Nonlinear Congestion Control

Deepak Bansal & Hari Balakrishnan
MIT Lab for Computer Science
http://nms.lcs.mit.edu/

End-to-end congestion control

Approach 1: I/D.

- Maintain window (rate) and control it.
  Probe for more capacity:
  $$w \leftarrow I(w);\ \text{e.g., } w+1, 2w, \text{ etc.}$$
  React to congestion:
  $$w \leftarrow D(w);\ \text{e.g., } w/2.$$
- I(w) and D(w) are usually *linear*.
- Loss *indication* is tracked.

Approach 2: Loss-rate monitoring.

- Monitor loss rate p.
  $$rate \leftarrow f(p).$$
- Loss *rate* is tracked.
- E.g., equation-based congestion control (EBCC; [Floyd *et al.* 2000]).

TCP-friendliness

- For any congestion control algorithm, C, let λ be the end-to-end throughput and p its loss rate (#losses/#sent), with multiple losses in the same RTT treated as one.

  If $\lambda \approx K/\sqrt{p}$, then C is TCP-friendly.

- Approach 2 (loss-rate monitoring) is interesting because:
  - Sources can avoid drastic reductions in transmission rate.
  - It can be TCP-friendly (EBCC).

- What about Approach 1?

Questions

- Can we generalize linear controls to understand the I/D trade-offs better?
  - Probing aggressiveness vs. reacting responsiveness.
- What are the properties of this generalization?
  - E.g., Can it be stable? TCP-friendly? Fair to TCP?
- How do different (TCP-friendly) I/D algorithms interact with each other when competing for resources?

Binomial algorithms

- AI: $w(t+R) = w(t) + \alpha; \alpha > 0$
  MD: $w(t+\delta t) = w(t) - \beta w(t); 0 < \beta < 1.$
  Here, α is independent of $w(t)$.

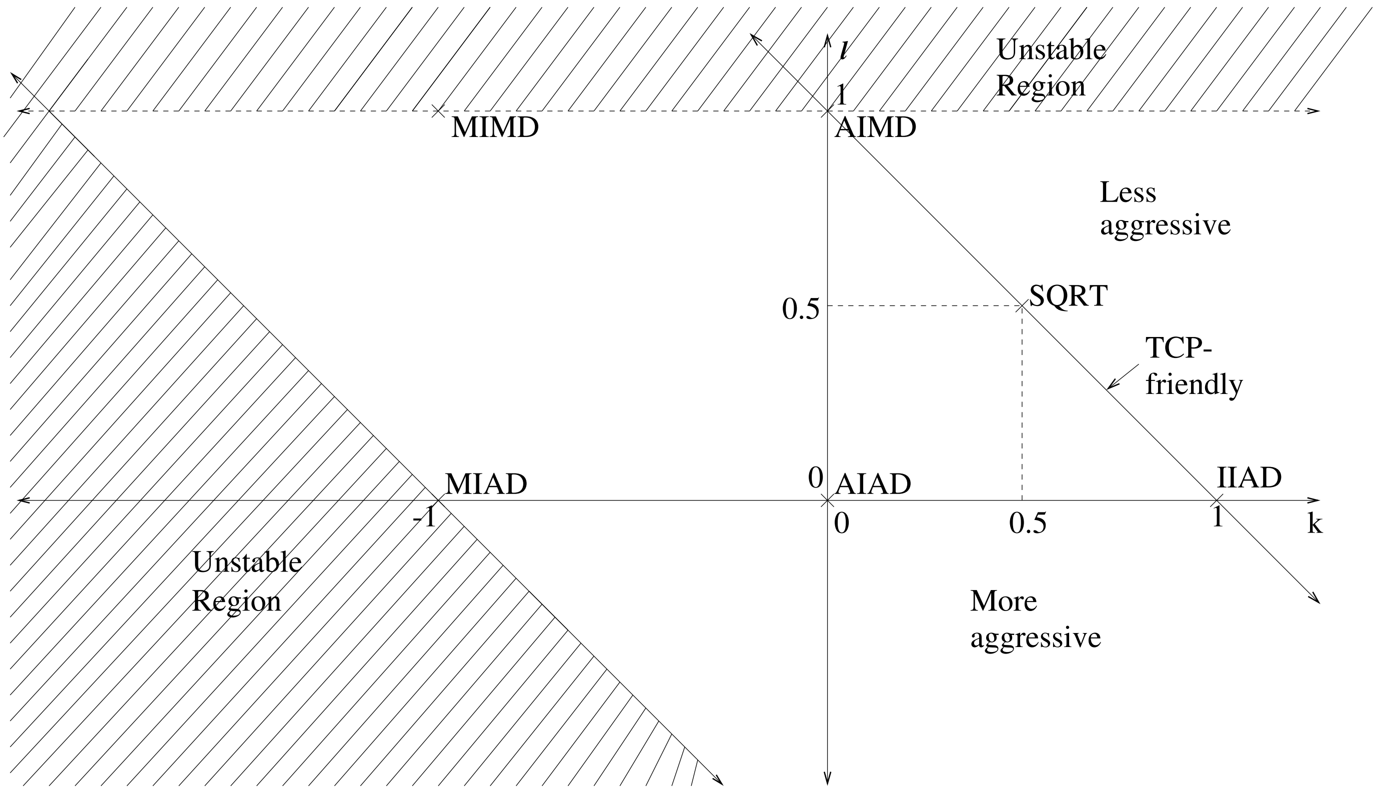
- Consider:
  I: $w(t+R) = w(t) + \alpha / w(t)^k; \alpha > 0$
  D: $w(t+\delta t) = w(t) - \beta w(t)^l; 0 < \beta < 1.$

- When $k > 0$, the larger the window, the smaller the relative increase.

- Binomial algorithms generalize linear ones.

$$
\begin{aligned}
(k=0, l=1) &\rightarrow \text{AIMD} \\
(k=-1, l=1) &\rightarrow \text{MIMD} \\
(k=0, l=0) &\rightarrow \text{AIAD} \\
(k=-1, l=0) &\rightarrow \text{MIAD}
\end{aligned}
$$

The (k, l) space



- Recap: increase by α / w^k; decrease by βw^l.
- TCP-friendly if $k + l = 1$.
- $k + l > 1 \Rightarrow$ less aggressive than AIMD.
  $k + l < 1 \Rightarrow$ more aggressive than AIMD.
  $k + l \leq -1$ or $l > 1 \Rightarrow$ unstable.

Examples

- <u>IIAD:</u> Inverse-increase, additive-decrease.
  - Here, $k = 1, l = 0$, i.e.,
    I: $w(t + R) = w(t) + \alpha / w(t)$,
    D: $w(t + \delta t) = w(t) - \beta$.

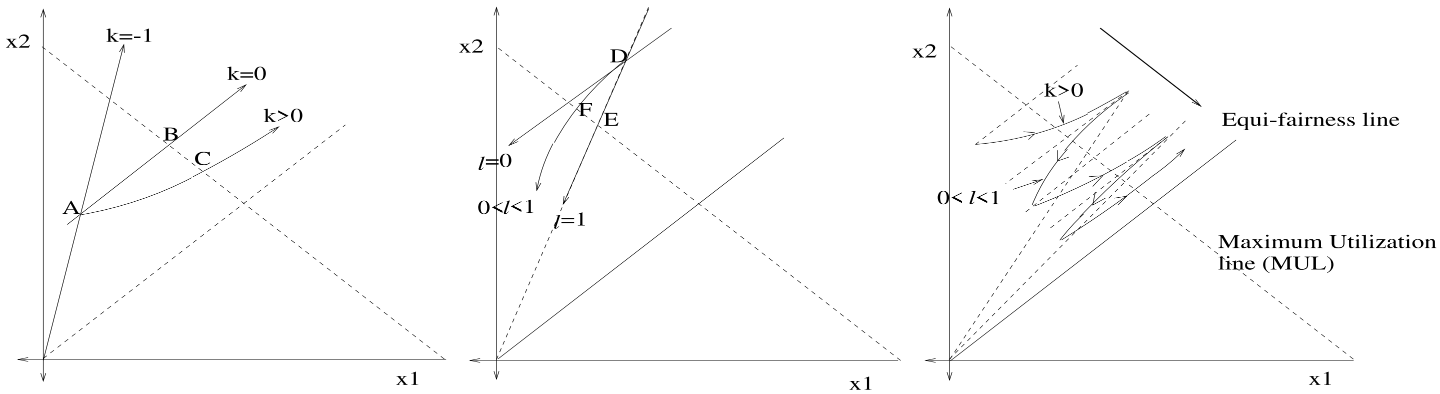
- <u>SQRT:</u> Square-root.
  - Here, $k = l = 1/2$, i.e.,
    I: $w(t + R) = w(t) + \alpha / \sqrt{w(t)}$,
    D: $w(t + \delta t) = w(t) - \beta \sqrt{w(t)}$.

- Window reductions on congestion are small; TCP-friendly because probing increases are small as well.

More properties

- If k, l are picked in the first quadrant of the (k, l)-space, then competing binomial algorithms with the same k, l converge to fairness.
- For a binomial algorithm, $\lambda \propto 1/p^{\frac{1}{k+l+1}}$.
- α, β can be found such that binomial algorithms compete fairly with each other, if the network equalizes drop-rates.
- Over drop-tail gateways, binomial algorithms do not compete fairly with each other.

“Phase plot” evolution



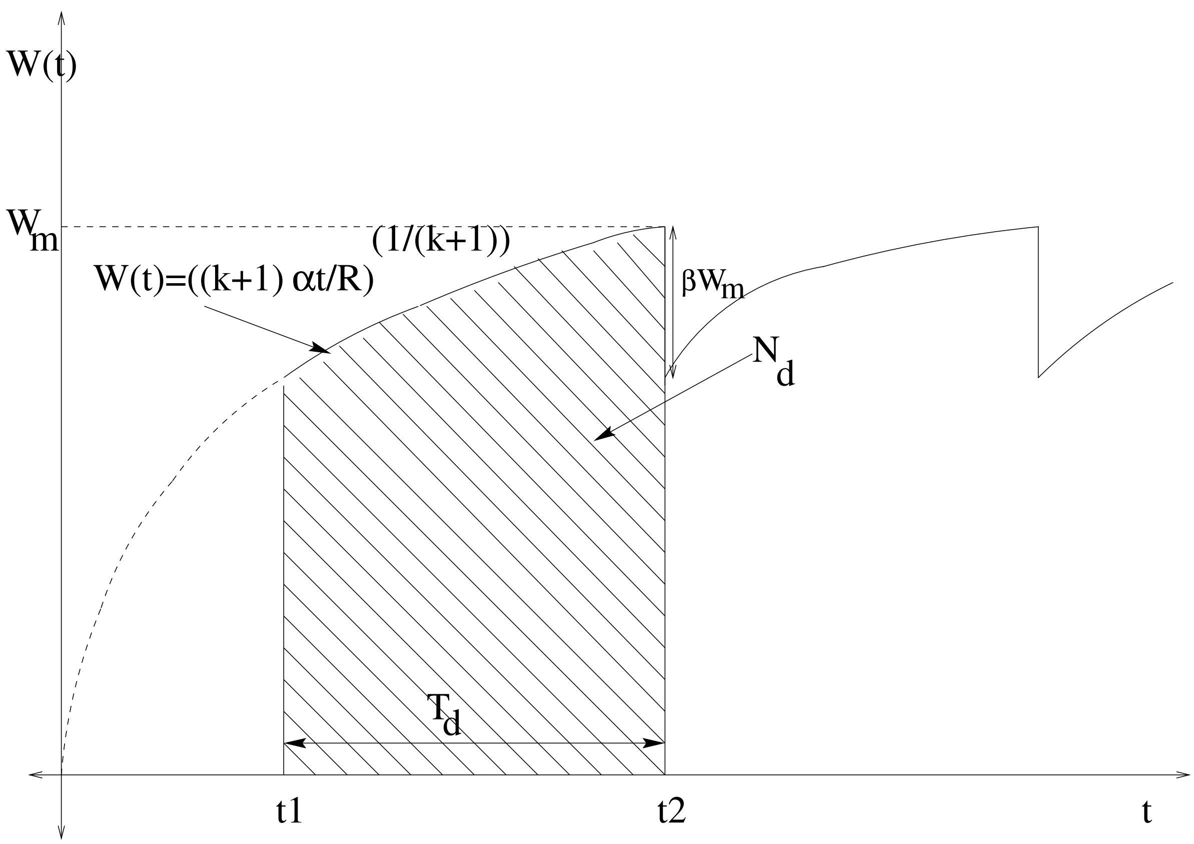
- Additive-increase improves fairness; multiplicative-decrease preserves fairness.

- A binomial algorithm does converge under a synchronized-feedback assumption if
  $k, l \geq 0$ and $k + l > 0$.

- Open question: model for demonstrating convergence under desynchronized feedback.

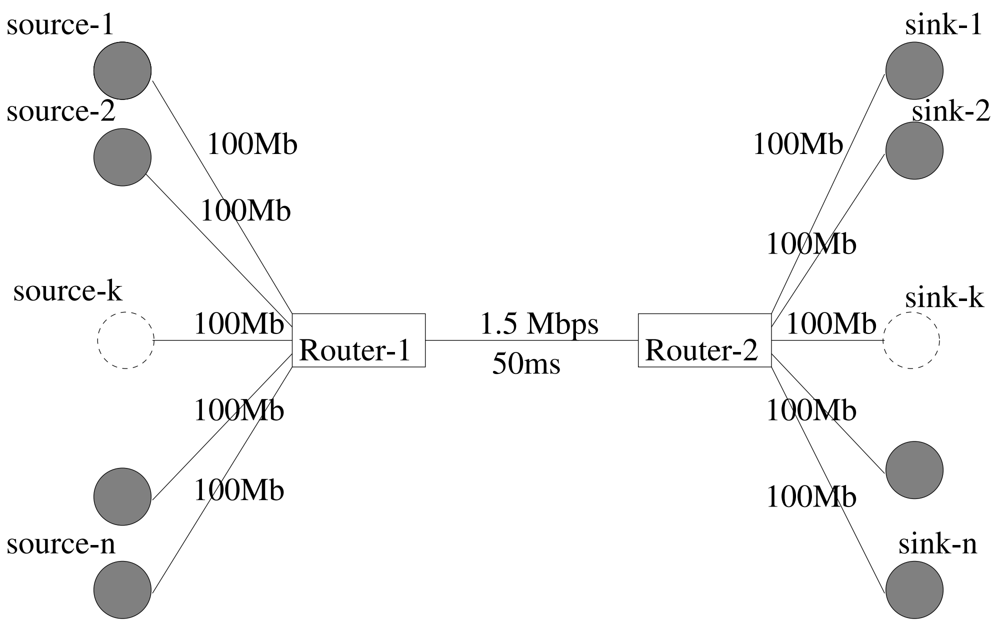
$\lambda \propto 1/p^{\frac{1}{k+l+1}}$

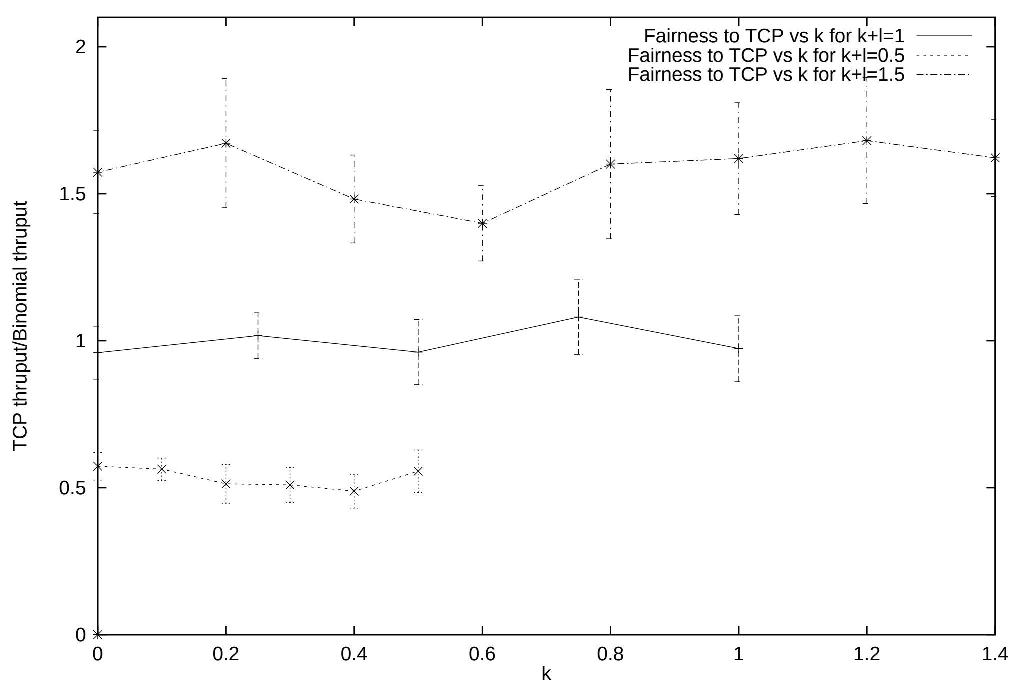
- Under a fluid approximation, $\frac{dw}{dt} = \frac{\alpha}{w^k.R}$
  $\Rightarrow \frac{w^{k+1}}{k+1} = \frac{\alpha t}{R} + C$



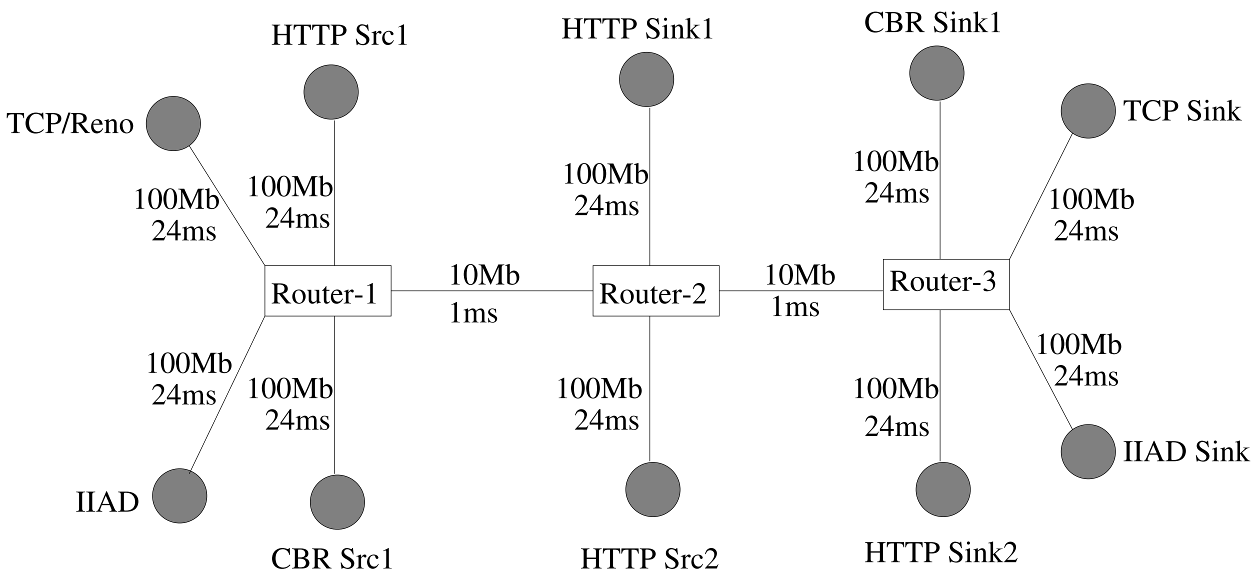
- $\lambda \approx (\frac{\alpha}{\beta})^{1/(k+l+1)} \frac{1}{Rp^{1/(k+l+1)}}$

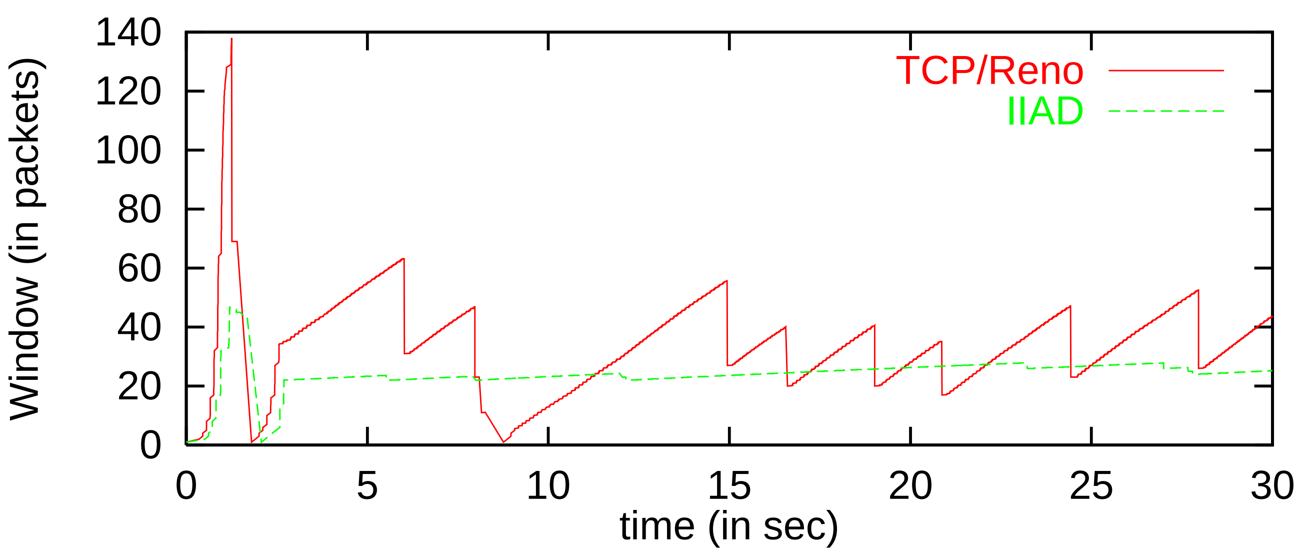
Co-existence with TCP AIMD



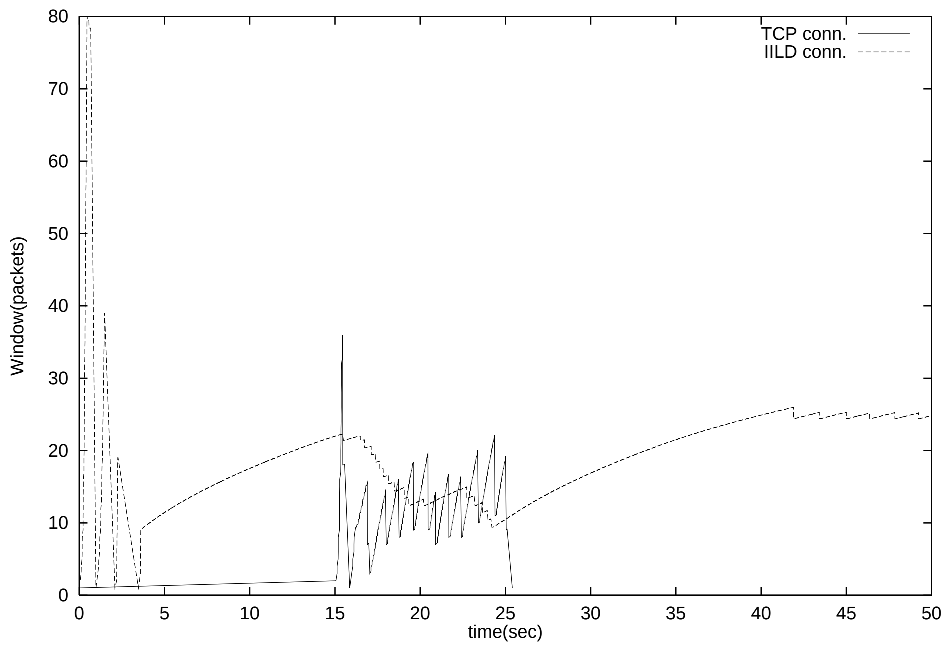


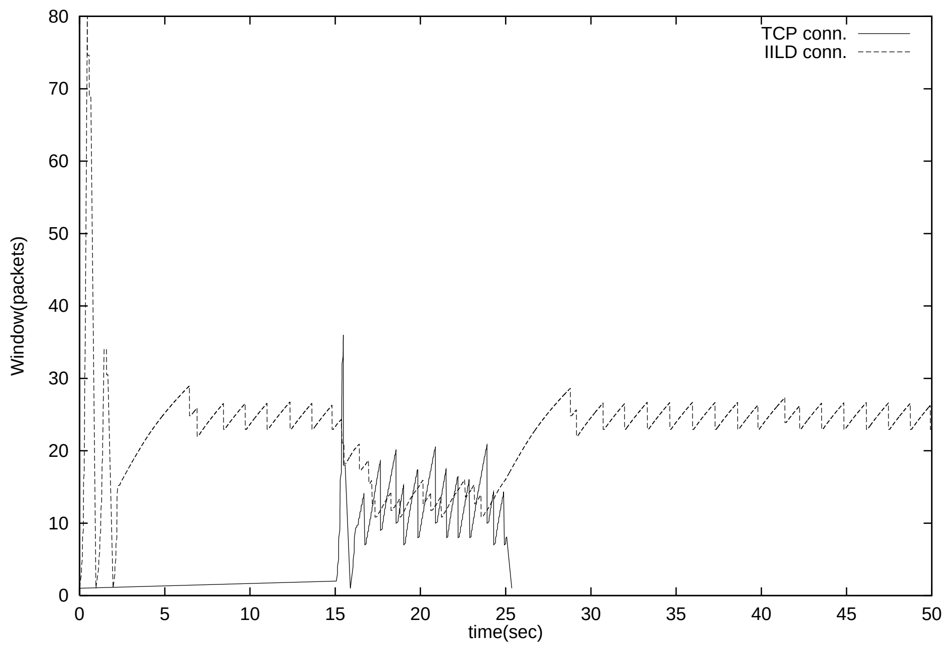
Window evolution in simulation



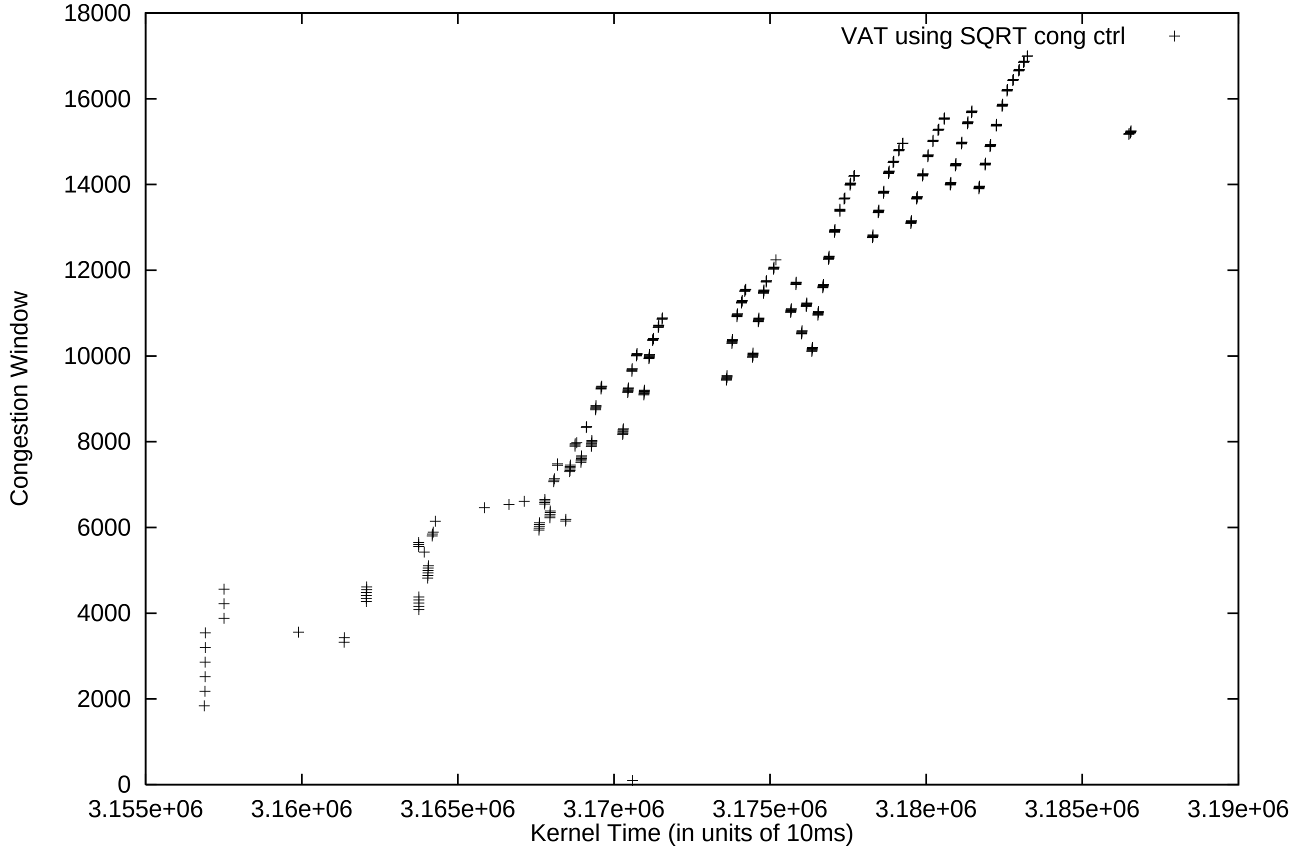


Impulse-response





SQRT in the Internet

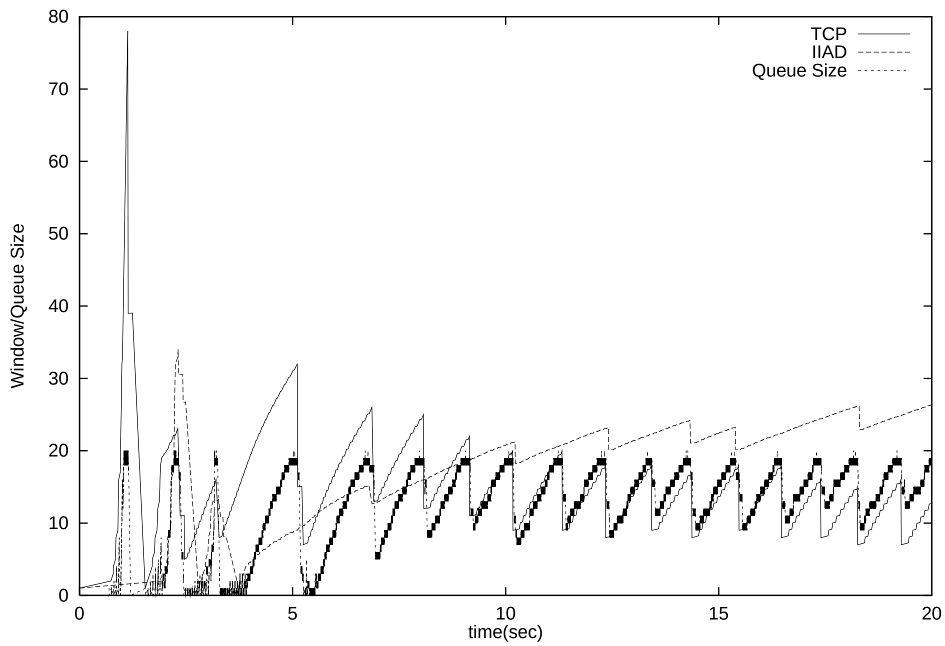


TCP-friendliness isn't quite enough

- Across drop-tail bottlenecks, non-AIMD, TCP-friendly binomial algorithms with $0 \leq k, l \leq 1$ often out-perform TCP AIMD.
- The theory isn't wrong—the problem is with drop-tail gateways.
- Drop-tail buffer provisioning tuned for TCP AIMD ($b \times RTT$) leads to higher buffer occupancy for a binomial algorithm.
- Deploy active queue management like RED!

Buffer occupancy

- With TCP AIMD, a window decrease flushes the buffer for the connection, allowing the competition to use it.
- In general, a non-AIMD binomial algorithm does not flush the buffer on a decrease.



If AIMD is the answer...

What is the question? (Shenker)

- TCP AIMD is the best TCP-friendly algorithm for elastic applications (like FTP).
- Of all the TCP-friendly binomial algorithms, it is the most efficient in terms of probing efficiency.
- But other binomial algorithms seem promising for other kinds of applications (e.g., streaming audio/video).

Summary

- I: $w(t+R) = w(t) + \alpha / w(t)^k; \alpha > 0$
  D: $w(t+\delta t) = w(t) - \beta w(t)^l; 0 < \beta < 1.$

- $k + l = 1, l \leq 1 \Rightarrow$ TCP-friendly
  trade-off between I and D functions.

  Schemes like IIAD and SQRT avoid drastic window reductions, but are TCP-friendly.

- TCP-friendly binomial control algorithms interact well with TCP AIMD across RED (or equivalent) gateways.

- Several issues remain open: desynchronized convergence, variable bottlenecks, EBCC comparisons, oscillatory behavior, stability across multiple co-existing algorithms.