

# **A report on a few steps in the evolution of congestion control**

Sally Floyd

June 10, 2002

IPAM Program in Large Scale Communication Networks

## Topics:

- High-speed TCP.
- Faster Start-up?
- AQM: Adaptive RED
- Evaluation of AQM mechanisms.
- A proposal about models and simulations
- Other open questions?

## **Architectural sub-themes:**

- A goal of incremental deployment in the current Internet.
- Steps must go in the fundamentally correct, long-term direction, not be short-term hacks.
- Robustness in heterogeneous environments valued over efficiency of performance in well-defined environments.
- A skepticism towards simple models.
- Learning from actual deployment is an invaluable step.
- The Internet will continue to be decentralized and fast-changing.

## **HighSpeed TCP:**

Joint work with Sylvia Ratnasamy and Scott Shenker.

Additional investigations with Evandro de Souza and Deb Agarwal.

URLs:

<http://www.icir.org/floyd/papers/draft-floyd-tcp-highspeed-00c.txt>

<http://www.icir.org/floyd/papers/draft-floyd-tcp-slowstart-00b.txt>

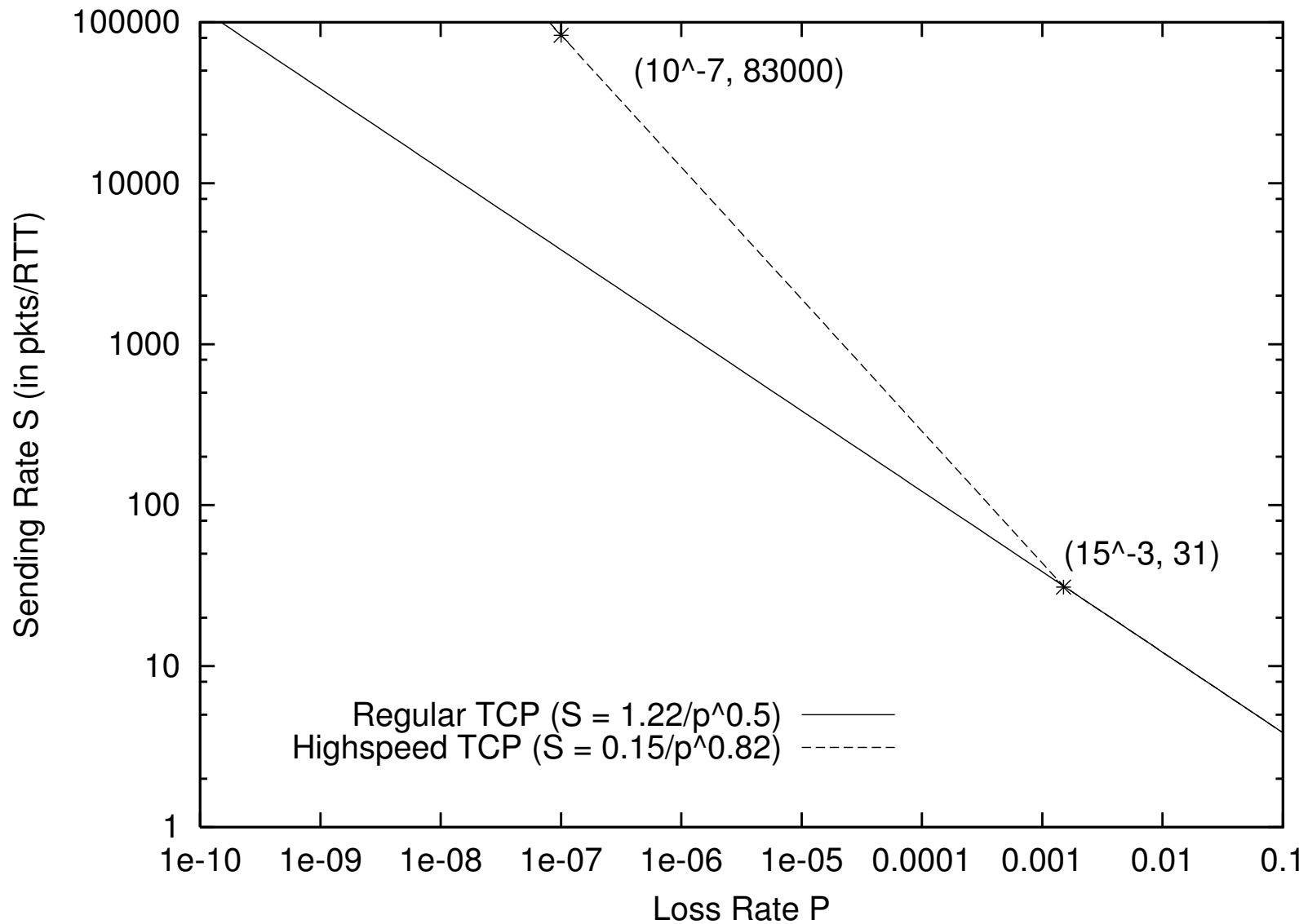
## HighSpeed TCP: The problem.

- TCP's average congestion window is roughly  $1.2/\sqrt{p}$  packets.
- Maintaining an average cwnd of at least  $1.2 * 10^k$  packets requires a packet loss/corruption rate of at most  $10^{-2k}$ .
- Given 1500-byte packets and a 100 ms RTT, filling a 10 Gbps pipe would correspond to a congestion window of  $W = 83,333$  packets.
  - At least 1.6 hours between packet drops.
- We can do better, even with only the current feedback from routers.

## HighSpeed TCP: Is this a pressing problem?

- Nope. In practice, users do one of the following:
  - Open up  $N$  parallel TCP connections; or
  - Use MulTCP (roughly like an aggregate of  $N$  virtual TCP connections).
- However, we think it is possible to do much better, with:
  - Better flexibility (no  $N$  to configure);
  - Better scaling;
  - Better slow-start behavior;
  - Competing more fairly with current TCP(for environments where TCP is able to use the available bandwidth).

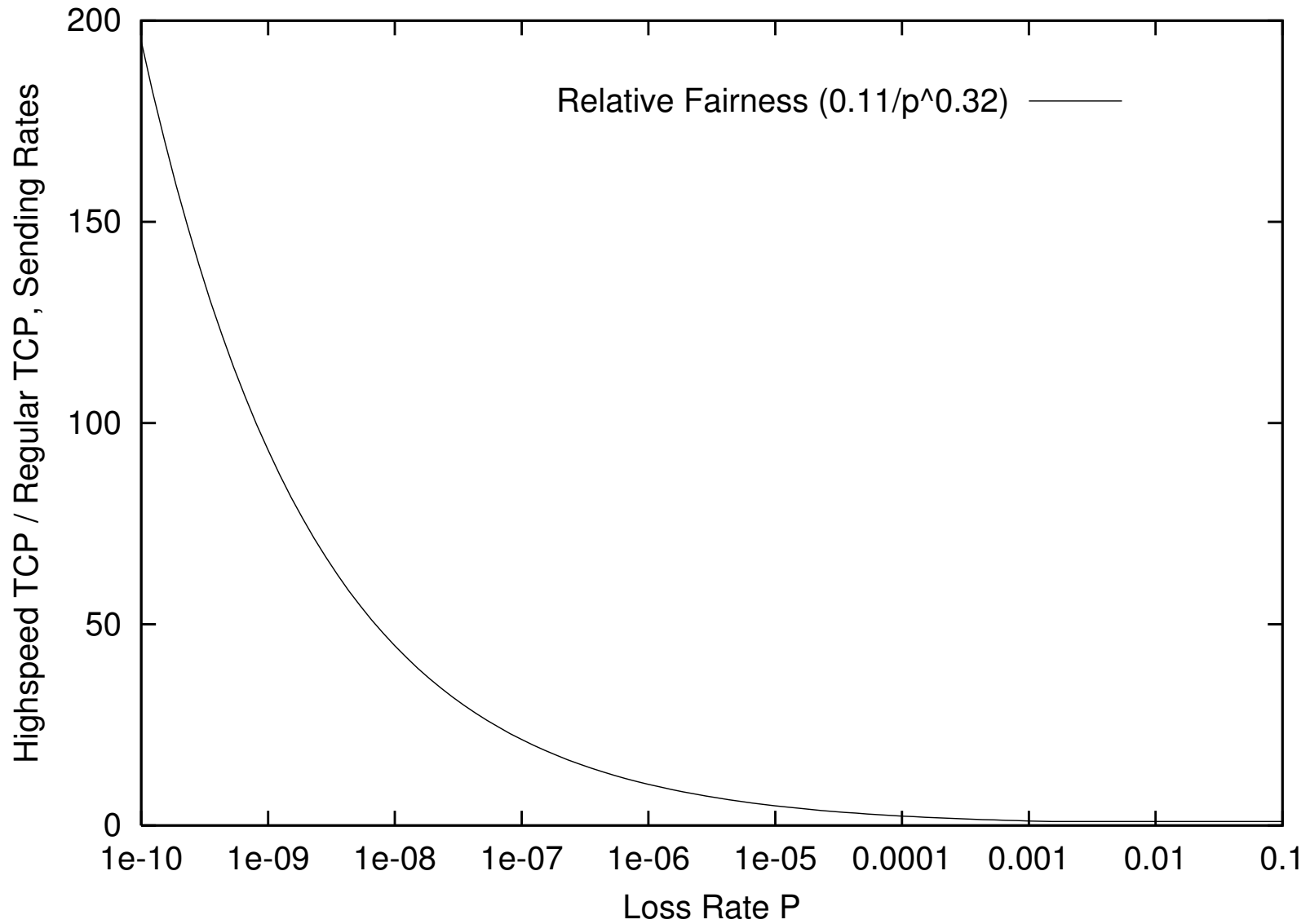
## HighSpeed TCP: use a modified response function.



## HighSpeed TCP: Simulations in NS.

- ./test-all-tcpHighspeed in tcl/test.
- The parameters specifying the response function:
  - Agent/TCP set low\_window\_ 31
  - Agent/TCP set high\_window\_ 83000
  - Agent/TCP set high\_p\_ 0.0000001
- The parameter specifying the decrease function at high\_p\_:
  - Agent/TCP set high\_decrease\_ 0.1

## HighSpeed TCP: Relative fairness.



## HighSpeed TCP: modifying slow-start:

- Slow-starting up to a window of 83,000 packets doesn't work well.
  - Tens of thousands of packets dropped from one window of data.
  - Slow recovery for the TCP connection.
- The answer:
  - Agent/TCP set `max_ssthresh_N`
  - During the initial slow-start, increase the congestion window by at most N packets in one RTT.

## **Faster Start-up?**

From a proposal by Amit Jain.

No URL yet.

## Faster Start-up: Larger Initial Sending Rate

- An IP option in the SYN packet gives the sender's desired initial sending rate.
  - Routers on the path decrement a counter,
  - and decrease the allowed initial sending rate, if necessary.
- If all routers on the path participated:
  - The receiver tells the sender the allowed initial sending rate in the SYN/ACK packet, in the transport header.
- This is from a proposal by Amit Jain (from Netscaler).

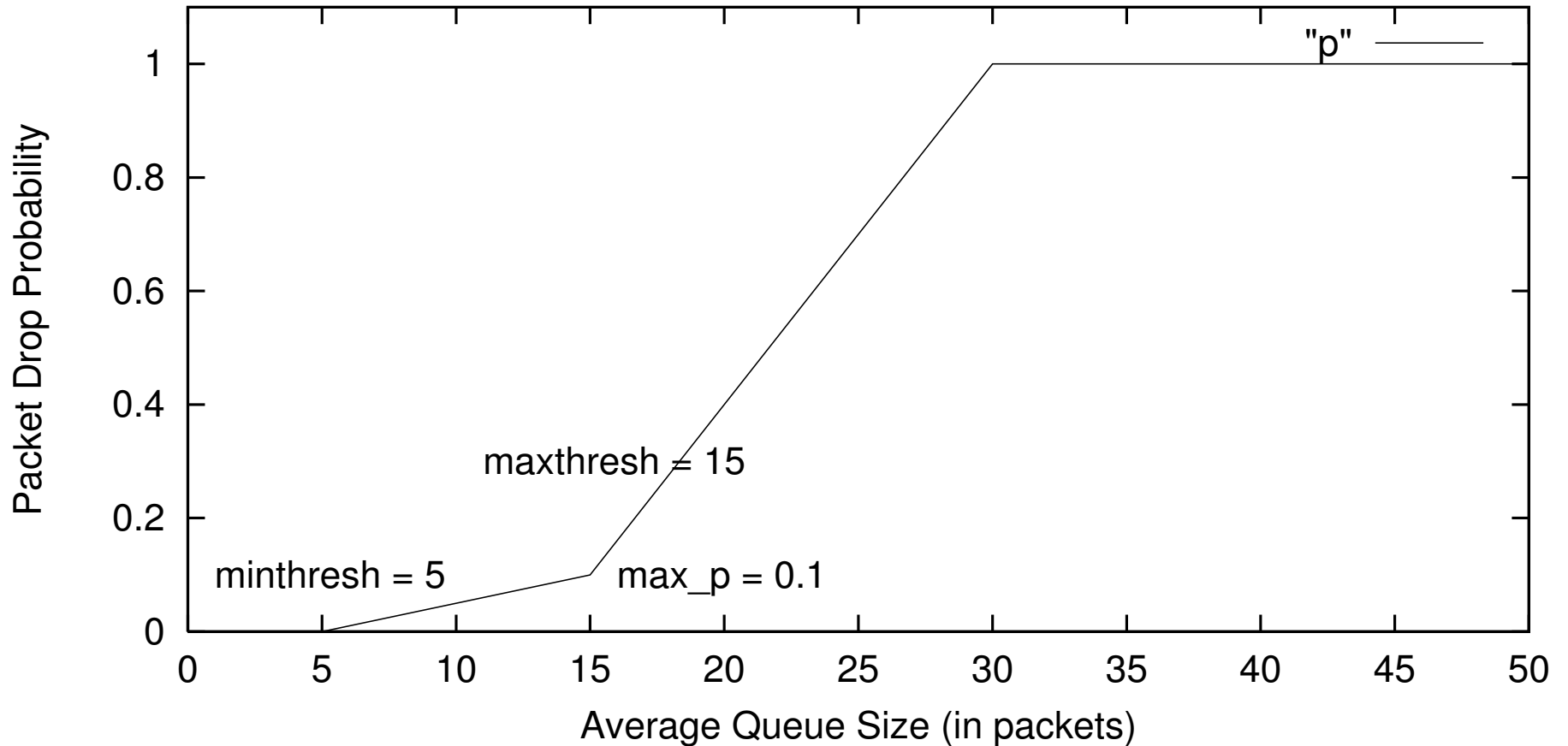
## **Adaptive RED**

Joint work with Ramakrishna Gummadi and Scott Shenker.

URL (with simulation scripts):

<http://www.cs.berkeley.edu/~ramki/adaptiveRED/>

## The One-Page Primer on RED:



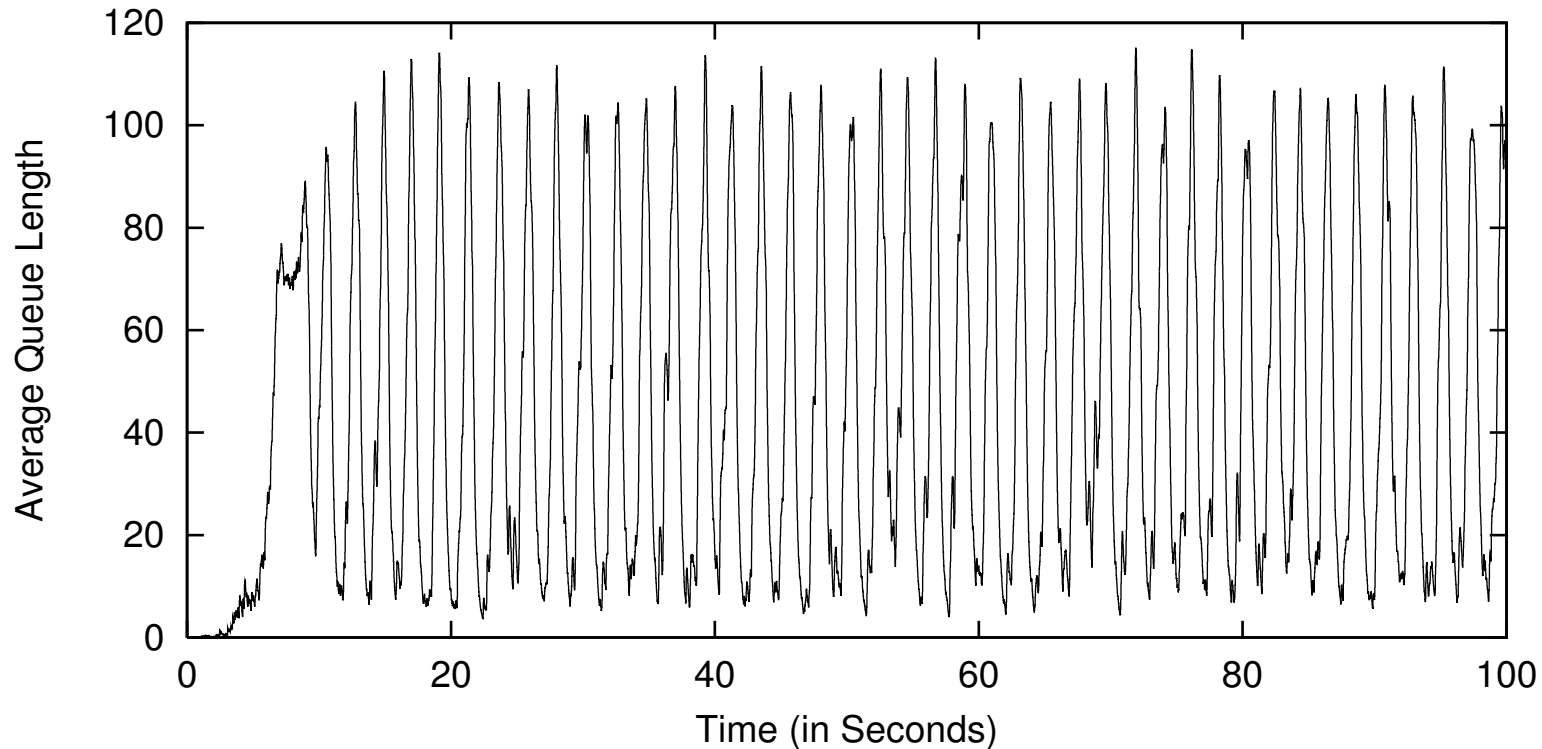
For the average queue size:

$$ave_q \leftarrow (1 - w_q)avg_q + w_q q$$

## Adaptive RED: adapting $max_p$

- The original Adaptive RED proposal is from Feng et al., 1997.
  - Adjusts  $max_p$  to keep the average queue between  $min_{th}$  and  $max_{th}$ .
- We have a new implementation of Adaptive RED, adapting  $max_p$ .
- Automatic setting of  $w_q$  as a function of the link bandwidth.
- Automatic setting of  $min_{th}$  and  $max_{th}$  as a function of the target queue size.

## Simulation with malignant oscillations:

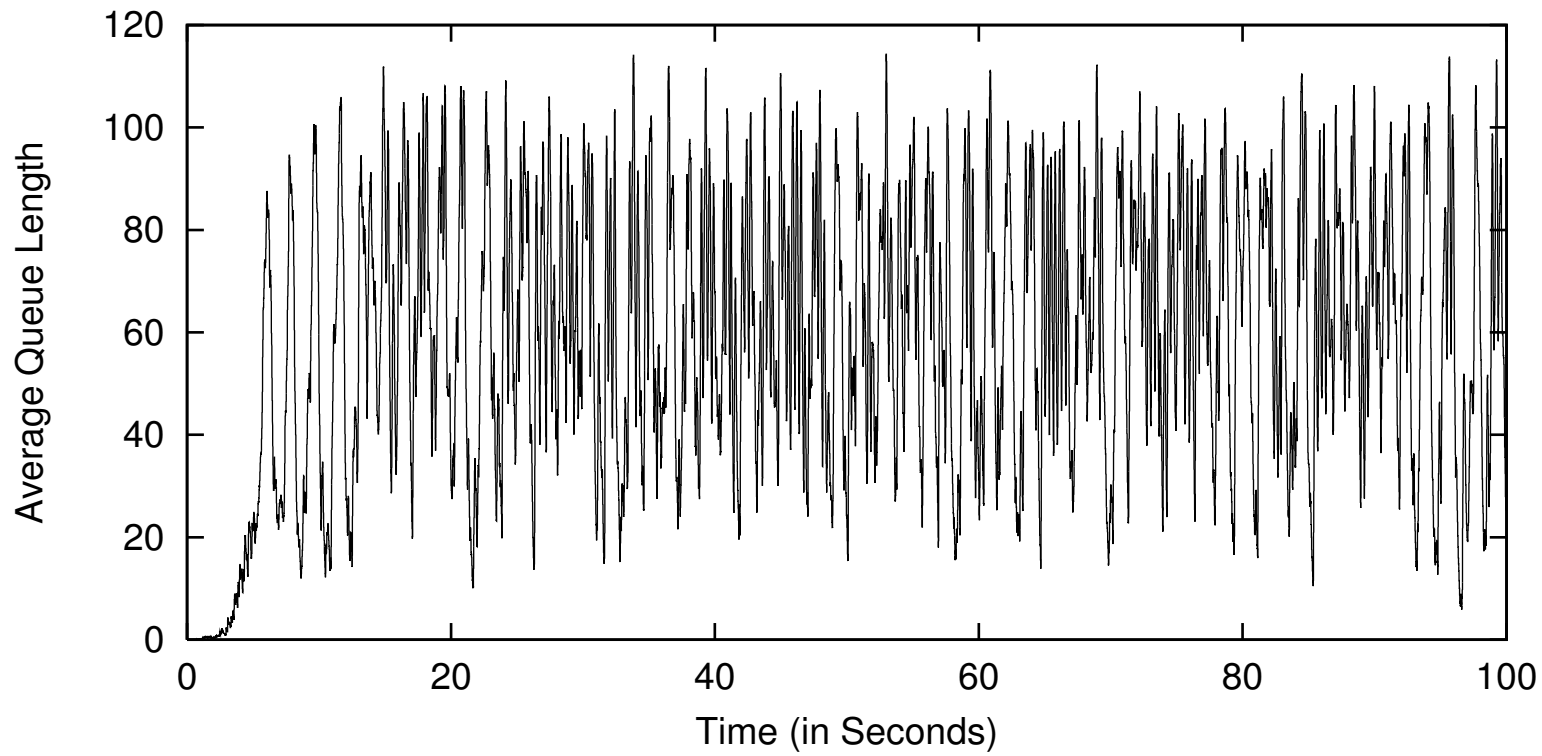


RED, one-way long-lived traffic,  $w_q=0.002$ .

No web traffic, no reverse-path traffic.

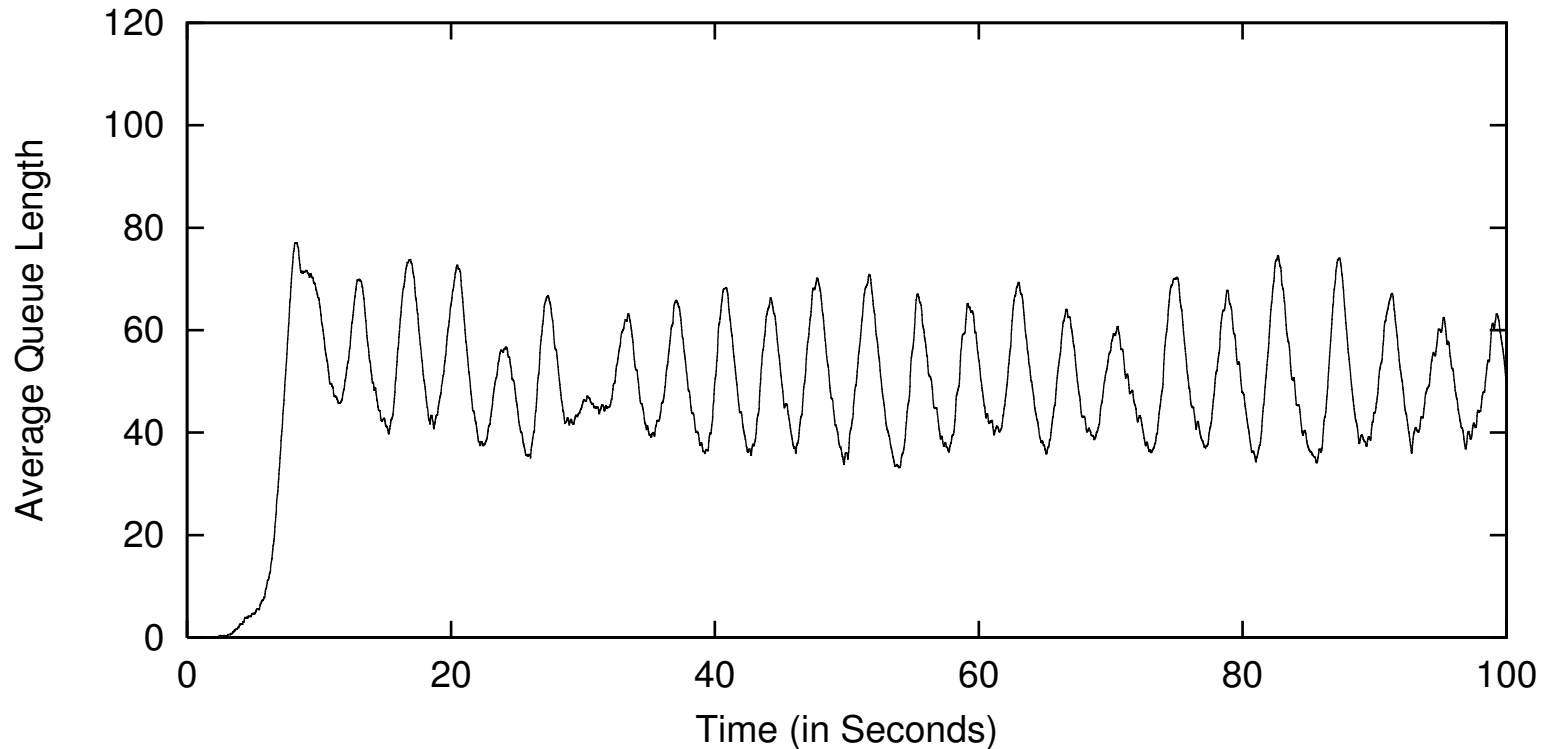
15 Mbps link, 250 ms round-trip time.

90% link utilization.



RED, mostly one-way long-lived traffic,  $w_q=0.002$ .  
Web traffic and reverse-path traffic added.  
15 Mbps link, 250 ms round-trip time.

## Simulation with benign oscillations:

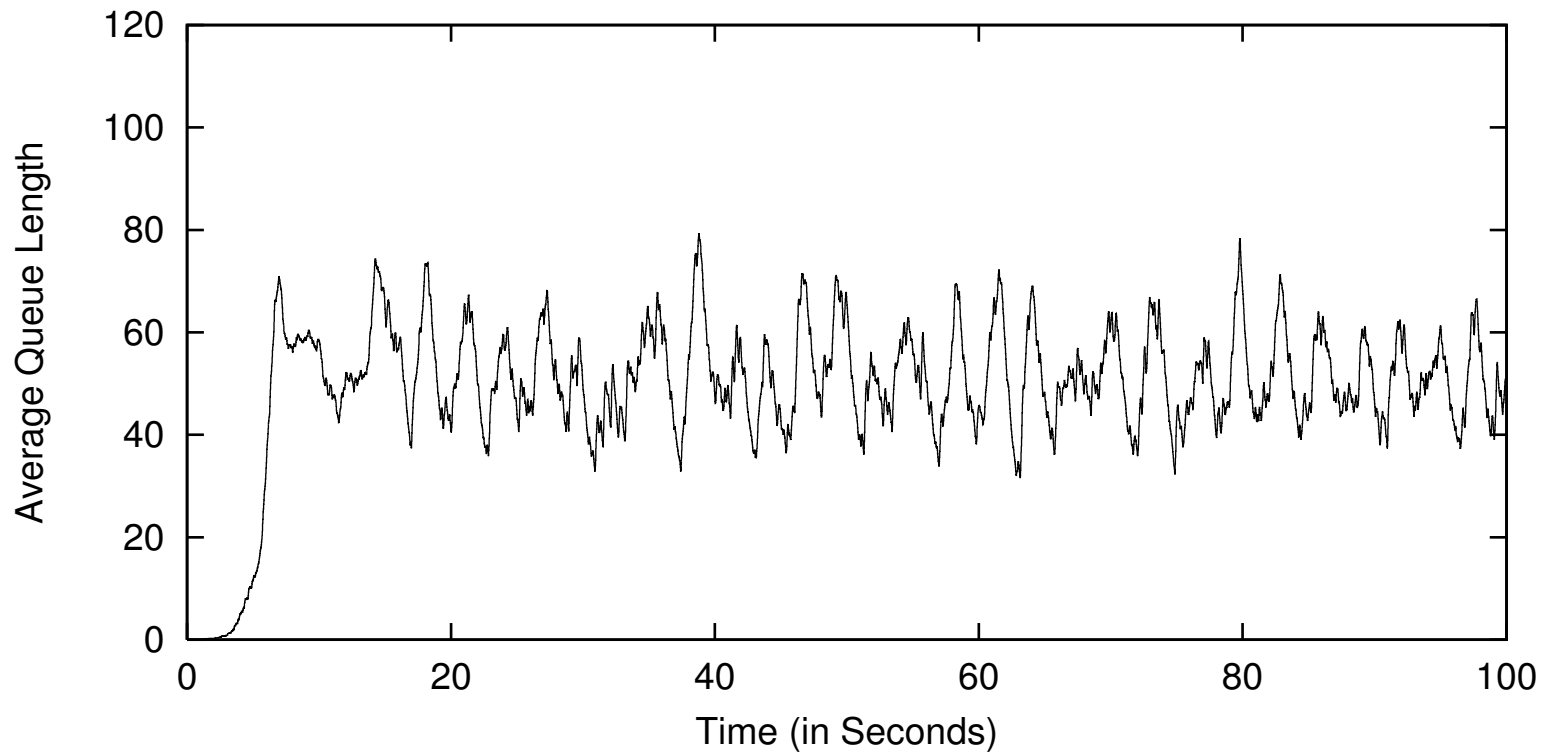


Adaptive RED, one-way long-lived traffic,  $w_q=0.00027$ .

No web traffic, no reverse-path traffic.

15 Mbps link, 250 ms round-trip time.

96.8% link utilization.

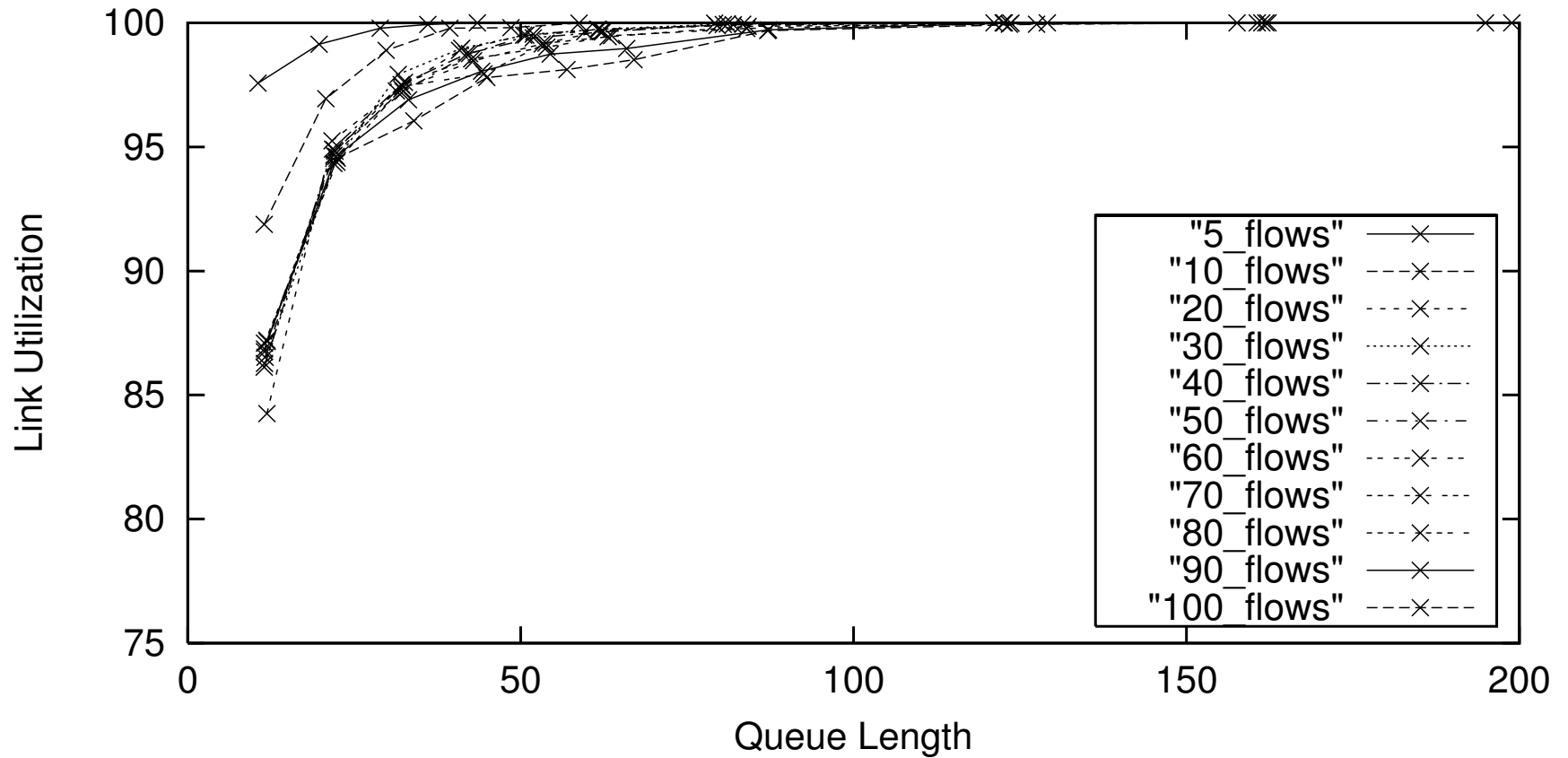


Adaptive RED, mostly one-way long-lived traffic,  $w_q=0.00027$ .  
Web traffic and reverse-path traffic added.  
15 Mbps link, 250 ms round-trip time.

## **The optimal average queue size?**

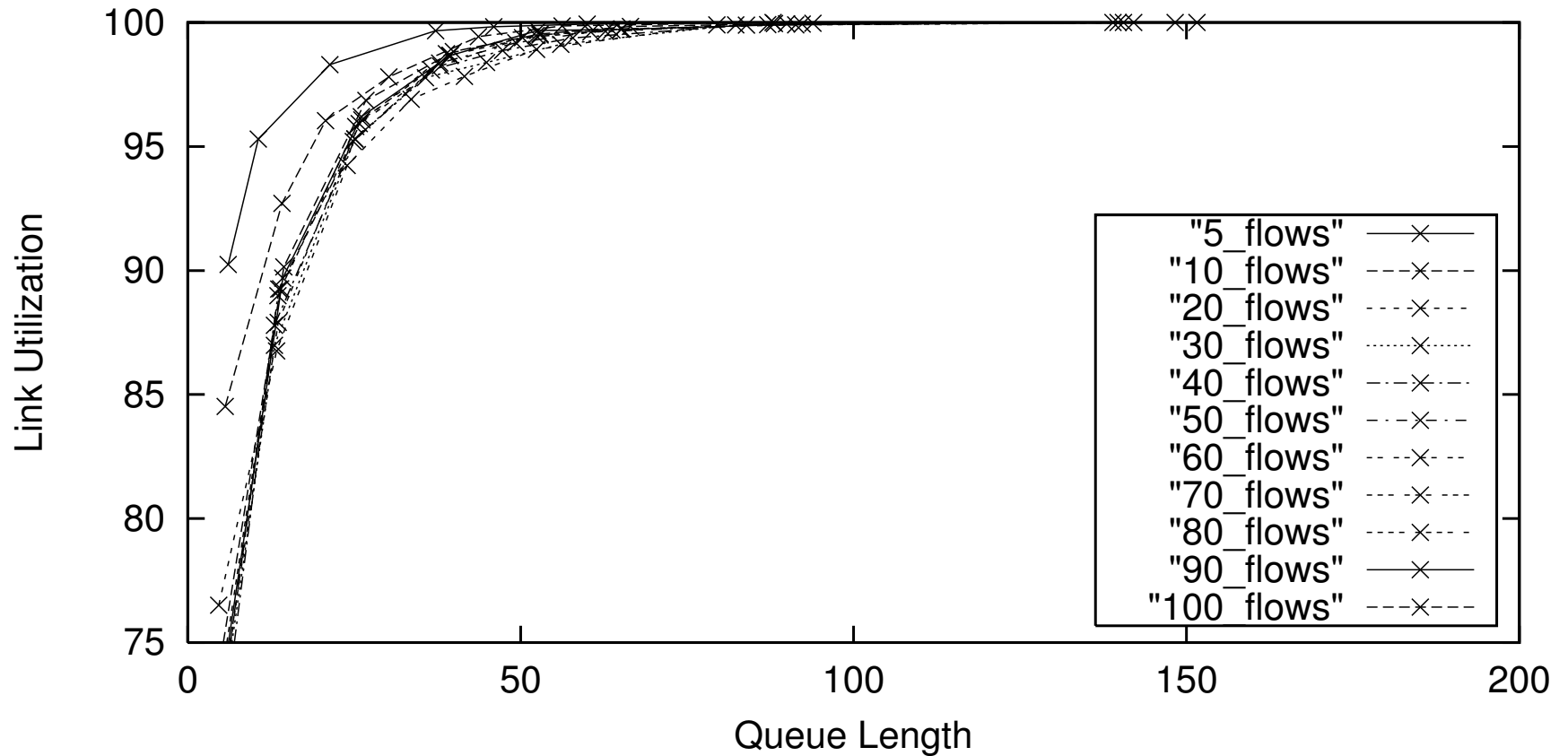
- It depends on the desired tradeoff at the router between high utilization and low delay.
- It is heavily affected by traffic, topology, etc.
- We don't know the optimal average queue size.

## Mostly long-lived traffic, Adaptive RED.



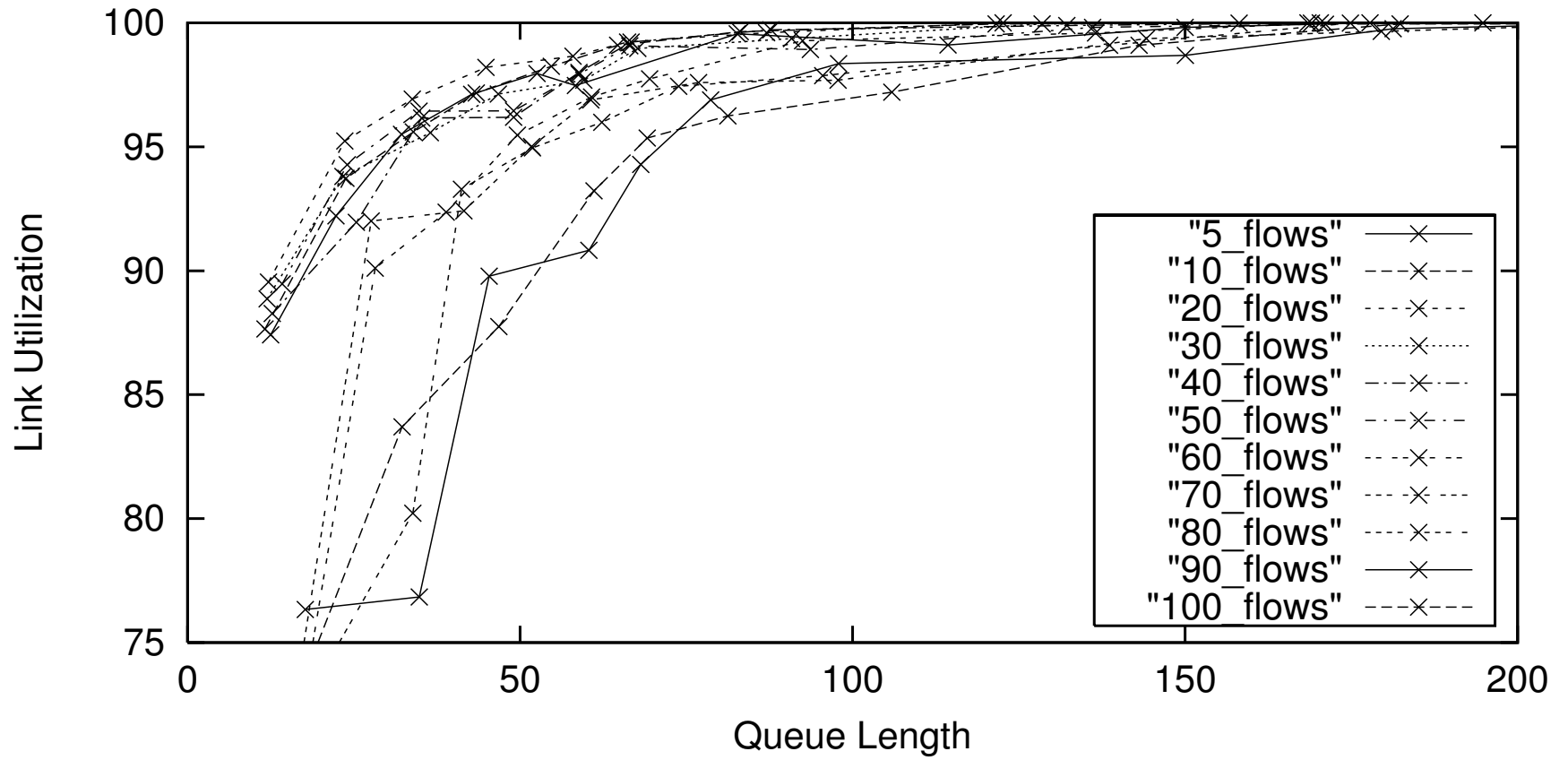
Traffic includes some web mice, and reverse-path traffic.

## Mostly long-lived traffic, Drop Tail.

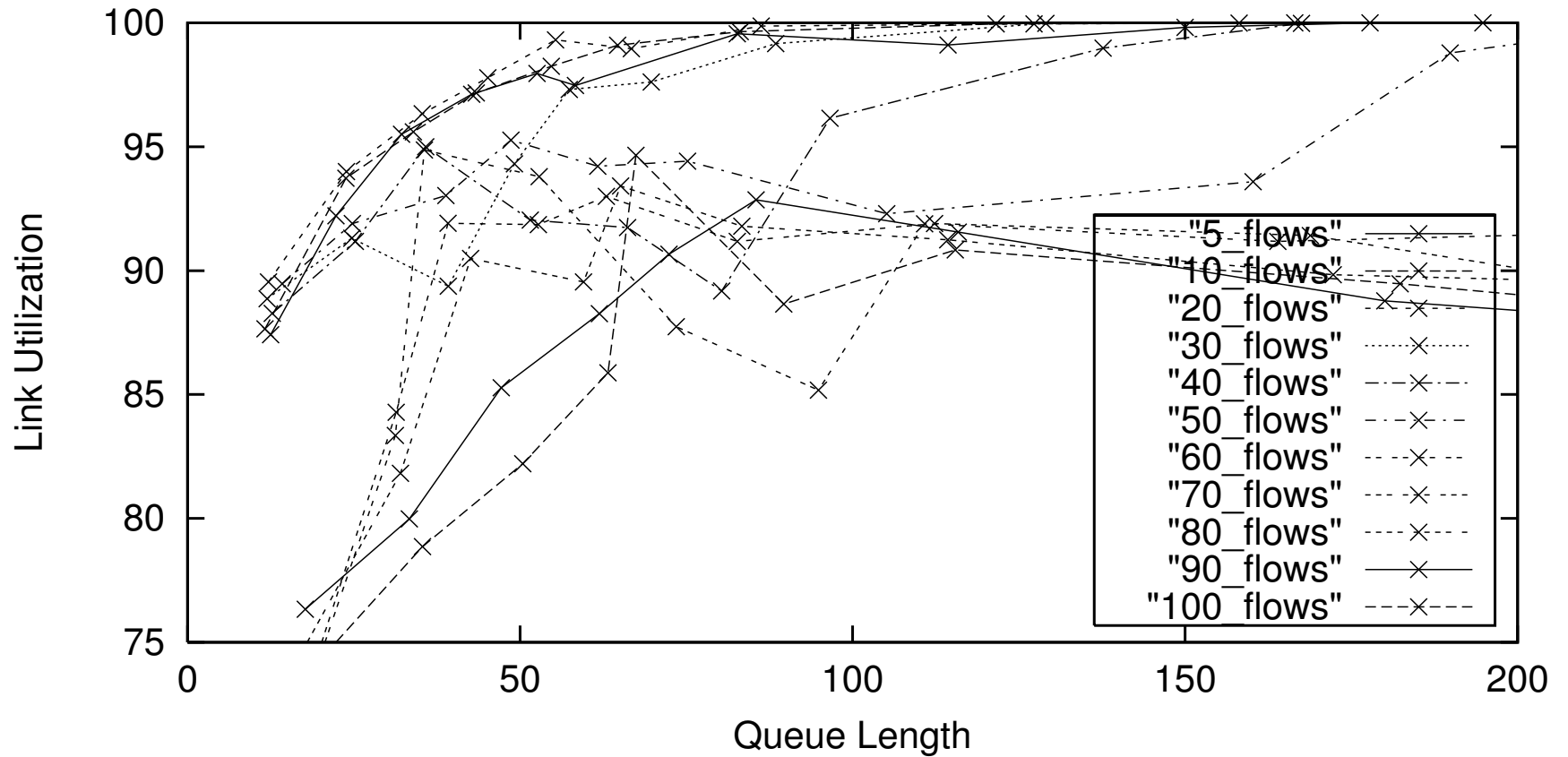


We haven't plotted anything for fairness, our third metric.  
(RED and Drop Tail often differ in fairness.)

## Long-lived and web traffic, Adaptive RED.



# Long-lived and web traffic, Adaptive RED, with reverse-path delay.



The reverse-path queue is configured the same as the forward-path queue:

**An aside: creating worst-case oscillations with TCP and AQM:**  
(from last night)

Assume a "time constant" for average queue size estimator of  $T$  sec.  
( $T = 1$  second, for Adaptive RED).

Then the "resonant frequency" is roughly  $3T$  seconds.

We want to choose the number of flows  $N$  so that the packet drop rate  $p$  gives a congestion control epoch of  $3T$  seconds, so that the natural frequency of TCP matches the resonant frequency of the RED/ARED queue.

Given link bandwidth  $B$  pkts/sec, RTT  $R$  seconds,  $N$  flows:  
The average bandwidth per flow is  $BR/N$  pkts/RTT.

Facts about TCP:

With packet drop rate  $p$ , the average window  $W$  is  $1.2/\sqrt{p}$  pkts, and there are  $\frac{2}{3}W$  RTTs in a congestion control epoch.

So we want  $\frac{2}{3}W = 3T/R$ , or  $W = 4.5T/R$ .

So choose  $N$  so that:  $BR/N = 4.5T/R$ , or  $N = BR^2/(4.5T)$ .

Conjecture: For  $T = 1$  sec.,  $R = 0.1$  sec.,  $B = 1000$  pkts/sec (8Mbps for 1KB pkts), the worst case oscillations occur with  $N = 10/0.45 = 22$  flows.

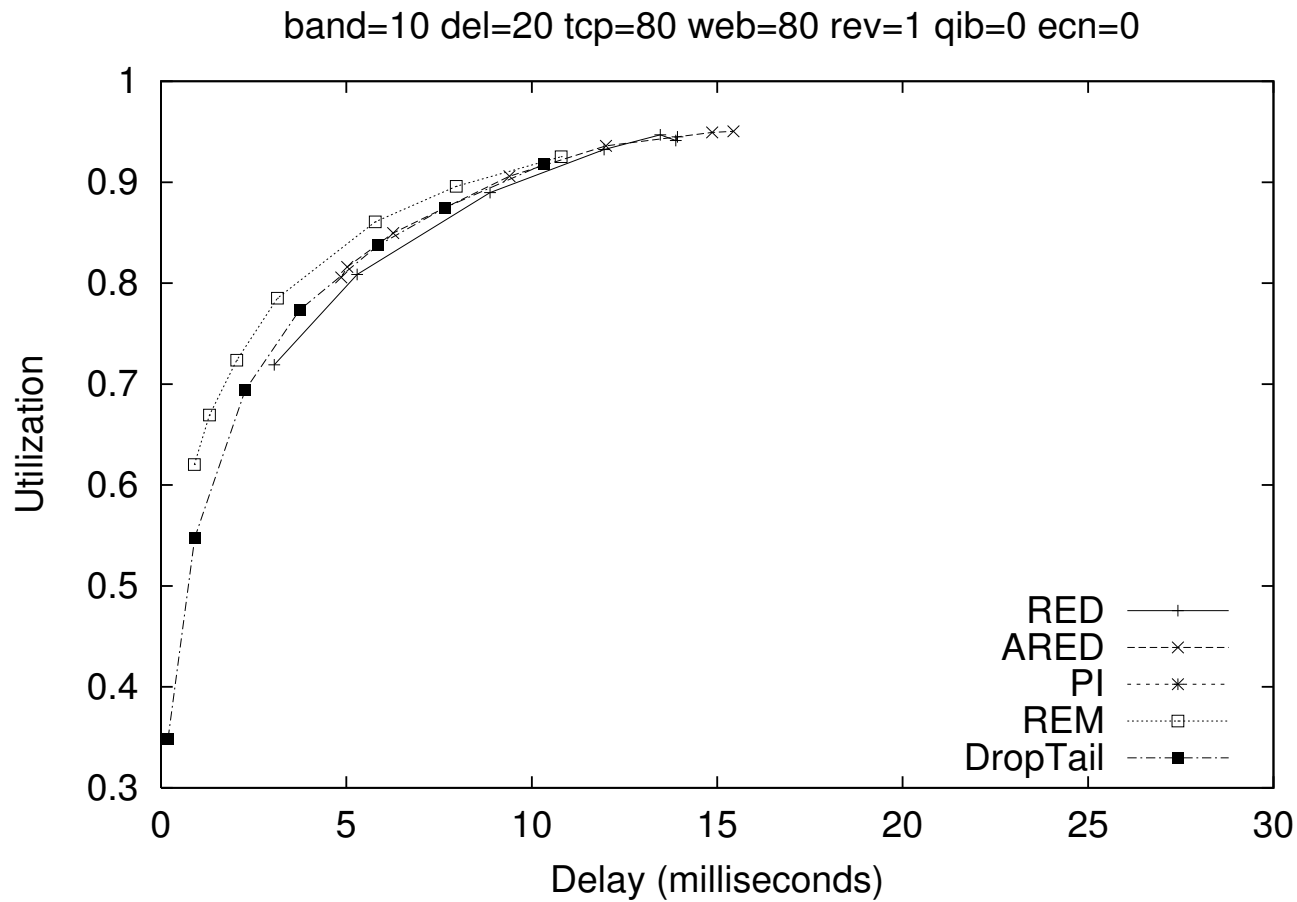
## **An Evaluation of AQM**

Joint work with Jitendra Padhye and Scott Shenker.

No URL yet...

## An Evaluation of AQM

- On many simulation scenarios, Adaptive RED, AVQ, Drop-Tail PI, RED, REM give similar performance.
- Drop-Tail and AVQ generally have higher packet drop rates.
- RED and Adaptive RED can have undesirable oscillations with very large round-trip times.
- REM and PI can perform poorly in scenarios with mostly web traffic, or with changes in the level of congestion.



Steady-state simulations, with some web traffic and reverse-path traffic, and 40-320 ms RTTs. Queue in packets.

## **Developing and Evaluating Models:**

Joint work with Eddie Kohler

URL for one part:

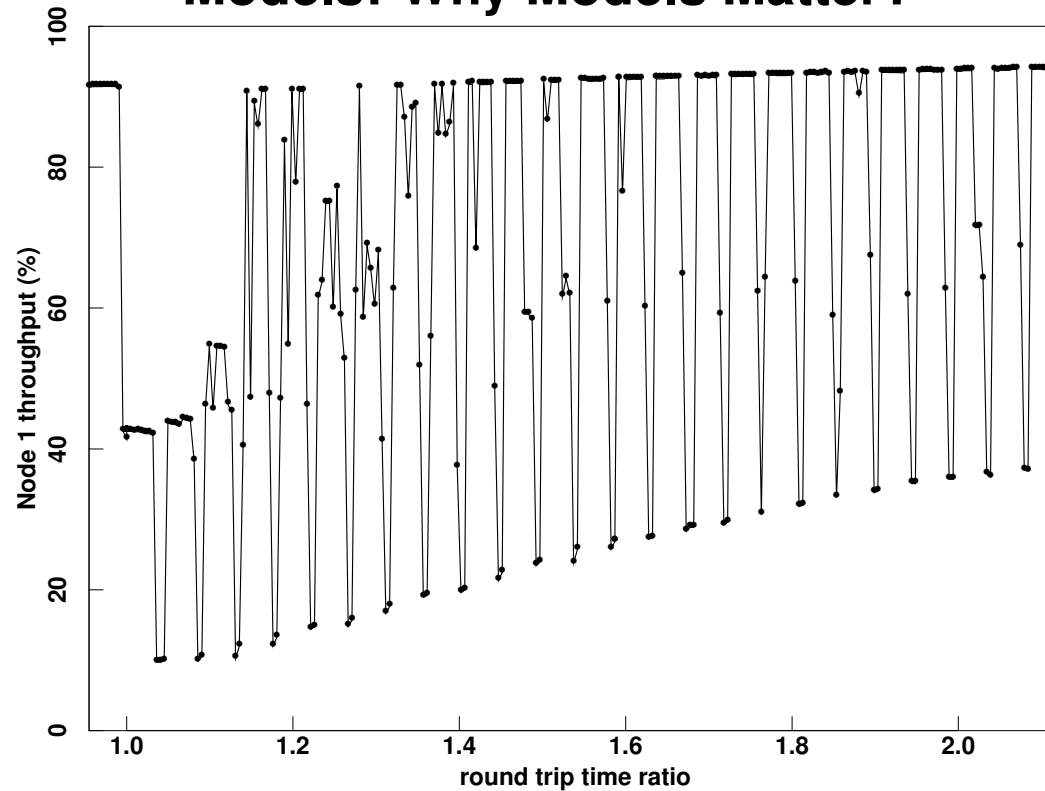
Building Models for Aggregate Traffic on Congested Links

<http://www.icir.org/models/>

## **Models: A Proposal about Developing Models and Simulation Scenarios**

- What measurement studies are needed for improving our models?
  - E.g., where is the congestion in the Internet? What are the ranges of round-trip times? Is more needed on traffic generation? What about reverse-path congestion?
- How do we translate results from measurement studies into our models?
- How do we know what features are critical to include in our models?
- Can we do more to improve our shared understanding of best practices for models and for simulation scenarios?

## Models: Why Models Matter?



Flow 1's throughput as a function of the ratio of the two flows' round-trip times, in a scenario with one-way traffic and Drop-Tail queue management. [From "On Phase Effects", 1992.]

## Open questions?

- How do things in one part of the network (e.g., buffer sizes, flash crowds) affect behavior in other parts?
- How do we shed light on tradeoffs between delay and throughput?
- What are the inherent limitations of one bit of congestion feedback, if any? (E.g., in terms of how aggressive flows can be.)
- What are the inherent limitations of not making reservations, and not keeping per-flow state in the network? (E.g., in terms of how aggressive flows can be.)
- What will the tradeoffs be when we have very fast networks, often with very high available bandwidth, and flows could often send all of their data in a fraction of a RTT?