Improving fairness of TCP congestion avoidance

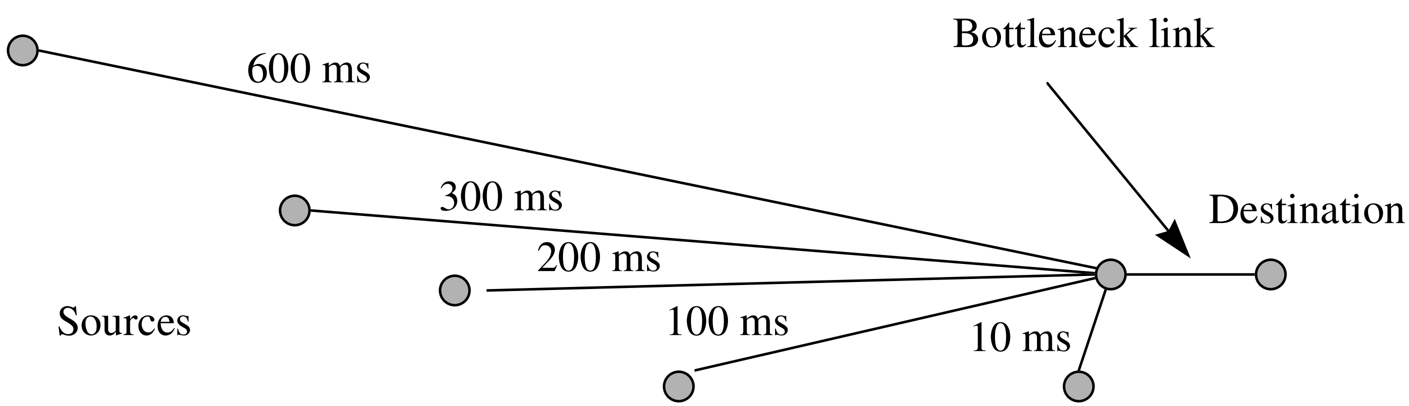
Tom Henderson

UC Berkeley, Hughes Research Labs

(continuation of work with Emile Sahouria, UCB)

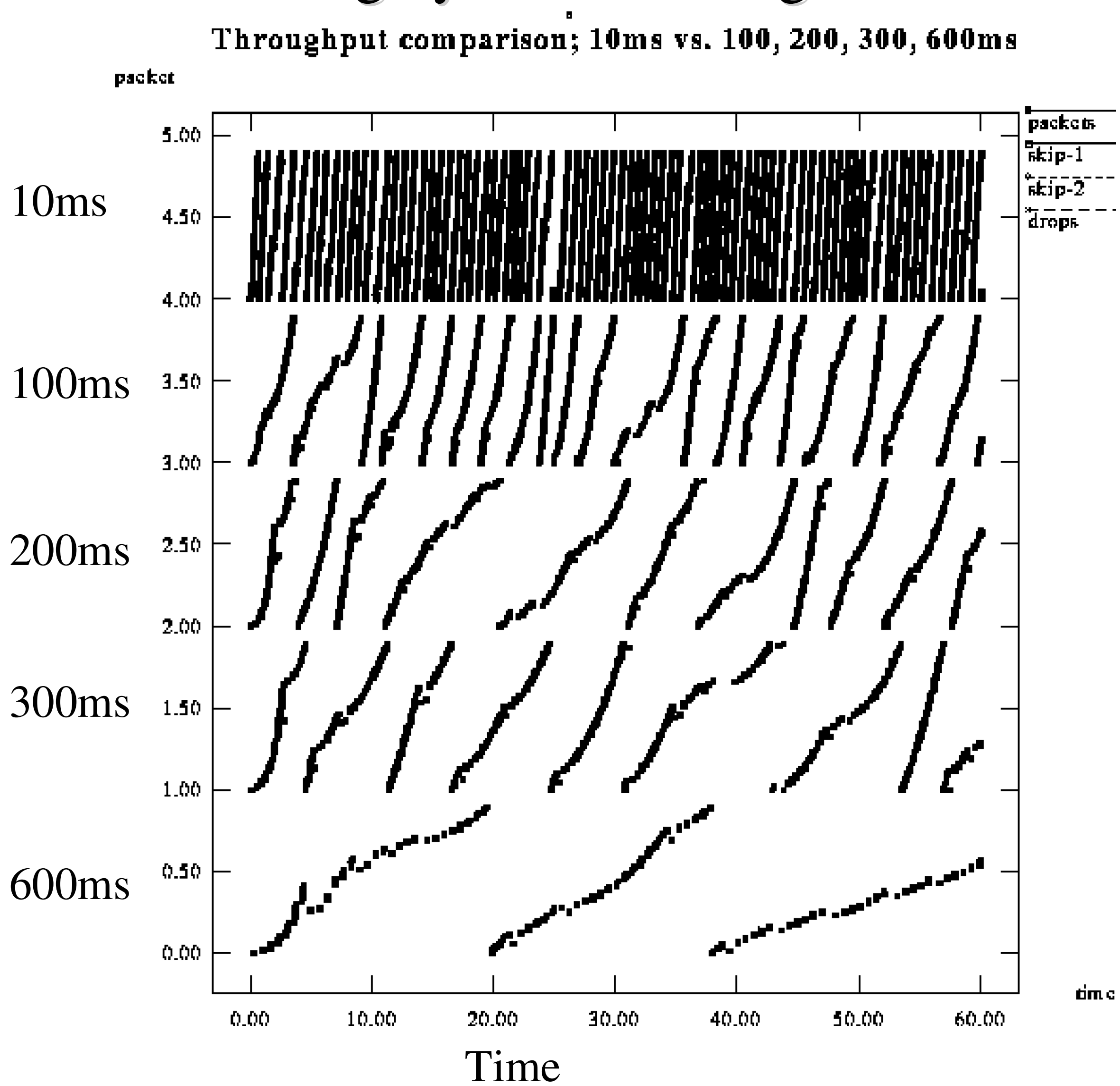
Introduction

What happens when multiple connections, each with different round trip delays, share a bottleneck link?



Introduction (cont.)

- TCP is highly unfair to long RTT connections



Introduction (cont.)

Long RTT connections cannot open their windows fast enough to squeeze data through the congested link.

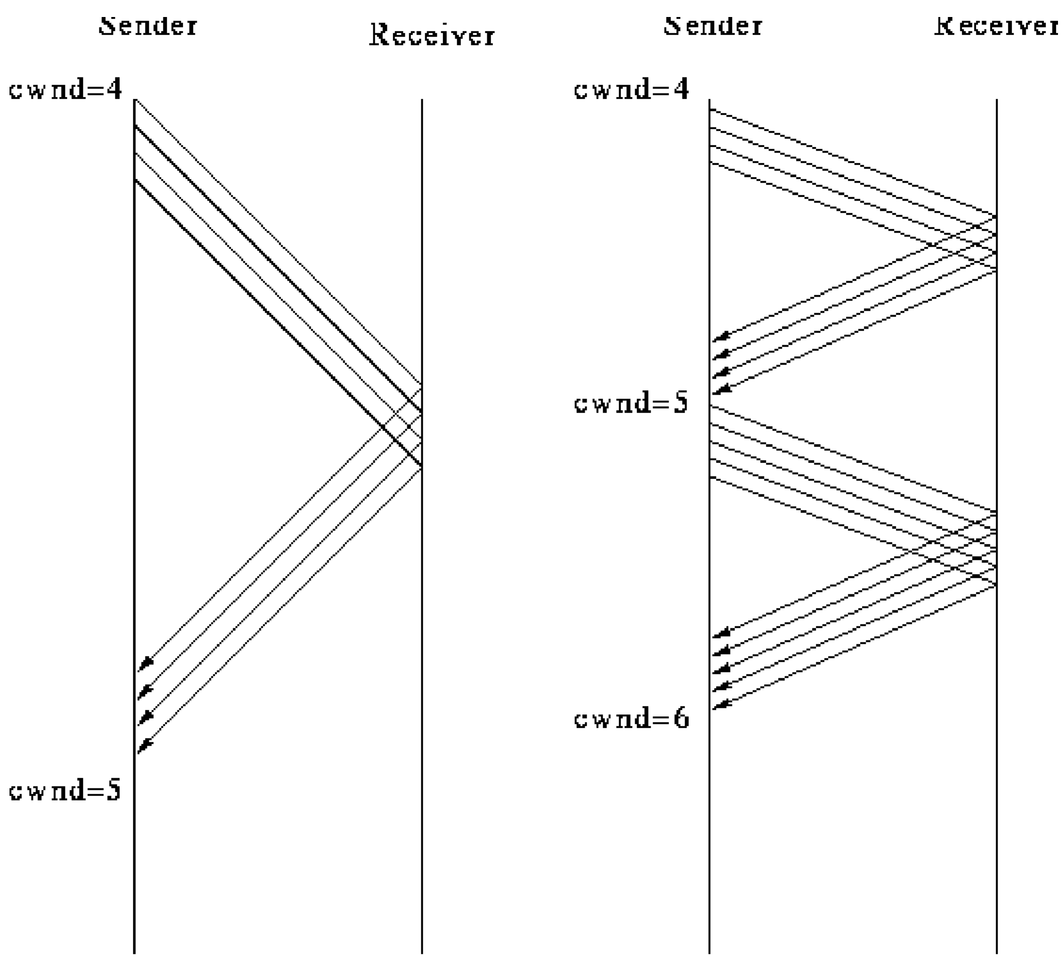
- Hypothesis: the problem is in the "additive increase" of TCP congestion avoidance

```
if (acked_new_data) {
    cwnd += 1/cwnd
}
```

- There is a $1/rtt^2$ bias in additive increase

Introduction (cont.)

Why it is insufficient to simply adjust the rate at which the congestion window builds



Related work

- Mankin [90], Floyd and Jacobson [92], Lakshman [97] have previously noted the problem of unfairness (no solution)
- Alternative techniques for improving TCP performance in hostile environments:
  - Striping [WWW browsers, Allman 96]
  - Spoofing [Balakrishnan 95]
  - Splitting [Bakre 94, Durst 96]
- Our approach aims for interoperable changes at the endpoints only
  - Stay close to standard TCP flow control

Research questions

1. To what extent can long delay connections improve their throughput in bottleneck situations by more aggressive additive window increases?

   - Improved throughput

2. Is the more aggressive behavoir “antisocial” towards peer connections?

   - Improved fairness, comparable utilization

Methodology

- ns-2 simulator used
  - TCP “Newreno”, “Sack”, and “Fack”
  - FIFO and RED queueing
- A number of foreground connections were analyzed in the presence of background emulated web connections
- Experiments used to validate simulation

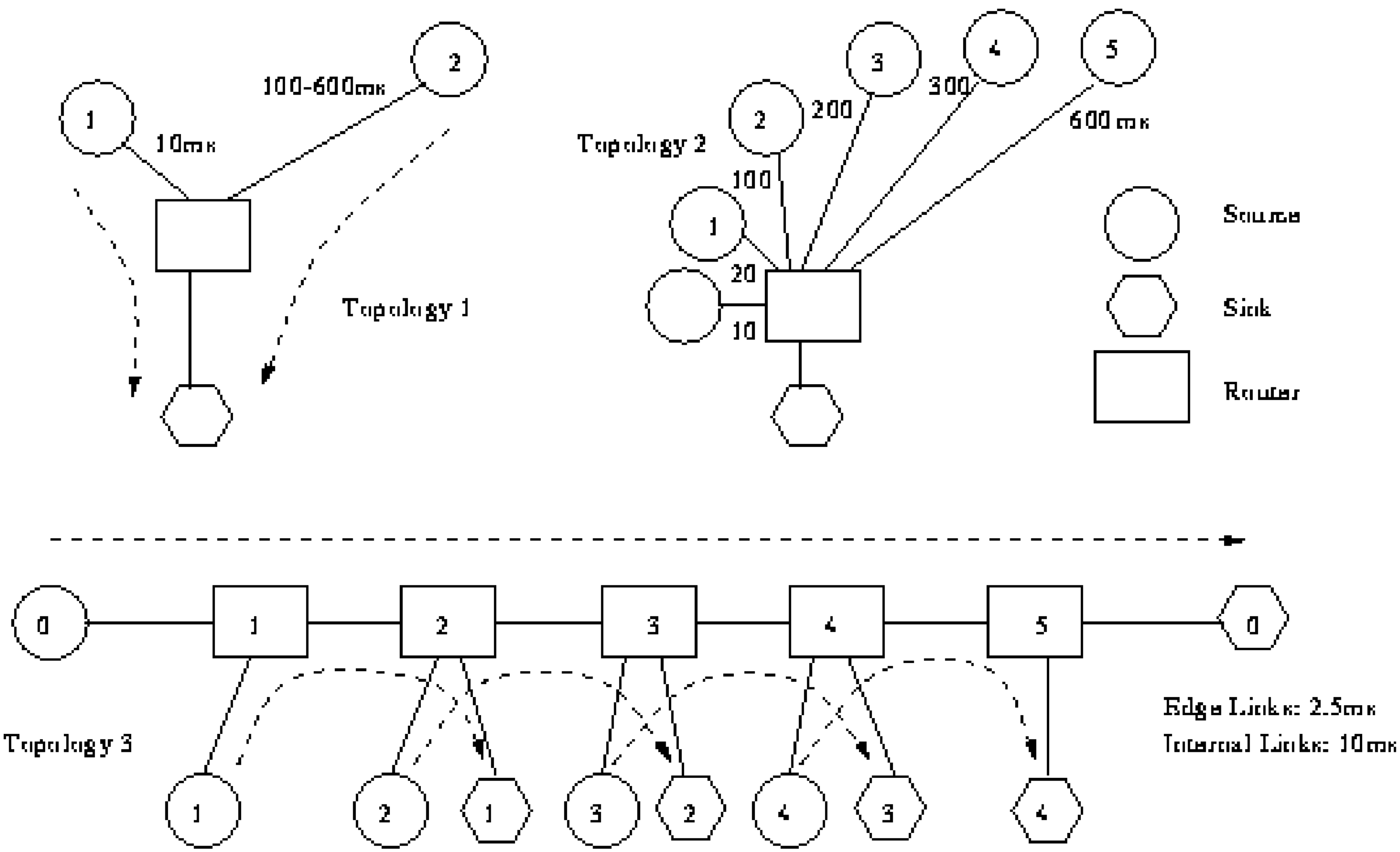
Performance metrics

Two metrics used:

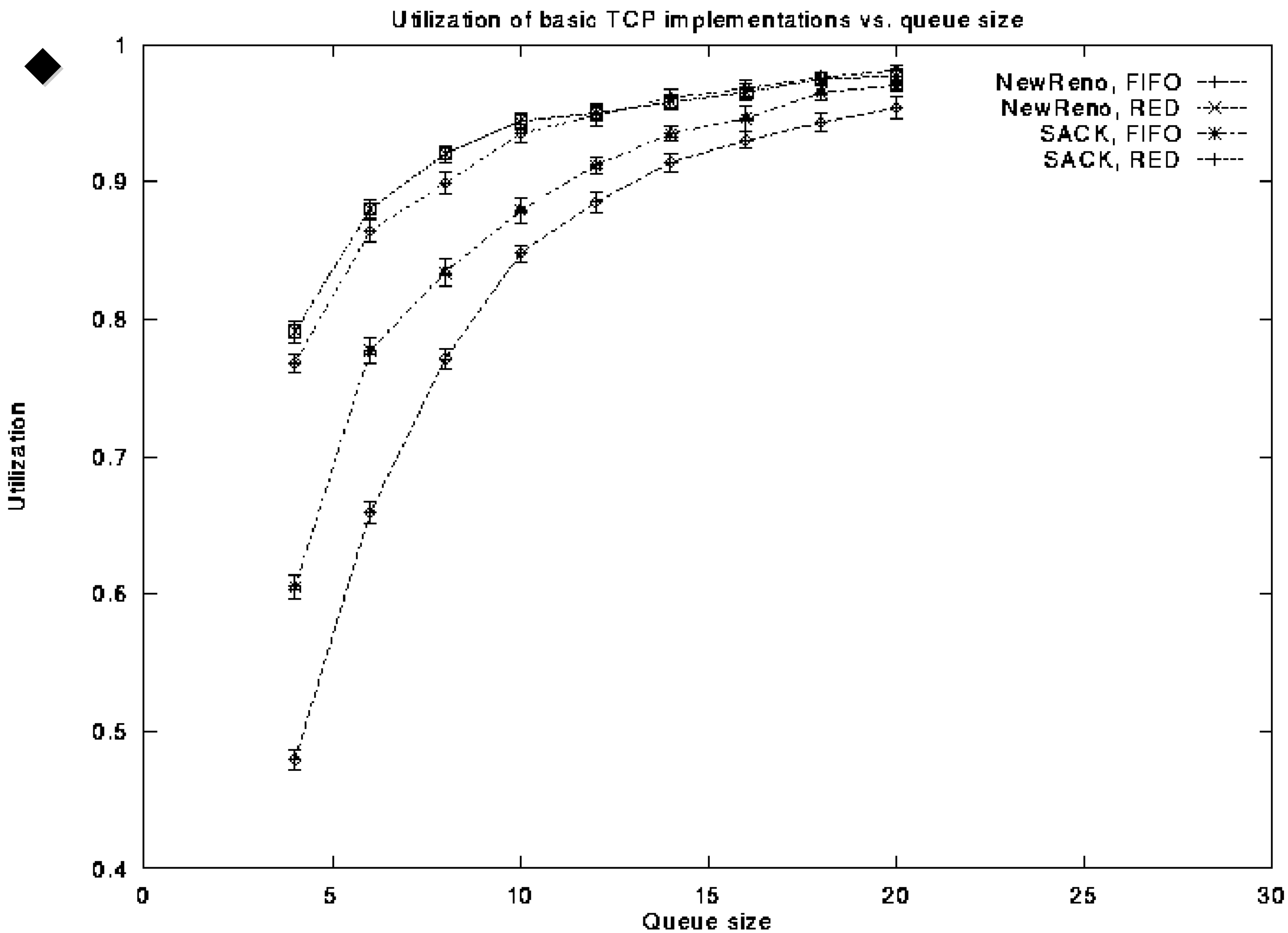
- Utilization of bottleneck link (retransmissions do not count)
- "Jain's fairness" [Chiu and Jain 89]:

$$Fairness = \frac{(\sum_{k=1}^{n} b_i)^2}{n * (\sum_{k=1}^{n} b_i^2)}$$

Network topologies considered

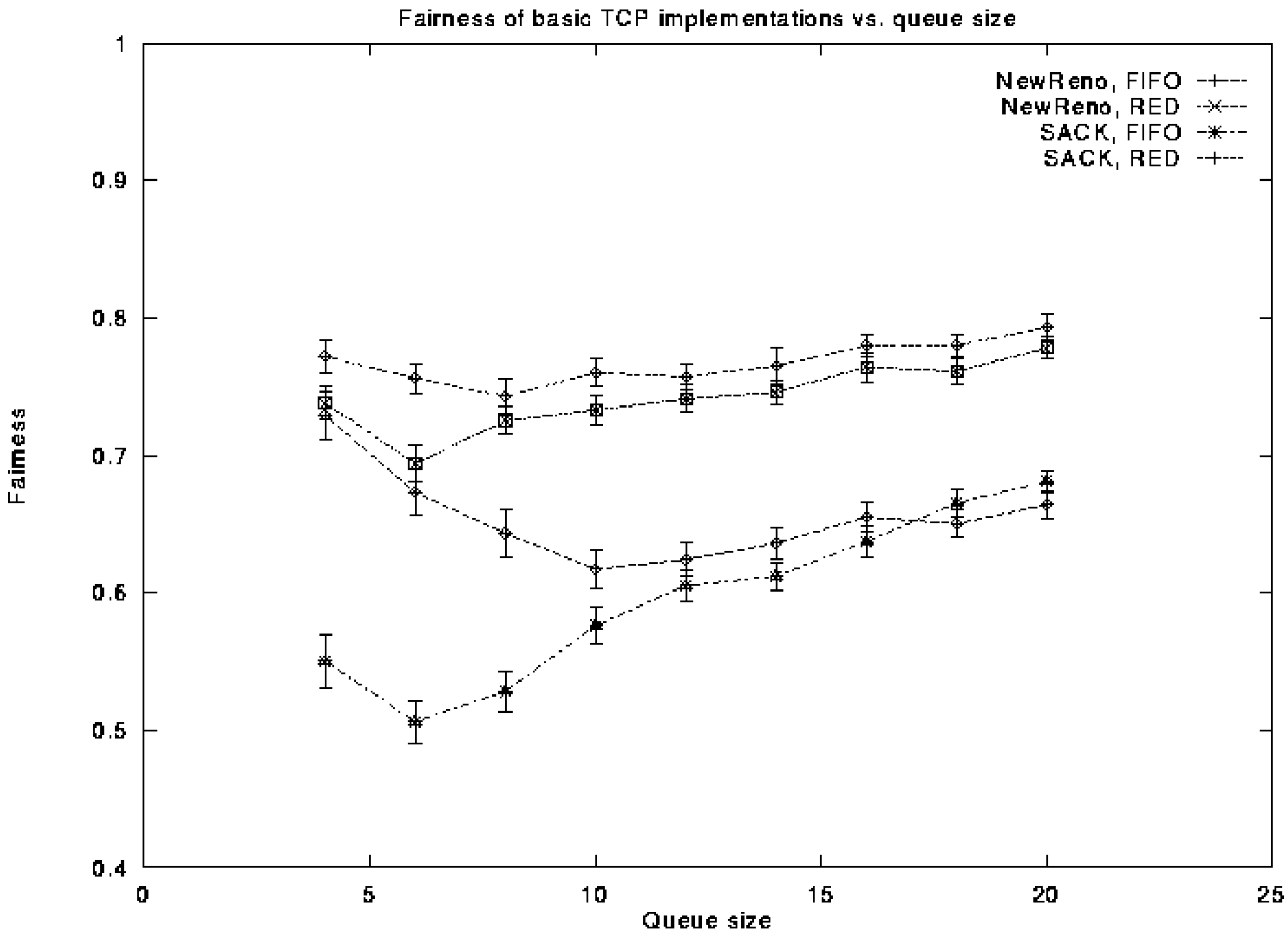


Benchmarks

◆ 

Utilization of basic TCP implementations vs. queue size
NewReno, FIFO
NewReno, RED
SACK, FIFO
SACK, RED
Utilization
Queue size
0
5
10
15
20
25
30
0.4
0.5
0.6
0.7
0.8
0.9
1

Benchmarks (cont.)



Modifications to code:

Our initial attempts: remove the $1/_{rtt^2}$ bias

- Newreno TCP:

```
if (acked_new_data) {
        cwnd += c*rtt*rtt/cwnd
   }
```

- Fack TCP: similar changes, with addition of forward acknowlegment to Sack [Mathis 96]

How to pick constant “c”?

- Initially, we traced actual TCP connections to find a range of constants to evaluate
  - A given value of “c” corresponds to aggressiveness of a particular RTT
- We also obtained better results by bounding aggressiveness by “slow start”
- Code becomes:

```
if (acked_new_data) {
     cwnd += min(c*rtt*rtt/cwnd, 1 segment)
  }
```

Improvements in throughput

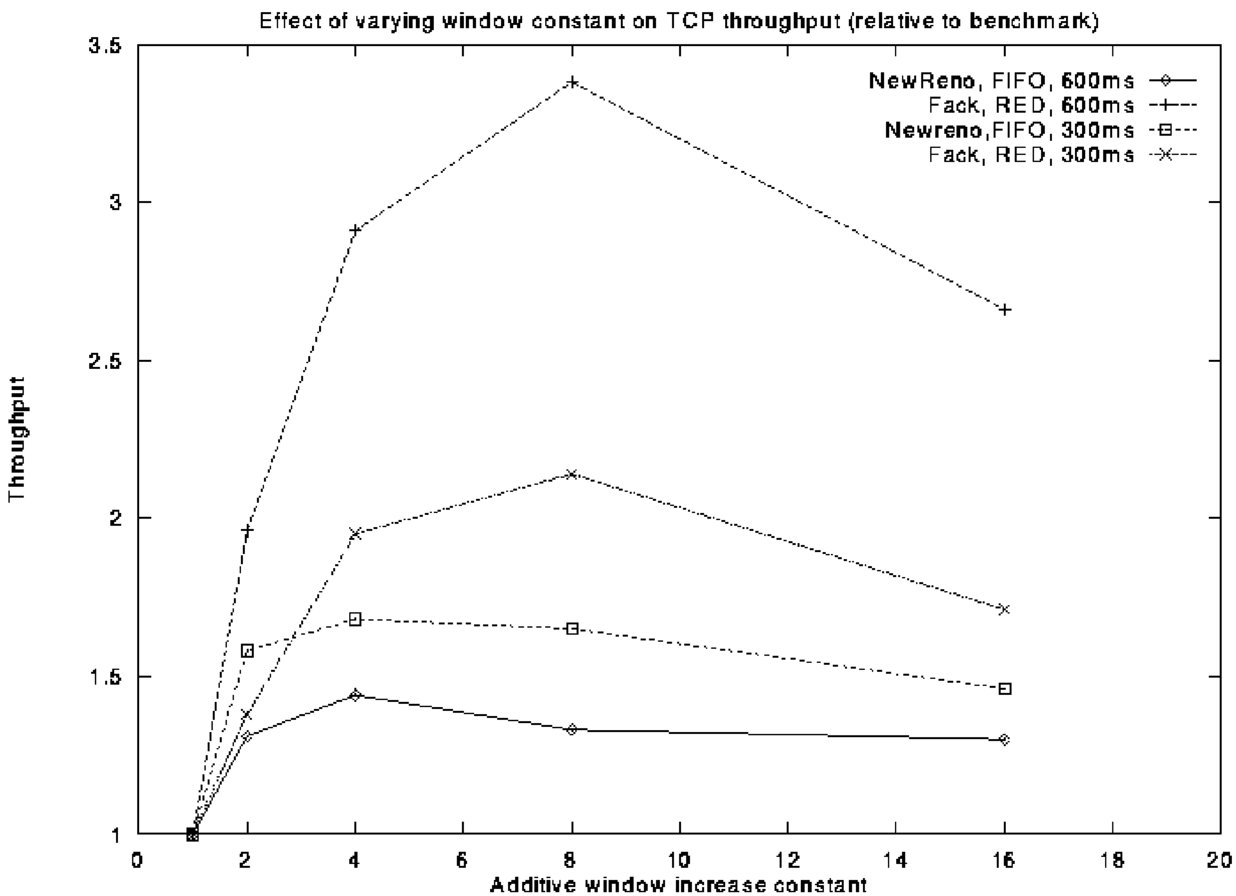
General results:

- Independent of the RTT, a value of $c = 4/rtt^2$ generally works best
- Connections which are overaggressive end up shutting themselves down due to timeouts
- Code becomes:

```
if (acked_new_data) {
      cwnd += min(4/cwnd, 1 segment)
   }
```

Improvements in throughput

Topology 2, relative throughput as a function of window constant



Effects on fairness

General results:

- More aggressive TCPs obtain their extra bandwidth primarily from heavy users
- Overall fairness index <u>always</u> improved in all simulation runs

Effects on fairness (cont.)

Representative results from topology 2

TCP type		Newreno			Sack + Fack	
Queue type	FIFO, length = 10			RED, length = 10		
	benchmark	Modified 300	Modified 600	benchmark	Modified 300	Modified 600
Utilization	0.846	0.832	0.844	0.930	0.926	0.912
Fairness	0.598	0.650	0.612	0.754	0.787	0.870
Throughput (kb/s)						
600ms	**31.6**	33.4	**45.6**	**53.6**	51.5	**156.4**
300ms	**54.2**	**91.1**	54.9	**103.7**	**202.2**	109.1
200ms	91.4	86.5	88.4	134.4	121.8	150.2
100ms	156.6	157.5	155.9	218.2	187	170.1

- **Bold type** indicates connection which was more aggressive.

Experiments

- Internet experiments to multiple sites worldwide
  - 1 MB file sent by two Berkeley machines simultaneously
- Obtained a 20% average throughput improvement with modified TCP (20 runs so far)
- Limitations of experimental method:
  - If no congestion, both connections run full throttle
  - No way to measure fairness

Implementation

- Timer granularity issues (how to accurately measure RTT)
- Receiver modifications to speed up the sender
- Traffic smoothing may yield additional improvements

Conclusions

Fairness can be improved simply by more aggressive sender behavior, but the achievable gains are limited

- Too aggressive TCPs tend to shut themselves down before doing serious damage to other TCP connections
- More aggressive TCPs are not “antisocial” in that they steal most of their bandwidth from other connections
- Overall utilization holds relatively steady