A Comparison of Equation-Based and AIMD Congestion Control

Sally Floyd, Mark Handley, and Jitendra Padhye

Sept. 4-6, 2000
Workshop on the Modeling of Congestion Control Algorithms
Paris

Why look at non-TCP congestion control mechanisms?

- The congestion control mechanisms in TCP are well-suited for applications such as file transfer:
  - with the goal of transferring a file in the shortest possible time;
  - given the restrictions of fairness and the avoidance of congestion collapse.

- Some applications would prefer to avoid TCP's characteristic halving of the congestion window:
  - and would be willing to pay the price of a longer transfer time.

- Not considered:
Applications exempt from end-to-end congestion control.

Candidates for TCP-compatible but slowly-responding congestion control:

- Equation-based congestion control
  - For example, TFRC (TCP-Friendly Rate Control).

- TCP based on AIMD (Additive-Increase, Multiplicative-Decrease)
  - With appropriate increase and decrease parameters a and b.

- Modified variants of TCP (e.g., rate-based).

- Binomial congestion control algorithms that are extensions of AIMD.

- And a number of others...

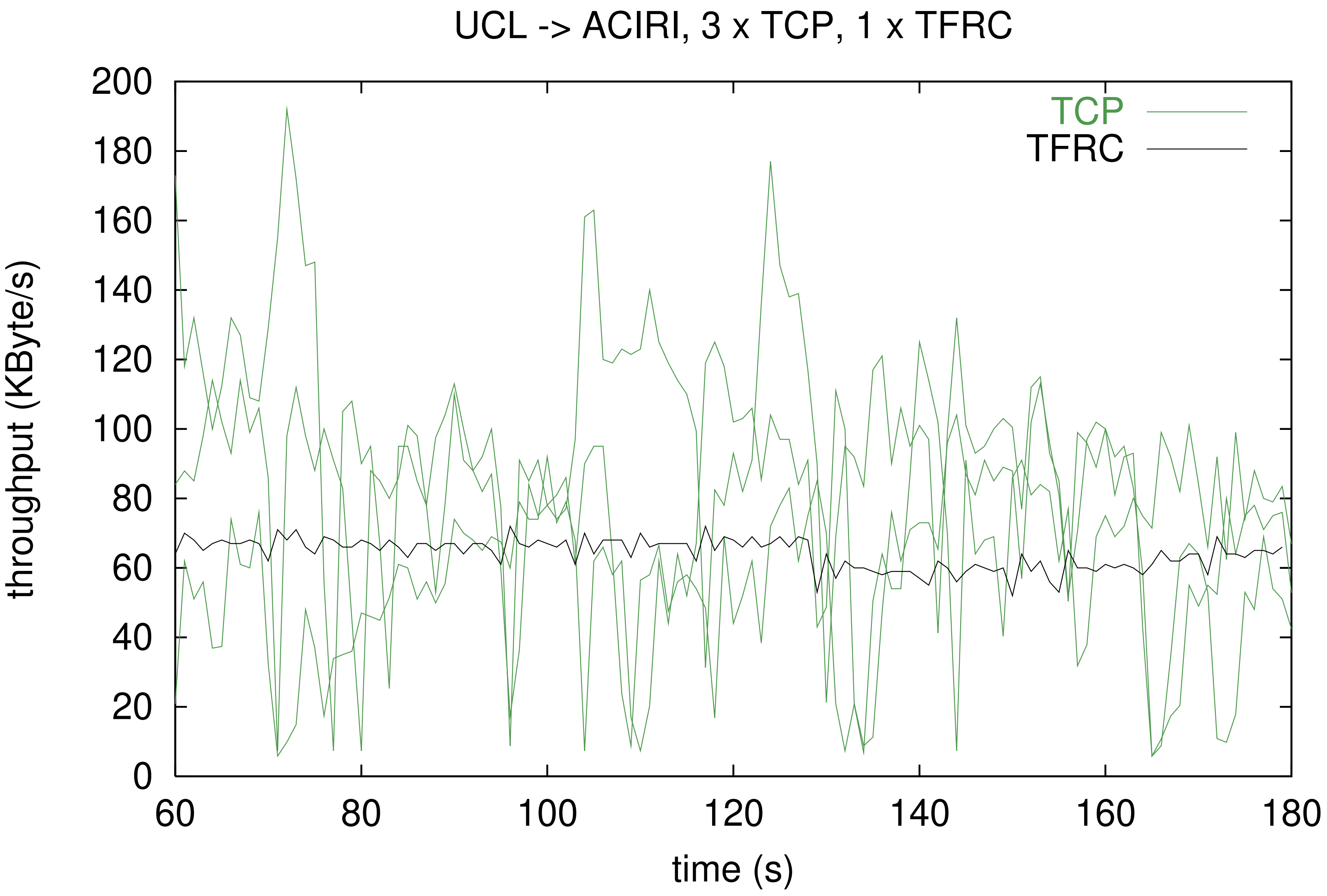
Equation-based congestion control: TFRC

- Uses TCP's equation for the acceptable sending rate as a function of the loss rate and RTT.

- The receiver estimates the loss event rate over the most recent eight loss intervals.

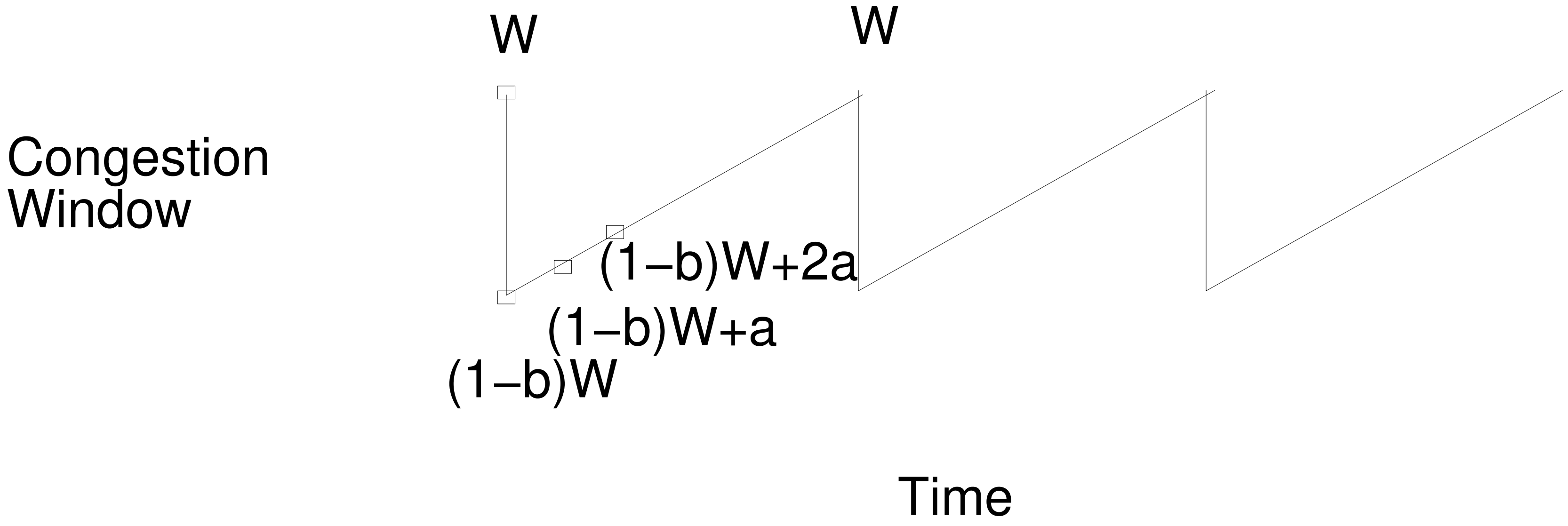
- We believe that TFRC is a viable congestion control scheme for many streaming media applications (many of which don't currently use end-to-end congestion control).

Experiments with TFRC



AIMD(a, b):

- The deterministic steady-state model:



AIMD congestion window in steady-state.

- Decrease window from W to $(1-b)W$.

- Increase window from W to $W+a$ packets each roundtrip time.

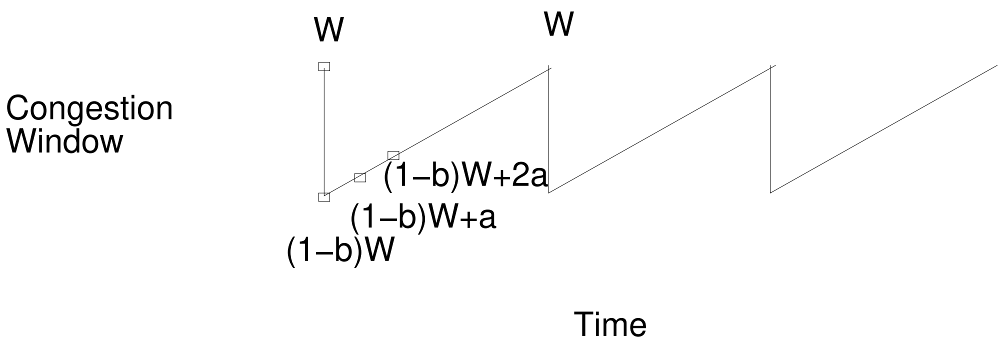
AIMD(a, b), continued:

- The average sending rate S is $(1-b/2)W$ packets per RTT.

- Each cycle has one drop in about $\frac{b(2-b)}{2a}W^2$ packets.

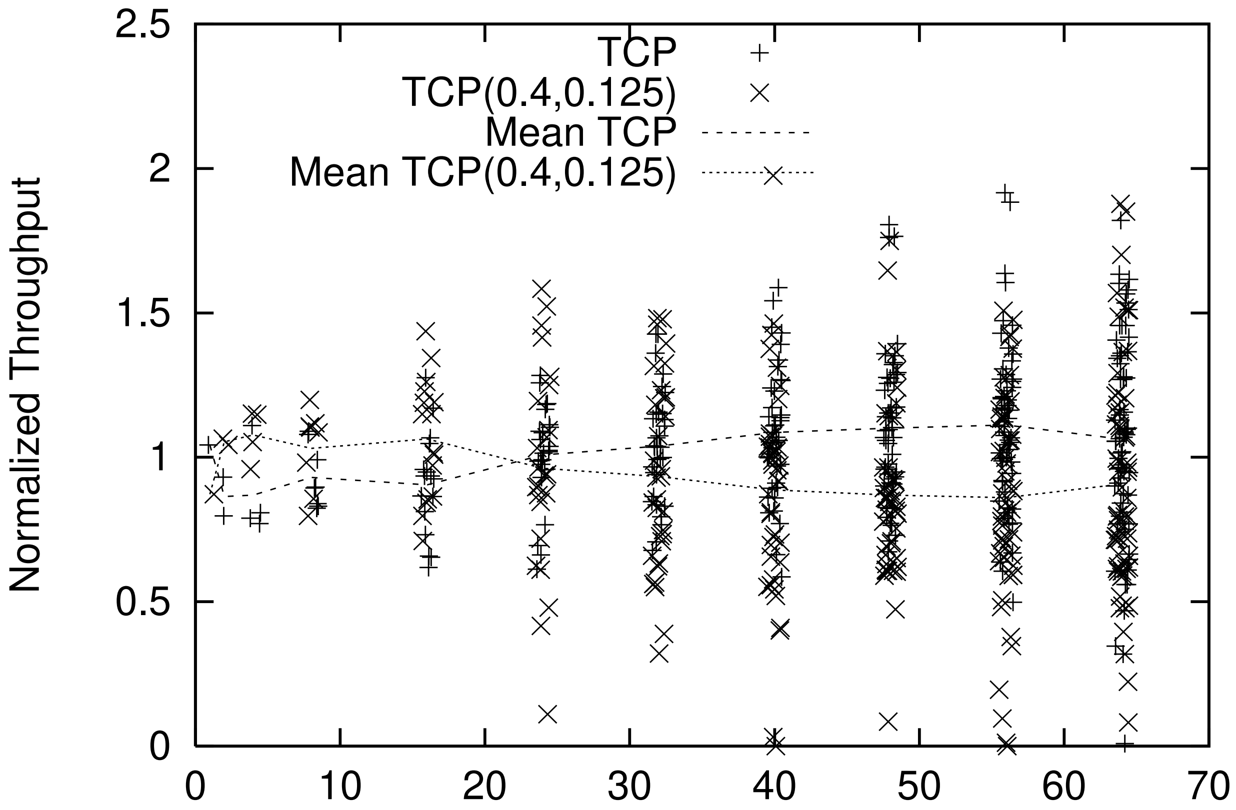
- Therefore, $W \approx \sqrt{\frac{2a}{b(2-b)p}}$, and $S \approx \sqrt{\frac{(2-b)a}{2bp}}$.

- For TCP, $S \approx \sqrt{1.5/p}$.
  – So AIMD(a, b) should compete fairly with AIMD(1, 1/2), in the deterministic model, if $a = \frac{3b}{(2-b)}$.



TCP(1/5, 1/8), or TCP(2/5, 1/8)?

- The previous slide suggests that TCP(1/5, 1/8) should be TCP-compatible, but actually, TCP(2/5, 1/8) looks better in simulations:



Comparing Aggressiveness and Responsiveness:

- Aggressiveness:

  Max increase in sending rate (in packets/RTT) in one RTT.

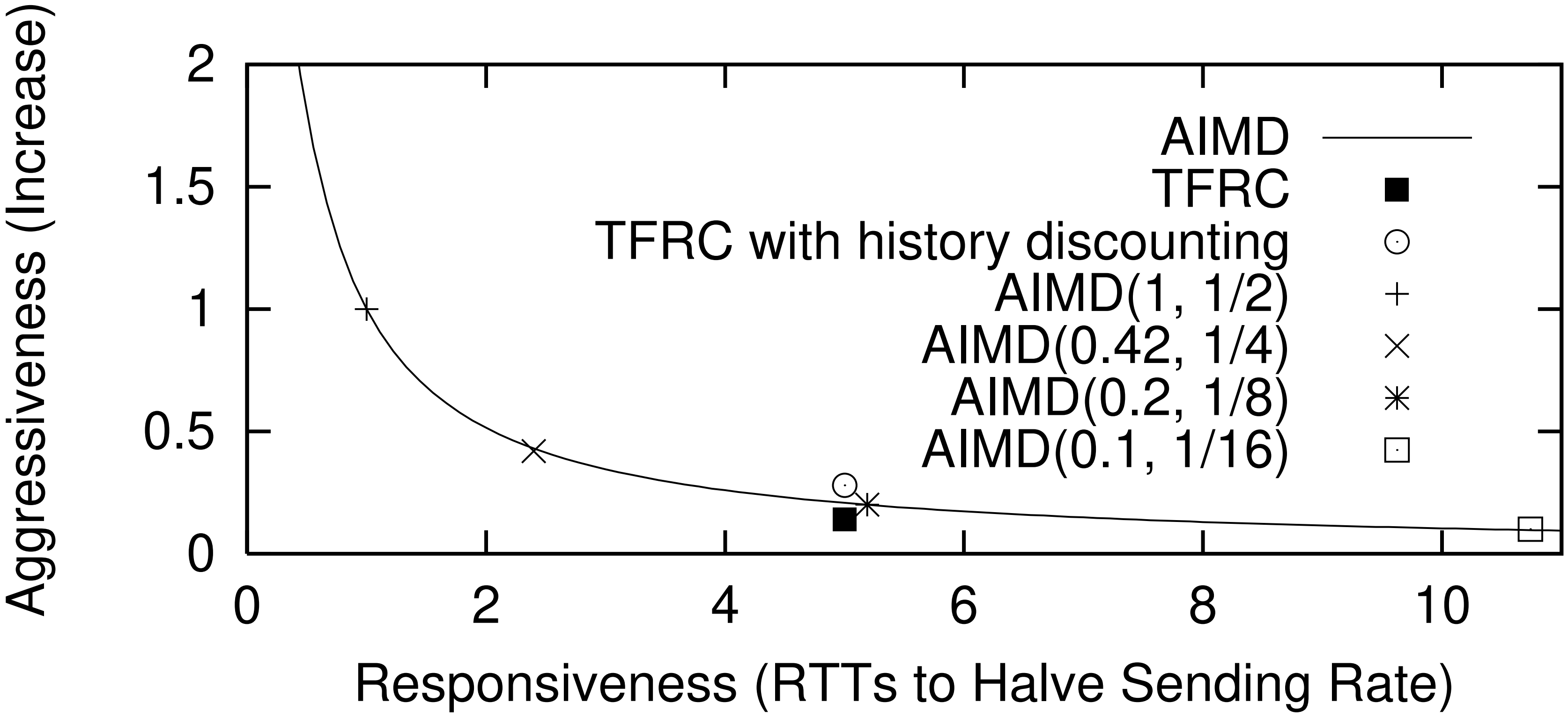
  - AIMD(a, b) or TCP(a, b): a
  - TFRC: 0.14$-$0.28 packets

- Responsiveness:

  RTTs of sustained congestion for halving the sending rate.

  - AIMD(a, b): $\log_{1-b} 0.5$
  - TCP(a, b): in heavy congestion, ack-clocking and retransmit timeouts give quick reductions in the sending rate
  - TFRC: 5 RTTs

Comparing Aggressiveness and Responsiveness, cont.:

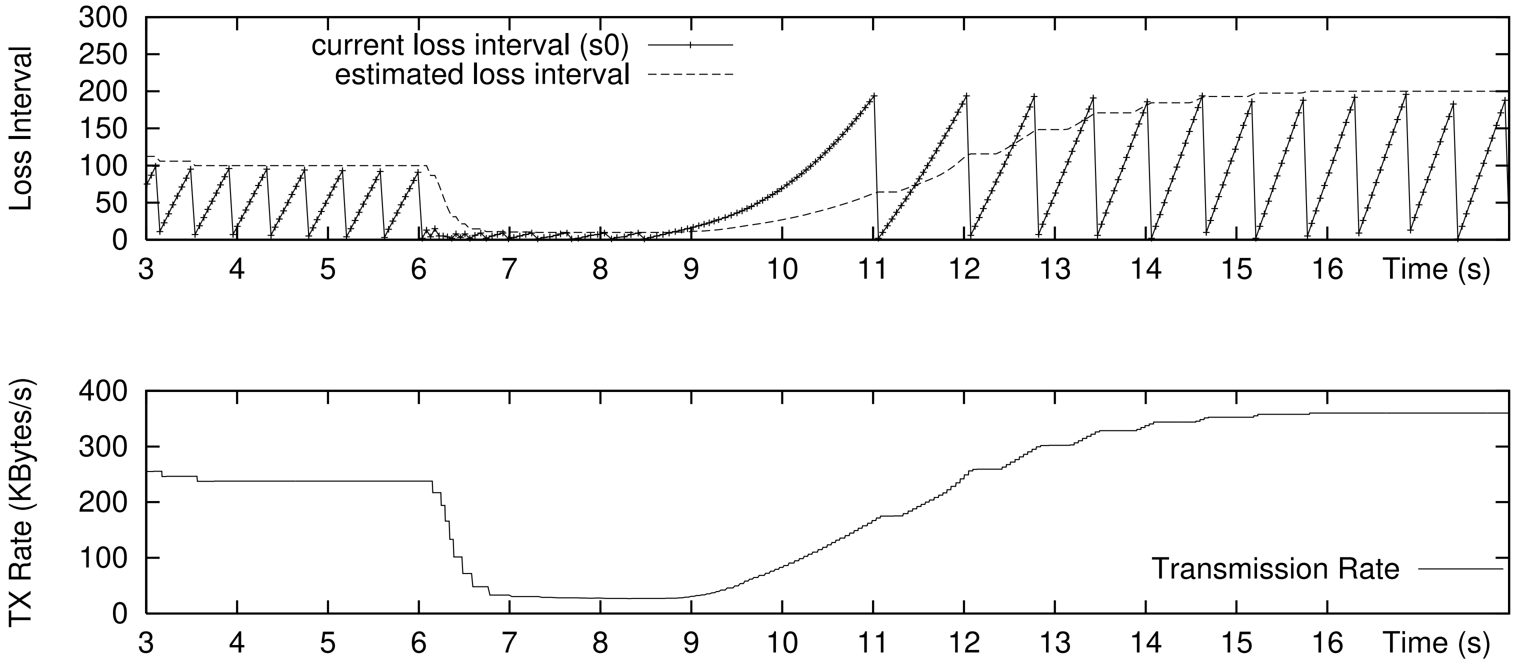


Comparing Smoothness and Responsiveness:

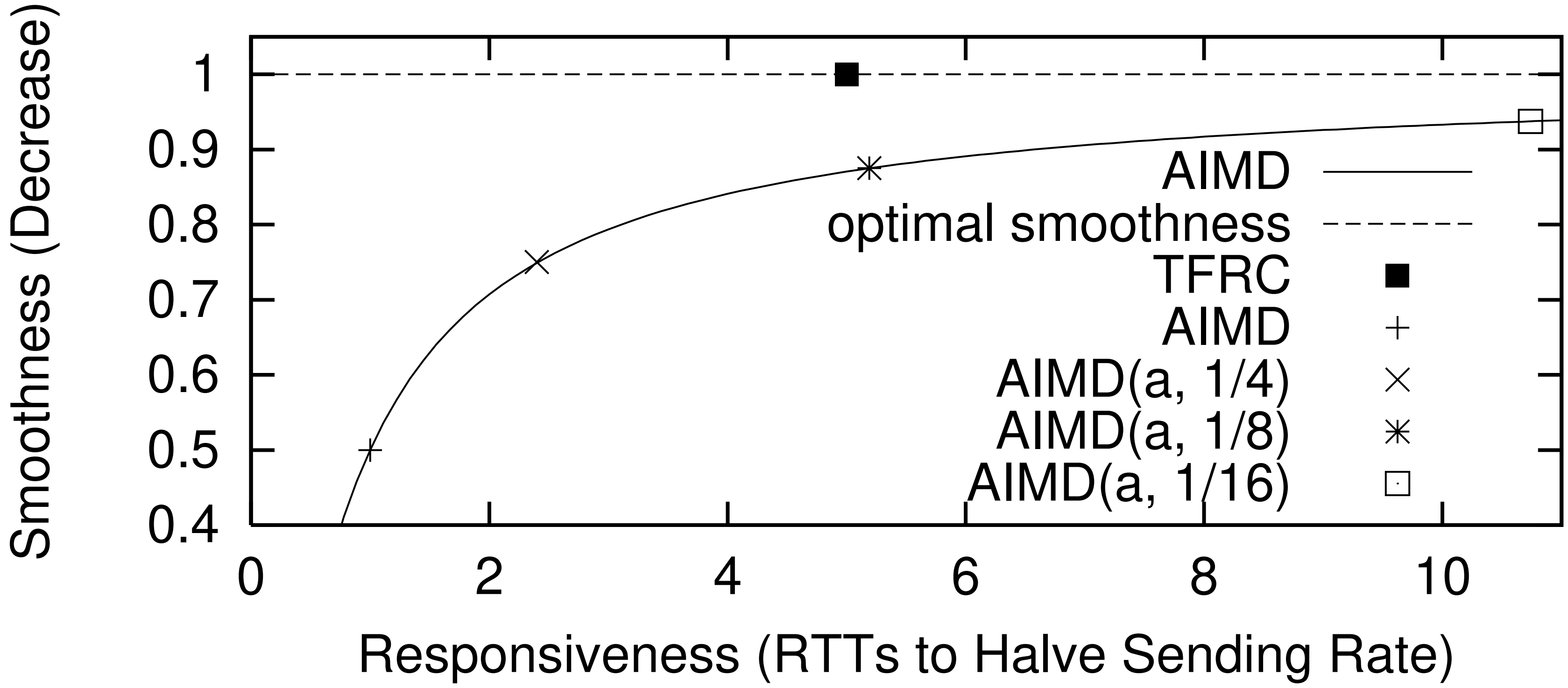
- Smoothness:

Largest reduction of sending rate in one RTT, in the deterministic model.

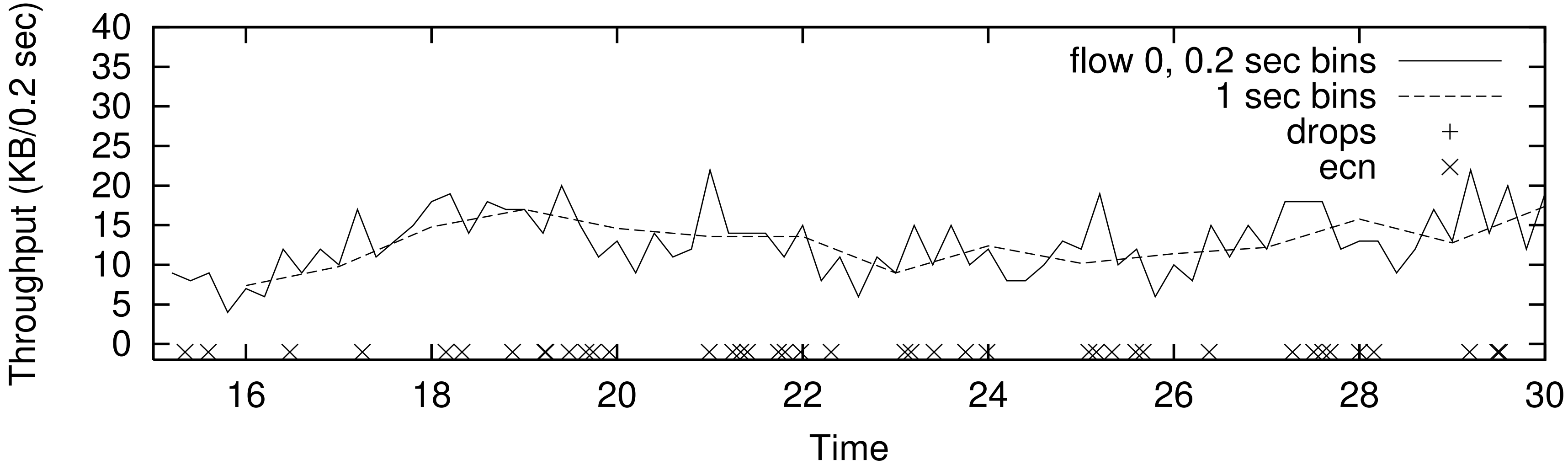
- TCP(a, b): $1 - b$
- TFRC: 1

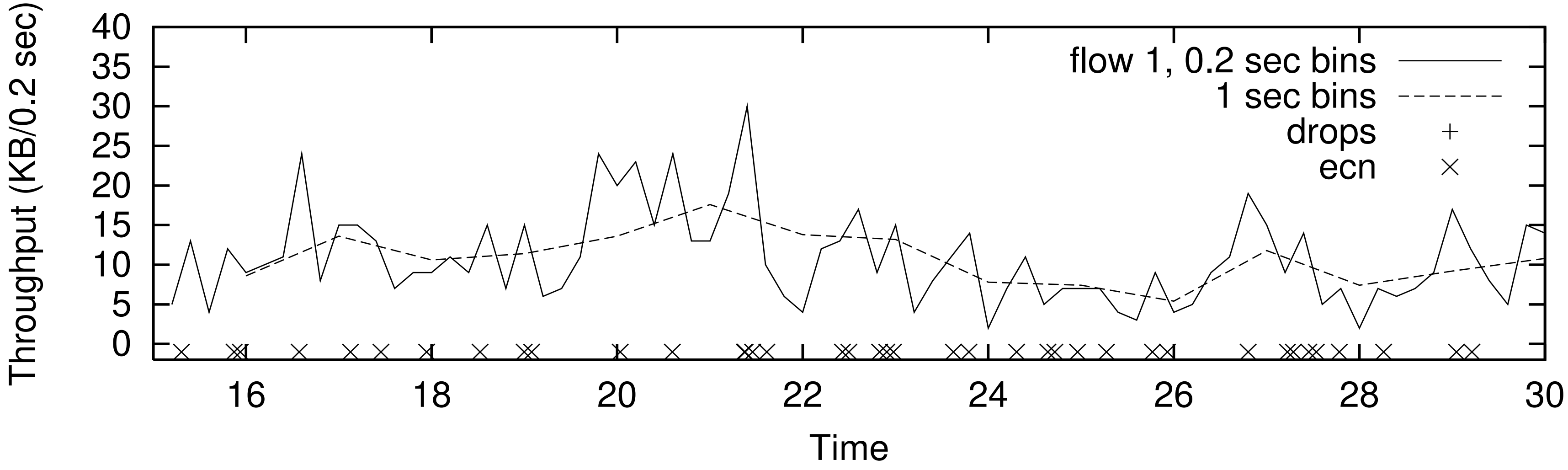


Comparing Smoothness and Responsiveness, etc.:

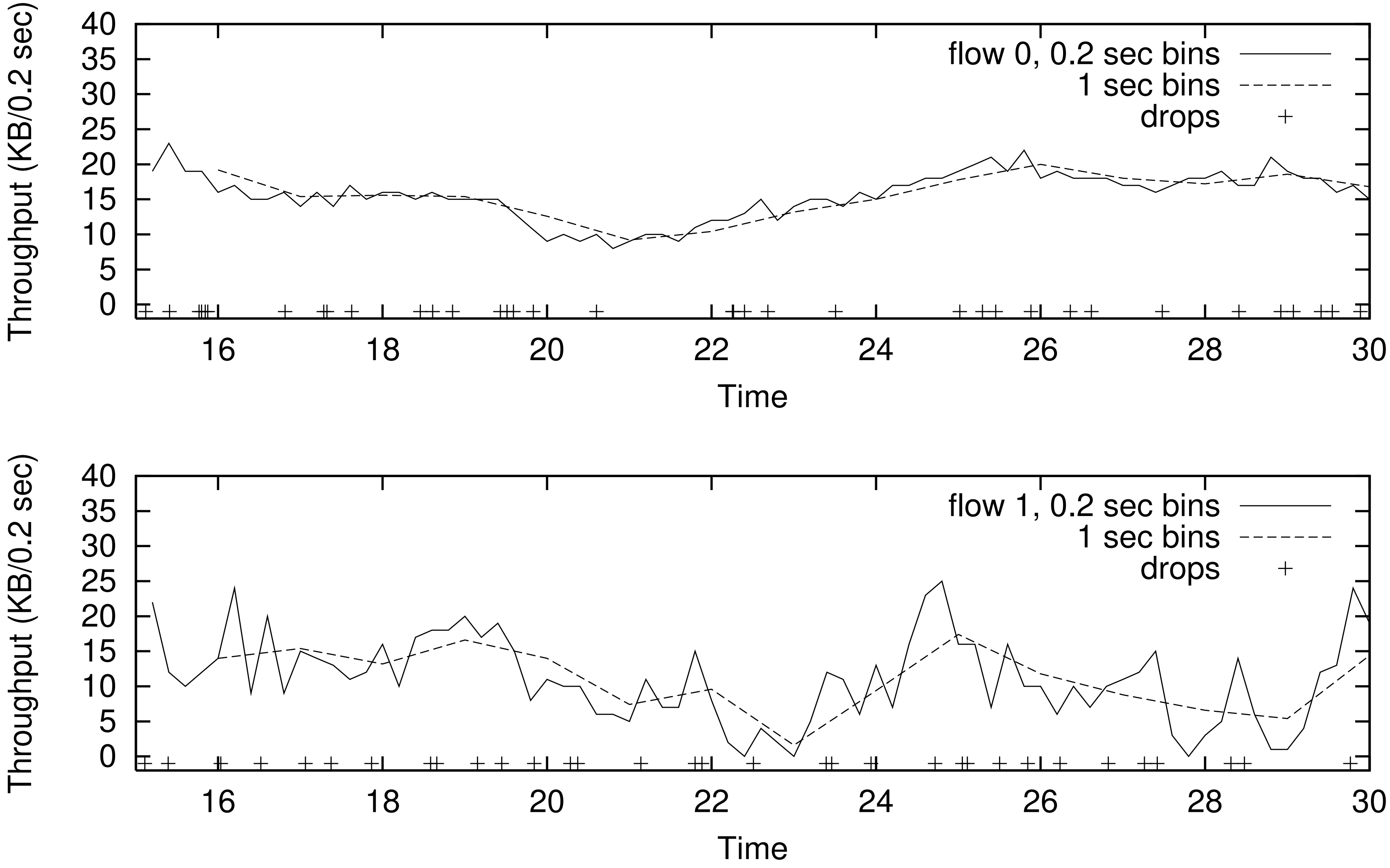


Simulation with TCP[2/5, 1/8] (top) and TCP (bottom)





Simulation with TFRC (top) and TCP (bottom)



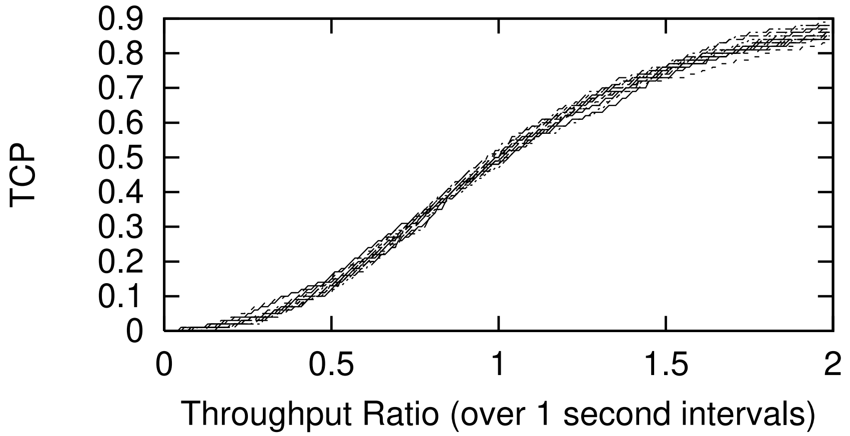
Measuring the throughput ratio

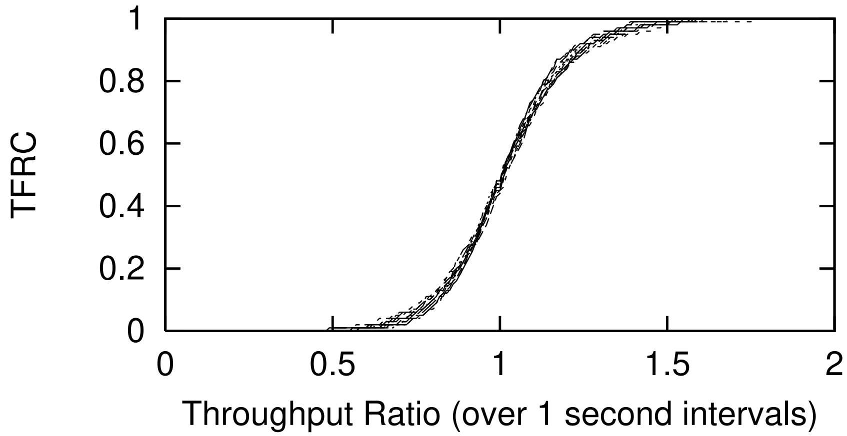
- T_i: the sending rate for a flow over the i-th time interval.

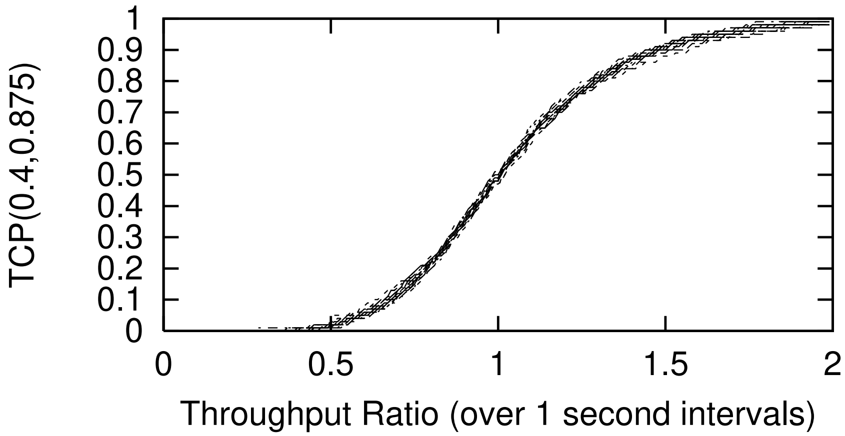
- Throughput ratio for the i-th interval: $\frac{T_i}{T_{i-1}}$

- For a flow, we look at the distribution of throughput ratios, for 0.2-second, 1-second, and 10-second time intervals.

The cumulative distribution of throughput ratios







Summary:
Why are we bothering with TFRC, instead of using TCP(2/5, 1/8)?

- TFRC gives a smoother sending rate than TCP(2/5, 1/8).

- TFRC uses less traffic on the return path than TCP:
  - TFRC uses a feedback message once per RTT.
  - TCP uses ACK packets.

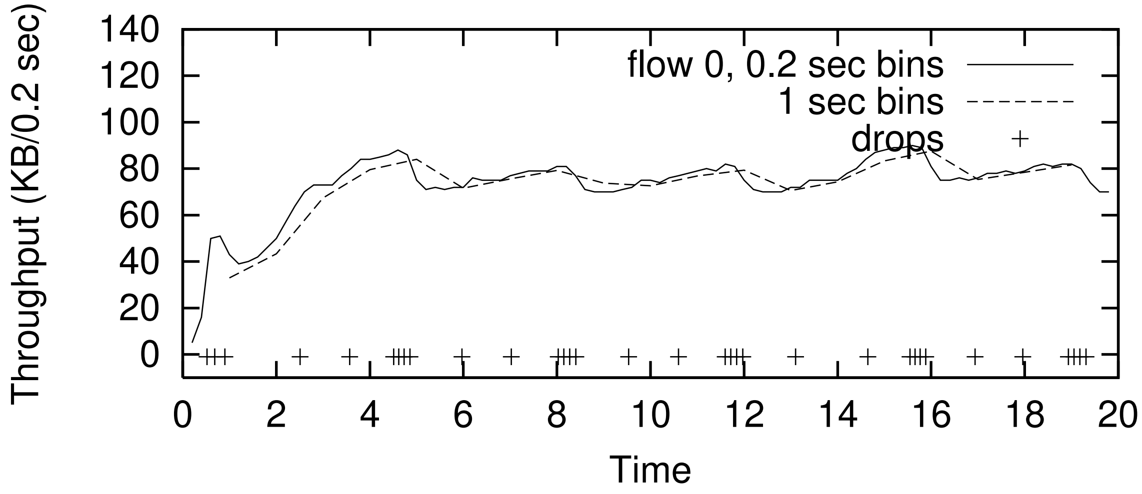
- TFRC is a promising building block for multicast congestion control.
  - Receiver-based.
  - Sender slowly adjusts its sending rate based on feedback from receivers.

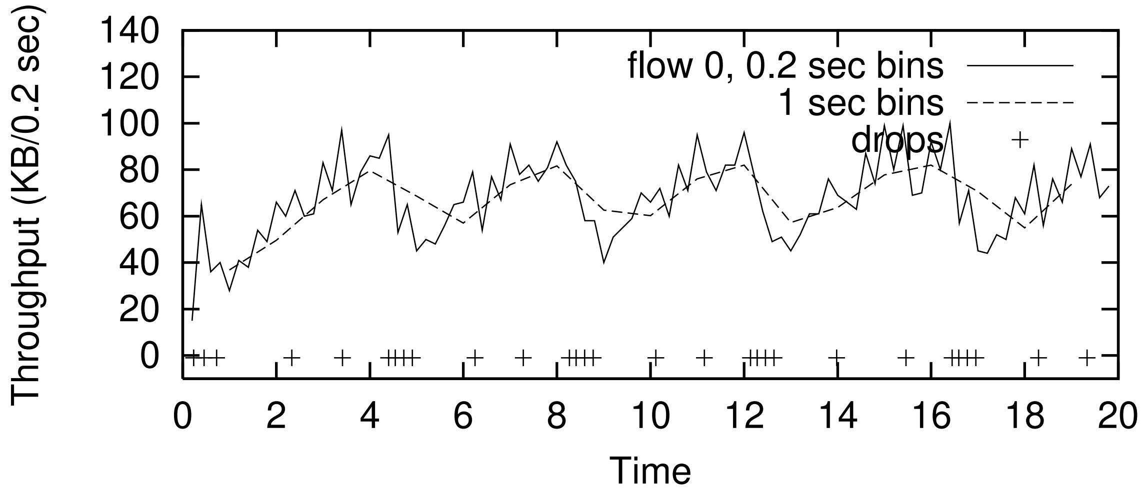
Exploring additional differences between memoryless and memory-based congestion control

- Memory-based congestion control:
  – Equation-based congestion control, such as TFRC, with its memory of past loss intervals.

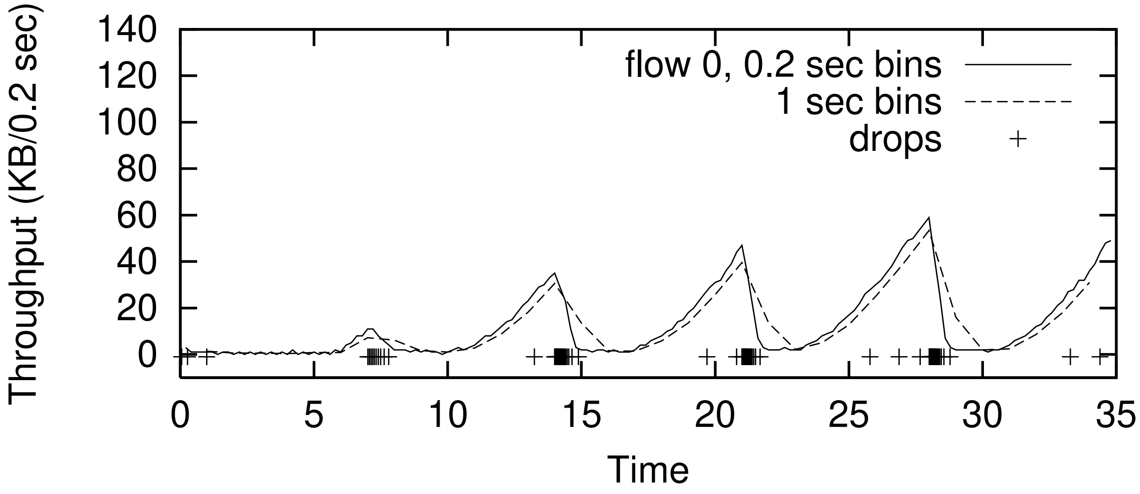
- Memoryless congestion control:
  – TCP, which responds to the presence or absence of congestion in the most recent RTT.
  – During a slow-start, TCP's slow-start threshold ssthresh retains some minimal memory of the previous congestion window.

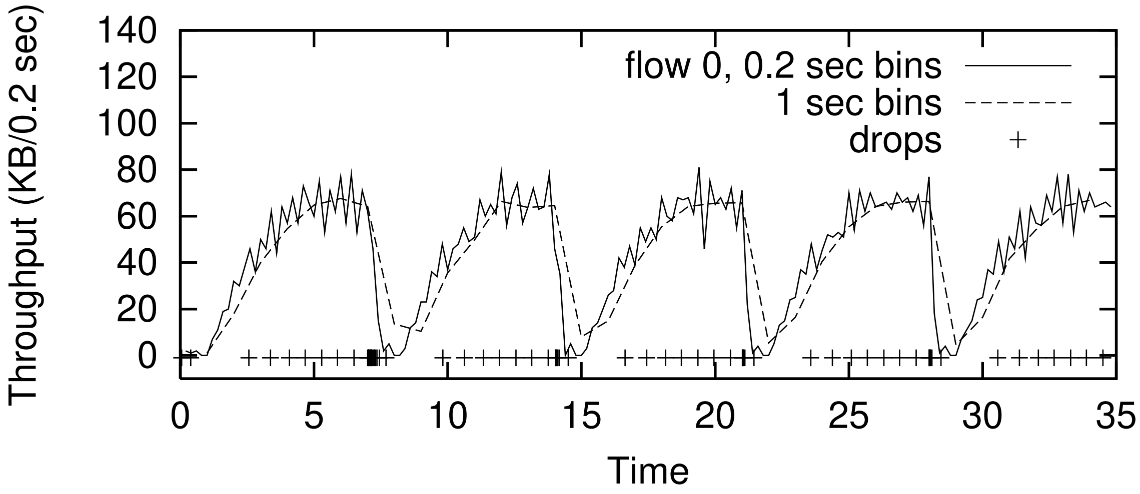
Simulations of TFRC (top) and TCP(2/5, 1/8) (bottom), with packet bursts perfectly adapted to TFRC's memory





Simulations of TFRC (top) and TCP(2/5, 1/8) (bottom), with packet bursts *not* perfectly adapted to TFRC's memory





Concerns about Slow Congestion Control (SlowCC):

A report on work in progress with Deepak Bansal,
Hari Balakrishnan, and Scott Shenker.

For this talk, SlowCC refers only to TCP(2/5, 1/8) or to TFRC.

Concerns about Slow Congestion Control (SlowCC):

• (1) Less aggressive: SlowCC's limited ability to take advantage of an increase in the available bandwidth.

• (2) Less responsive: Is there an extended period of transient congestion, with high packet drop rates, after a sharp decrease in the available bandwidth?

• (3) Fairness with TCP in a changing environment:
  – Long-term fairness with TCP?
  – Transient fairness with TCP?
  – Fairness to competing TCP web mice?

Aggressiveness: SlowCC's limited ability to take advantage of increases in the available bandwidth

- Assume that the bandwidth available to a flow is doubled.

- For TCP(a,b), the sending rate is increased by:
  – a/R packets/sec each RTT,
  – a/R^2 packets/sec each second.

- Therefore, for TCP(a,b) in congestion avoidance with a sending rate of λ pkts/sec, it takes $\lambda R^2/a$ seconds to double its sending rate.

- This "cost" of SlowCC cannot be improved by standard scheduling or queue management mechanisms at the routers.

Responsiveness: SlowCC's slow decrease in response to congestion

• In mild but persistent congestion (e.g., with ECN), TCP(a, b) decreases its sending rate by a multiplicative factor $(1 - b)$ each RTT.

• In severe congestion, TCP(a, b) decreases quickly, as a result of both ack-clocking and retransmit timeouts.

• For equation-based congestion control, the decrease rate is determined by the method for estimating the loss event rate.

• For TFRC, which averages over a number of loss intervals, the decrease rate is determined by the number of loss intervals in the average (8).

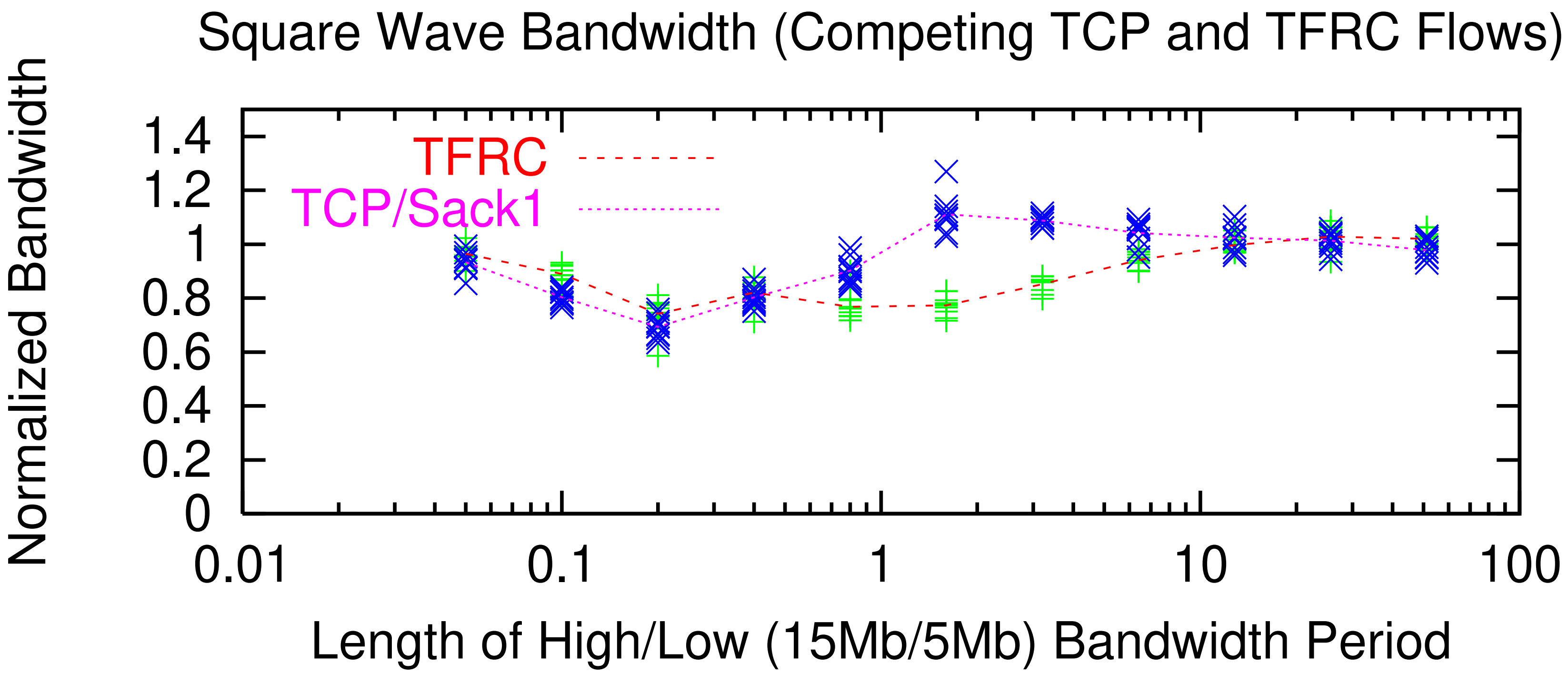
– For heavy congestion, TFRC's "ack-clocking-emulation" is the dominant mechanism determining the decrease rate.

Next: fairness results

- Long-term fairness with TCP in a changing environment?
- Transient fairness with TCP in a changing environment?
- Fairness to competing TCP web mice in a changing environment?

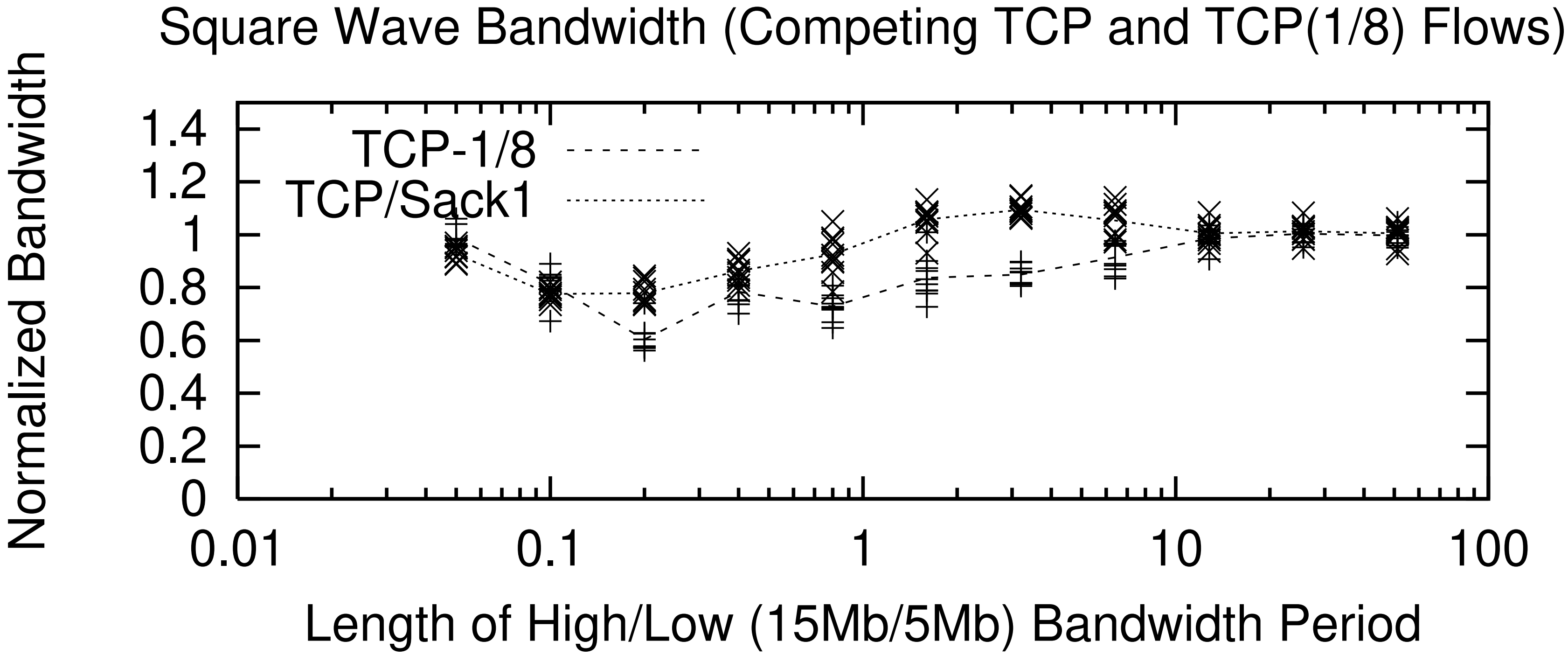
Summary: We could find no fairness reasons for not deploying SlowCC (that is, TCP(2/5, 1/8) or TFRC) in the current Internet.

SlowCC's long-term fairness with TCP in changing environments:



Simulations of TCP competing with TFRC.

SlowCC's long-term fairness with TCP in changing environments:



Simulations of TCP competing with TCP(2/5, 1/8).

Transient fairness with TCP in a changing environment:

- After a sudden doubling in the available bandwidth, TCP initially gets more than its share of the available bandwidth, when competing with TFRC or TCP(2/5, 1/8).

- When the available bandwidth returns to its initial value, the return to fairness is fairly prompt.

Fairness to competing TCP web mice in a changing environment: Results from current simulations

• TCP web mice competing with TFRC or TCP(2/5, 1/8) do no worse than web mice competing with TCP.

• TCP flows starting up against an existing TFRC or TCP(2/5, 1/8) flow quickly achieve their share of the link bandwidth.

Related work in progress:

- What about *very* slow congestion control, e.g., TCP(a, 1/256), or TFRC with a loss event rate measured over 256 loss intervals instead of 8?

  – These *very* slow congestion control mechanisms have significant costs in terms of transient fairness, and in terms of slowness in taking advantage of newly-available bandwidth.