Computer Networks
COL 334/672

Congestion Control

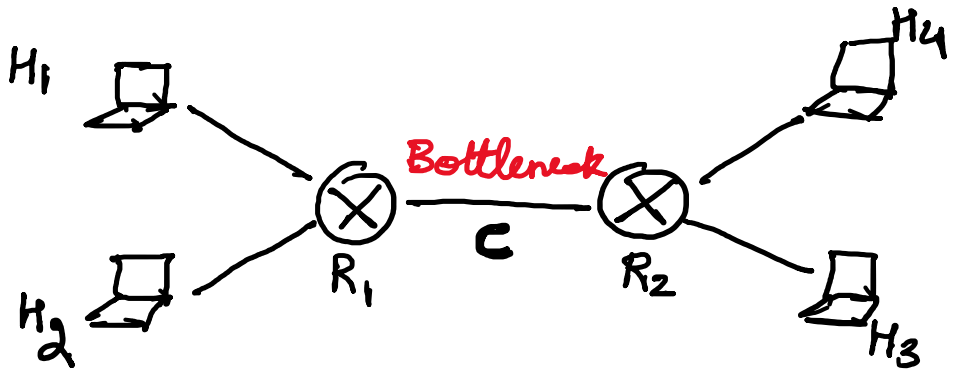
Tarun Mangla

Slides adapted from KR

Sem 1, 2024-25

Recap: TCP Congestion Control

- **What:** Adjust sending rate at end-host to avoid congestion in the network
- **Why:** congestion → wasted network resources
- **How:**
  - **Design Space:** Multiple approaches possible. TCP chose:
    - End-to-end
    - Distributed
    - Window-based
  - **General approach:**
    1. Probe b/w by increasing window
    2. Detect congestion
    3. Backoff by decreasing window
  - **Multiple ways:** AIMD is chosen (desirable stability properties, provides fairness)



Window Update in AIMD

$0 < \beta < 1$

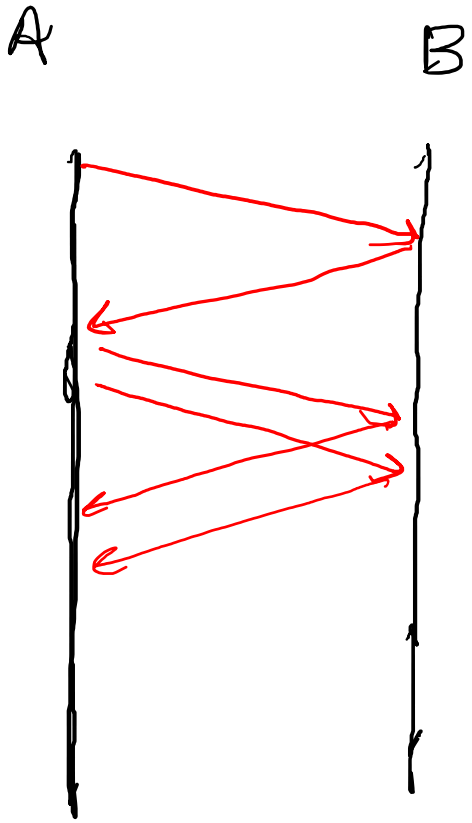
AIMD; AI: cwnd = cwnd + α, cwnd max > α > 1 = 1 MD: cwnd : $\frac{cwnd}{\beta}$

- **Goal**: For additive increase, update window size by 1 every RTT. But when?

cwnd = cwnd + 1/cwnd

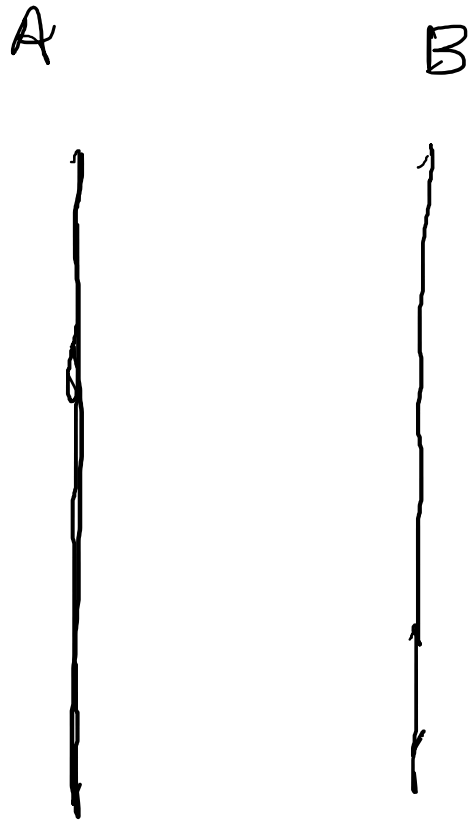
goal 1 MSS ↑ per RTT

cwnd = cwnd + $\left(\frac{MSS}{cwnd}\right)$ x MSS



Window Update in AIMD

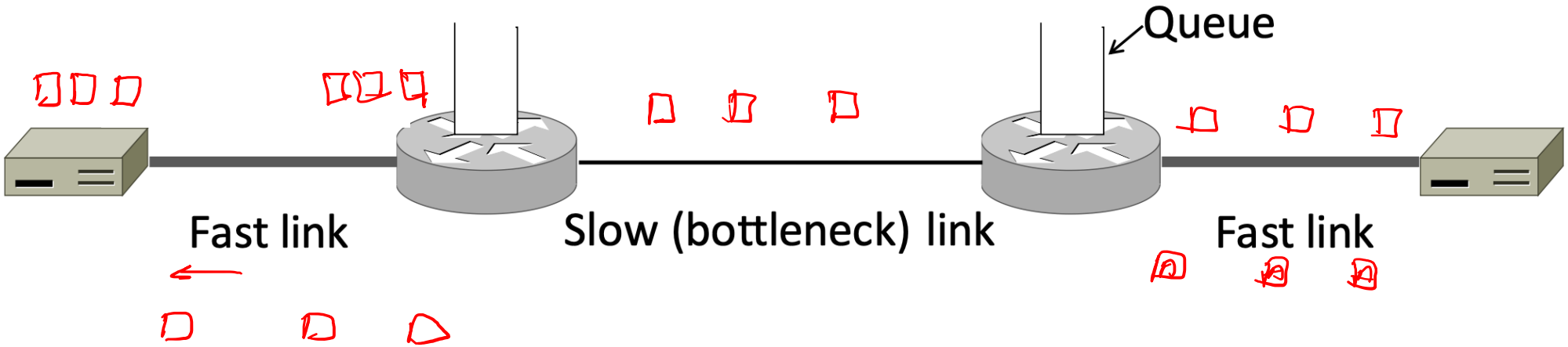
- **Goal**: For additive increase, update window size by 1 every RTT. But when?
  - Window updated as ACKs arrive
- How much should be the update?



- Because TCP updates based on ACKs, TCP is self-clocking Benefit: Self-clocking smooths packet sending rate

TCP Self Clocking

- ACK clocking smooths packet sending rate

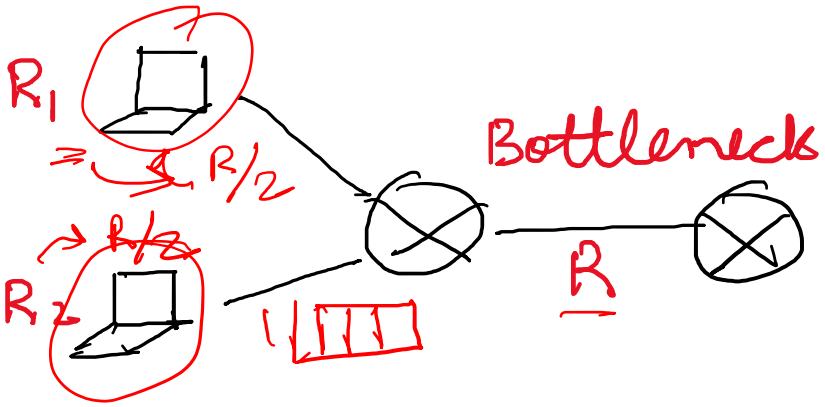


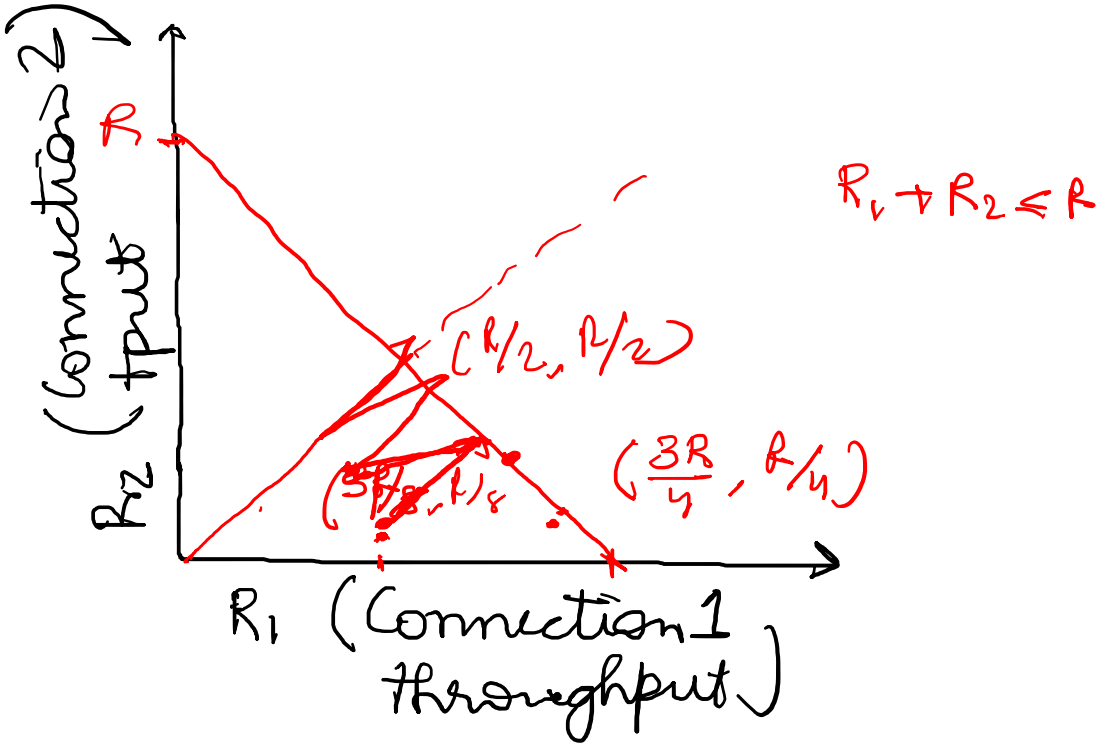
AIMD is TCP Fair

Active queue Management (AQM)
↳ Random Early Drop (RED)

Example: two competing TCP sessions:

② Is this mechanism TCP friendly in all scenarios?
AIMD: is not RTT-fair





Beginning of the Connection

cwnd = 10^4 IW = 1

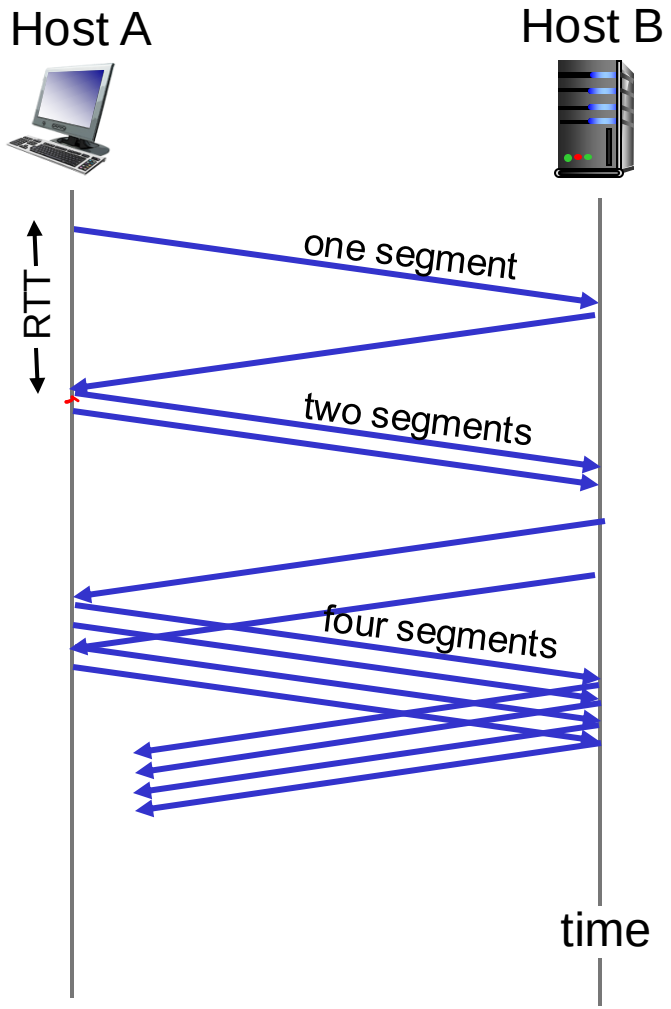
RTT = 10^{-2} s

How much time = 10^2

- What *cwnd* size should we start with?
  - **Goal**: We want to quickly near the *right* rate.
  - A linear increase with a small value of cwnd is painfully slow!
    - exponential
    - start w/ a large value :
- What are the options?
  - Start with a large value of cwnd
  - Start with a small value of cwnd but increase it faster

TCP slow start

- when connection begins, increase rate exponentially until first loss event:
  - initially **`cwnd`** = 1 MSS
  - double **`cwnd`** every RTT
  - done by incrementing **`cwnd`** for every ACK received
- *summary:* initial rate is slow, but ramps up exponentially fast

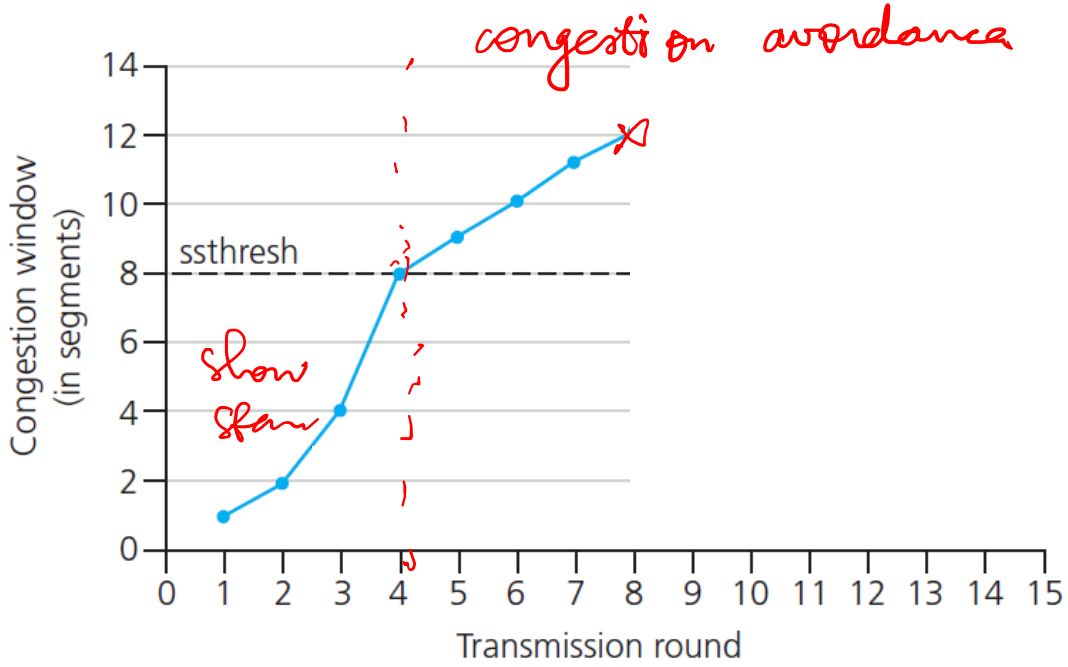


Until when?

TCP Slow Start and Congestion Avoidance

- Uses a threshold, *ssthresh*, after which TCP enters congestion avoidance

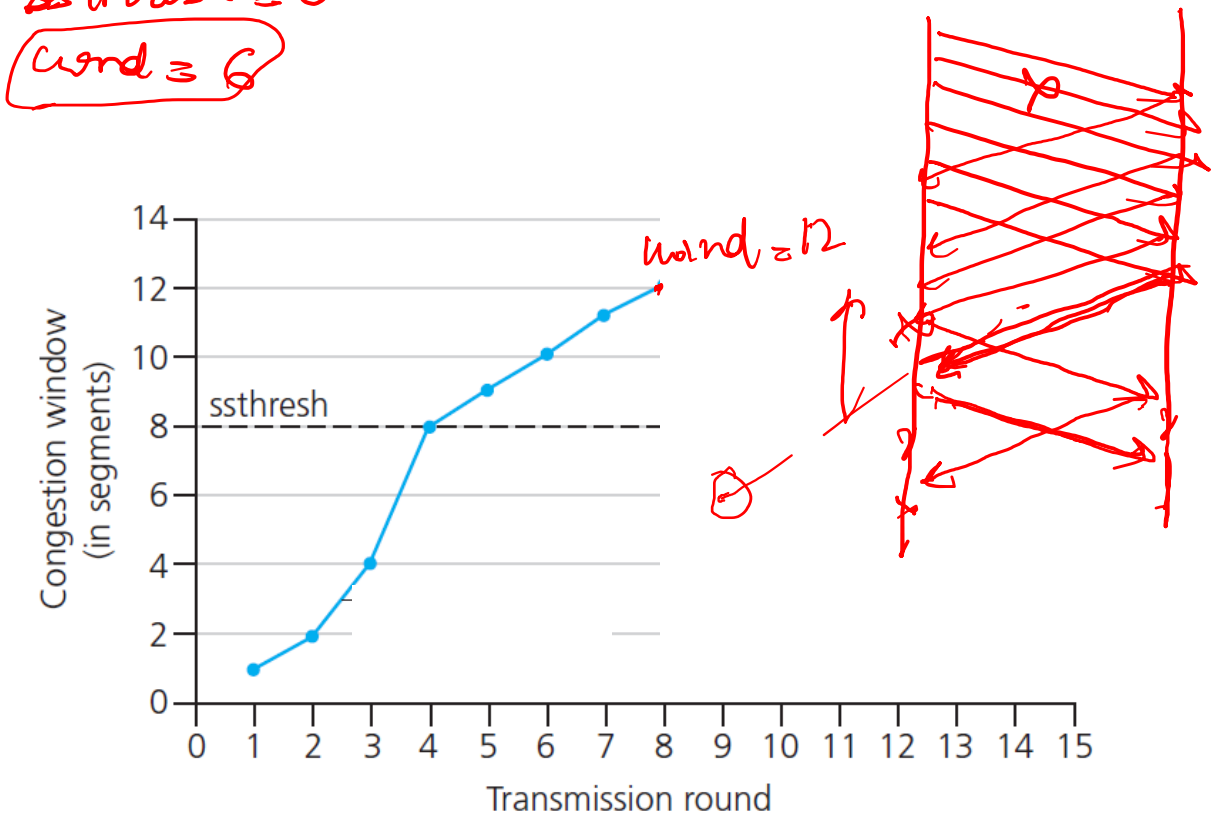
- *What happens when a loss occurs?*



What happens when a loss occurs?

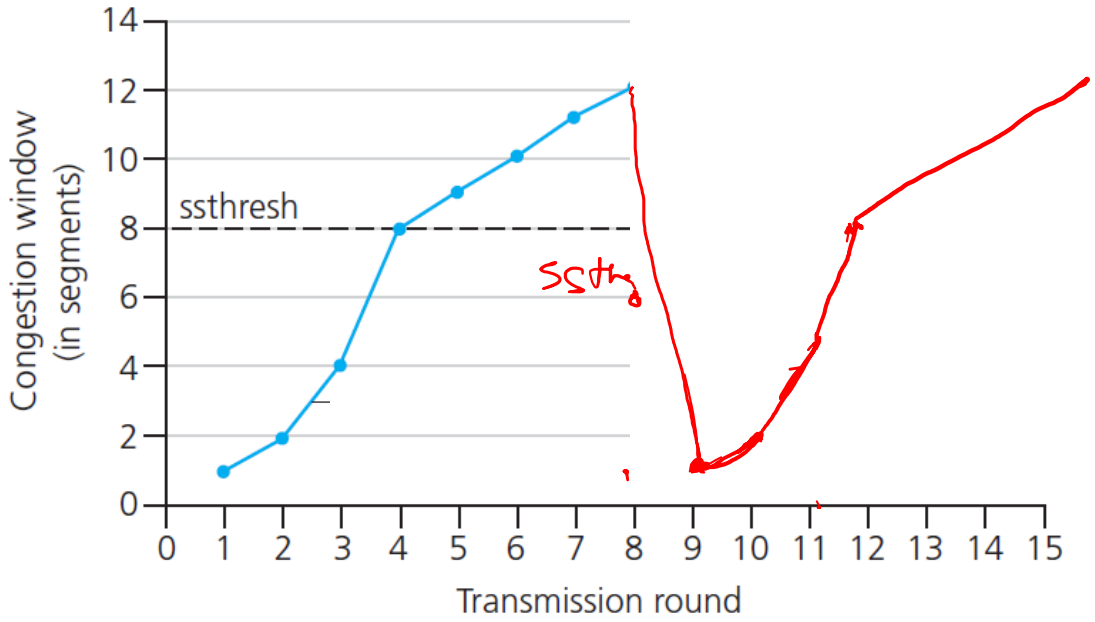
- When loss is due to triple duplicate ACK: congestion is not severe!
  - Set *ssthresh to cwnd/2*
  - On receipt of another duplicate ACK, send 1 new segment
  - Once a new ACK arrives, set *cwnd = ssthresh (or cwnd/2)*
  - Enter congestion avoidance phase

TCP Reno: Fast retransmit, fast recovery



What happens when a loss occurs?

- When loss is due to timeout: severe congestion!!
  - Set *ssthresh to cwnd/2*
  - Reset cwnd to 1
  - Enter slow start phase



What happens when a loss occurs?

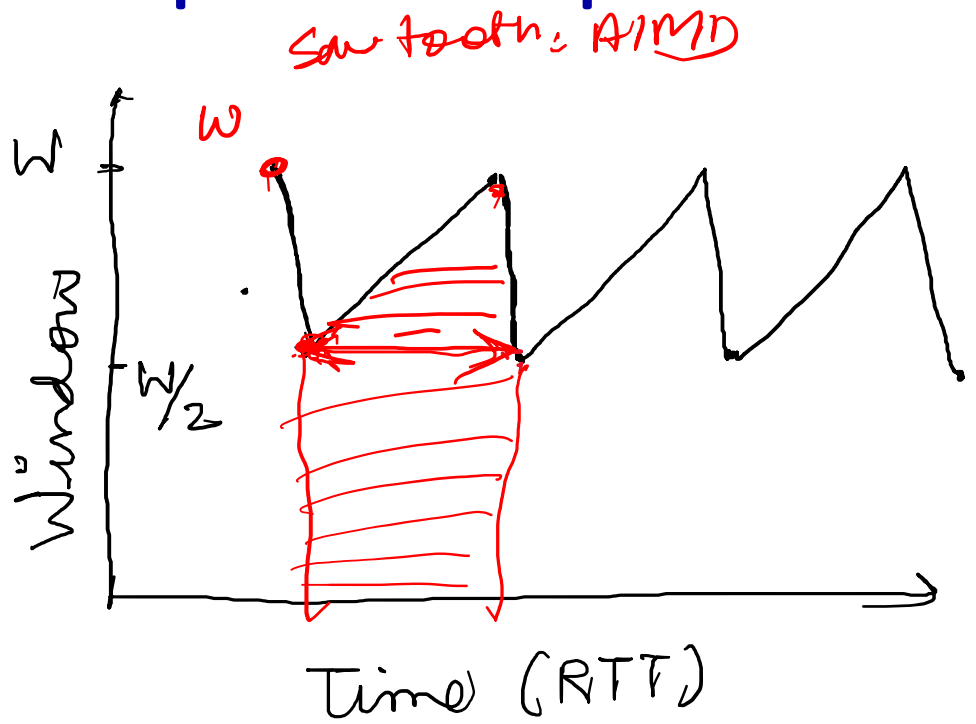
- Loss is due to timeout: severe congestion!!
  - Set *ssthresh to cwnd/2*
  - Reset cwnd to 1
  - Enter slow start phase

- Loss is due to triple duplicate ACK: congestion is not severe!
  - Set *ssthresh to cwnd/2*
  - On receipt of another duplicate ACK, send 1 new segment
  - Once a new ACK arrives, set *cwnd = ssthresh (or cwnd/2)*
  - Enter congestion avoidance phase

TCP Reno Throughput: Macroscopic Description

- Throughput: area under the curve

Average Tput : $\frac{3W}{4}$

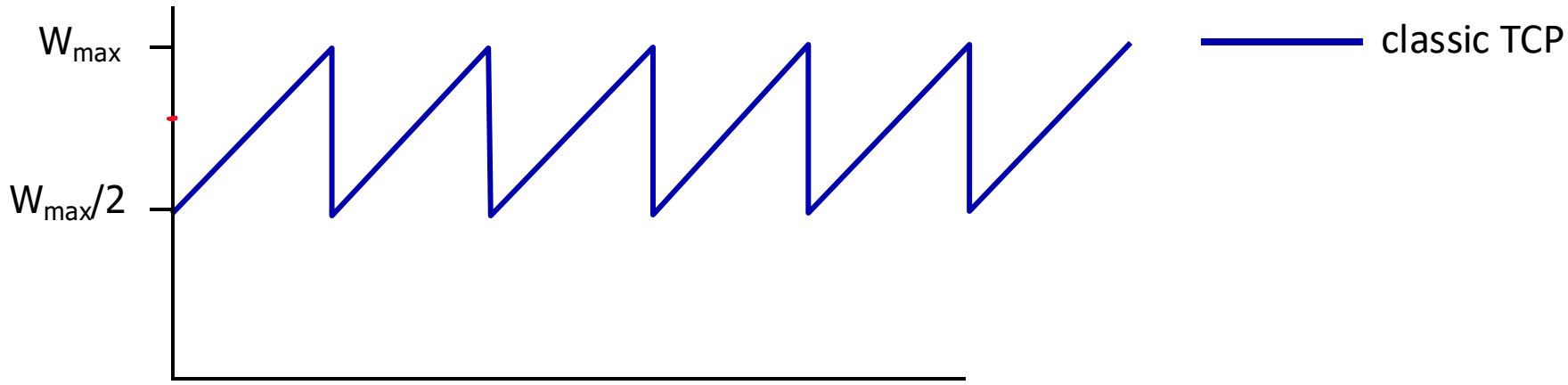


Inefficient for networks with high bandwidth delay product!

Can we do faster?

Is there a better way than AIMD to "probe" for usable bandwidth?

- Insight/intuition:
  - W_{max}: sending rate at which congestion loss was detected
  - congestion state of bottleneck link probably (?) hasn't changed much
  - after cutting rate/window in half on loss, initially ramp to to W_{max} *faster*, but then approach W_{max} more *slowly*



TCP CUBIC

- K: point in time when TCP window size will reach W_{max}
  - K itself is tunable
- increase W as a function of the *cube* of the distance between current time and K

$$W(t) = C(t-K)^3 + W_{max}$$

$$K = \sqrt[3]{\frac{W_{max}\beta}{C}}$$

- TCP CUBIC default in Linux, most popular TCP for popular Web servers

Attendance