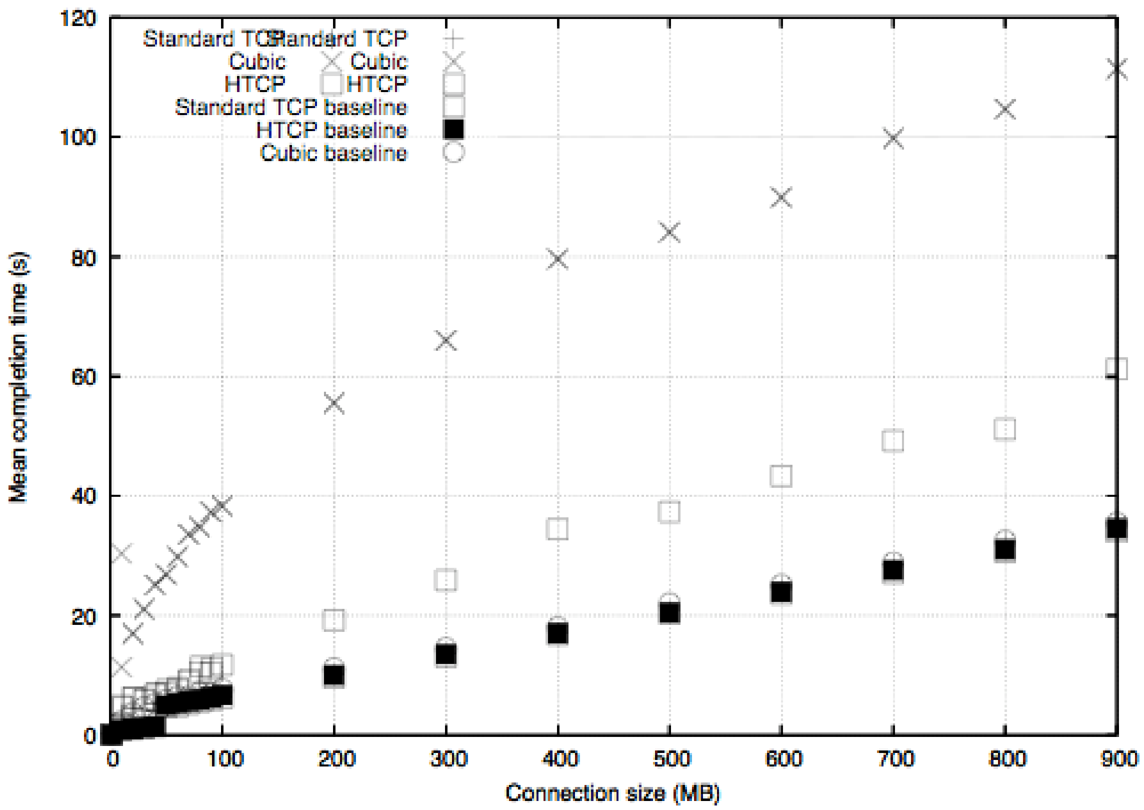
•Flows with low cwnds grab bandwidth less aggressively than flows with large cwnds cf High-Speed TCP

•Backoff factor of 0.8 (cf standard TCP backoff of 0.5) means that flows release bandwidth more slowly

Does slow convergence matter ?

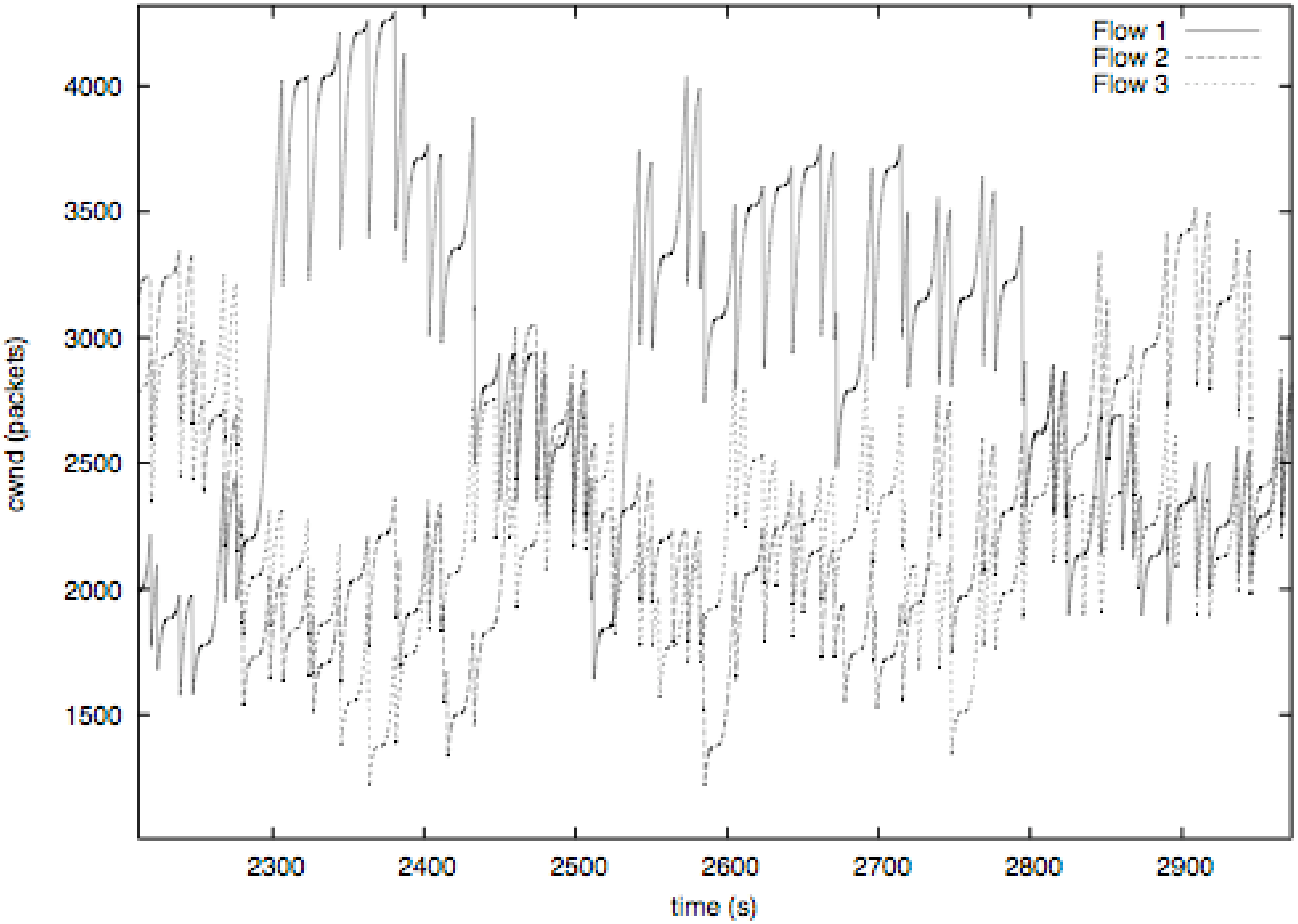
•Implies prolonged unfairness e.g. two identical file transfers may have very different completion times depending on the order in which they are started.

•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.

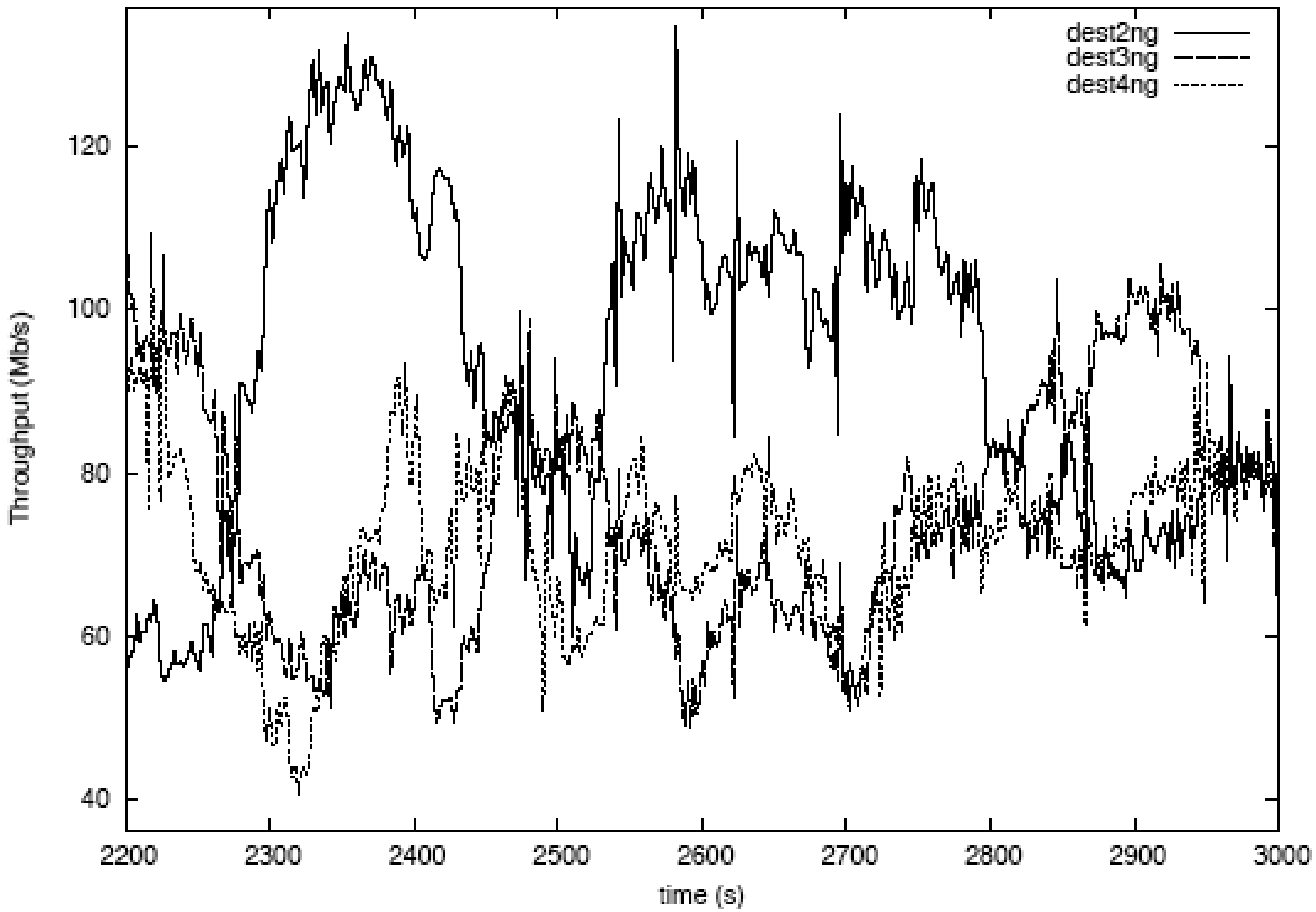


Does slow convergence matter ?

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

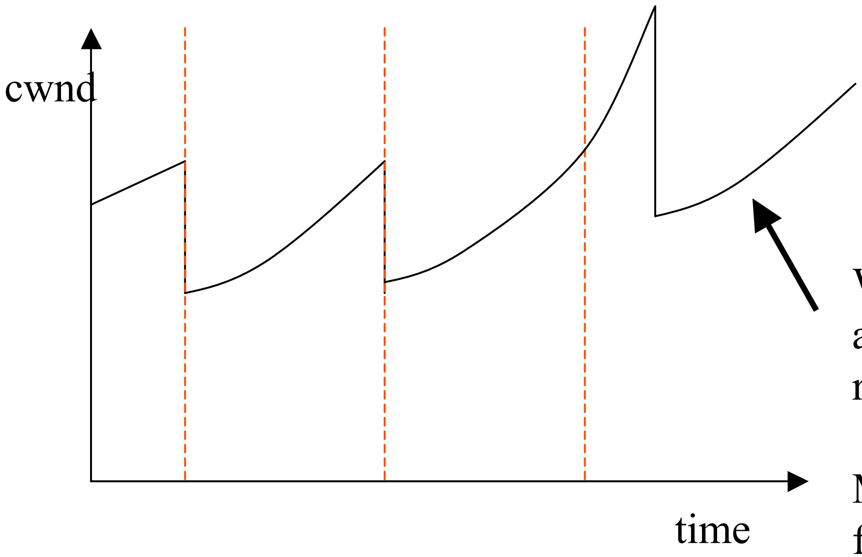


250Mb/s link, RTT 200ms, 3 long-lived Cubic flows. Link shared with 25 on-off sessions, Pareto connection size mean 100 packets, exponential off periods mean 10s.



dest2ng
dest3ng
dest4ng
Throughput (Mb/s)
time (s)
40
60
80
100
120
2200
2300
2400
2500
2600
2700
2800
2900
3000

Cost of "missing a drop"

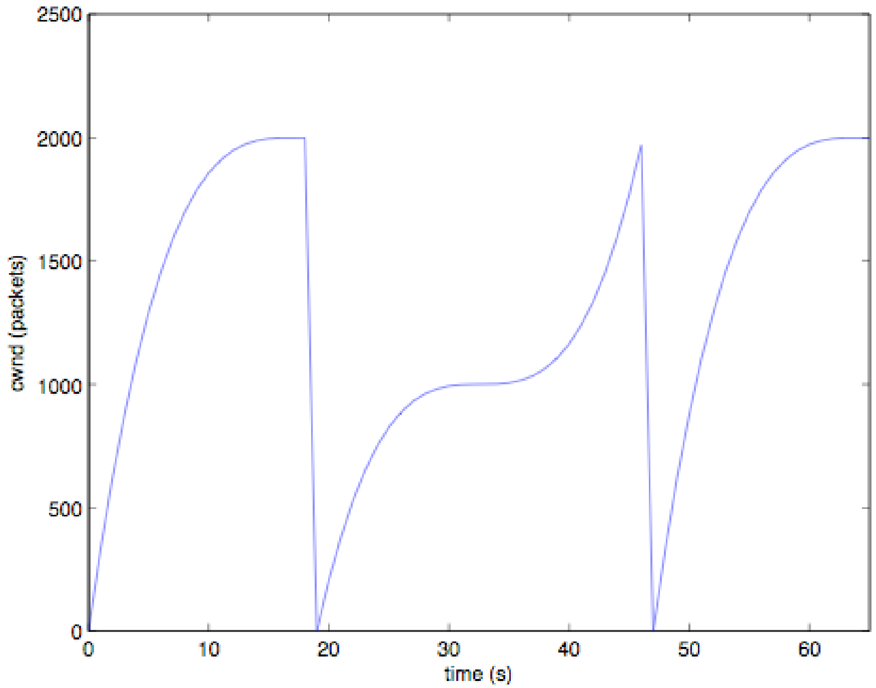


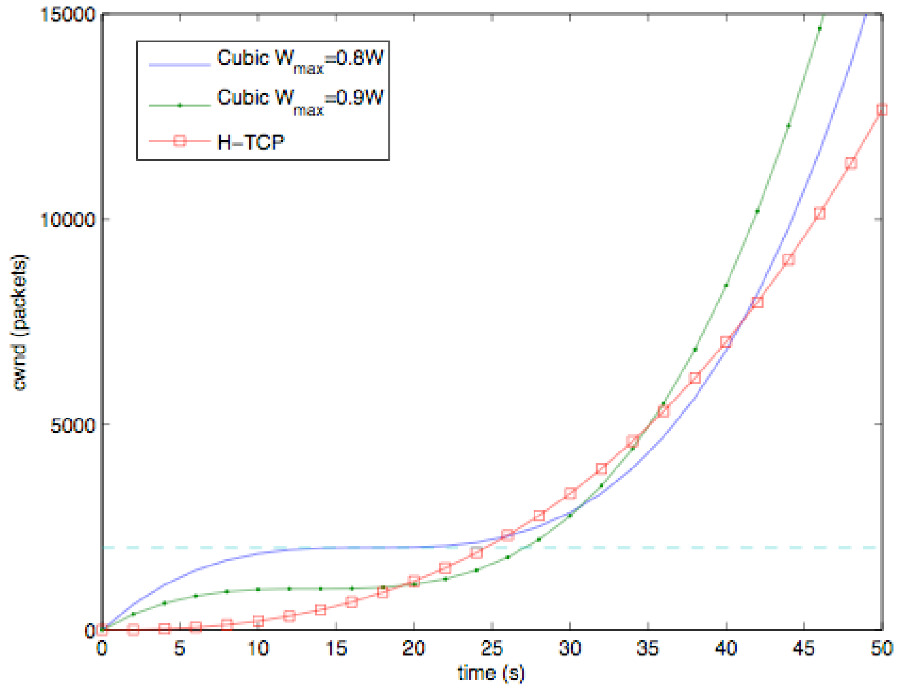
When "miss" a drop, an adaptive increase protocol can run away.

May lead to increased fluctuations in cwnd value.

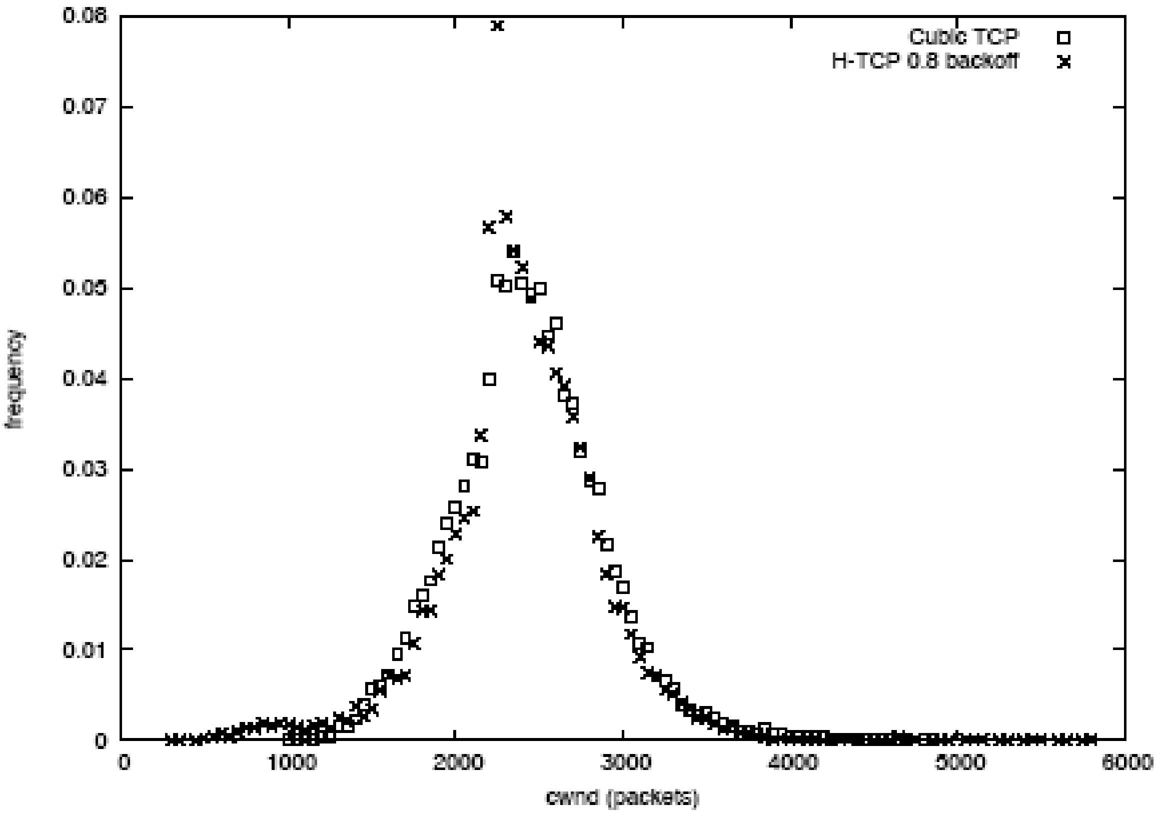
An issue for loss-based high-speed algorithms. All have aggressive increase functions of one form or another.

Cost of "missing a drop"

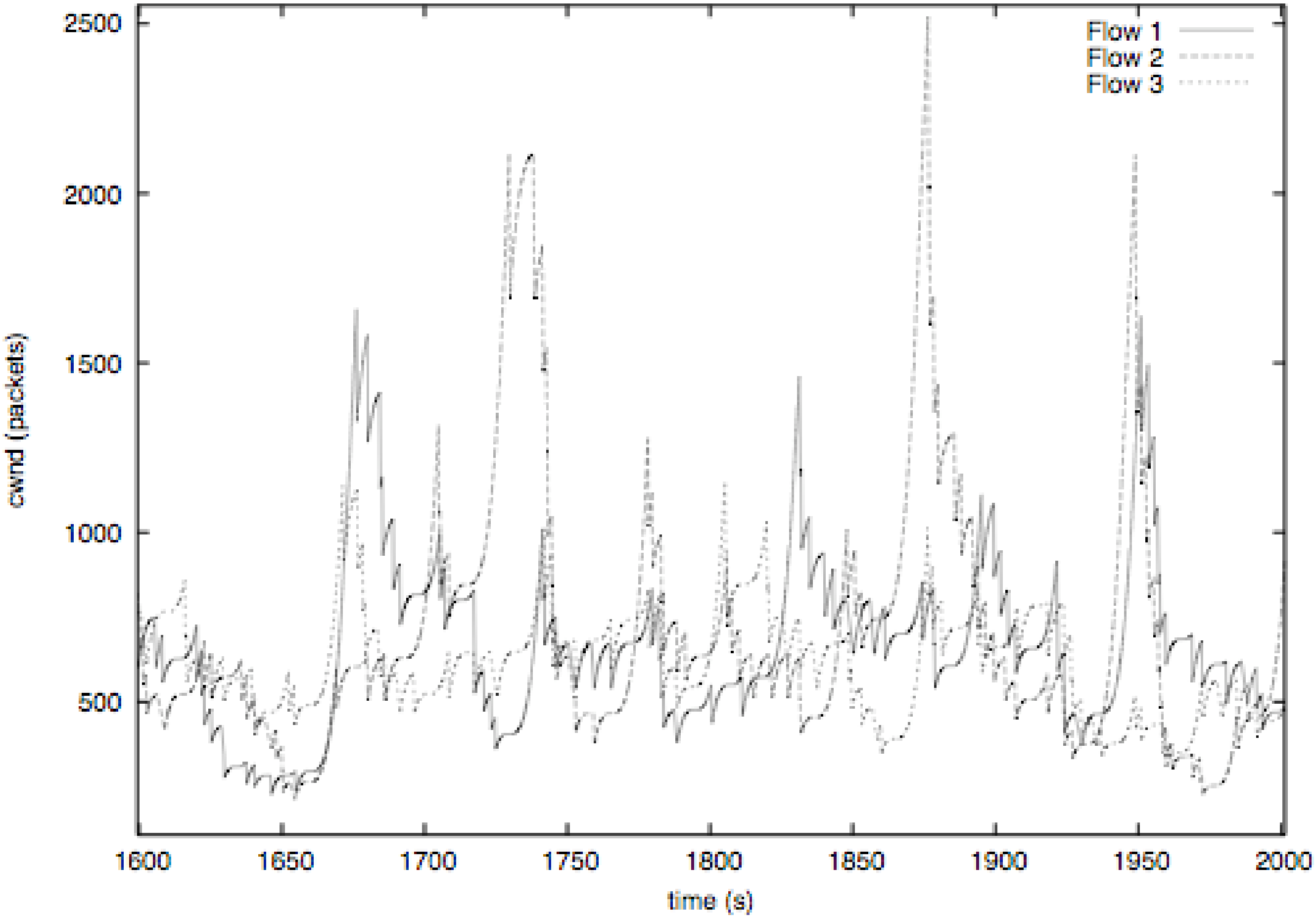




Cost of “missing a drop” & CoV

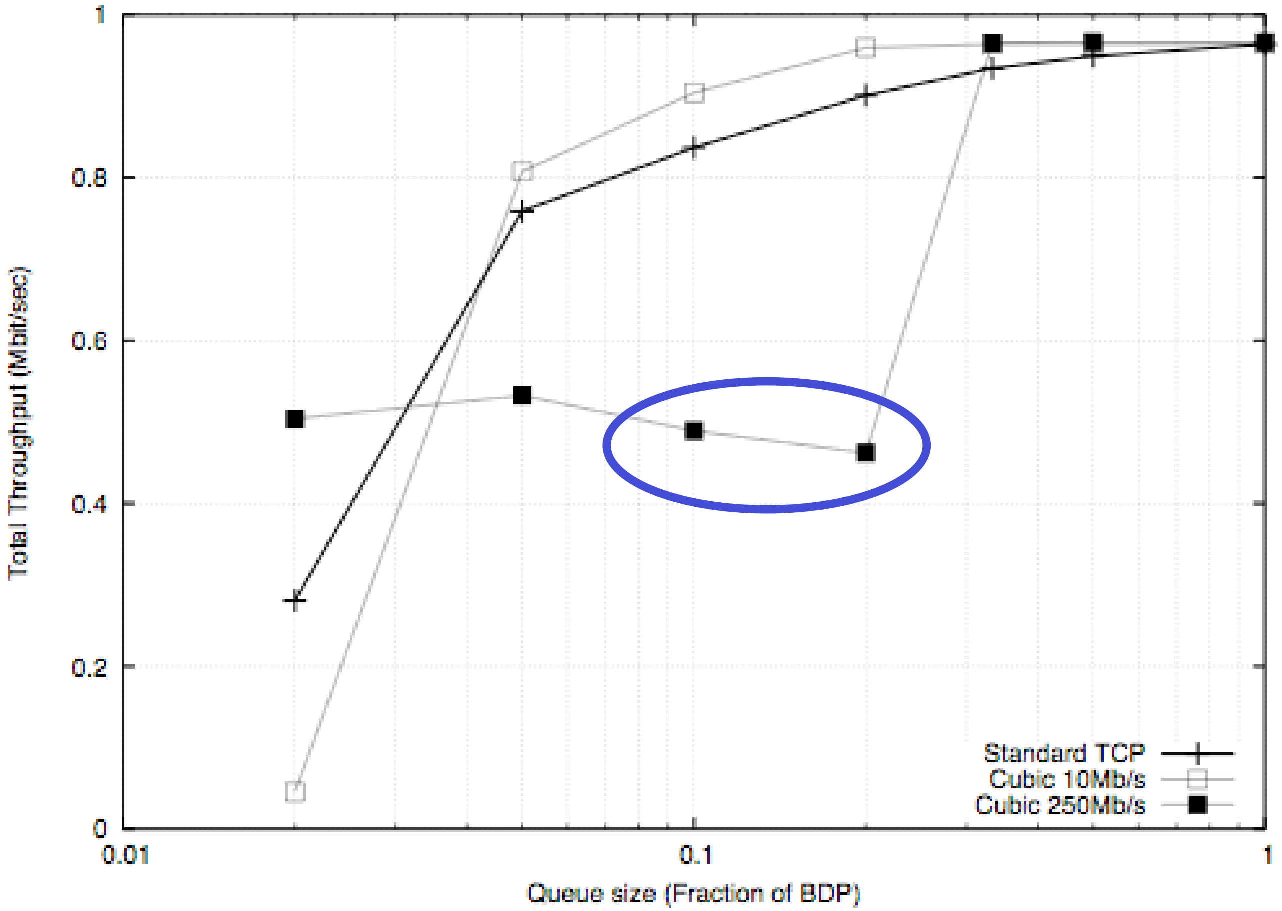


Cost of "missing a drop"



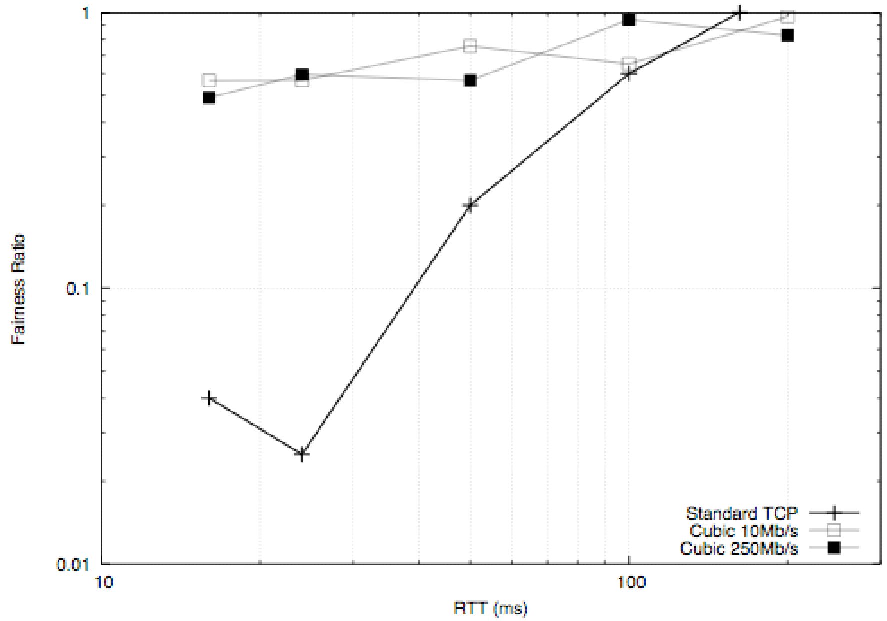
250Mb/s link, RTT 200ms, 3 long-lived Cubic flows. Link shared with 50 on-off sessions, Pareto connection size mean 100 packets, exponential off periods mean 10s.

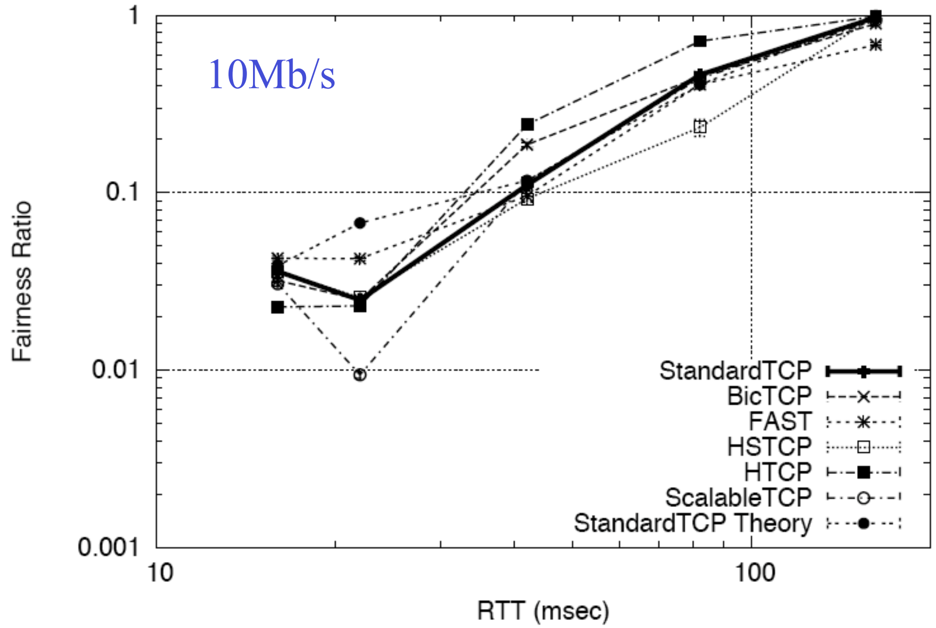
Link utilisation

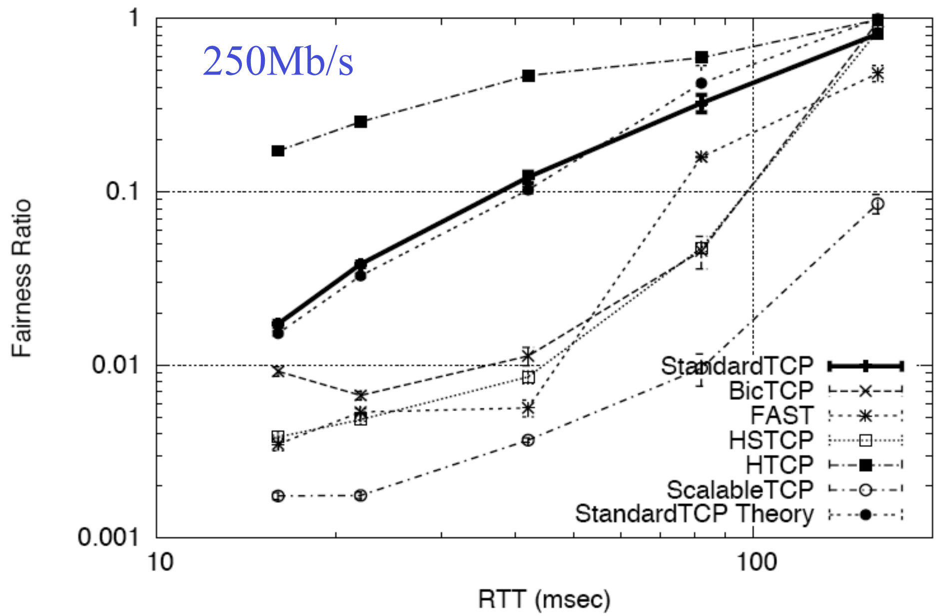


Link utilisation, two flows. 10Mb/s and 250Mb/s links, RTT 100ms

RTT Unfairness

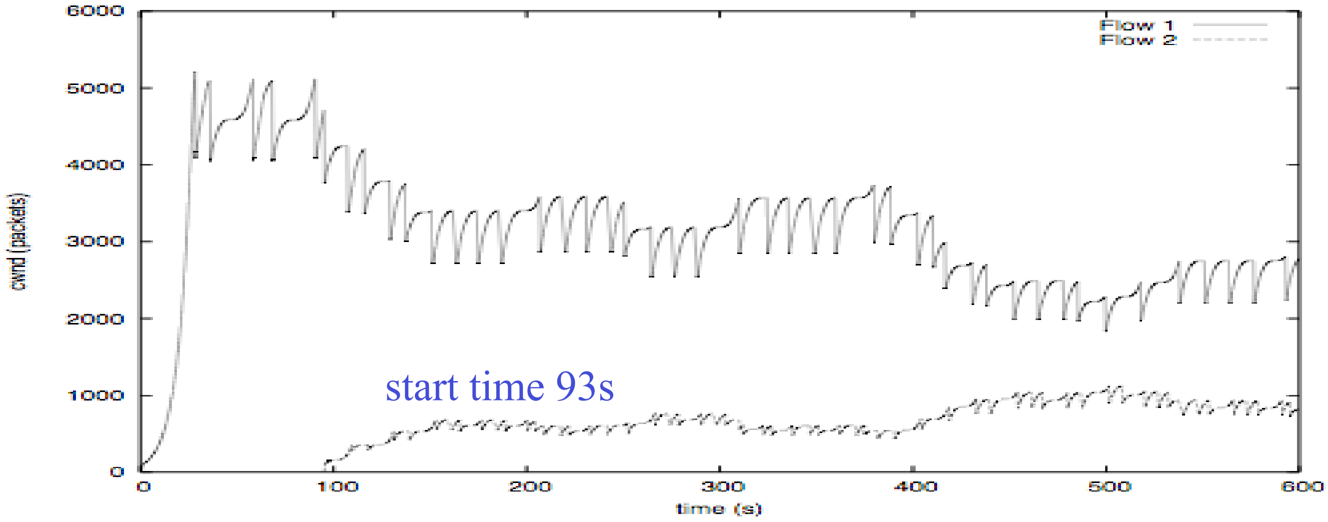


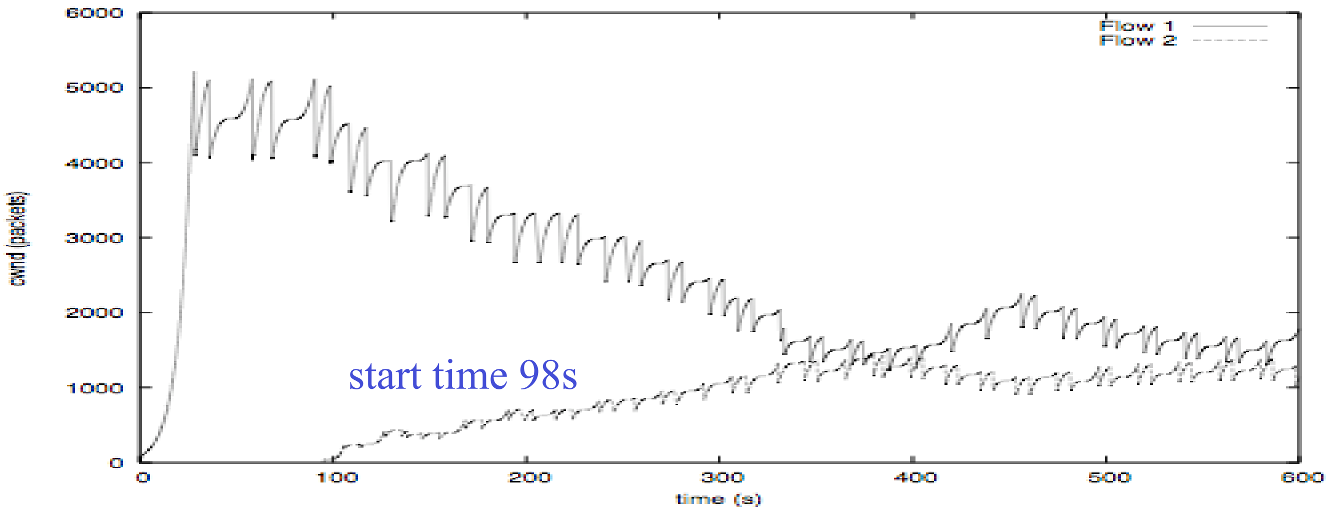




RTT Unfairness

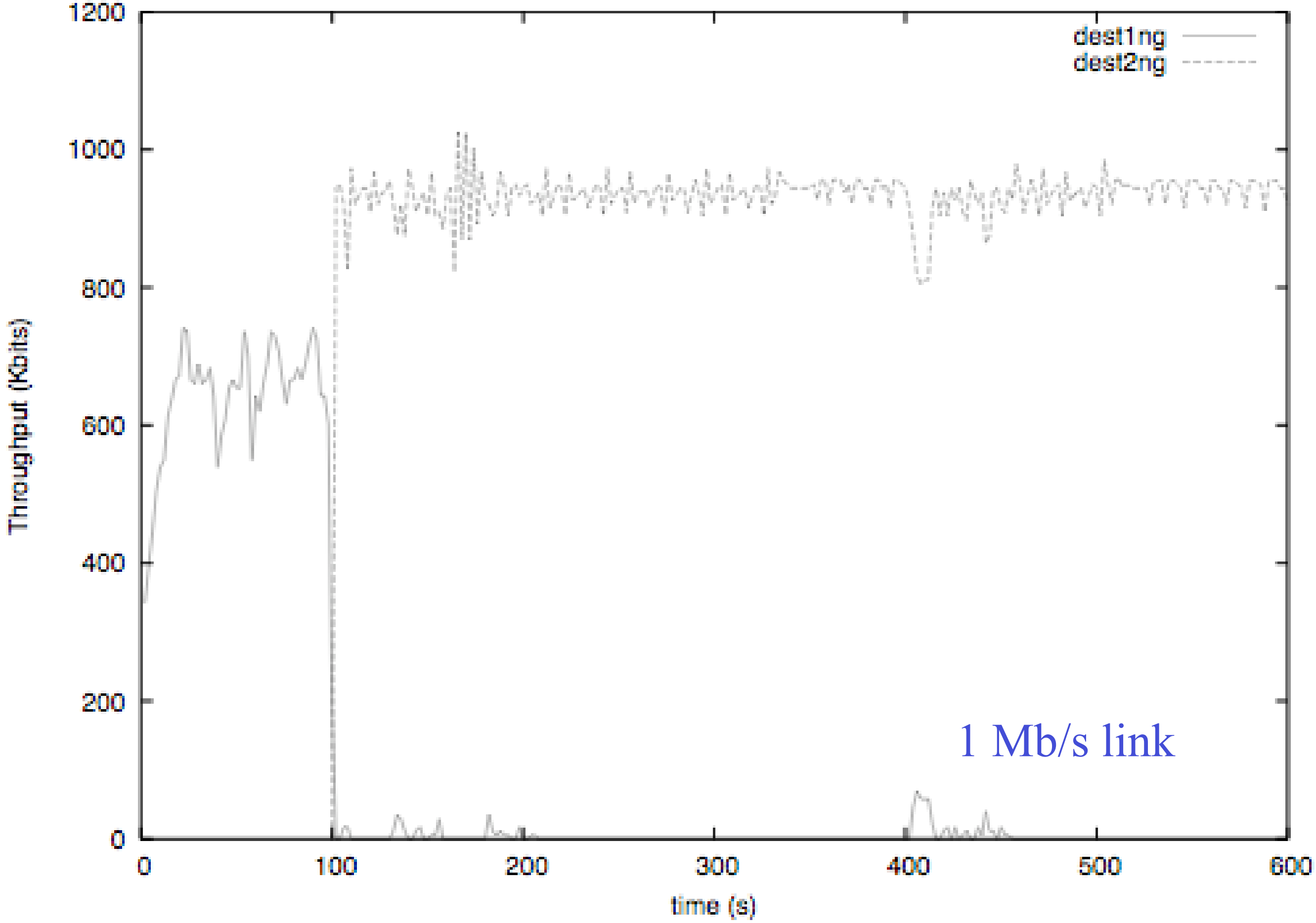
Fairness is sensitive to flow start times



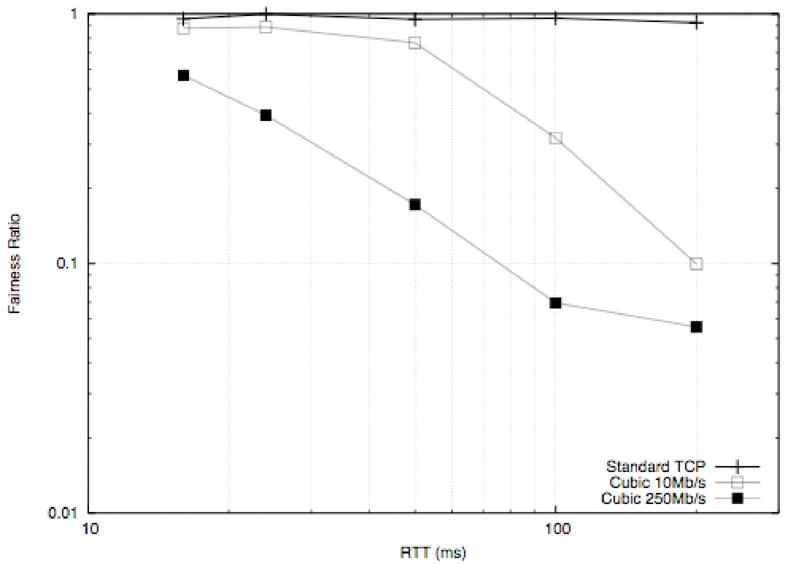


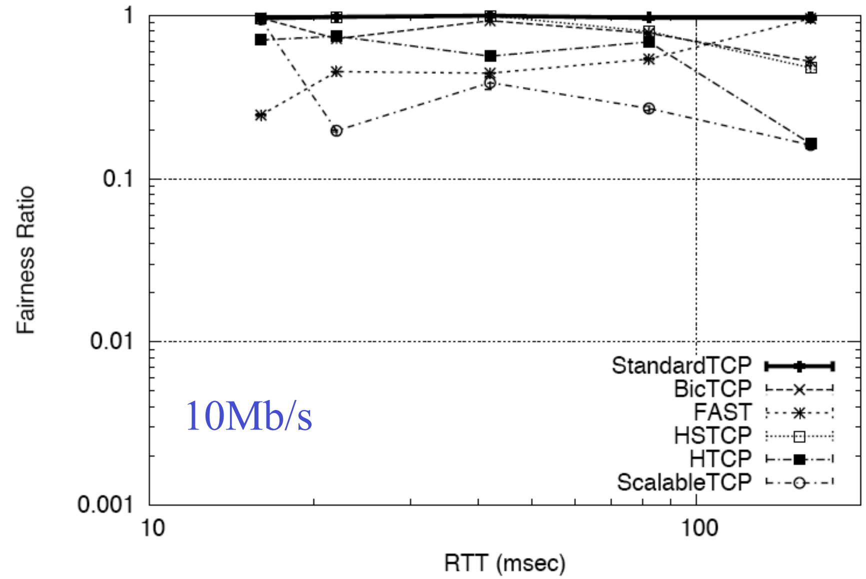
RTT Unfairness

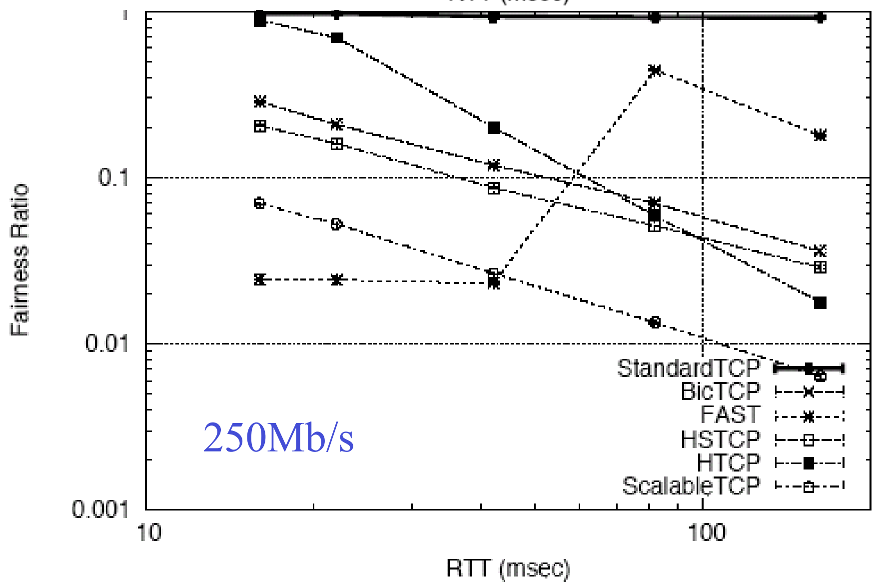
Quantisation issues at low cwnds



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







Summary

•Cubic TCP suffers from slow convergence - yields poor network responsiveness, prolonged unfairness between flows, increases unfairness between long and short lived flows.

•In common with other high-speed protocols, Cubic TCP uses an aggressive additive increase action to maintain short congestion epochs on high bandwidth-delay product paths. The associated cost of ``missing a drop" is similar for both Cubic TCP and HTCP.

•At high bandwidth-delay products Cubic TCP reverts to a linear increase function. This implies that congestion epoch duration eventually scales linearly with BDP (similarly to standard TCP).

•At higher speeds, for buffer sizes below 30\% BDP the link utilisation achieved by Cubic TCP falls to around 50\% of link capacity.

•For flows with different RTTs, Cubic exhibits unfairness that is dependent on the start time of the flows. It is unclear at present why this non-convergence behaviour occurs -- it may be due to a fundamental stability issue or perhaps associated with implementation issues.

•Demonstrate that even simple tests can be surprisingly revealing. Argue that it is vital to measure performance over a wide range of bandwidths, RTT's, queue sizes etc and study >1 competing flow. Do **not** claim that these results are exhaustive, only that they are a useful starting point.

Extra Slides

fast_convergence=0

250Mb/s, 200ms