
Network Transport Layer:

TCP/Reno Analysis, TCP Cubic, TCP/Vegas

Qiao Xiang

<https://qiaoxiang.me/courses/cnns-xmuf22/index.shtml>

11/08/2022

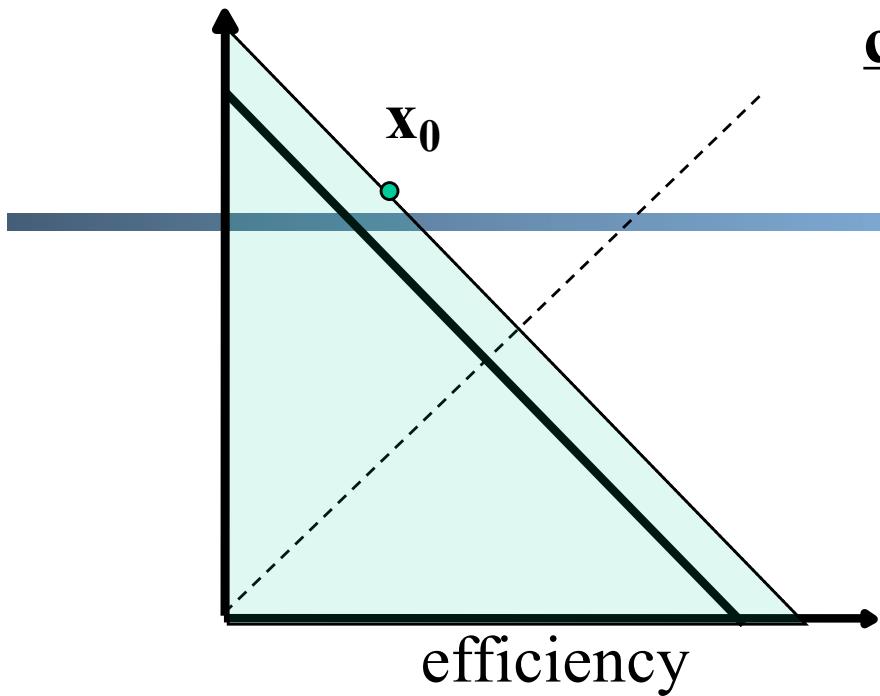
Admin.

- Lab 3 due today
 - By email to the TA
- Lab 4 posted, due on Dec. 8
- Midterm exam: 2:30-4:10pm, Nov. 10

Recap: Transport Design

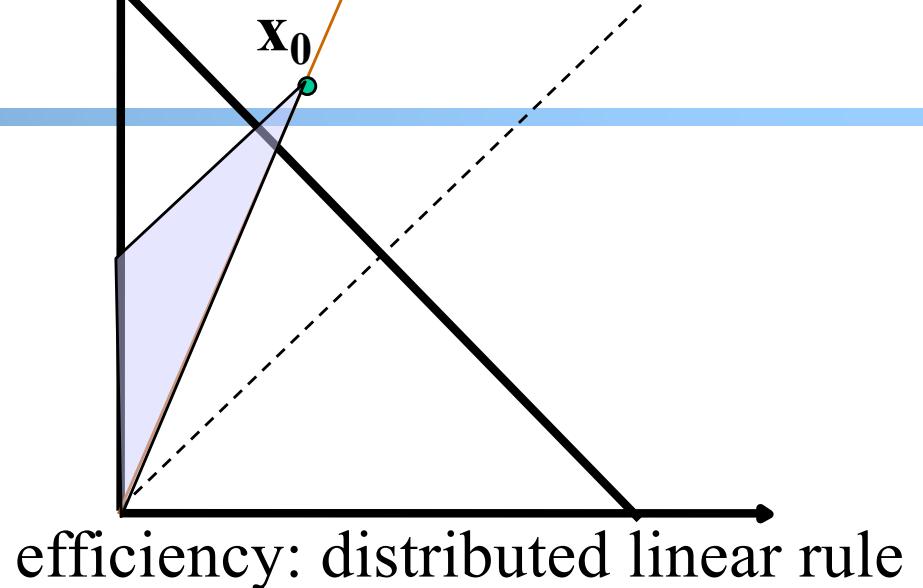
- Basic structure/reliability: sliding window protocols
- Determine the “right” parameters
 - Timeout
 - mean + variation
 - Sliding window size
 - Related w/ congestion control or more generally resource allocation
 - Bad congestion control can lead to congestion collapse (e.g., zombie packets)
 - Goals: **distributed** algorithm to achieve **fairness** and **efficiency**

congestion

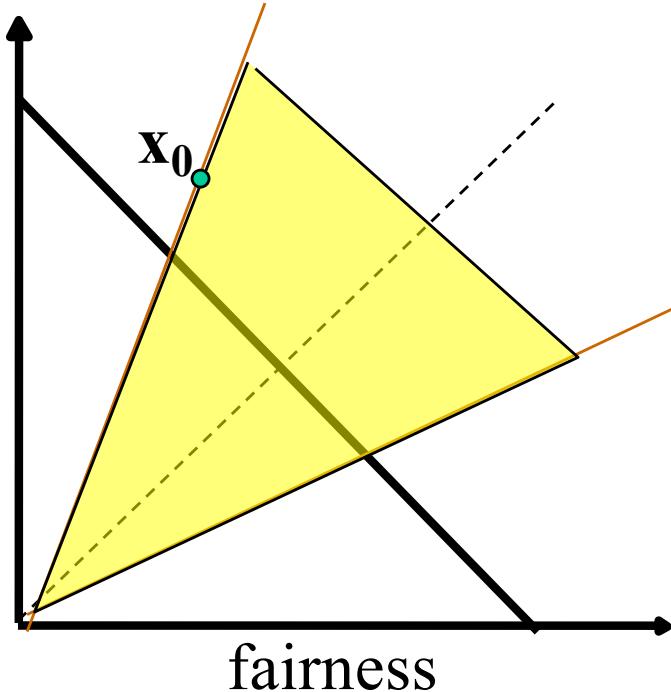


efficiency

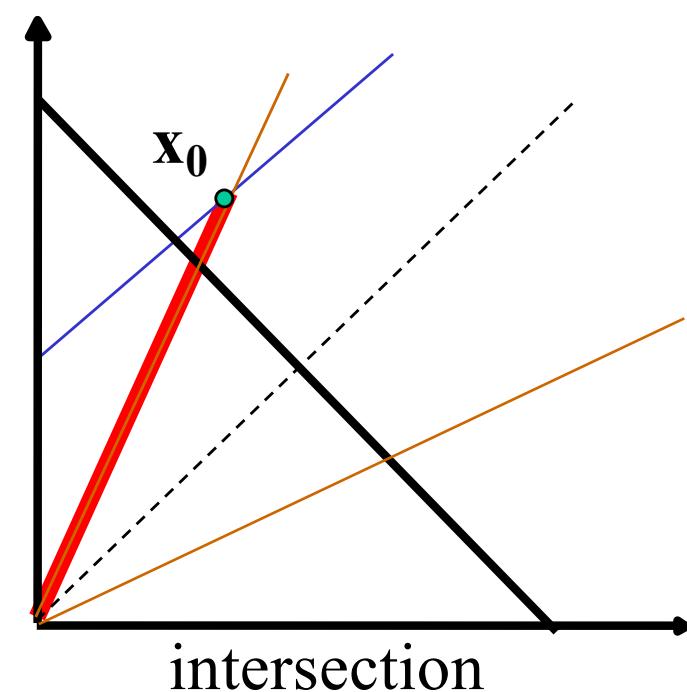
$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{if } d(t) = \text{no cong.} \\ a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.} \end{cases}$$



efficiency: distributed linear rule



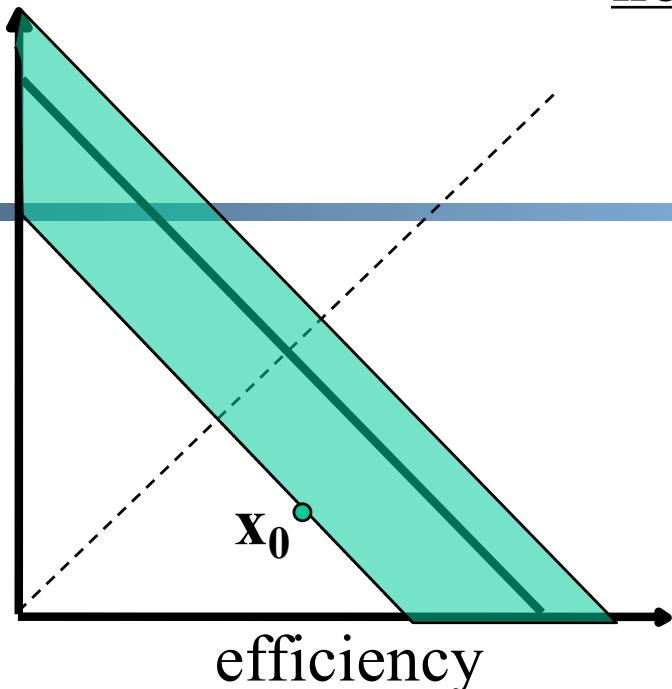
fairness



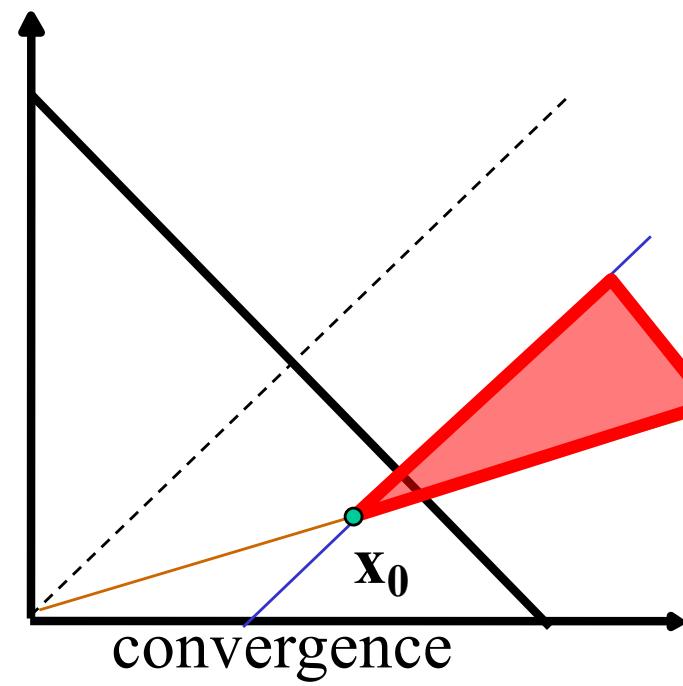
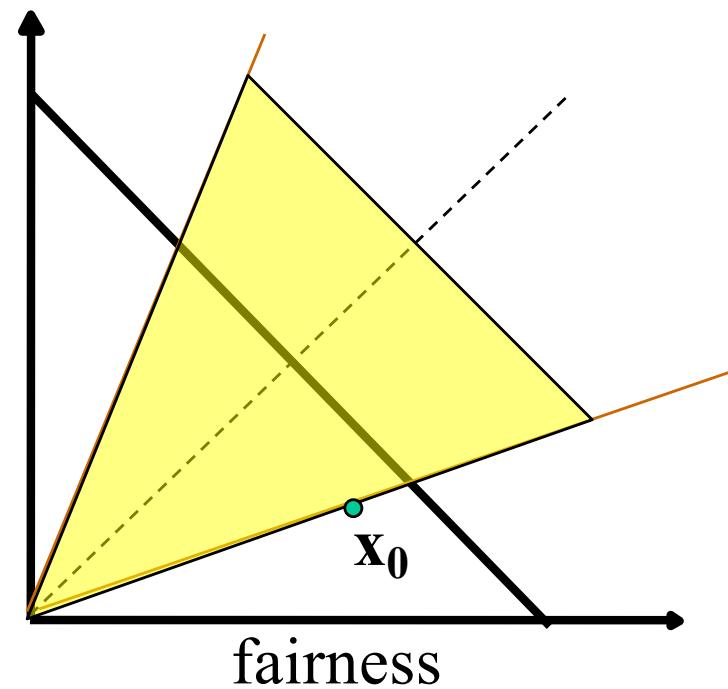
intersection

no-congestion

$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{if } d(t) = \text{no cong.} \\ a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.} \end{cases}$$



efficiency: distributed linear rule



Mapping A(M)I-MD to Protocol

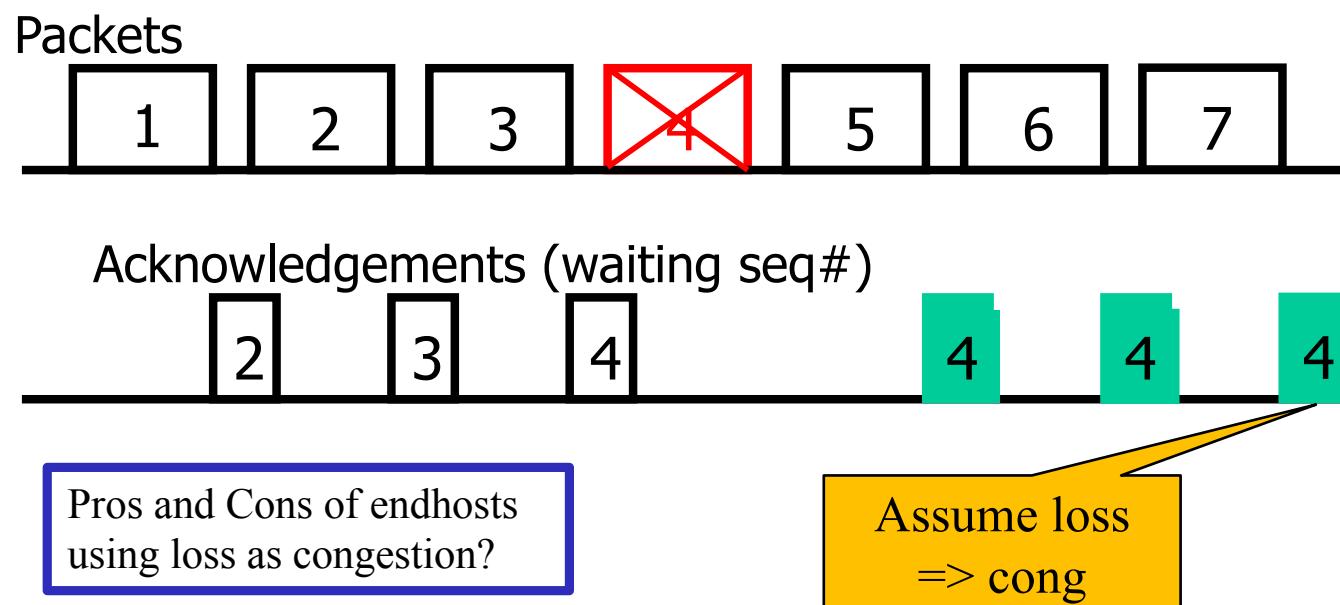
□ Basic questions to look at:

- How to obtain $d(t)$ --the congestion signal?
- What values do we choose for the formula?
- How to map formula to code?

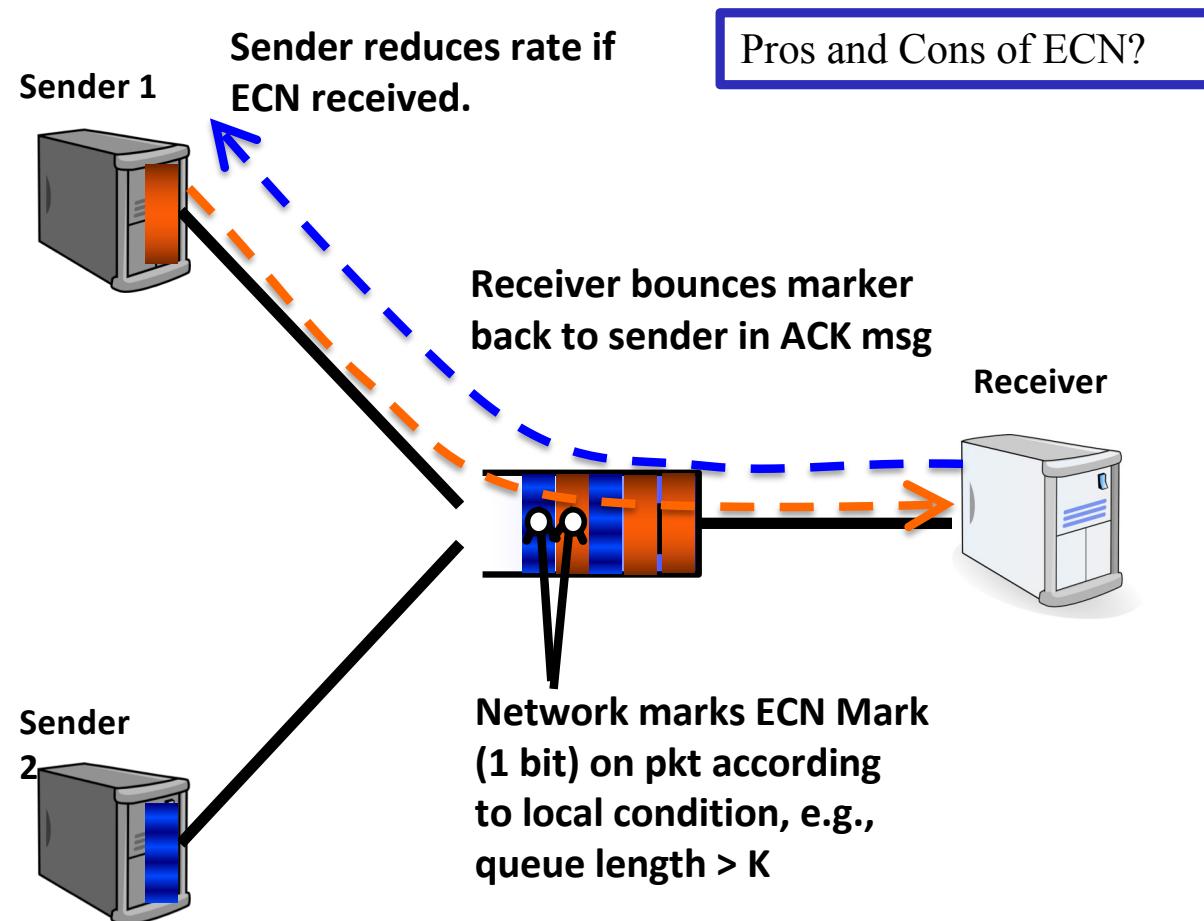
$$x_i(t+1) = \begin{cases} a_I + x_i(t) & \text{if } d(t) = \text{no cong.} \\ b_D x_i(t) & \text{if } d(t) = \text{cong.} \end{cases}$$

Obtain $d(t)$ Approach 1: End Hosts

Consider Loss as Congestion



Obtain $d(t)$ Approach 2: Network Feedback (ECN: Explicit Congestion Notification)



Mapping A(M)I-MD to Protocol

□ Basic questions to look at:

- How to obtain $d(t)$ --the congestion signal?
- What values do we choose for the formula?
- How to map formula to code?

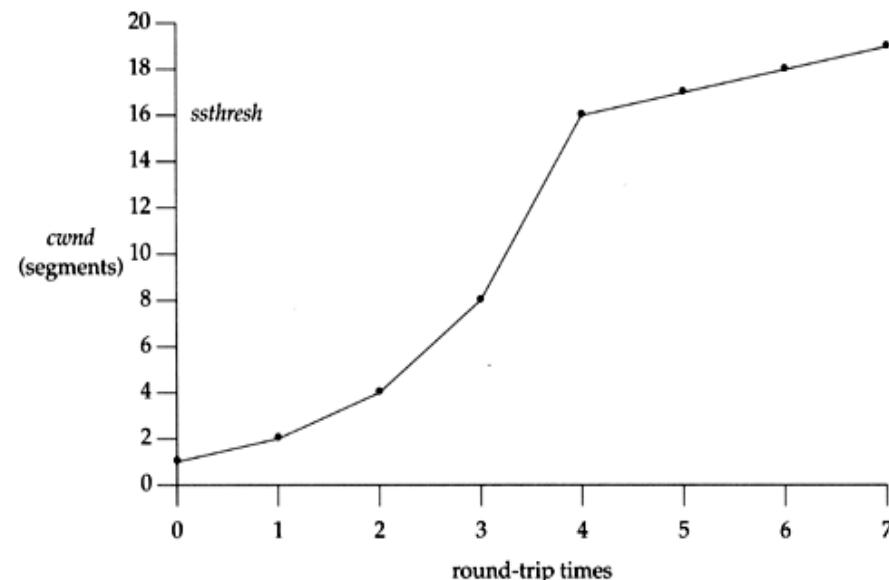
$$x_i(t+1) = \begin{cases} a_I + x_i(t) & \text{if } d(t) = \text{no cong.} \\ b_D x_i(t) & \text{if } d(t) = \text{cong.} \end{cases}$$

TCP/Reno Formulas

- Multiplicative Increase (MI)
 - double *the rate*: $x(t+1) = 2 x(t)$
- Additive Increase (AI)
 - Linear increase *the rate*: $x(t+1) = x(t) + 1$
- Multiplicative decrease (MD)
 - half *the rate*: $x(t+1) = 1/2 x(t)$

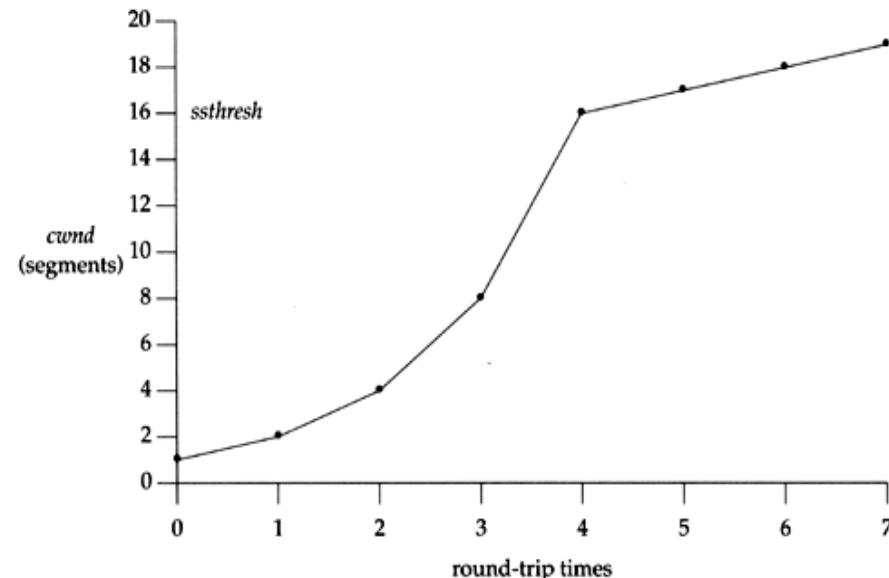
TCP/Reno Formula Switching (Control Structure)

- Two “phases”
 - slow-start
 - Goal: getting to equilibrium gradually but quickly, to get a rough estimate of the optimal of $cwnd$
 - Formula: MI
 - congestion avoidance
 - Goal: Maintains equilibrium and reacts around equilibrium
 - Formula: AI MD



TCP/Reno Formula Switching (Control Structure)

- Important variables:
 - **cwnd**: congestion window size
 - **ssthresh**: threshold between the slow-start phase and the congestion avoidance phase
- If $cwnd < ssthresh$
 - MI
- Else
 - AIMD



MI: Slow Start

- Algorithm: MI
 - double *cwnd* every RTT until network congested
- Goal: getting to equilibrium gradually but quickly, to get a rough estimate of the optimal of *cwnd*

MI: Slow-start

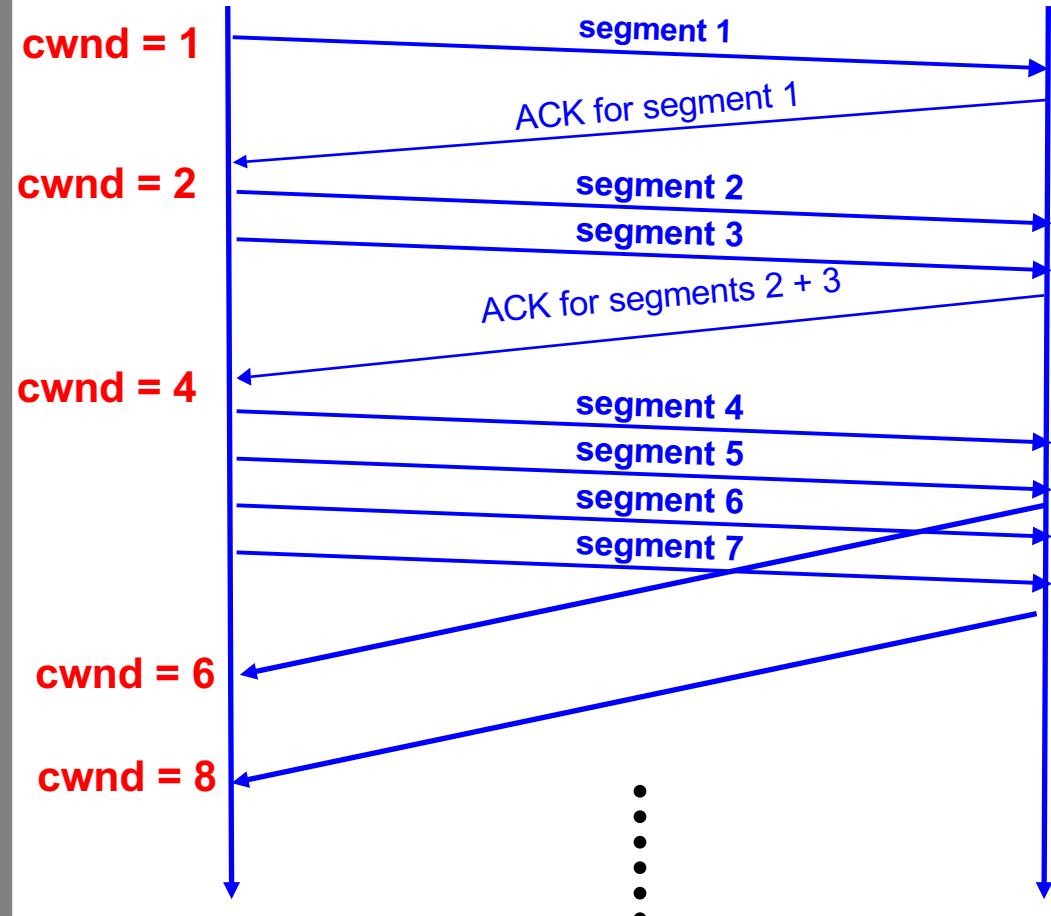
Initially:

$cwnd = 1;$

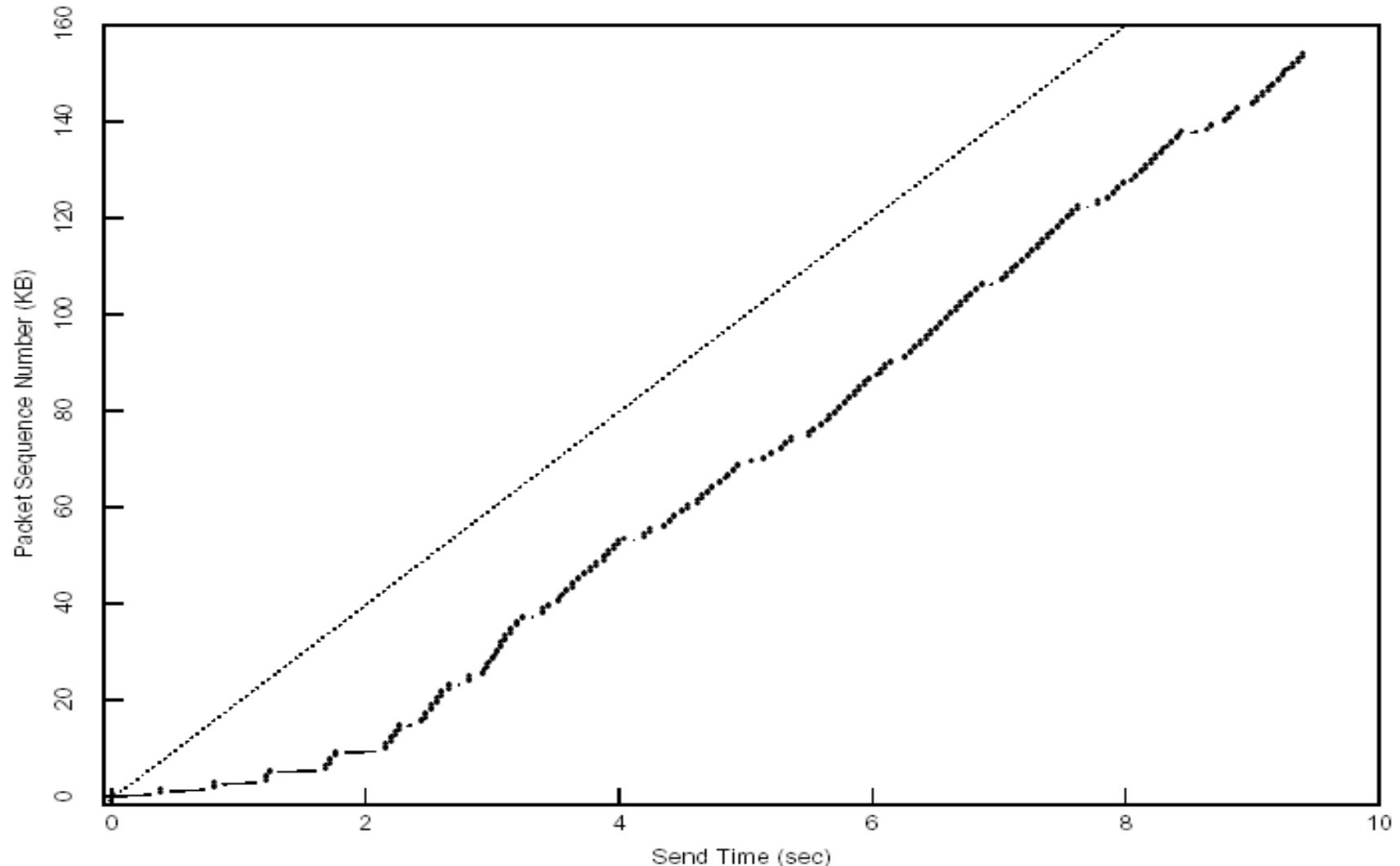
$ssthresh = \text{infinite (e.g., 64K)}$;

For each newly ACKed segment:

```
if ( $cwnd < ssthresh$ )
    /* MI: slow start */
     $cwnd = cwnd + 1;$ 
```



Startup Behavior with Slow-start



AIMD: Congestion Avoidance

- Algorithm: AIMD
 - increases window by 1 per round-trip time (how?)
 - cuts window size
 - to half when detecting congestion by 3DUP
 - to 1 if timeout
 - if already timeout, doubles timeout

- Goal: Maintains equilibrium and reacts around equilibrium

TCP/Reno Full Alg

Initially:

cwnd = 1;

ssthresh = infinite (e.g., 64K);

For each newly ACKed segment:

if (cwnd < ssthresh) // slow start: MI

cwnd = cwnd + 1;

else

// congestion avoidance; AI

cwnd += 1/cwnd;

Triple-duplicate ACKs:

// MD

cwnd = ssthresh = cwnd/2;

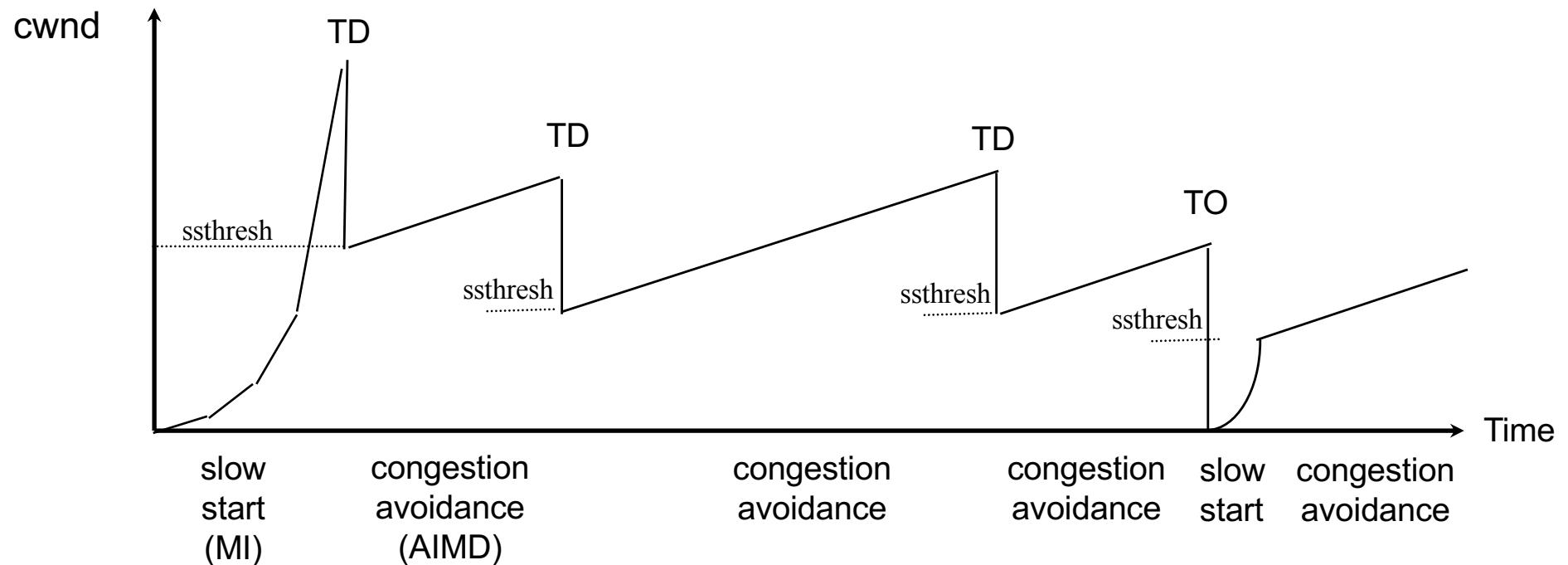
Timeout:

ssthresh = cwnd/2; // reset

cwnd = 1;

(if already timed out, double timeout value; this is called exponential backoff)

TCP/Reno: Big Picture



TD: Triple duplicate acknowledgements

TO: Timeout

Outline

- Admin and recap
 - Transport congestion control
 - what is congestion (cost of congestion)
 - basic congestion control alg.
 - TCP/Reno congestion control
 - design
- *analysis*

Objective

- To understand
 - the throughput of TCP/Reno as a function of RTT (RTT), loss rate (p) and packet size
 - the underlying queue dynamics

- We will analyze TCP/Reno under two different setups

TCP/Reno Throughput Analysis

- Given mean packet loss rate p , mean round-trip time RTT, packet size S
- Consider only the congestion avoidance mode (long flows such as large files)
- Assume no timeout
- Assume mean window size is W_m segments, each with S bytes sent in one RTT:

$$\text{Throughput} = \frac{W_m * S}{\text{RTT}} \text{ bytes/sec}$$

Outline

- Admin and recap
- Transport congestion control
 - what is congestion (cost of congestion)
 - basic congestion control alg.
 - TCP/Reno congestion control
 - design
 - analysis
 - small fish in a big pond
 - loss rate given from the environment

TCP/Reno Throughput Modeling (Fixed, Given Loss Rate)

$$\Delta W = \begin{cases} \frac{1}{W} & \text{if the packet is not lost} \\ -\frac{W}{2} & \text{if packet is lost} \end{cases}$$

$$\text{mean of } \Delta W = (1-p)\frac{1}{W} + p(-\frac{W}{2}) = 0$$

$$\Rightarrow \text{mean of } W = \sqrt{\frac{2(1-p)}{p}} \approx \frac{1.4}{\sqrt{p}}, \text{ when } p \text{ is small}$$

$$\Rightarrow \text{throughput} \approx \frac{1.4S}{RTT\sqrt{p}}, \text{ when } p \text{ is small}$$

This is called the TCP throughput sqrt of loss rate law.

Exercise: Application of Analysis

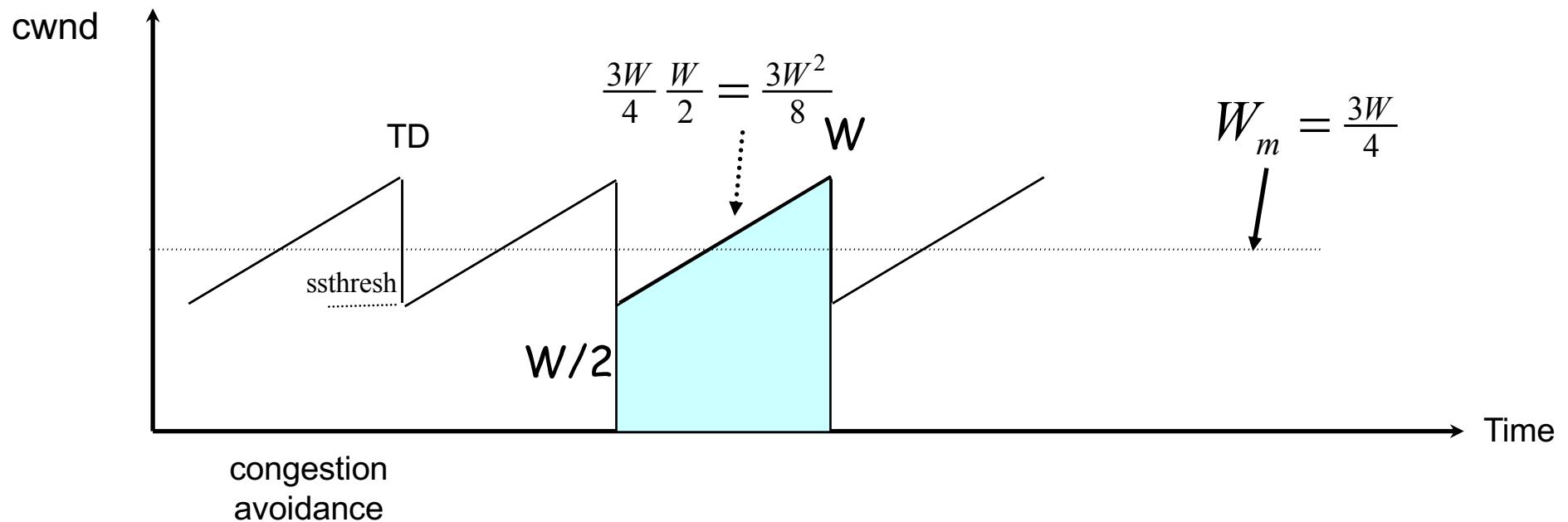
- State of art network link can reach 100 Gbps. Assume packet size 1250 bytes, RTT 100 ms, what is the highest packet loss rate to still reach 100 Gbps?

tcp-reno-tput.xlsx

Outline

- Admin and recap
- Transport congestion control
 - what is congestion (cost of congestion)
 - basic congestion control alg.
 - TCP/Reno congestion control
 - design
 - analysis
 - small fish in a big pond
 - **big fish in small pond**
 - growth causes losses

TCP/Reno Throughput Modeling: Relating W with Loss Rate p



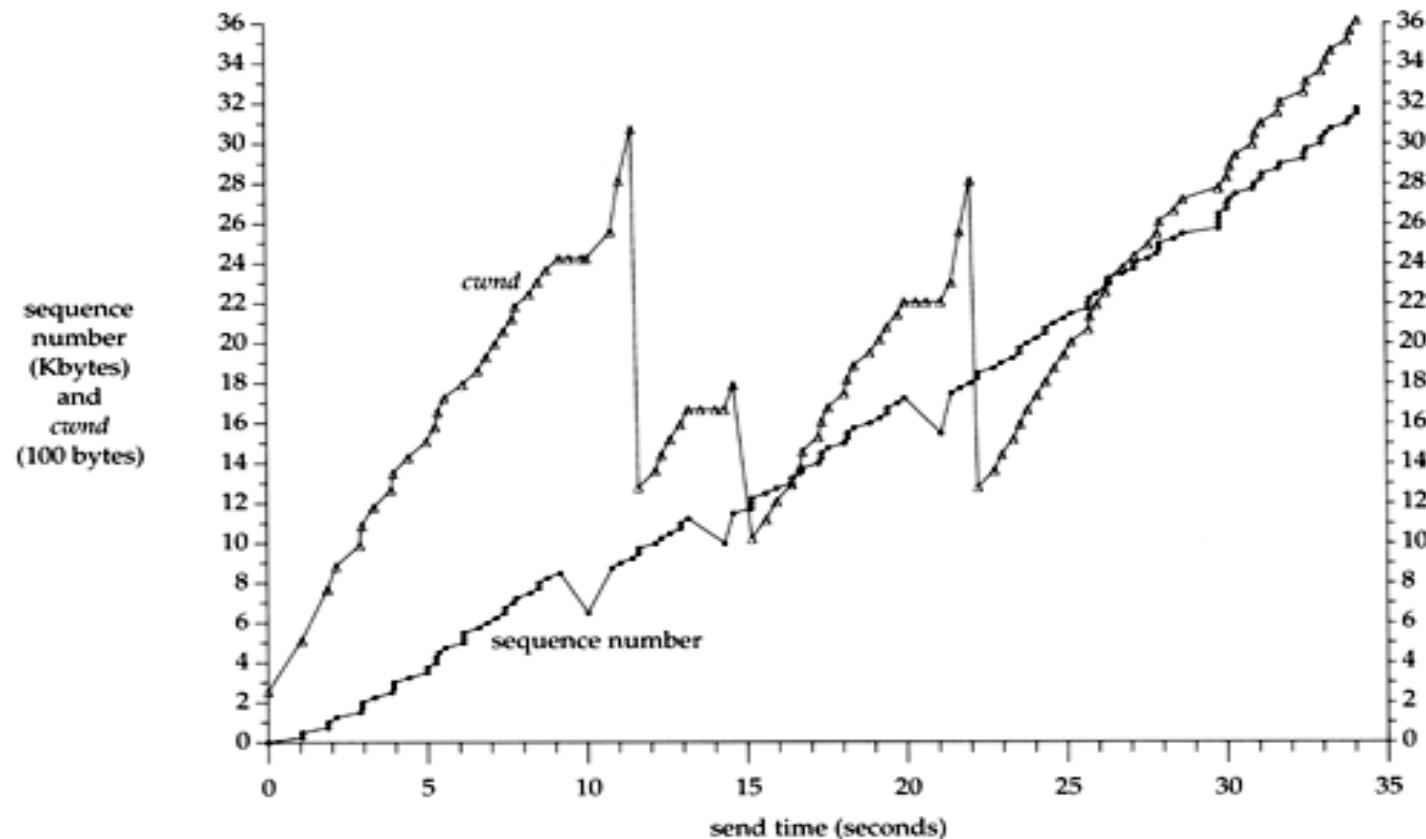
$$\text{Total packets sent per cycle} = (W/2 + W)/2 * W/2 = 3W^2/8$$

$$\text{Assume one loss per cycle} \Rightarrow p = 1/(3W^2/8) = 8/(3W^2)$$

$$\Rightarrow W = \frac{\sqrt{8/3}}{\sqrt{p}} = \frac{1.6}{\sqrt{p}}$$

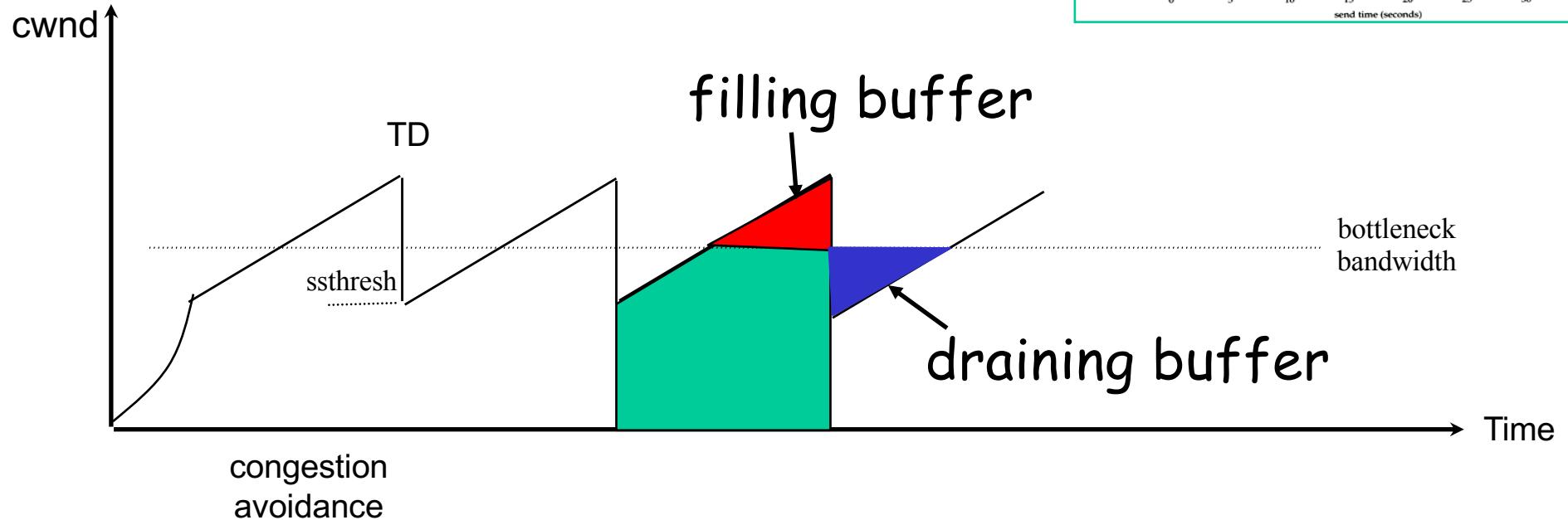
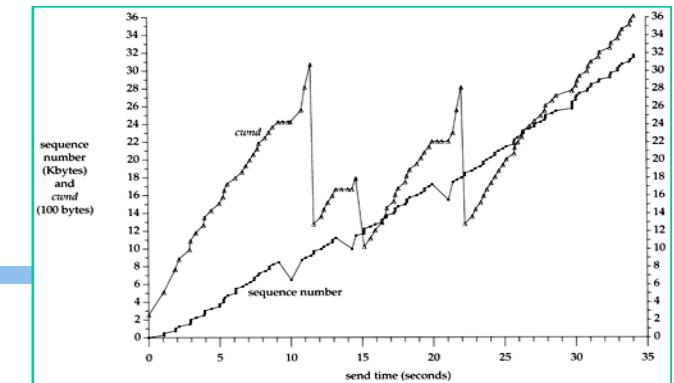
$$\Rightarrow \text{throughput} = \frac{S}{RTT} \frac{3}{4} \frac{1.6}{\sqrt{p}} = \boxed{\frac{1.2S}{RTT \sqrt{p}}} \quad 26$$

A Puzzle: cwnd and Rate of a TCP Session



Question: although cwnd fluctuates widely (i.e., cut to half), why can the sending rate stay relatively smooth?

TCP/Reno Queueing Dynamics



If the buffer at the bottleneck is large enough, the buffer is never empty (not idle), during the cut-to-half to "grow-back" process.

Exercise: How big should the buffer be to achieve full utilization?

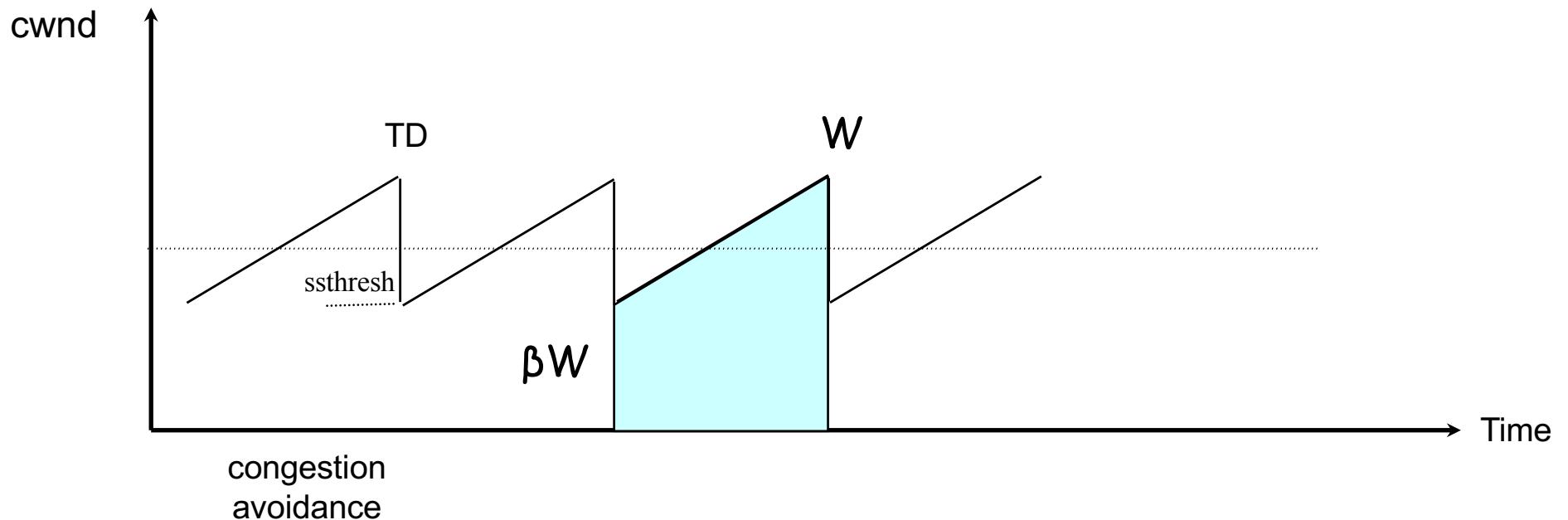
Design

- Assume a generic AIMD alg:
 - increase to $W + \alpha$ after each successful RTT
 - reduce to βW after each loss event

- Q: What value β gives higher utilization
(assume small/zero buffer)?

- Q: Assume picking a high value β , how to make the alg TCP friendly (same throughput as $\alpha=1, \beta=0.5$)?

Generic AIMD and TCP Friendliness



$$\text{Total packets sent per cycle} = \frac{\beta W + W}{2} \cdot \frac{(1-\beta)W}{\alpha} = \frac{(1-\beta)(1+\beta)}{2\alpha} W^2$$

$$\text{Assume one loss per cycle } p = \frac{2\alpha}{(1-\beta)(1+\beta)W^2} \quad W = \sqrt{\frac{2\alpha}{(1-\beta)(1+\beta)p}}$$

$$\text{tput} = \frac{W_m S}{RTT} = \frac{S}{RTT} \cdot \frac{(1+\beta)W}{2} = \frac{S}{RTT} \sqrt{\frac{\alpha(1+\beta)}{2(1-\beta)p}}$$

$$\text{TCP friendly} \Rightarrow \alpha = 3 \frac{1-\beta}{1+\beta}$$

Outline

- Admin and recap
- Transport congestion control
 - what is congestion (cost of congestion)
 - basic congestion control alg.
 - TCP/Reno congestion control
 - TCP Cubic

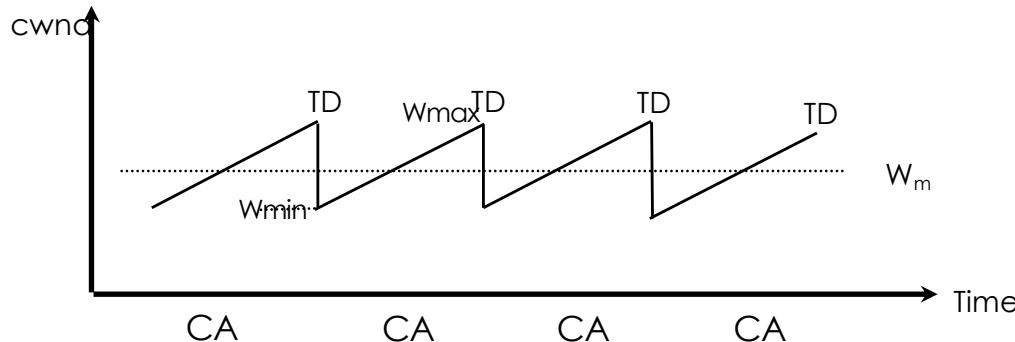
TCP Cubic

- Designed in 2008
- Default for Linux
- Most sockets in MAC appear to use cubic as well
 - `sw_vers`
 - `sysctl -a`

TCP Cubic Goals

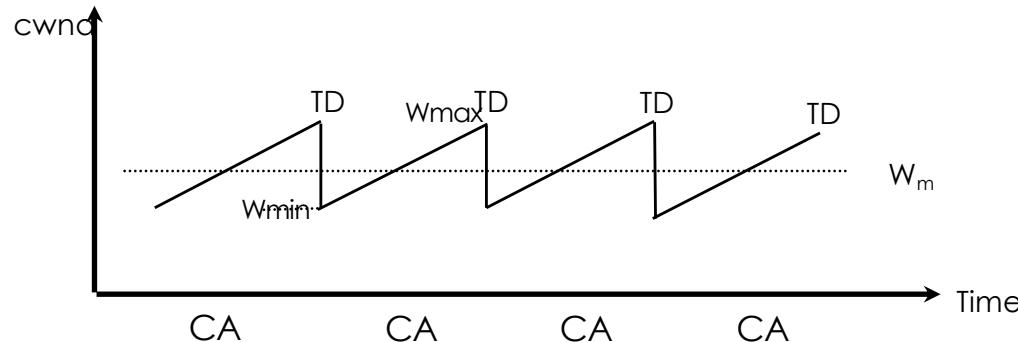
- Improve TCP efficiency over fast, long-distance links
Smaller reduction, longer stay at BDP, faster than linear increase---cubic function
- TCP friendliness
Follows TCP if TCP gives higher rate
- Fairness of flows w/ different RTTs
Window growth depends on real-time (from congestion-epoch through synchronized losses)

TCP BIC Algorithm



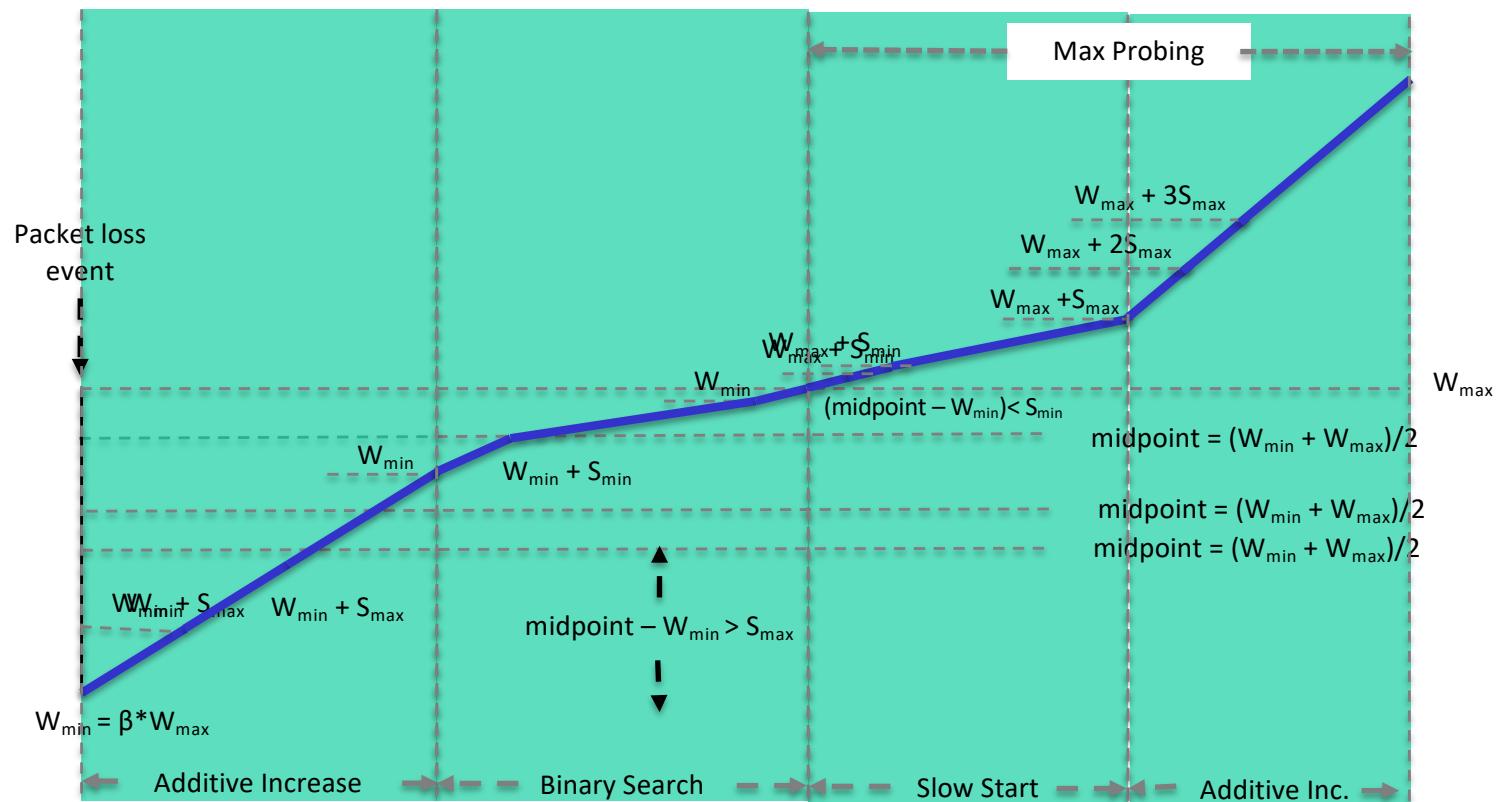
- Setting
 - W_{max} = cwnd size before reduction
 - Too big
 - $W_{min} = \beta * W_{max}$ - just after reduction, where β is multiplicative decrease factor
 - Small
- Basic idea
 - binary search between W_{max} and W_{min}

TCP BIC Algorithm: Issues



- Pure binary search (jump from W_{min} to $(W_{max} \text{ and } W_{min})/2$) may be too aggressive
 - Use a large step size S_{max}
- What if you grow above W_{max} ?
 - Use binary growth (slow start) to probe more

TCP BIC Algorithm



TCP BIC Algorithm

```
while (cwnd < Wmax) {  
    if ( (midpoint - Wmin) > Smax )  
        cwnd = cwnd + Smax  
    else  
        if ((midpoint - Wmin) < Smin)  
            cwnd = Wmax  
        else  
            cwnd = midpoint  
    if (no packet loss)  
        Wmin = cwnd  
    else  
        Wmin = β*cwnd  
        Wmax = cwnd  
    midpoint = (Wmax + Wmin)/2  
}
```

Additive Increase

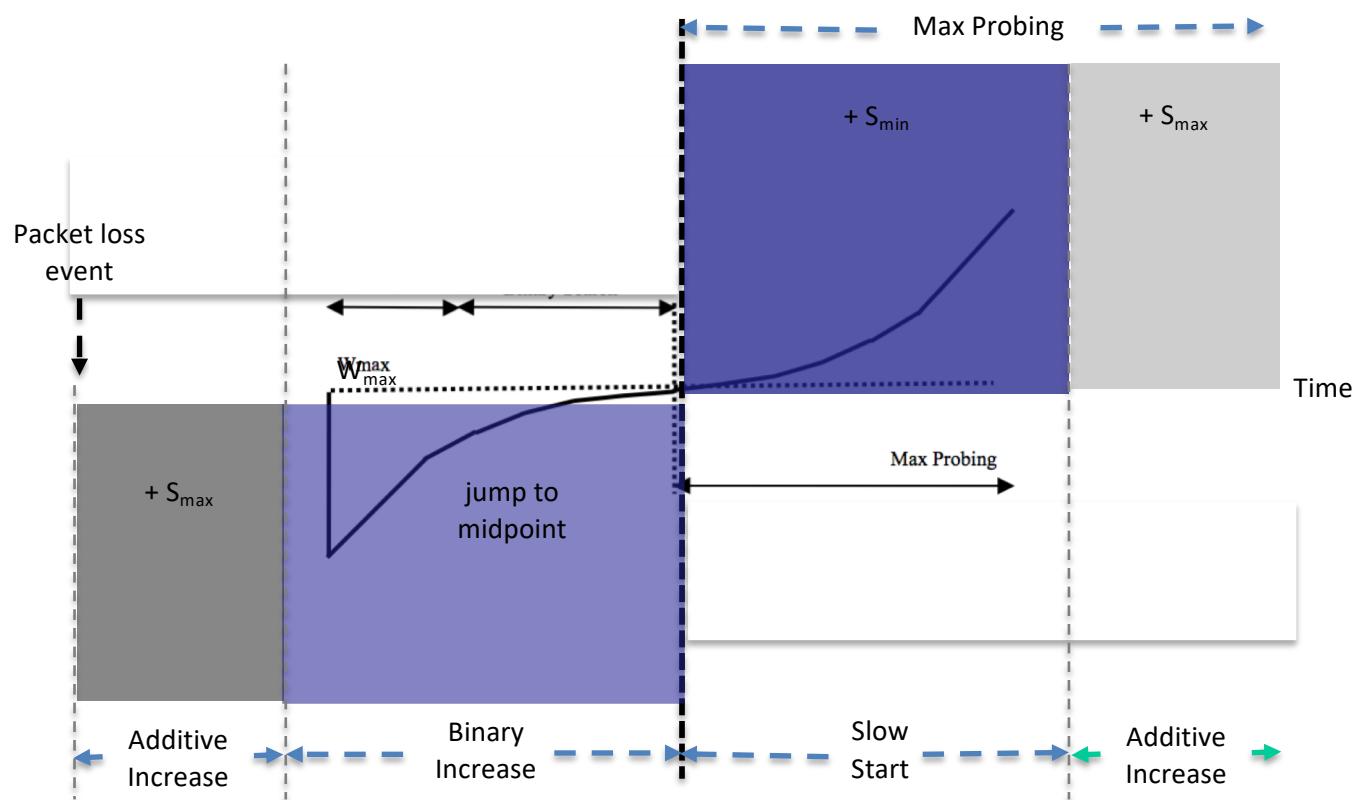
Binary Search

TCP BIC Algorithm: Probe

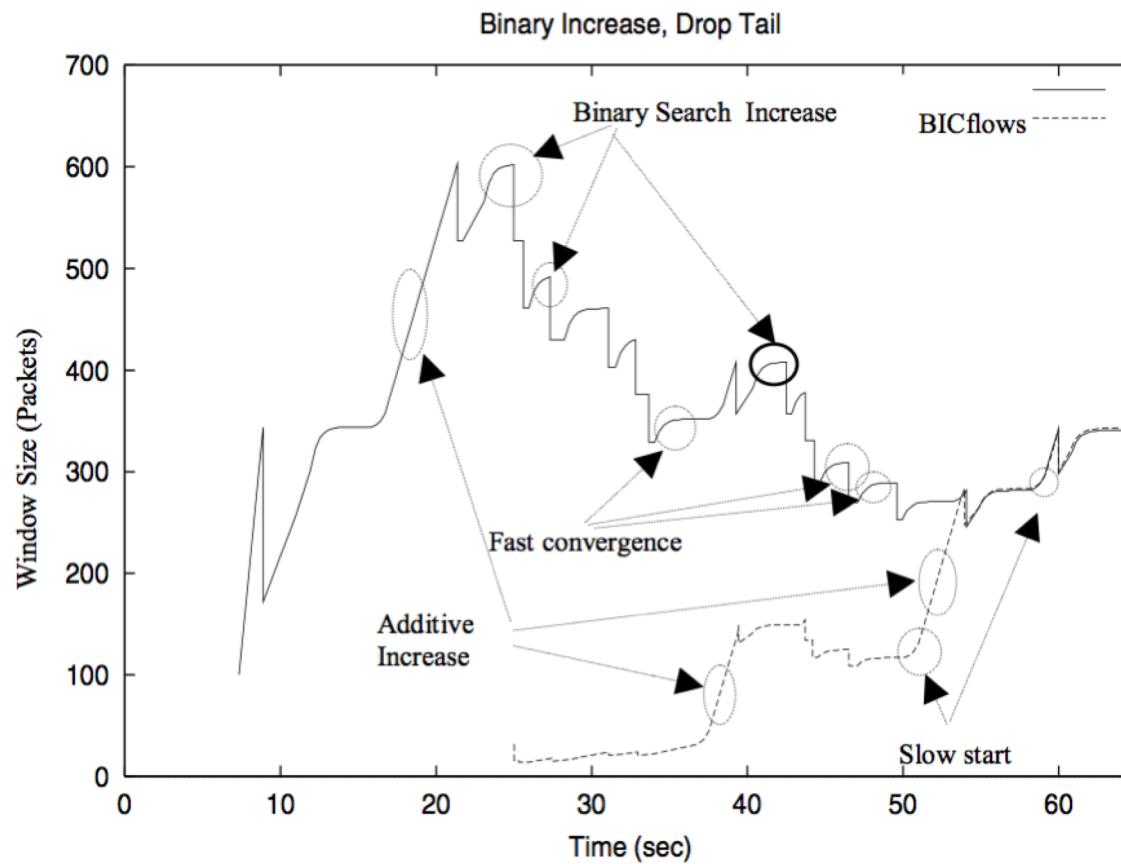
```
while (cwnd >= Wmax) {  
    if (cwnd < Wmax + Smax)  
        cwnd = cwnd + Smin  
    else  
        cwnd = cwnd + Smax → Slow growth  
    if (packet loss)  
        Wmin = β * cwnd  
        Wmax = cwnd  
    } → Fast growth  
}
```

Max Probing

TCP BIC - Summary



TCP BIC in Action



TCP BIC Analysis

□ Advantages

- *Faster convergence at large gap*
- *Slower growth at convergence to avoid timeout*

□ Issues

- Still depend on RTT
- Complex growth function

More details: <http://www.land.ufrj.br/~classes/coppe-redes-2007/projeto/BIC-TCP-infocom-04.pdf>

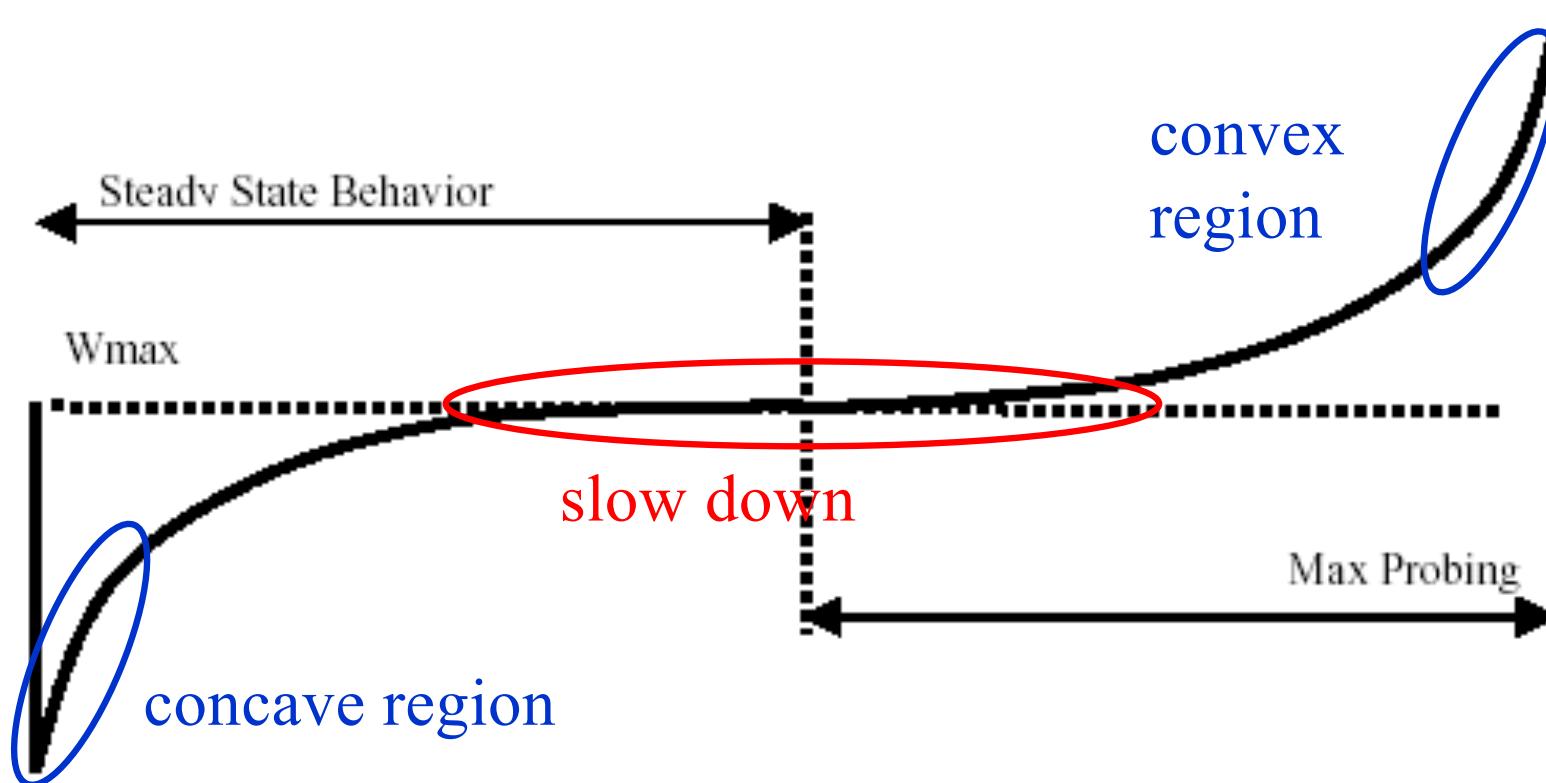
Cubic High-Level Structure

- If (received ACK && state == cong avoid)
 - Compute $W_{\text{cubic}}(t+RTT)$.
 - If $cwnd < W_{\text{TCP}}$
 - Cubic in TCP mode
 - If $cwnd < W_{\text{max}}$
 - Cubic in concave region
 - If $cwnd > W_{\text{max}}$
 - Cubic in convex region

$$\beta' = 1 - \beta$$

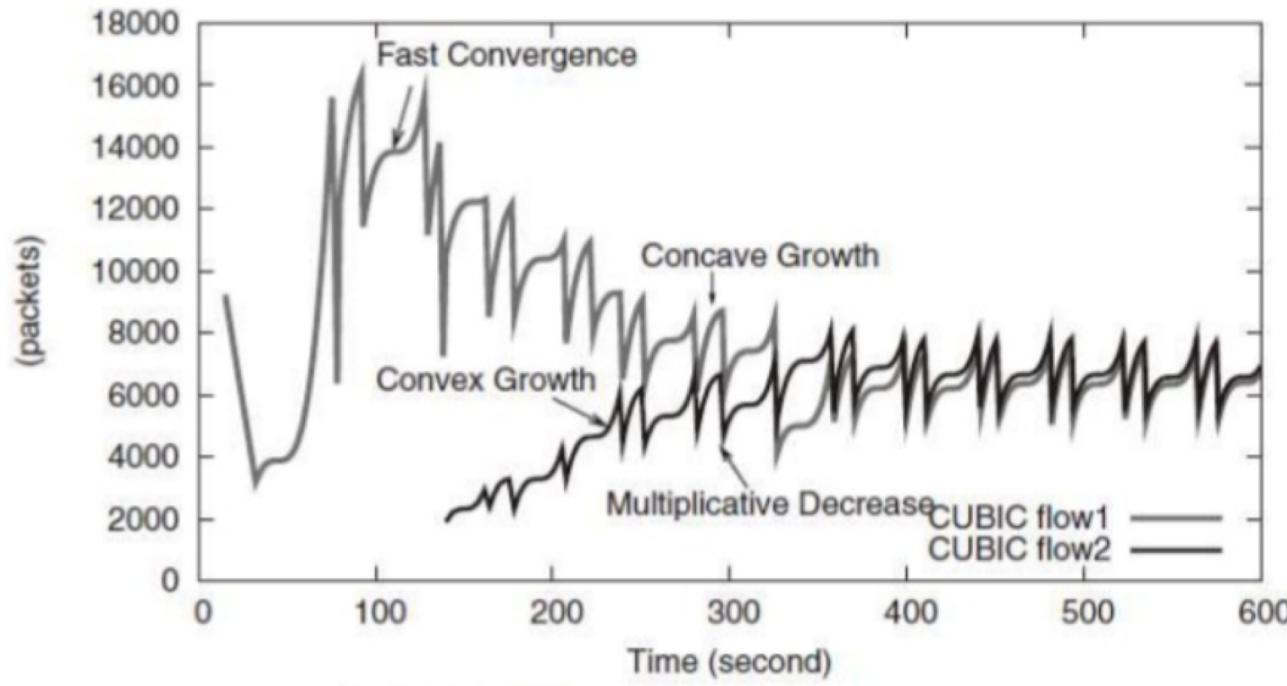
The Cubic function

$$W_{tcp(t)} = W_{max} \beta' + 3 \frac{1-\beta'}{1+\beta'} \frac{t}{RTT}$$

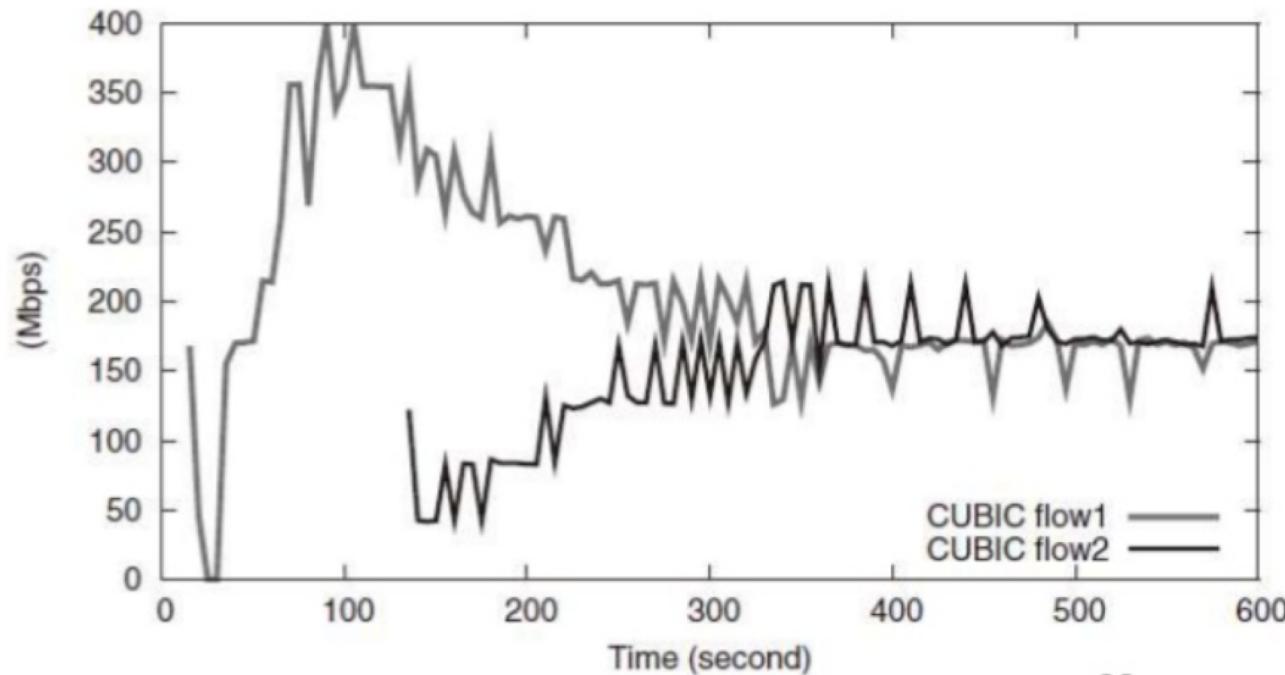


$$W_{cubic} = C(t - K)^3 + W_{max} \quad K = \sqrt[3]{W_{max} \beta / C}$$

where C is a scaling factor, t is the elapsed time from the last window reduction, and β is a constant multiplication decrease factor



(a) CUBIC window curves.

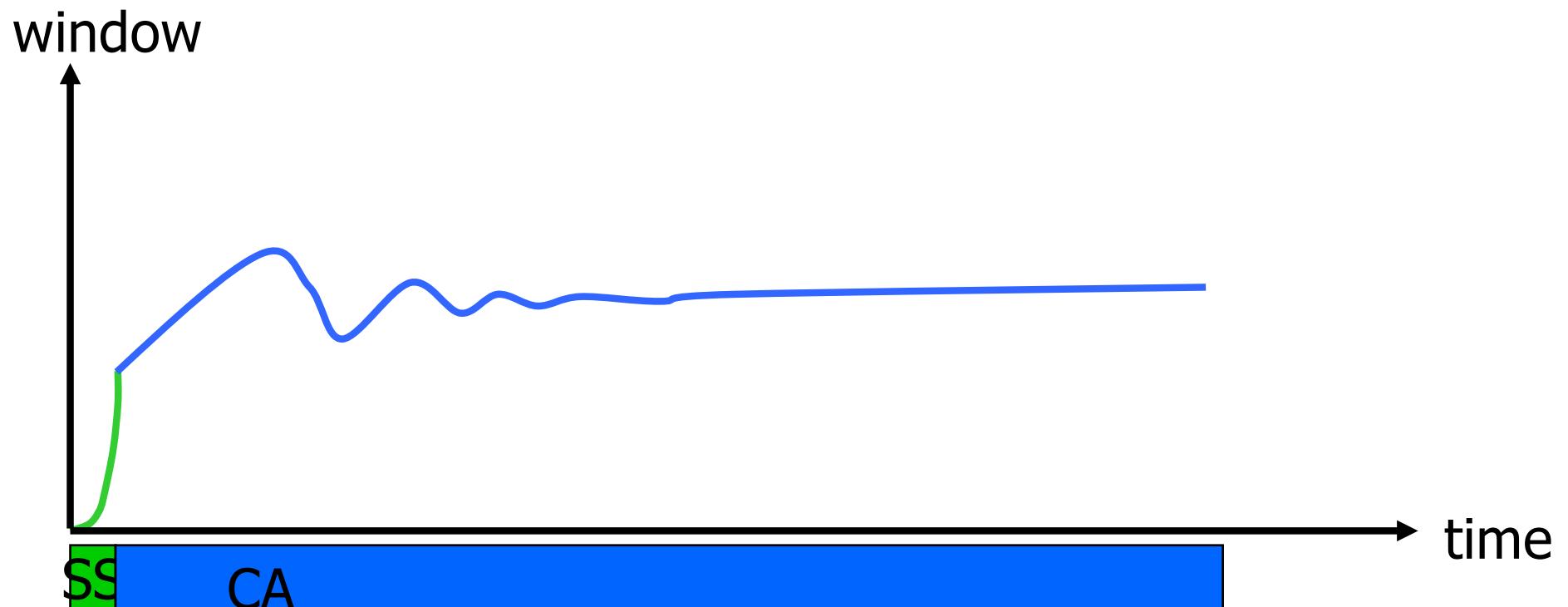


(b) Throughput of two CUBIC flows.²³

Outline

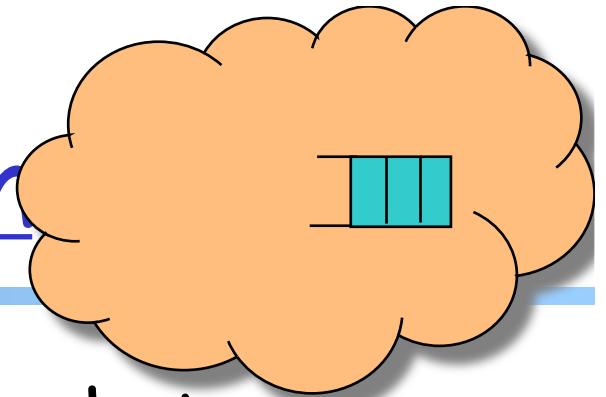
- Admin and recap
- Transport congestion control
 - what is congestion (cost of congestion)
 - basic congestion control alg.
 - TCP/Reno congestion control
 - TCP Cubic
 - TCP/Vegas

TCP/Vegas (Brakmo & Peterson 1994)

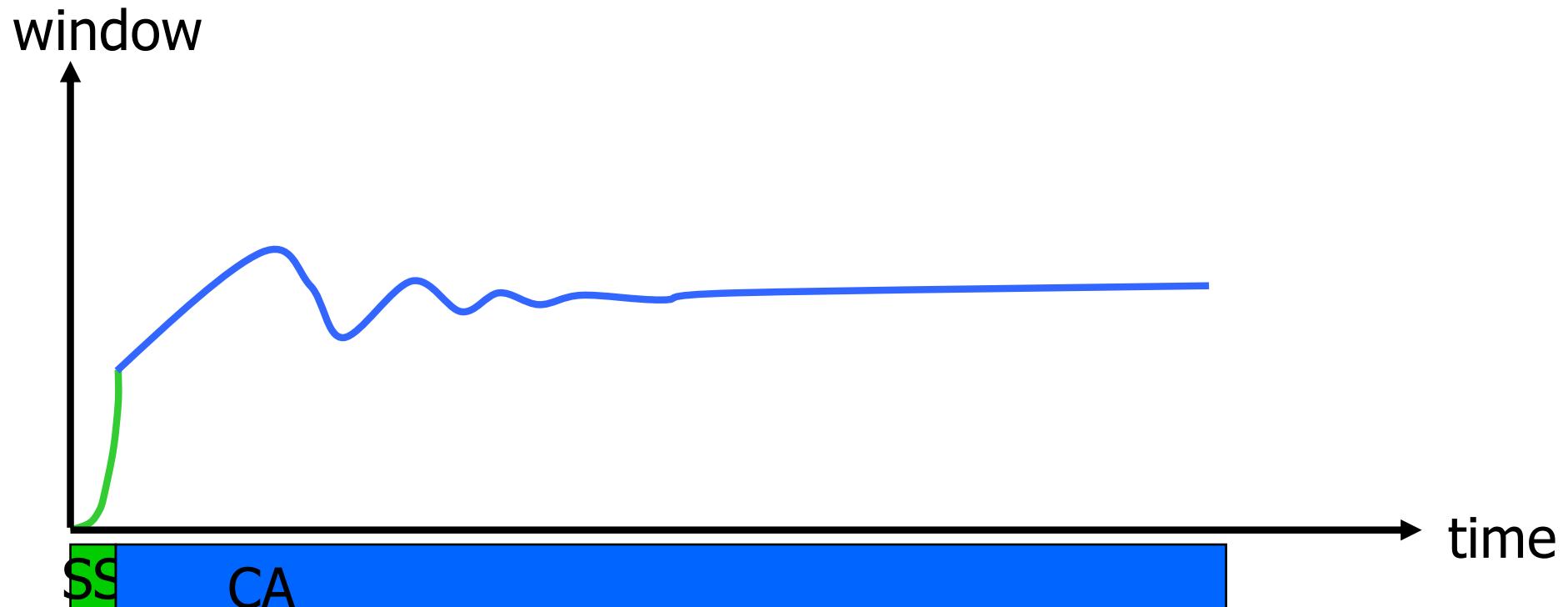


- Idea: try to detect congestion by **delay before loss**
- Objective: not to overflow the buffer; instead, try to maintain a **constant** number of packets in the bottleneck queue

TCP/Vegas: Key Questions



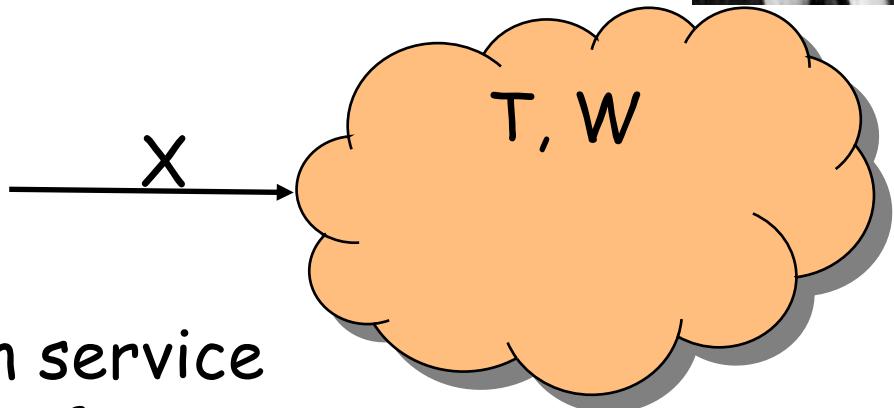
- How to estimate the number of packets queued in the bottleneck queue?



Recall: Little's Law

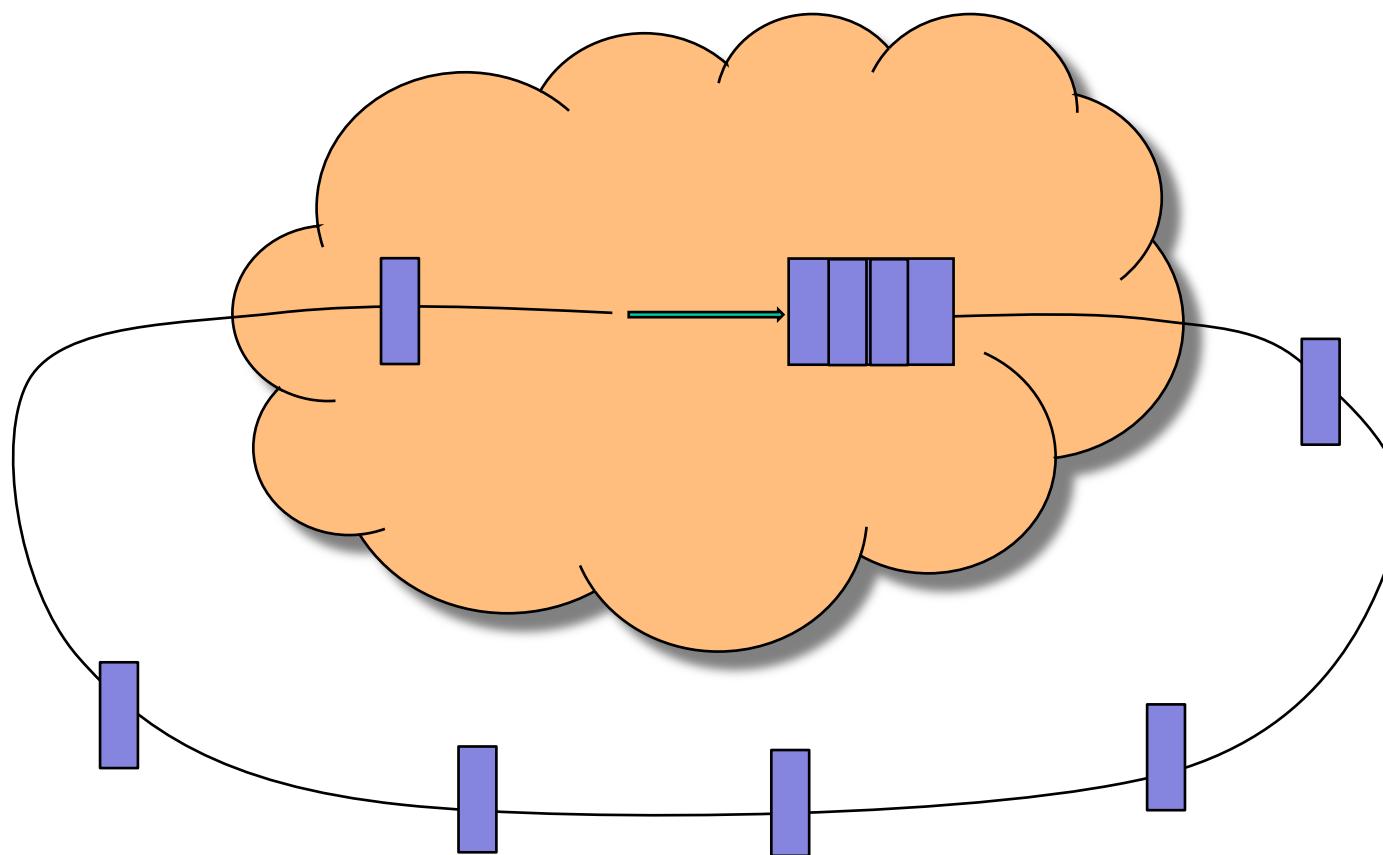


- For any system with no or (low) loss.
- Assume
 - mean arrival rate X , mean service time T , and mean number of requests in the system W
- Then relationship between W , X , and T :

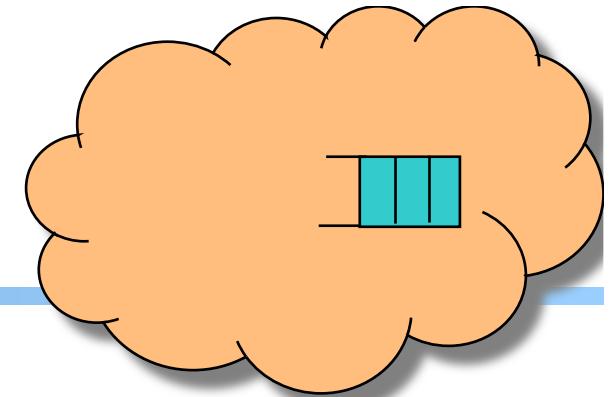


$$W = XT$$

Estimating Number of Packets in the Queue



TCP/Vegas CA algorithm



$$T = T_{\text{prop}} + T_{\text{queueing}}$$

Applying Little's Law:

$$x_{\text{vegas}} T = x_{\text{vegas}} T_{\text{prop}} + x_{\text{vegas}} T_{\text{queueing}},$$

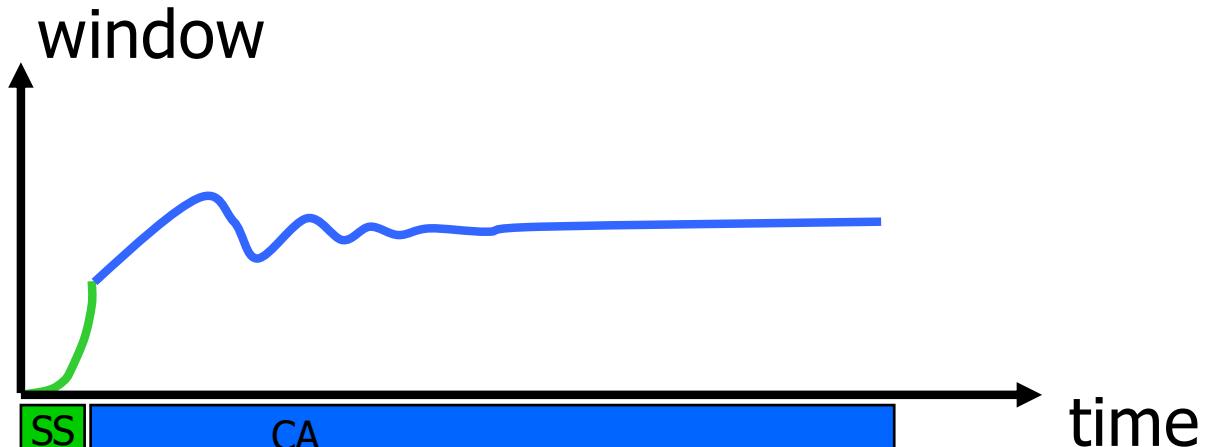
where $x_{\text{vegas}} = W / T$ is the sending rate

Then number of packets in the queue is

$$\begin{aligned}x_{\text{vegas}} T_{\text{queueing}} &= x_{\text{vegas}} T - x_{\text{vegas}} T_{\text{prop}} \\&= W - W/T T_{\text{prop}}\end{aligned}$$

TCP/Vegas CA algorithm

maintain a
constant
number of
packets in the
bottleneck
buffer



```
for every RTT
{
    if  $w - w/RTT \leq RTT_{min}$  <  $\alpha$  then  $w++$ 
    if  $w - w/RTT \geq RTT_{min}$  >  $\alpha$  then  $w--$ 
}
for every loss
     $w := w/2$ 
```

queue size

Discussions

- If two flows, one TCP Vegas and one TCP reno run together, how may bandwidth partitioned among them?

- Issues that limit Vegas deployment?