Network Transport Layer:
Network Resource Allocation Framework

Qiao Xiang, Congming Gao, Qiang Su

https://sngroup.org.cn/courses/cnns-xmuf25/index.shtml

11/11/2025

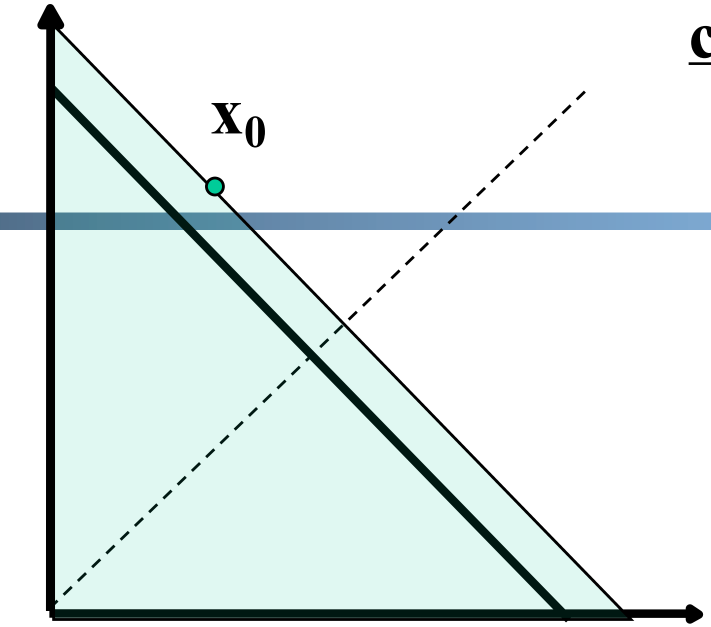
This deck of slides are heavily based on CPSC 433/533 at Yale University, by courtesy of Dr. Y. Richard Yang.

Outline

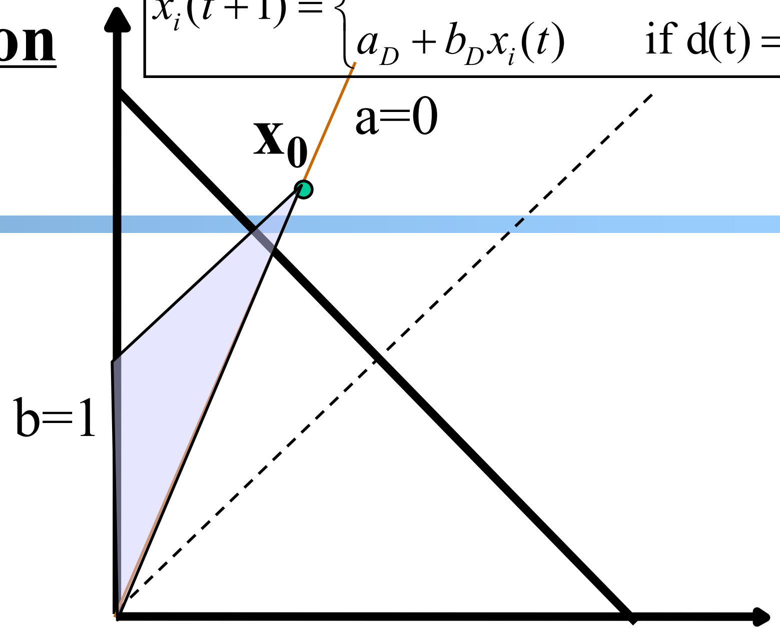
- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic
  - TCP/Vegas
  - network wide resource allocation
    - general framework
    - objective function: axiom derivation of network-wide objective function
    - algorithm: general distributed algorithm framework
    - application: TCP/Reno TCP/Vegas revisited

congestion

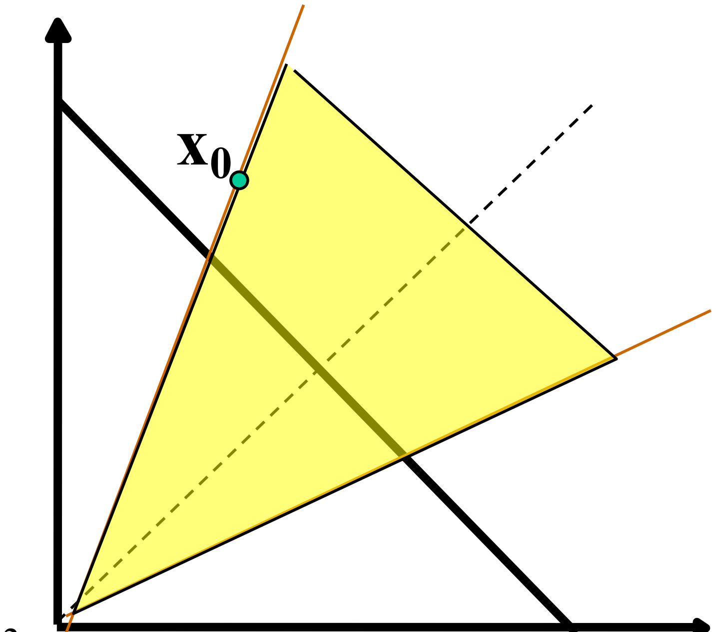
$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{if } d(t) = \text{no cong.} \\ a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.} \end{cases}$$



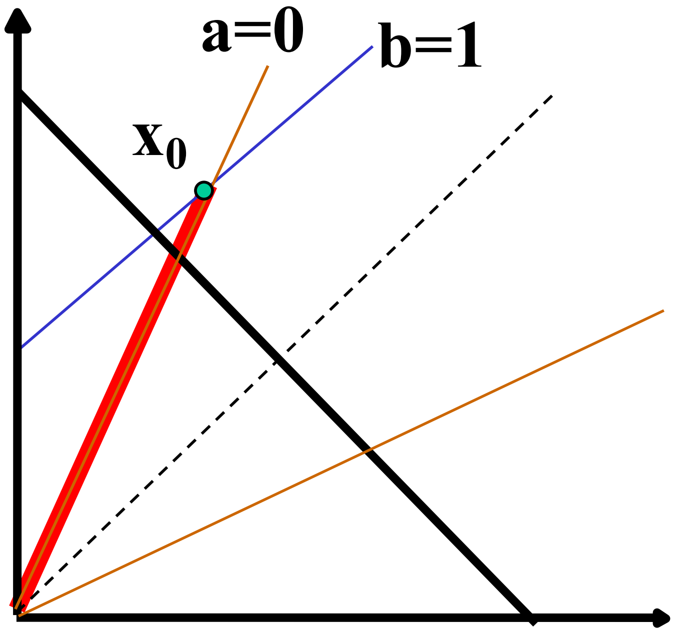
efficiency



efficiency: distributed linear rule



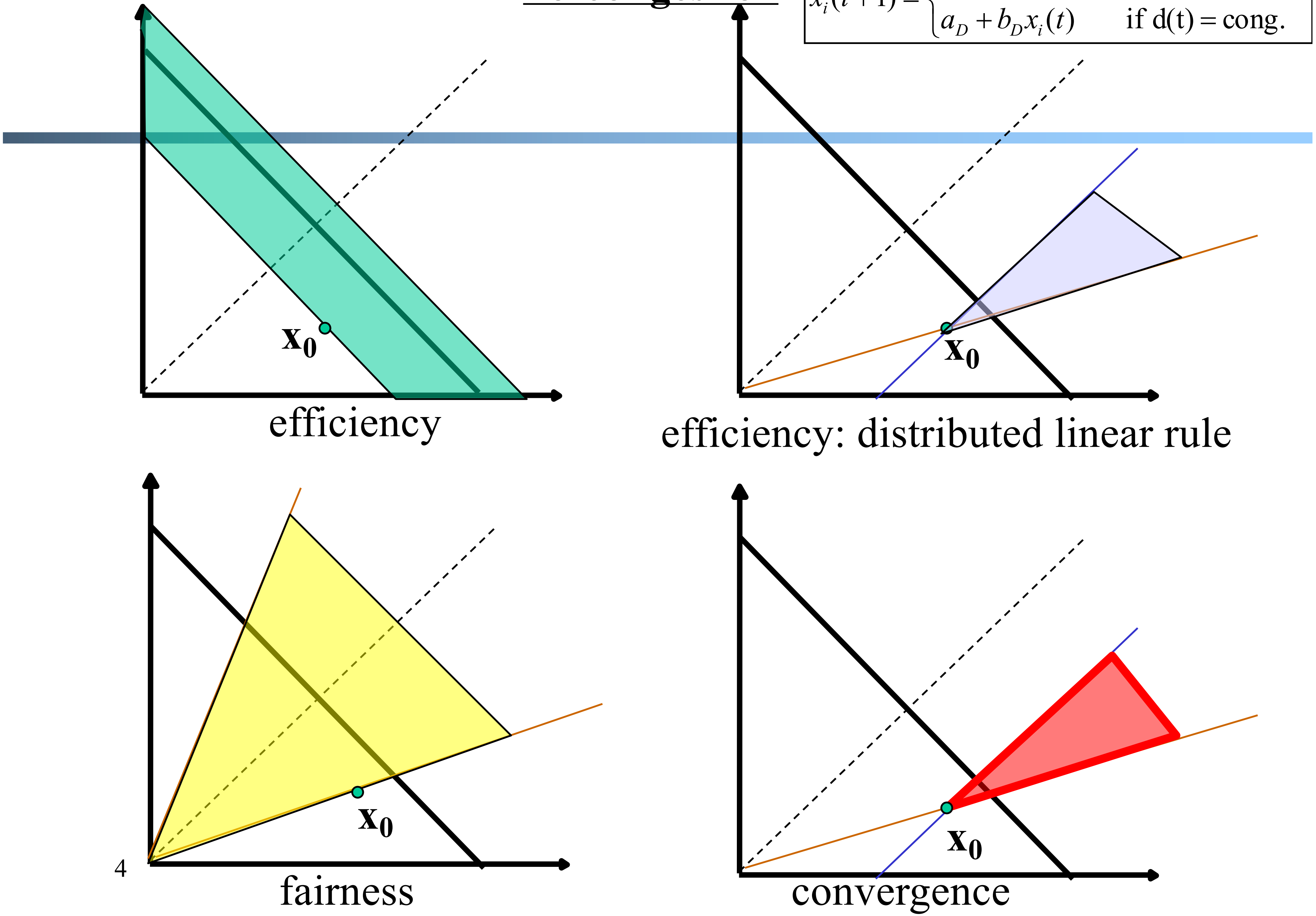
fairness



intersection

no-congestion

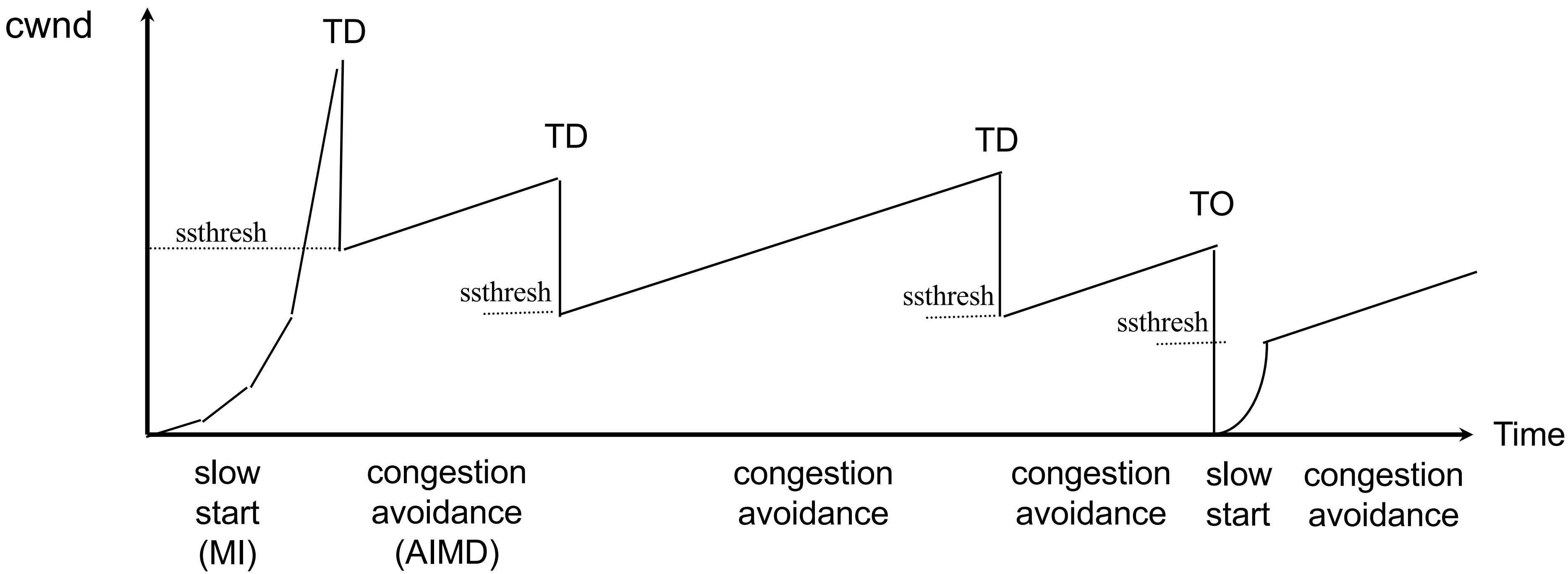
$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{if } d(t) = \text{no cong.} \\ a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.} \end{cases}$$



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:

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

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)

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

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

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

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

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

Exercise: Application of Analysis

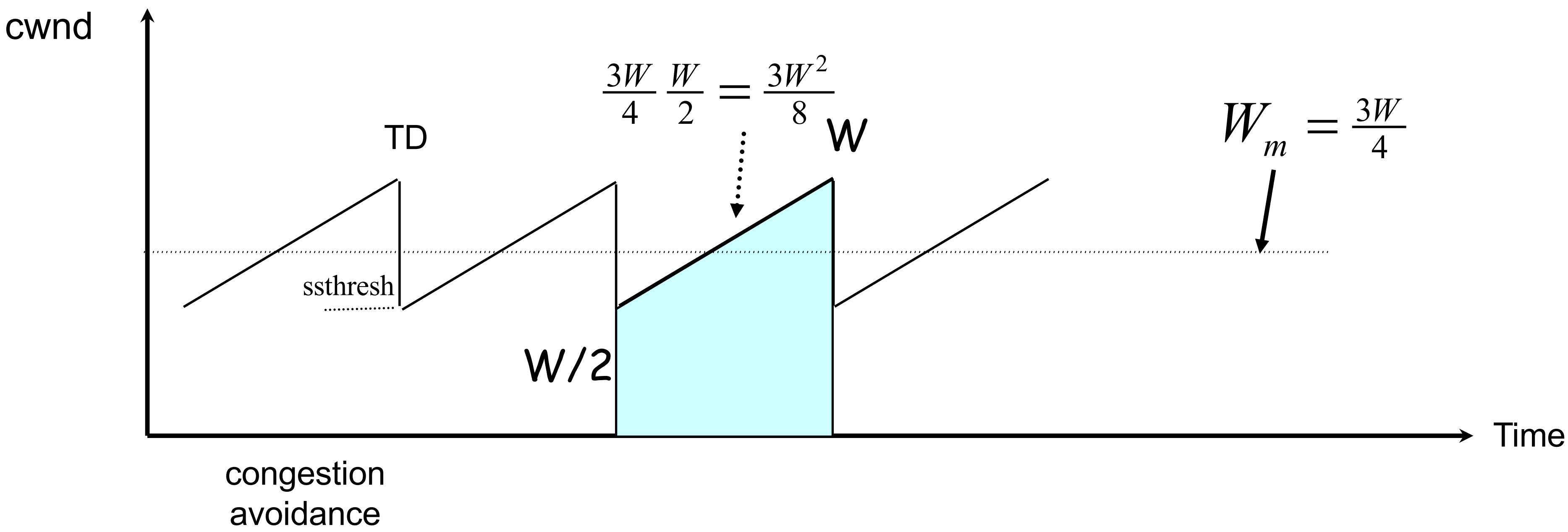
- State of art network link can reach 100 Gbps. Assume packet size 1250 bytes, RTT 100 ms, what is the highest packet loss rate to still reach 100 Gbps?

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



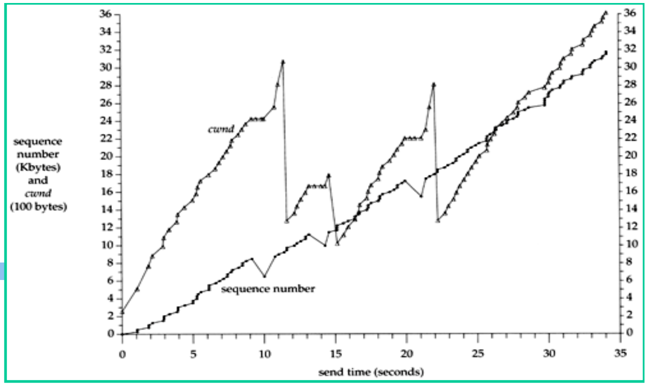
Total packets sent per cycle = (W/2 + W)/2 * W/2 = $3W^2/8$

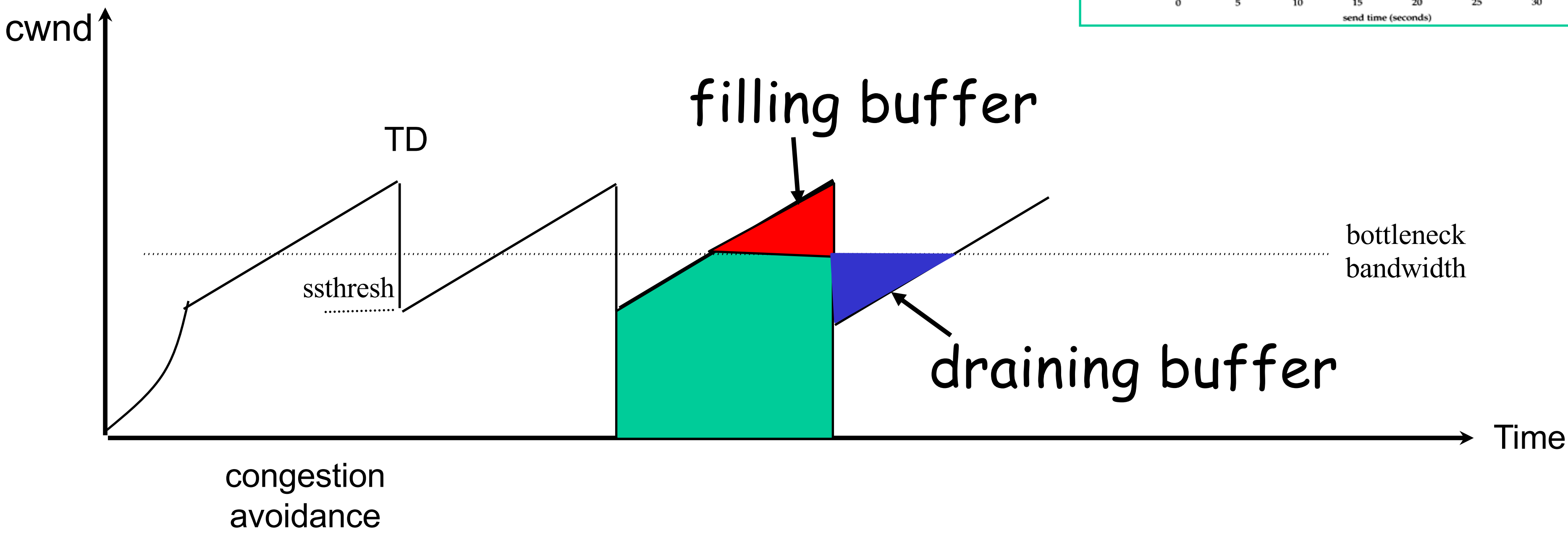
Assume one loss per cycle => p = $1/(3W^2/8) = 8/(3W^2)$

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

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

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?

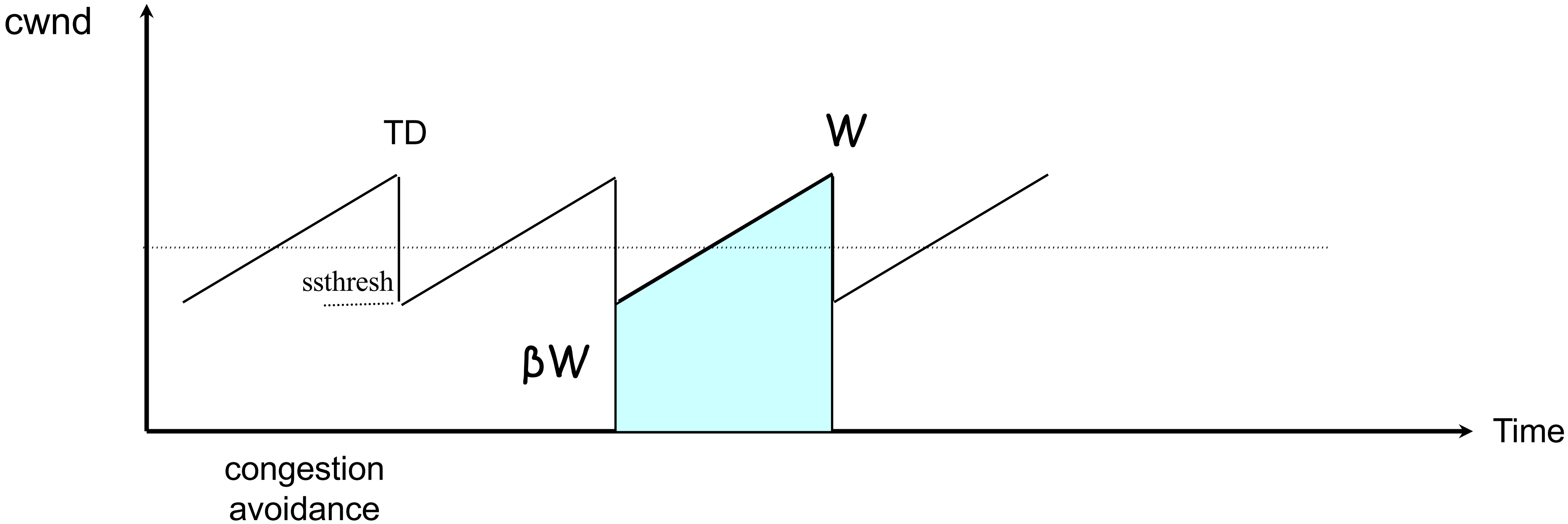
Design

- Assume a generic AIMD alg:
  - increase to W + α after each successful RTT
  - reduce to β W 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 α=1, β=0.5)?

Generic AIMD and TCP Friendliness



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

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

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

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

Outline

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

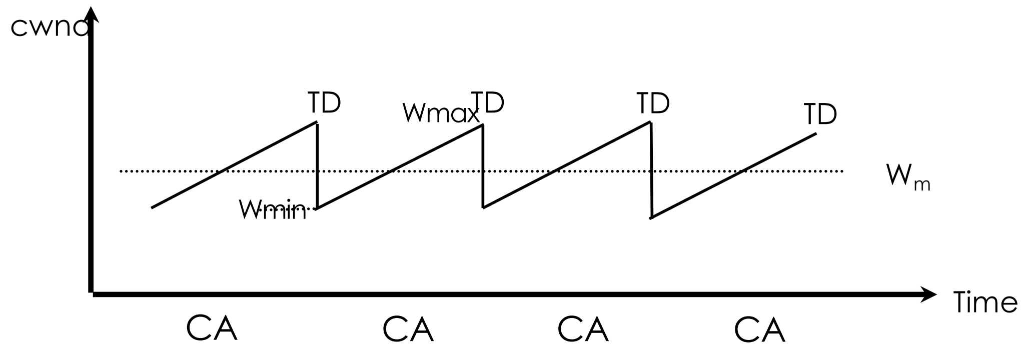
TCP Cubic

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

TCP Cubic Goals

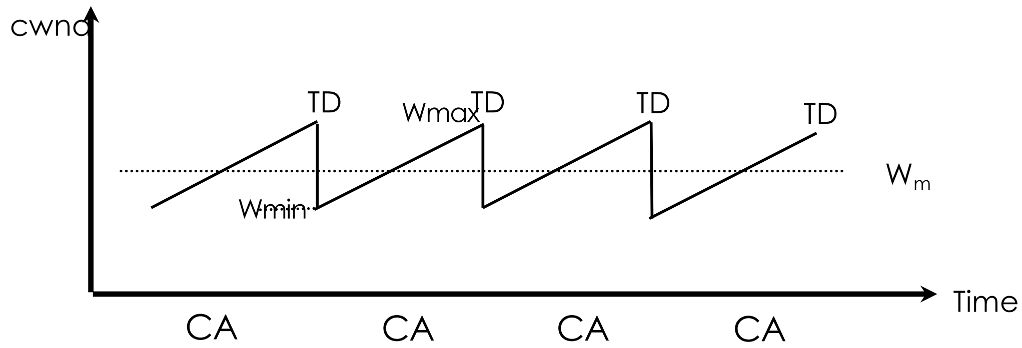
Improve TCP efficiency over fast, long-distance links	Smaller reduction, longer stay at BDP, faster than linear increase---cubic function
TCP friendliness	Follows TCP if TCP gives higher rate
Fairness of flows w/ different RTTs	Window growth depends on real-time (from congestion-epoch through synchronized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

TCP BIC Algorithm



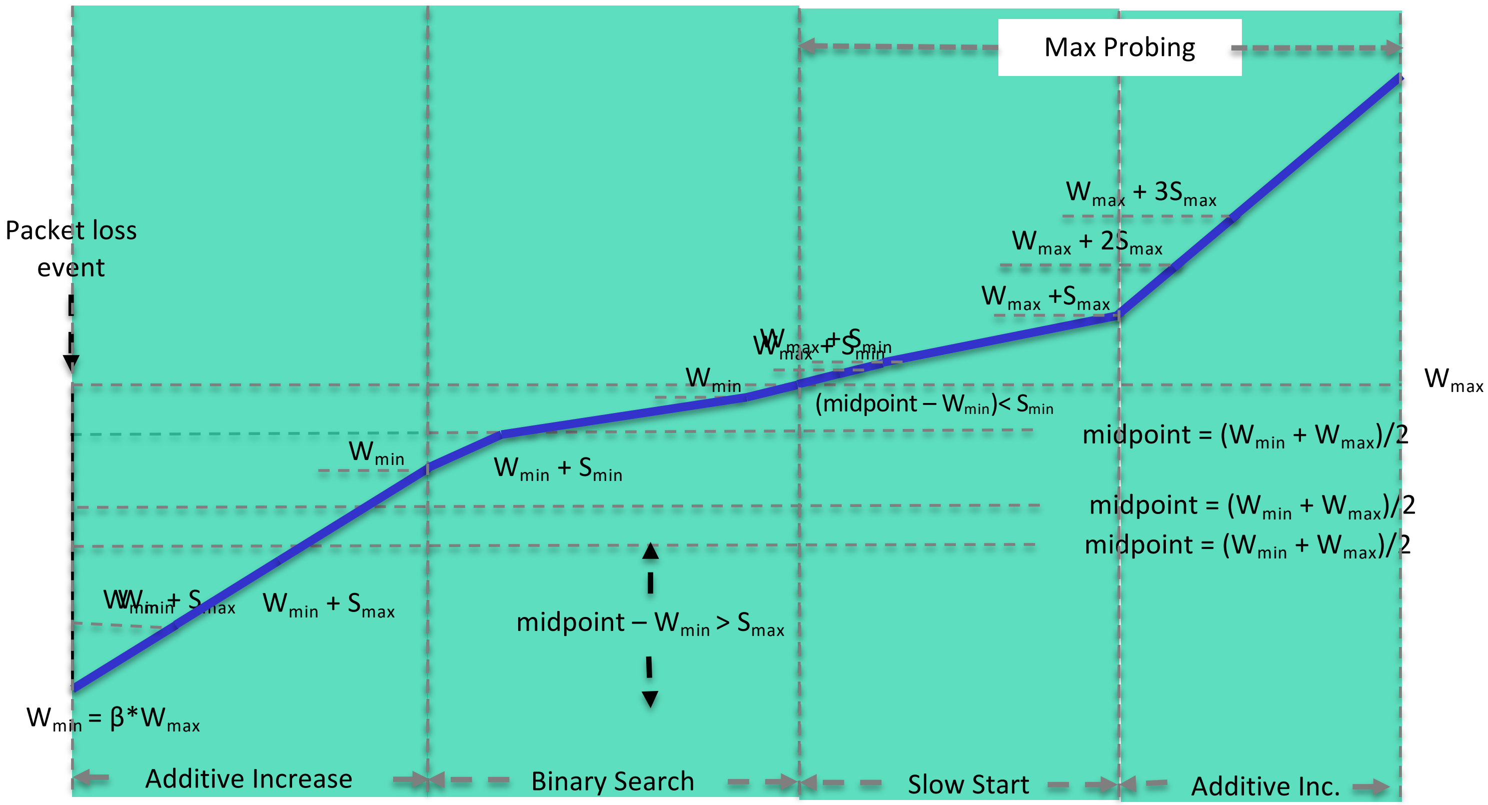
- Setting
  - W_{max} = cwnd size before reduction
    - Too big
  - $W_{min} = \beta * W_{max}$ – just after reduction, where β is multiplicative decrease factor
    - Small
- Basic idea
  - binary search between W_{max} and W_{min}

TCP BIC Algorithm: Issues



- Pure binary search (jump from W_{min} to (W_{max} and W_{min})/2) may be too aggressive
  - Use a large step size Smax
- 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

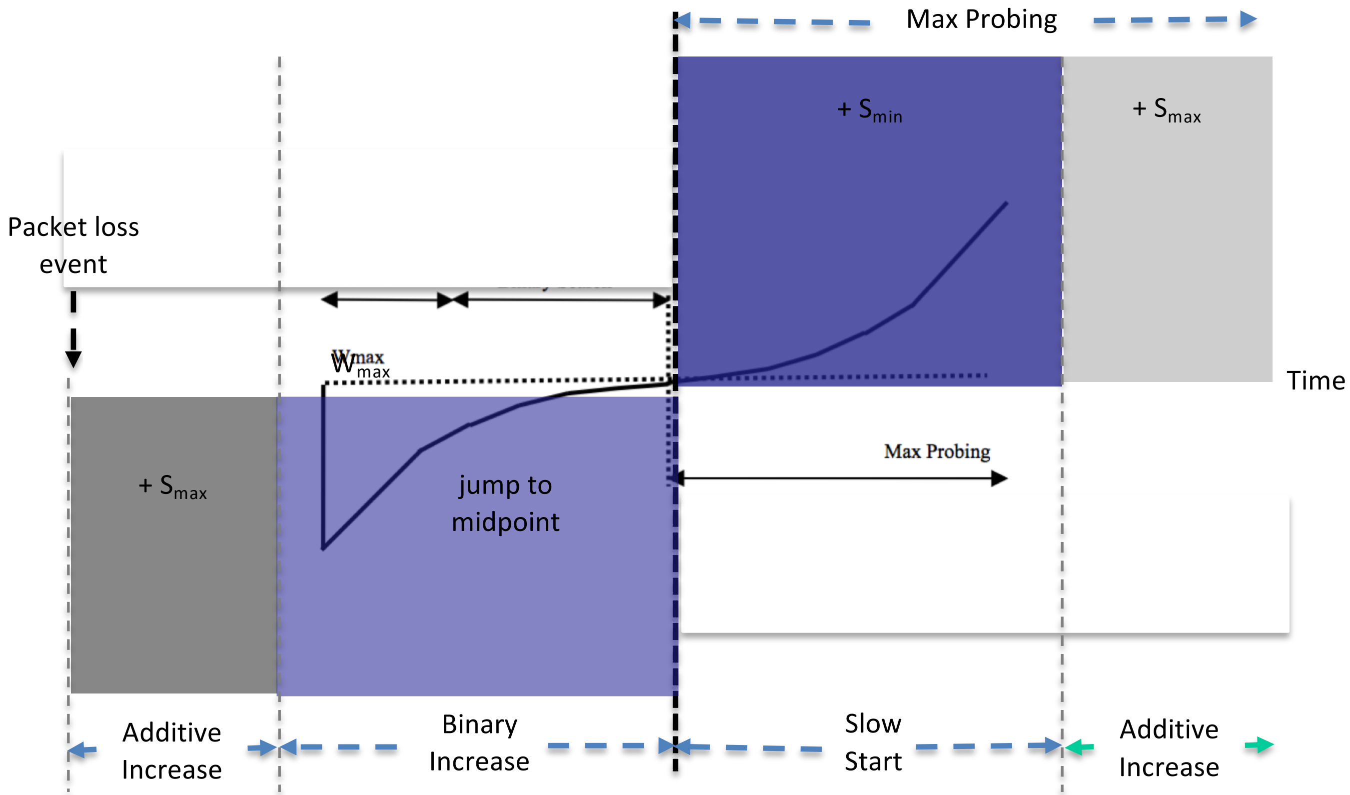
```
while (cwnd >= W_max){
  if (cwnd < W_max + S_max)
        cwnd = cwnd + S_min
   else
        cwnd = cwnd + S_max

   if (packet loss)
        W_min = β*cwnd
        W_max = cwnd
}
```

→ Slow growth

→ Fast growth

Max Probing

TCP BIC - Summary



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

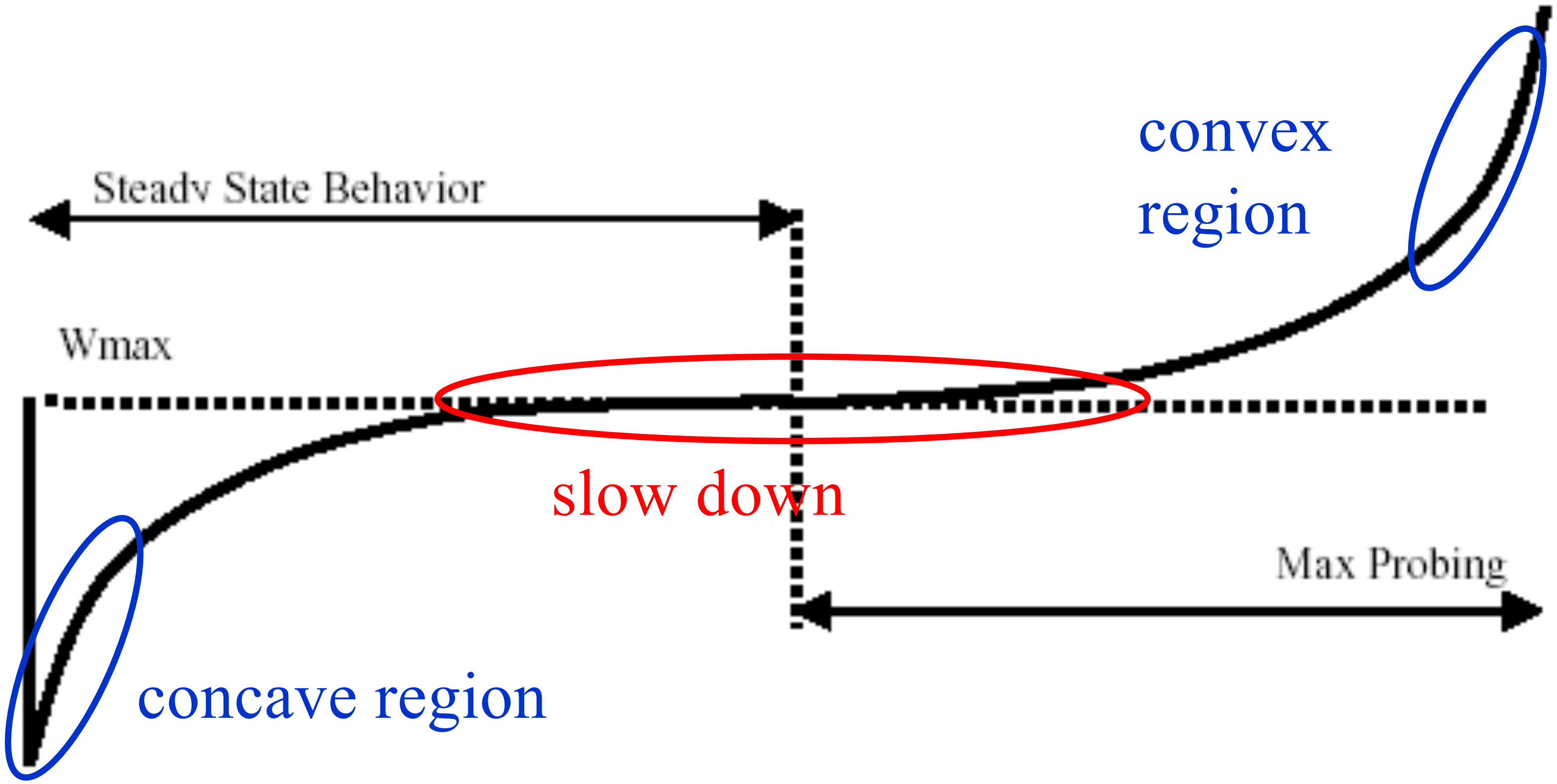
Cubic High-Level Structure

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

The Cubic function

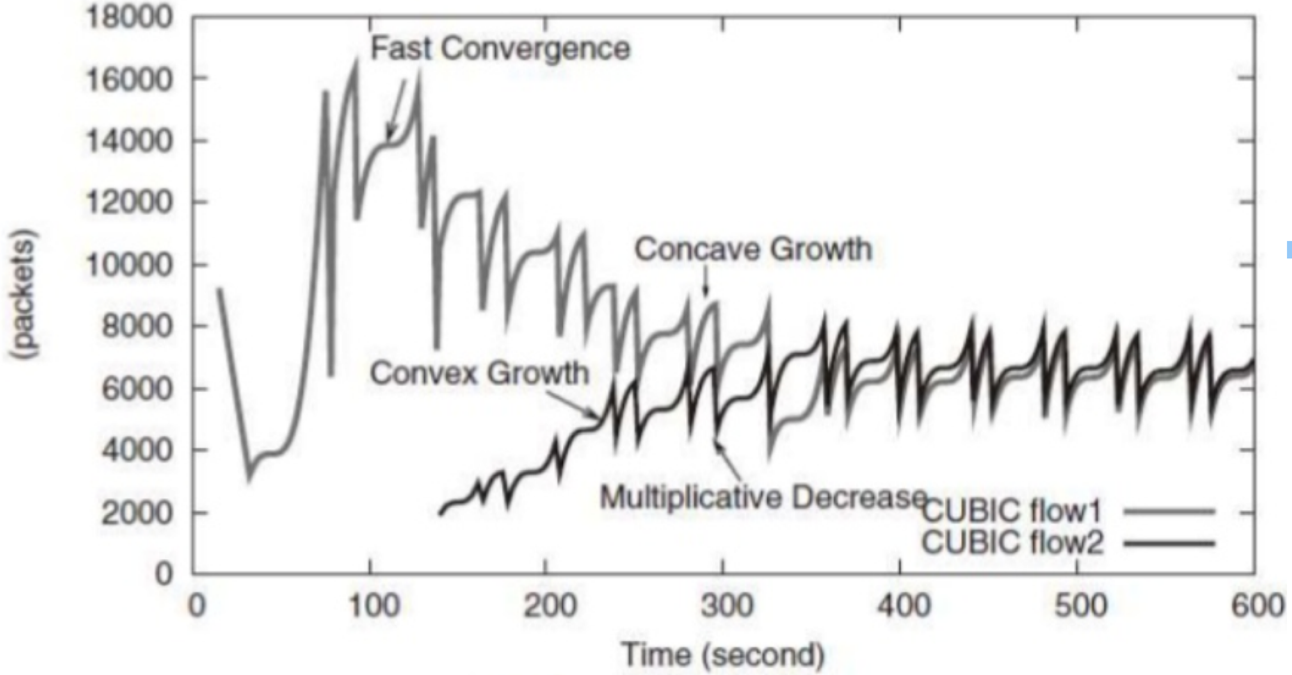
$$\beta' = 1 - \beta$$

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

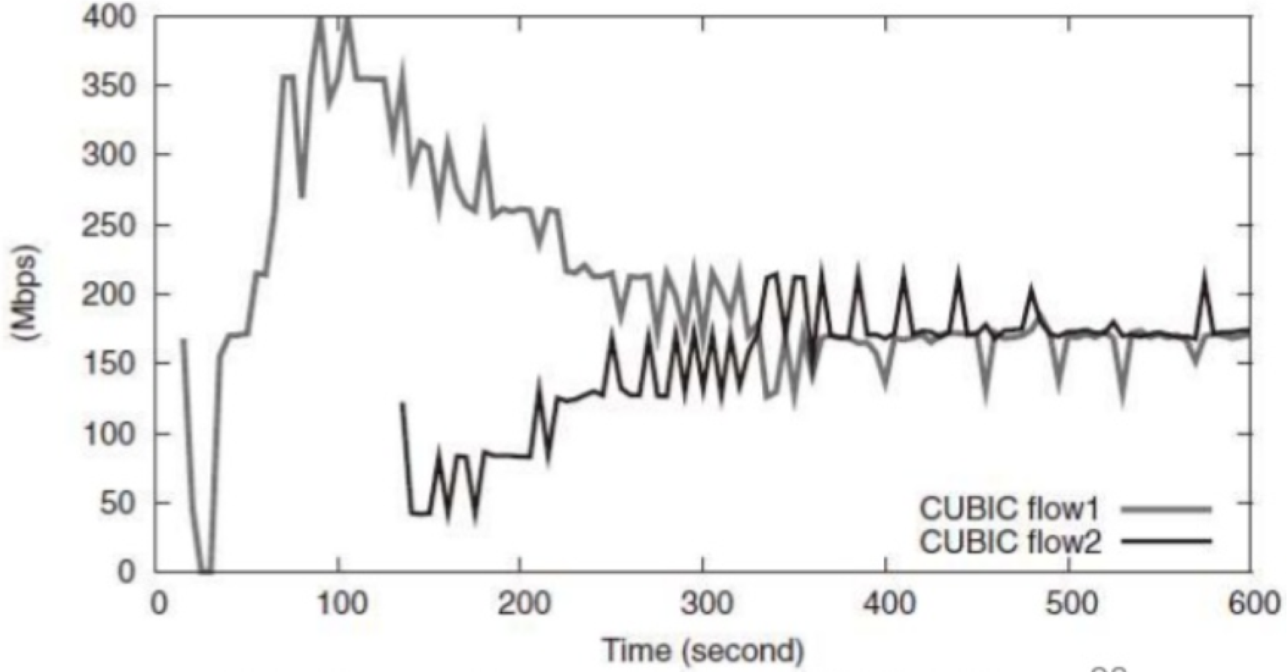


$$W_{cubic} = C(t-K)^3 + W_{max} \qquad K = \sqrt[3]{W_{max}\beta / C}$$

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.

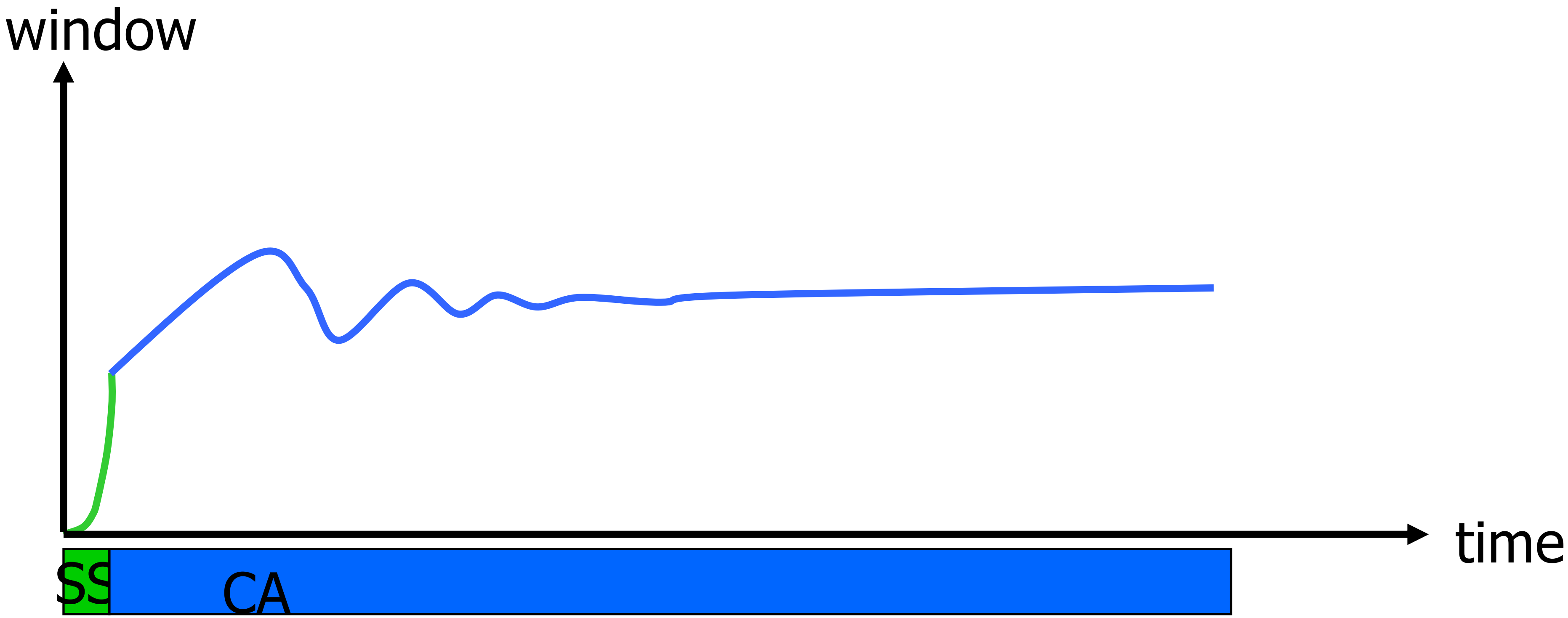


(b) Throughput of two CUBIC flows.

Outline

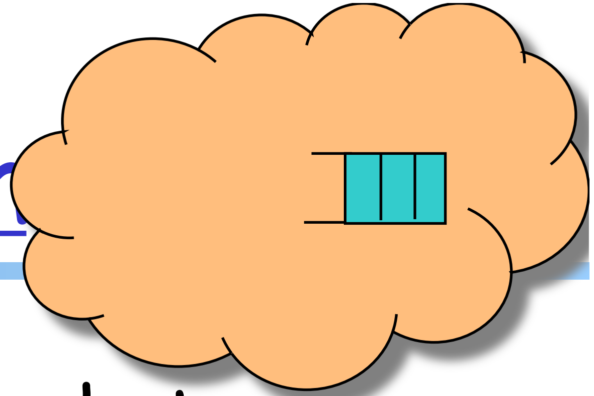
- Admin and recap
- Transport congestion control
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic
  - TCP/Vegas

TCP/Vegas (Brakmo & Peterson 1994)

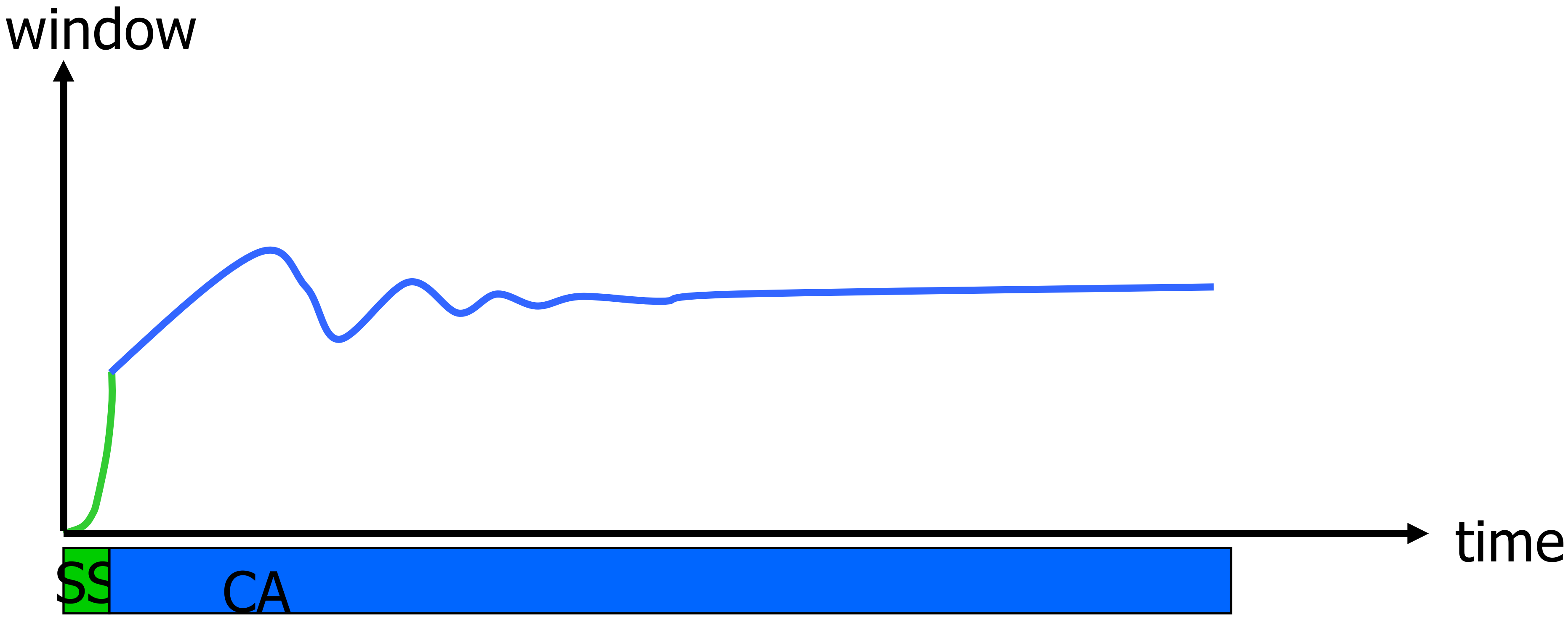


- Idea: try to detect congestion by delay before loss
- Objective: not to overflow the buffer; instead, try to maintain a *constant* number of packets in the bottleneck queue

TCP/Vegas: Key Question

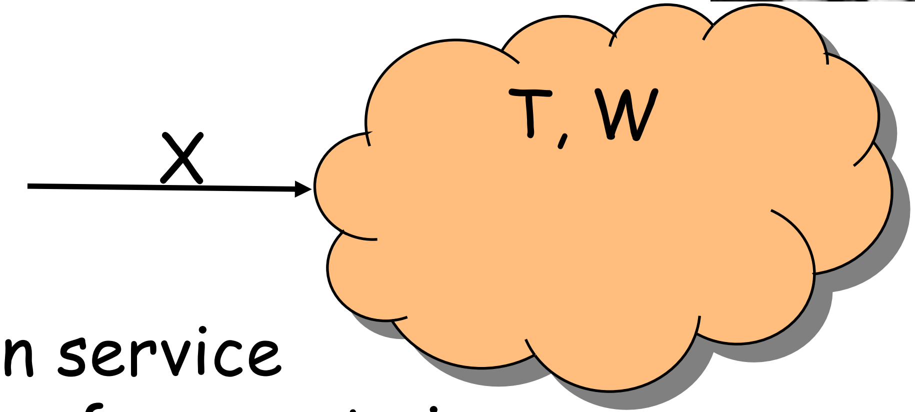


- How to estimate the number of packets queued in the bottleneck queue?



Recall: Little's Law

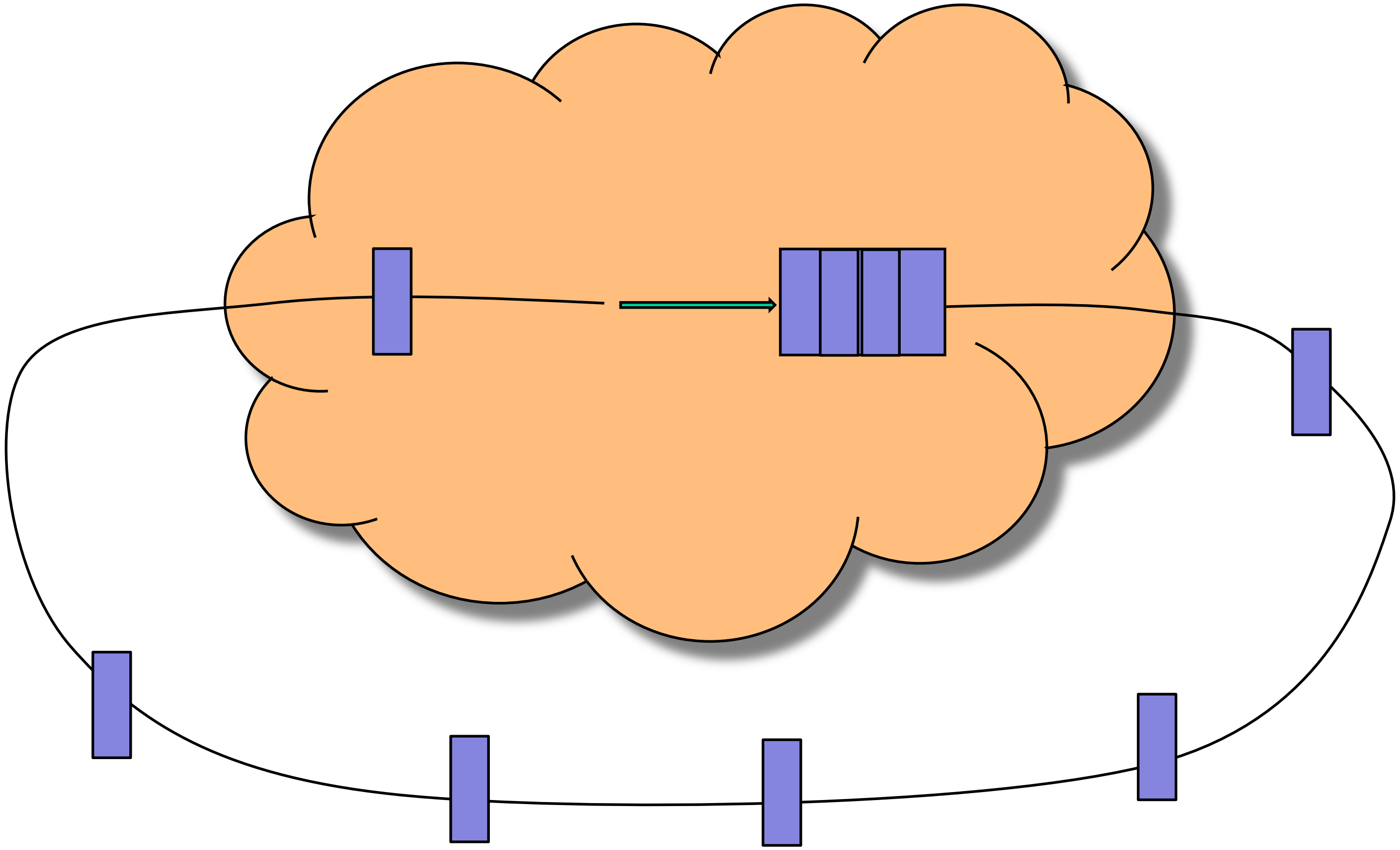
- For any system with no or (low) loss.
- Assume
  - mean arrival rate X, mean service time T, and mean number of requests in the system W



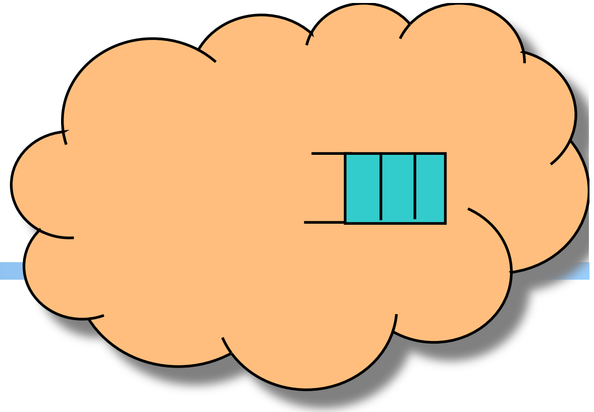
- Then relationship between W, X, and T:

$$W = XT$$

Estimating Number of Packets in the Queue



TCP/Vegas CA algorithm



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

Applying Little's Law:

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

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

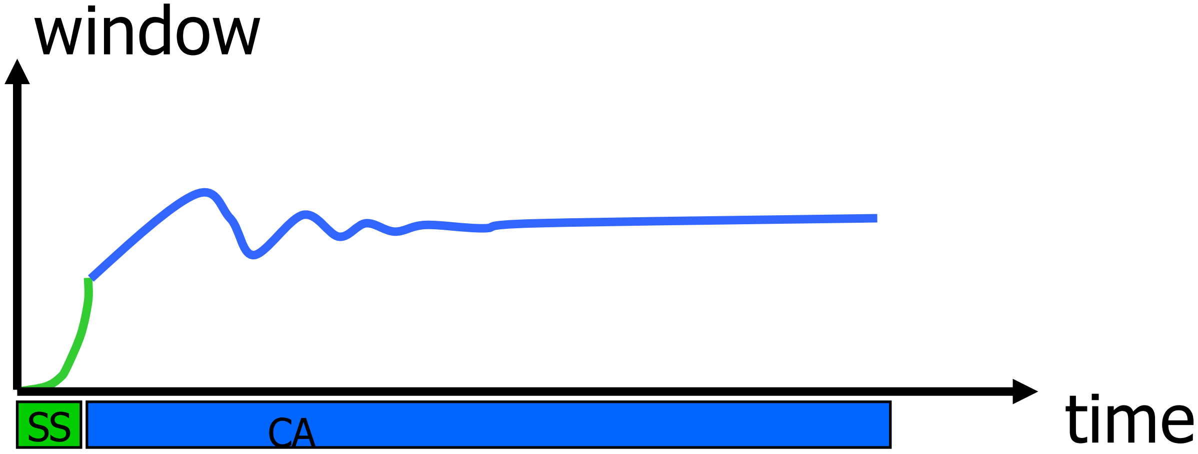
Then number of packets in the queue is

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

$$= W - W/T\;T_{prop}$$

TCP/Vegas CA algorithm

maintain a *constant* number of packets in the bottleneck buffer




```
for every RTT
{    if W – W/RTT RTTmin < α then W ++
     if W – W/RTT RTTmin > α 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?