

The following paper posted here is not the official IEEE published version. This material was presented at the IEEE International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks (9th : 2002 : New York, USA)

Copyright © 2002 IEEE.

Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

TCP



Matthew Roughan
AT&T Labs - Research
roughan@research.att.com

Ashok Erramilli
Qnetworx
Erramilli@qnetworx.com

Darryl Veitch
EMUlab – Ericsson and the University of Melbourne
d.veitch@ee.mu.oz.au

Outline



- TCP flow control
- Persistent source models
 - Square root p law
 - Fixed points
- Short-lived sources

Why Flow Control?



- October 1986, Internet had its first congestion collapse
- Link LBL to UC Berkeley
 - 400 yards, 3 hops, 32 Kbps
 - throughput dropped to 40 bps
 - factor of ~1000 drop!
- 1988, Van Jacobson proposed TCP flow control

Congestion Control

- TCP seeks to
 - Achieve high utilization
 - Avoid congestion
 - Share bandwidth
- Window flow control
 - Source rate = $\frac{W}{RTT}$ packets/sec
 - Adapt W to network (and conditions)
 $W = BW \times RTT$

Network Flow Control

- Source calculates **cwnd** from indication of network congestion
- Congestion indications
 - Losses
 - Delay
 - Marks
- Algorithms to calculate **cwnd**
 - Tahoe, Reno, Vegas, RED, REM ...

TCP Congestion Control

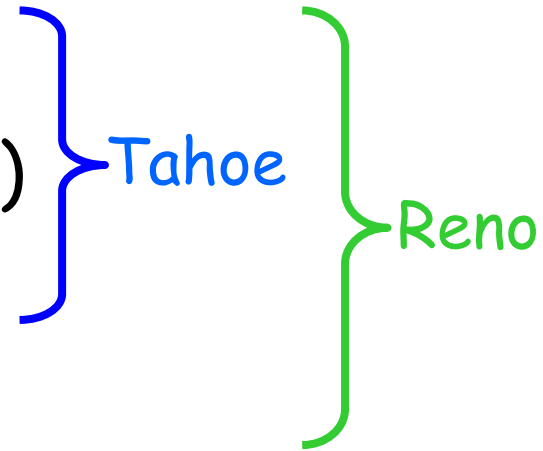
- Has four main parts

- Slow Start (SS)

- Congestion Avoidance (CA)

- Fast Retransmit

- Fast Recovery



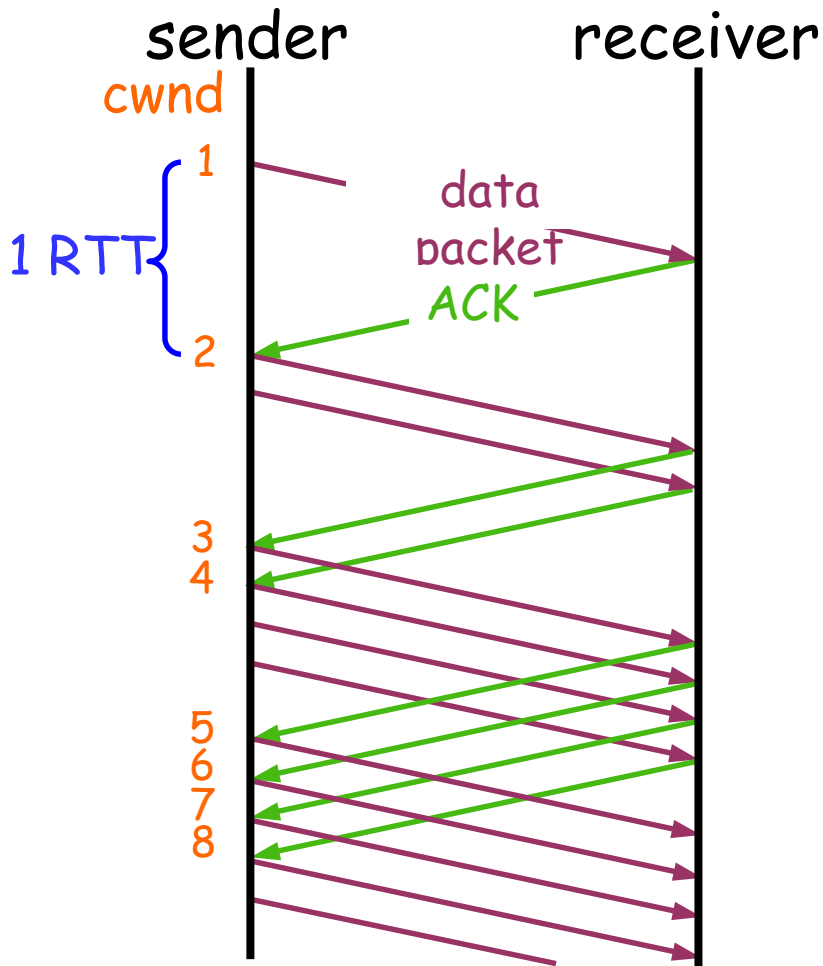
- **ssthresh**: slow start threshold
determines whether to use SS or CA

- Assume packet losses are caused by congestion

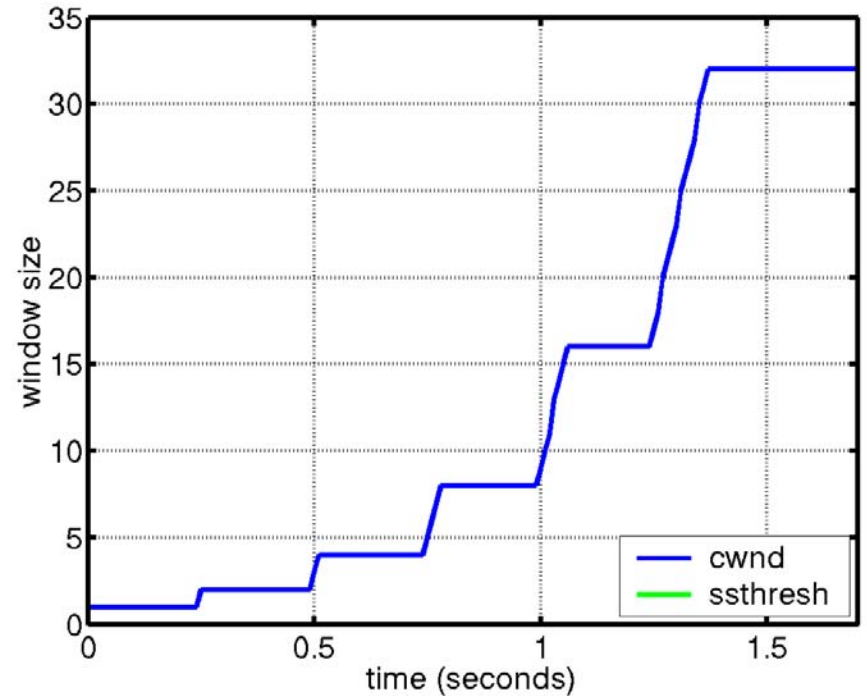
Slow Start

- Start with $cwnd = 1$ (slow start)
- On each successful ACK increment $cwnd$
 $cwnd \leftarrow cwnd + 1$
- Exponential growth of $cwnd$
each RTT: $cwnd \leftarrow 2 \times cwnd$
- Enter **CA** when $cwnd \geq ssthresh$

Slow Start



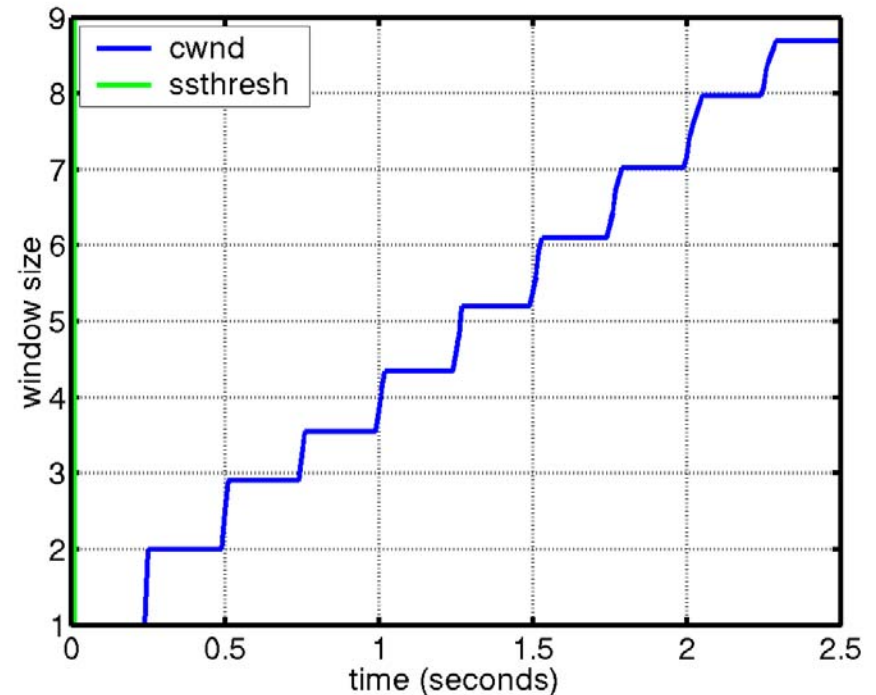
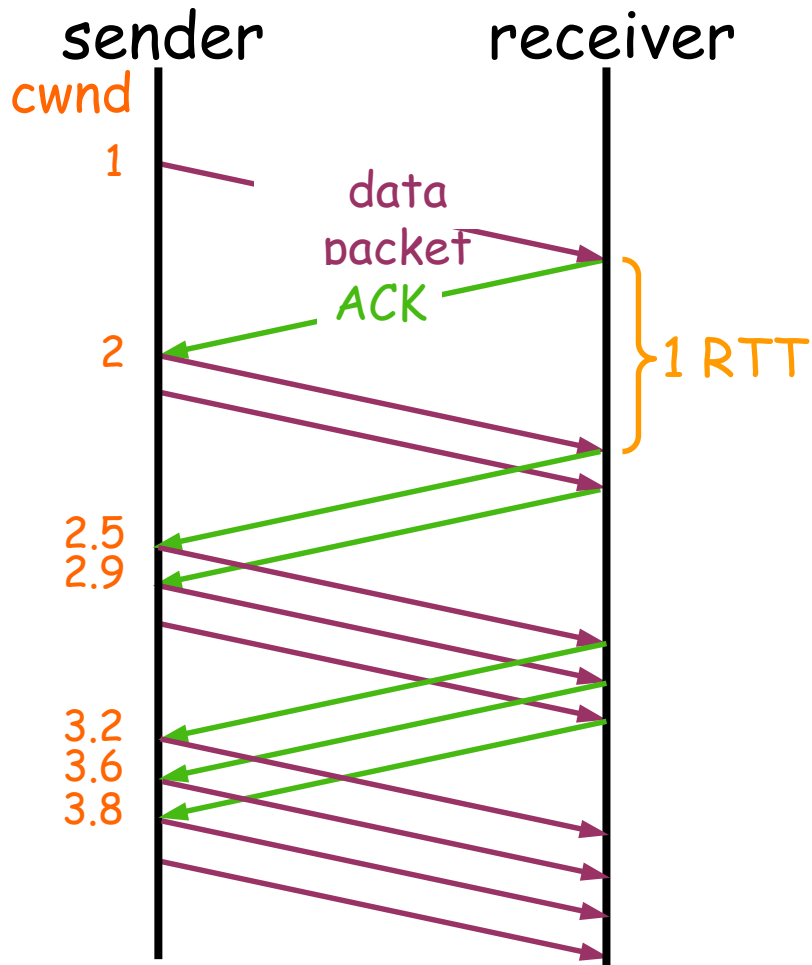
$cwnd \leftarrow cwnd + 1$ (for each ACK)



Congestion Avoidance

- Starts when $cwnd \geq ssthresh$
- On each successful ACK:
 $cwnd \leftarrow cwnd + 1/cwnd$
- Linear growth of $cwnd$
each RTT: $cwnd \leftarrow cwnd + 1$

Congestion Avoidance



$cwnd \leftarrow cwnd + 1/cwnd$ (for each ACK)

Packet Loss

- **Assumption:** loss indicates congestion
- Packet loss detected by
 - Retransmission TimeOuts (RTO timer)
 - Duplicate ACKs (at least 3)

Packets



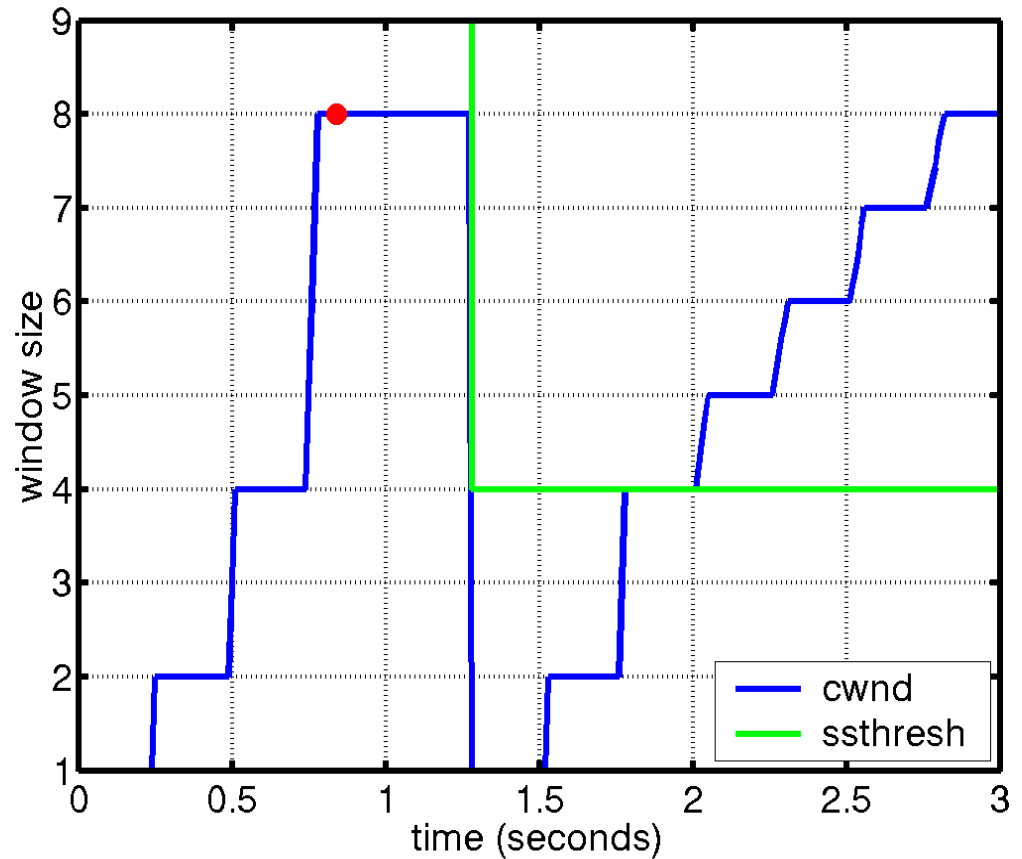
Acknowledgements



Timeout

$ssthresh \leftarrow cwnd/2$

$cwnd = 1$



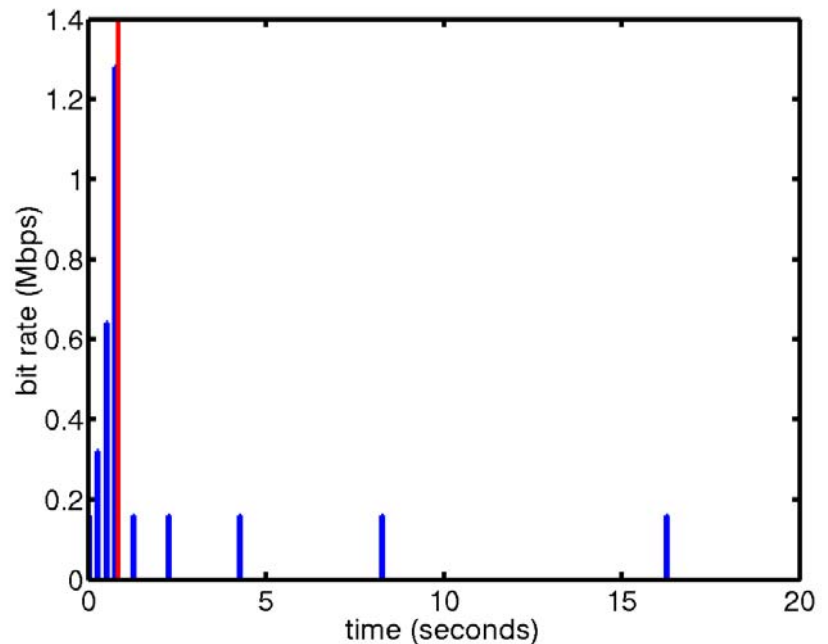
Fast Retransmit

- Wait for a timeout is quite long
- Immediately retransmits after 3 dupACKs without waiting for timeout
- Adjusts ssthresh
 - $flight_size = \min(awnd, cwnd)$
 - $ssthresh \leftarrow \max(flight_size/2, 2)$
- Enter Slow Start ($cwnd = 1$)

Successive Timeouts

- When there is a timeout, double the RTO
- Keep doing so for each lost retransmission
 - Exponential back-off
 - Max 64 seconds¹
 - Max 12 retransmits¹

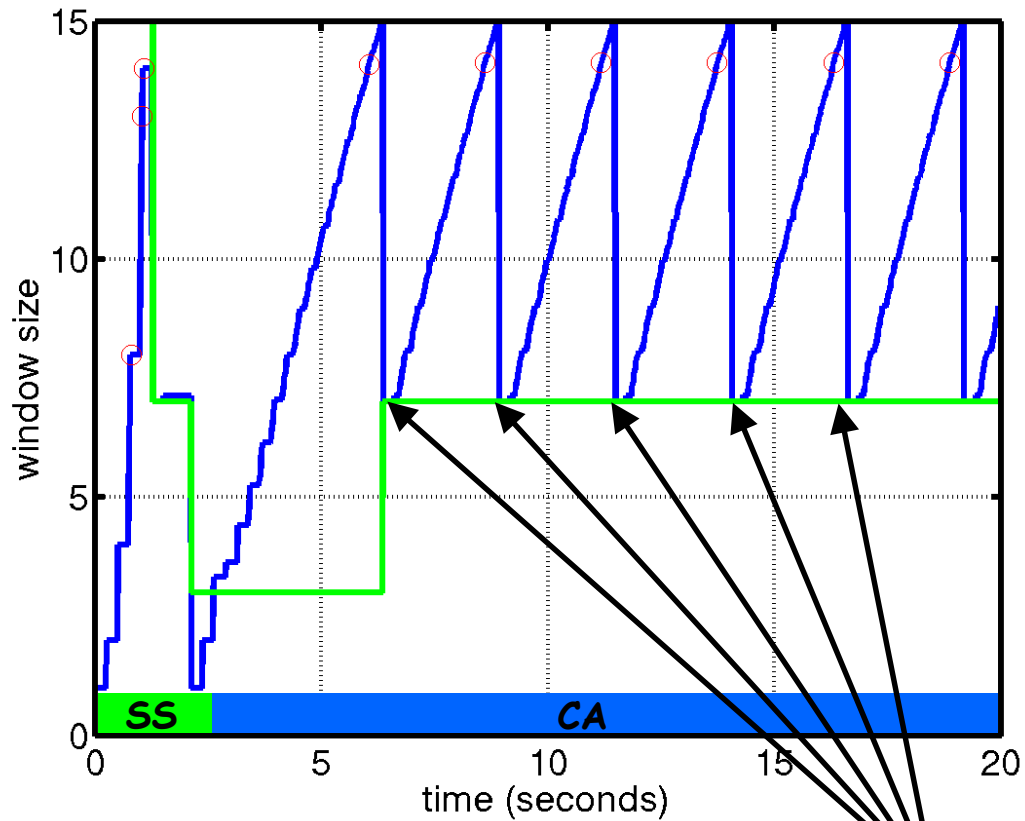
1 - Net/3 BSD



Fast recovery

- Motivation: prevent 'pipe' from emptying after fast retransmit
- Idea: each dupACK represents a packet having left the pipe (successfully received)
- Enter FR/FR after 3 dupACKs
 - Set $ssthresh \leftarrow \max(\text{flightsize}/2, 2)$
 - Retransmit lost packet
 - Set $cwnd \leftarrow ssthresh + ndup$ (window inflation)
 - Wait till $W = \min(\text{awnd}, cwnd)$ is large enough; transmit new packet(s)
 - On non-dup ACK (1 RTT later), set $cwnd \leftarrow ssthresh$ (window deflation)
- Enter CA

TCP Reno



Fast retransmission/fast recovery

RTO Calculation

- An accurate RTT measure is required to judge timeouts
- We can measure RTT by measuring the time to receive a packets ACK
- Use a smoothed RTT, S_{RTT} and the smoothed mean deviation D_{RTT}

$$RTO = S_{RTT} + 4 D_{RTT}$$

- Initial RTT should be > 3 seconds
 - Avoid spurious retransmission

Round Trip Time Estimation

- RTT is not known

- From <1 ms up to >1 second

- Need to know RTT to calculate RTO

- The measurement of RTT

$$S_{RTT} = S_{RTT} + g (M_{RTT} - S_{RTT})$$

$$D_{RTT} = D_{RTT} + h (|M_{RTT} - S_{RTT}| - D_{RTT})$$

- Need to minimize processing requirements

- Only 1 counter (regardless of how many packets are extant)

- Counter granularity is typically 500 ms

- Measurement equations have gain

$1/\sqrt{p}$ Law

- Equilibrium window size $w_s = \frac{a}{\sqrt{p}}$
- Equilibrium rate $x_s = \frac{a}{D_s \sqrt{p}}$
- Empirically constant $a \sim 1$
- Verified extensively through simulations and on Internet
- References
 - T.J.Ott, J.H.B. Kemperman and M.Mathis (1996)
 - M.Mathis, J.Semke, J.Mahdavi, T.Ott (1997)
 - T.V.Lakshman and U.Mahdow (1997)
 - J.Padhye, V.Firoin, D.Towsley, J.Kurose (1998)
 - J.Padhye, V.Firoin, D.Towsley (1999)

Implications



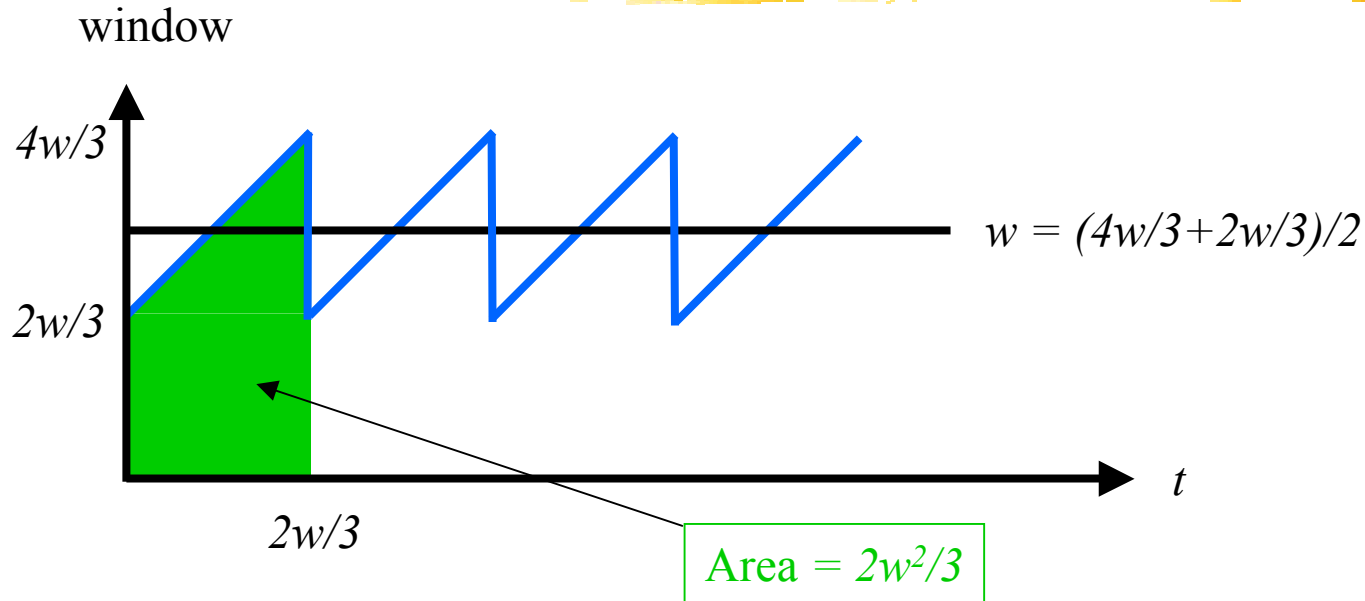
■ Applicability

- Additive increase multiplicative decrease (Reno)
- Congestion avoidance dominates
- No timeouts, e.g., SACK+RH
- Small losses
- Persistent, greedy sources
- Receiver not bottleneck

■ Implications

- Reno equalizes window
- Reno discriminates against long connections

Derivation (I)



- Each cycle delivers $2w^2/3$ packets
- **Assume:** each cycle delivers $1/p$ packets
 - Delivers $1/p$ packets followed by a drop
 - Loss probability = $p/(1+p) \sim p$ if p is small.
- Hence $w = \sqrt{3/2p}$

Derivation (II)

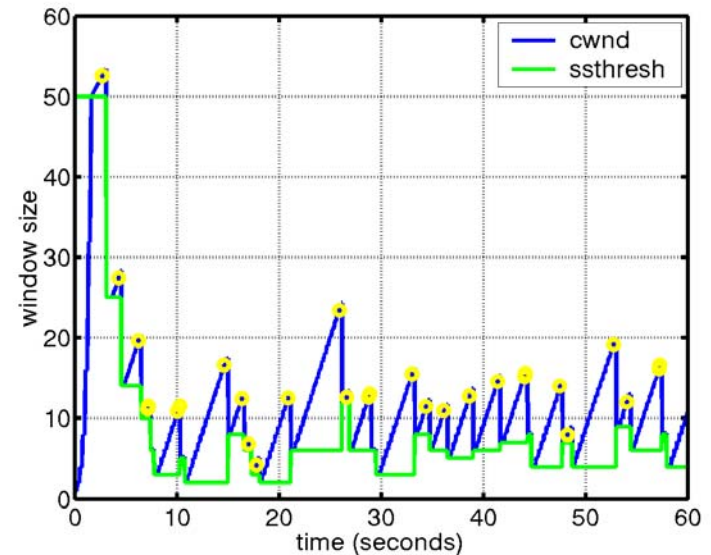
- **Assume:** loss occurs as Bernoulli process rate p
- **Assume:** spend most time in CA
- **Assume:** p is small
- w_n is the window size after

$$w_{n+1} = \begin{cases} w_n / 2, & \text{if a packet is lost (prob. } pw_n) \\ w_n + 1, & \text{if no packet is lost (prob. } (1 - pw_n)) \end{cases}$$

$$\bar{w} = \frac{\bar{w}}{2} p \bar{w} + (\bar{w} + 1)(1 - p \bar{w})$$

$$\bar{w}^2 \approx 2/p$$

$$\bar{w} \approx \sqrt{2/p}$$



Refinement

(Padhye, Firoin, Towsley & Kurose 1998)

- Renewal model including
 - FR/FR with Delayed ACKs (b packets per ACK)
 - Timeouts
 - Receiver awnd limitation

- Source rate

$$x_s = \min \left(\frac{W_r}{D_s}, \frac{1}{D_s \sqrt{\frac{2bp}{3}} + T_o \min \left(1, 3 \sqrt{\frac{3bp}{8}} \right) p(1+32p^2)} \right)$$

- When p is small and W_r is large, reduces to

$$x_s = \frac{a}{D_s \sqrt{p}}$$

Calculating Performance

■ Single link, capacity C , buffer B

■ Window size: $w = f(p)$

■ Loss rate: $p = g(w; C, B)$

■ Find w^* : $w^* = f(g(w^*; C, B))$

■ Example:

■ Window size: $w = 1/\sqrt{p}$

■ Loss rate approx. $p = \frac{[w-C]^+}{w}$

$$w^* = \frac{C + \sqrt{C^2 + 4}}{2}$$

Fixed Point Models

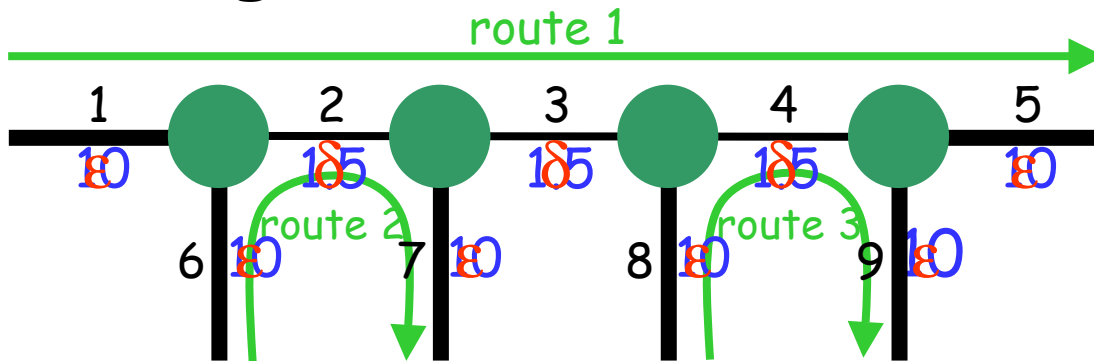
- Mean field theory
 - Solve for a particular source given the mean field
 - Use single source to approximate the mean field
- Generalize previous example
 - Multiple sources
 - Network
 - various routes, RTTs, capacities, ...
 - Arbitrary functions f , and g
- Solve using
 - Repeated substitution
 - Newton-Raphson

Network Formulation

- N links, R routes
- Capacity $c = \{c_j\}$ $j=1, \dots, N$
- Propagation time $t = \{t_j\}$ $j=1, \dots, N$
- Routing matrix $A = \{a_{ij}\}$ $j=1, \dots, N, i=1, \dots, R$
 - $a_{ij} = 1$, if link j is in route i
 - $a_{ij} = 0$, if link j isn't in route i
- Sources per route $n = \{n_i\}$ $i=1, \dots, R$
- MSS per route $m = \{m_i\}$ $i=1, \dots, R$
- Route send rate $s = \{s_i\}$ $i=1, \dots, R$
- Link loss rate $q = \{d_j\}$ $j=1, \dots, N$
- Route loss rate $p = \{p_i\}$ $i=1, \dots, R$

Example Network

- N congested bottlenecks (e.g. 2)



$$\mathbf{t} = (\varepsilon, \delta, \delta, \delta, \varepsilon, \varepsilon, \varepsilon, \varepsilon, \varepsilon)^{\dagger}$$

$$\mathbf{c} = (10, 1.5, 1.5, 1.5, 10, 10, 10, 10, 10)^{\dagger}$$

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Solution

- Estimate RTT delay from propagation time

$$d = 2At$$

(can use queueing delays)

- Route send rates

$$x(w) = (w \cdot n \cdot m) / d$$

- Link rates

$$b(w) = A^t x$$

- Link loss rate

$$q(w;c) = [b - c]^+ / b$$

(can use queueing losses)

- Route loss rate

$$p(w;c) = 1 - e^{A \ln(1 - q(w;c))}$$

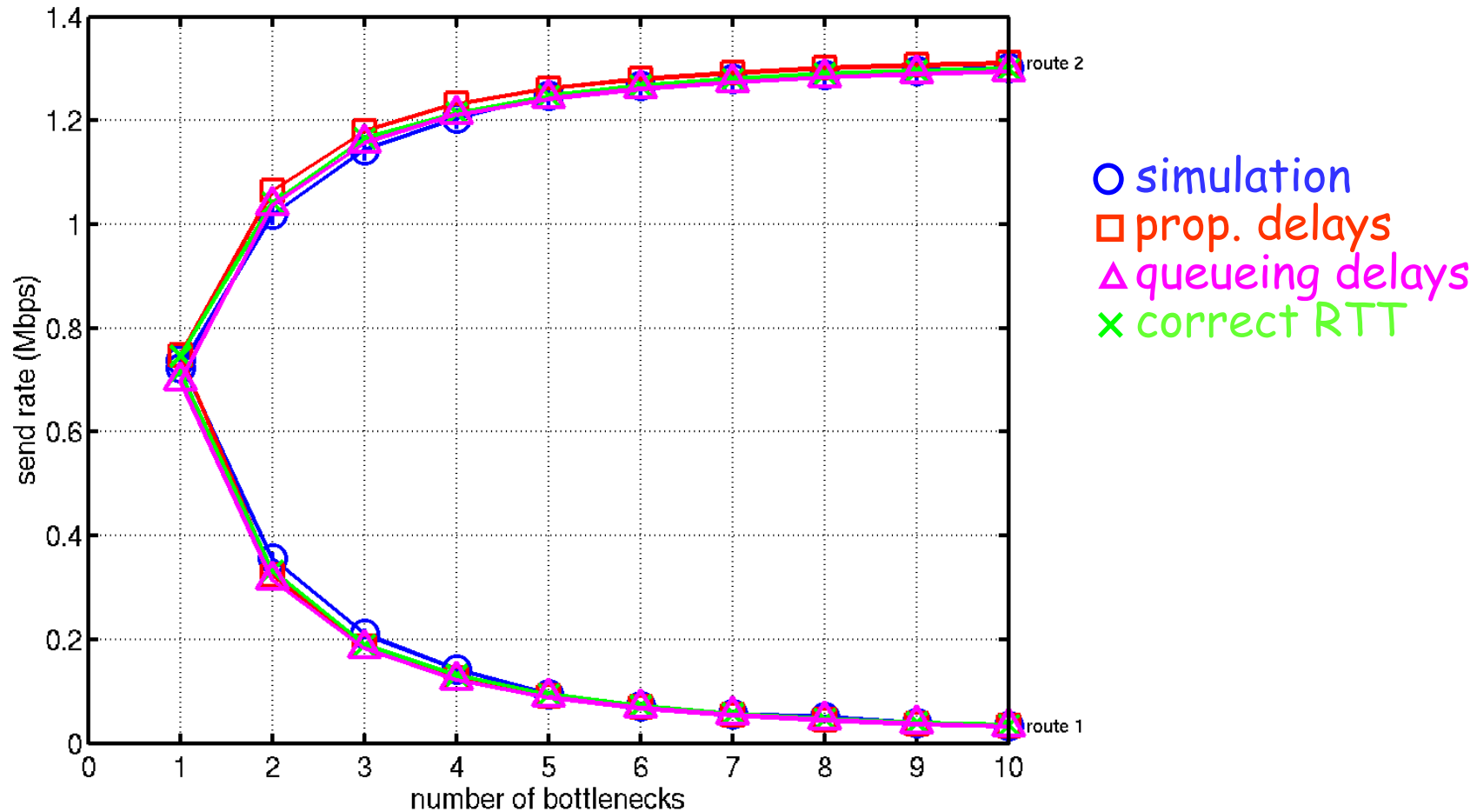
- Window size

$$W^2 p(w;c) - a = 0$$

(could use refined model,
or a transient model)

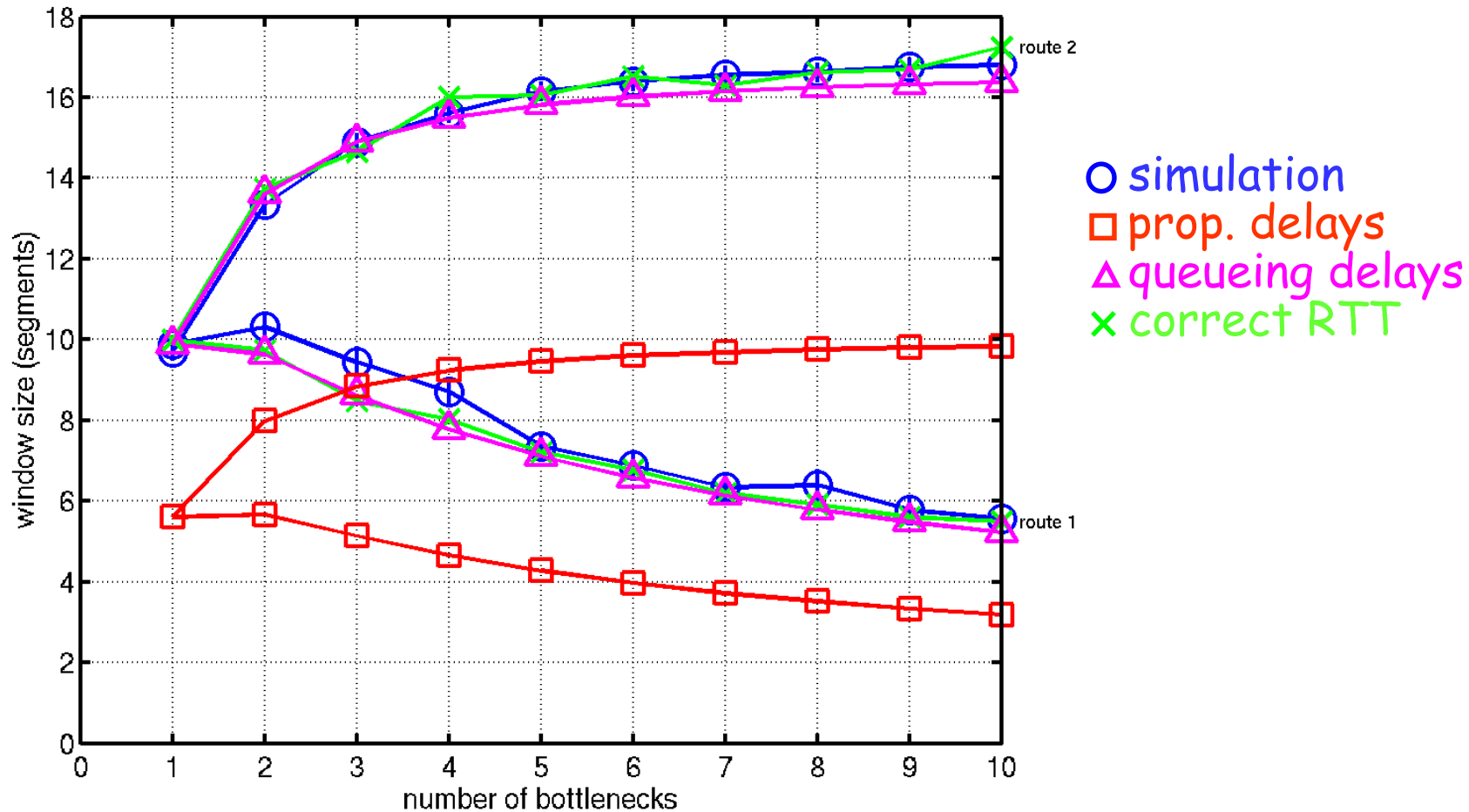
Numerical Example

Send rates



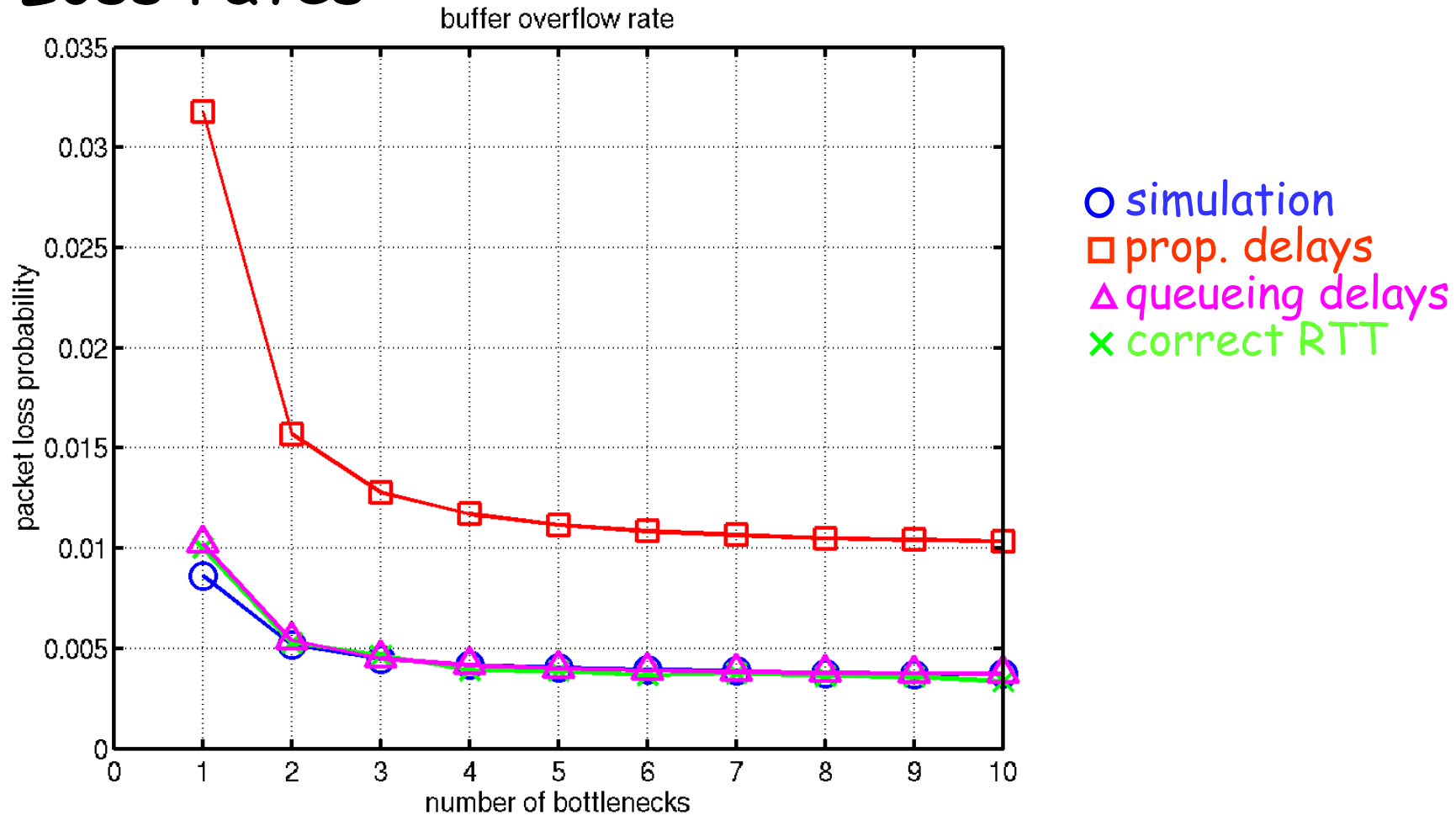
Numerical Example

Window sizes



Numerical Example

Loss rates



Short-lived sources

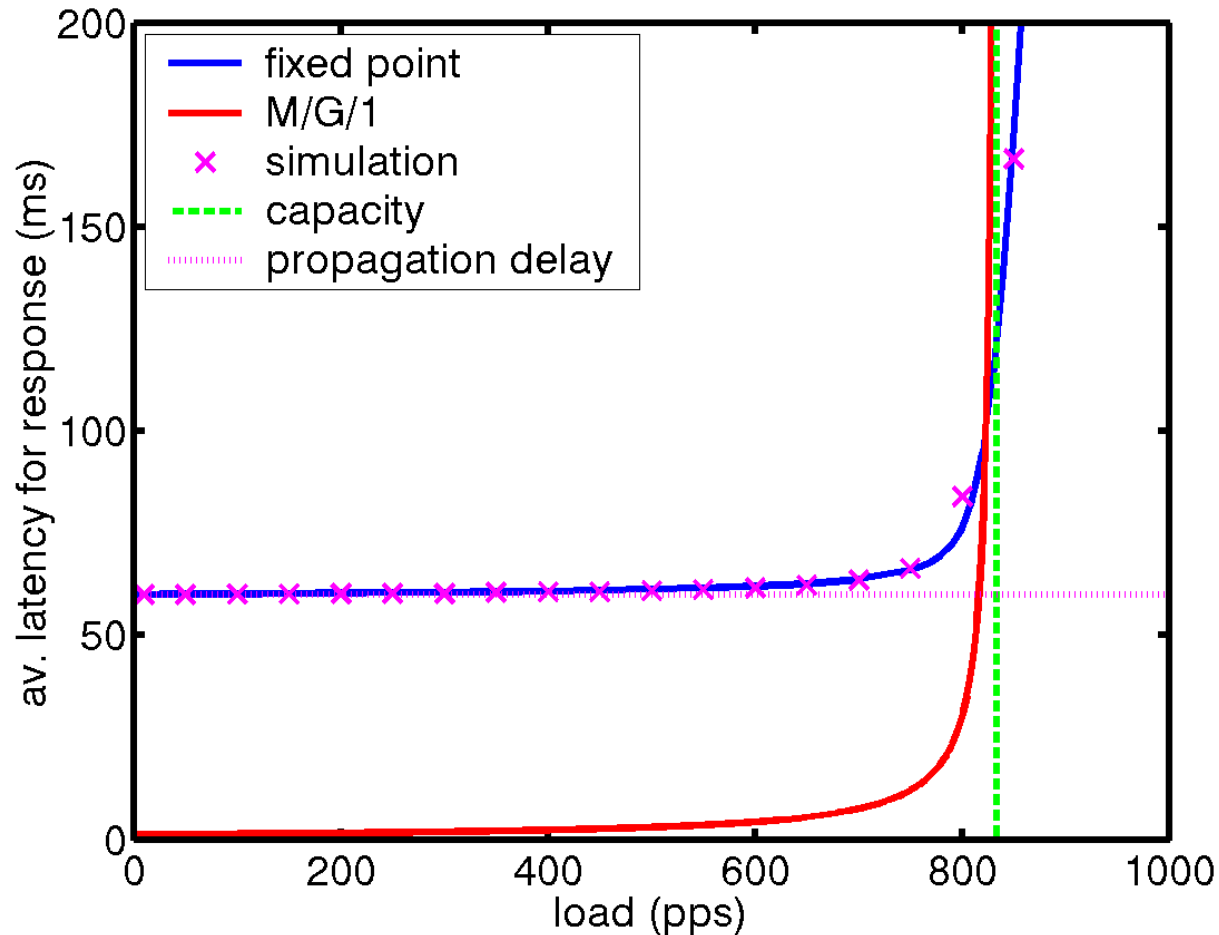
- Heavy-tailed distribution of flow sizes
 - Some really big files elephants
 - Many small files mice
- Persistent model only good for elephants
 - Concentrates on Congestion Avoidance
 - Short lived sources always in Slow Start
 - M/G/1 processor sharing suggested
 - Really we need a new model, e.g.
 - Cardwell, Savage and Anderson, Infocom 2000
 - Sikdar, Kalyanaraman and Vastola, IPCC 2001
 - Mellia, Stoica and Zhang, IEEE Communications Let. 2002

New approach

- Use the loss rate to estimate transfer latency (e.g. from Cardwell *et al*)
- Use transfer latency to compute the number of sessions in progress
 - $M/G/1$ processor sharing queue (for number of sources)
- Use the number of sessions in progress (and their duration) to estimate the load and thence the loss rate
 - $M/G/1/K$ FIFO model (for packets in each buffer)

Simple example

- Poisson arrivals of single packet transfers



Results



- Processor sharing
 - Doesn't get latency right for low load (can't get RTT)
 - Asymptote at capacity
- Even so result is not responsive to congestion!
- Can get a good measure from fixed point approach

Conclusion

- Can use fixed point methods to estimate performance for TCP flow controls
 - Persistent case (based on CA)
 - Short-lived case (based on SS)
- Nice because they generalize to networks
- Need to understand limitations of SS models for TCP window flow controls
 - RTT estimation used in RTO computation
 - In BSD simple because of 500ms timer