Lecture 21: Congestion Control

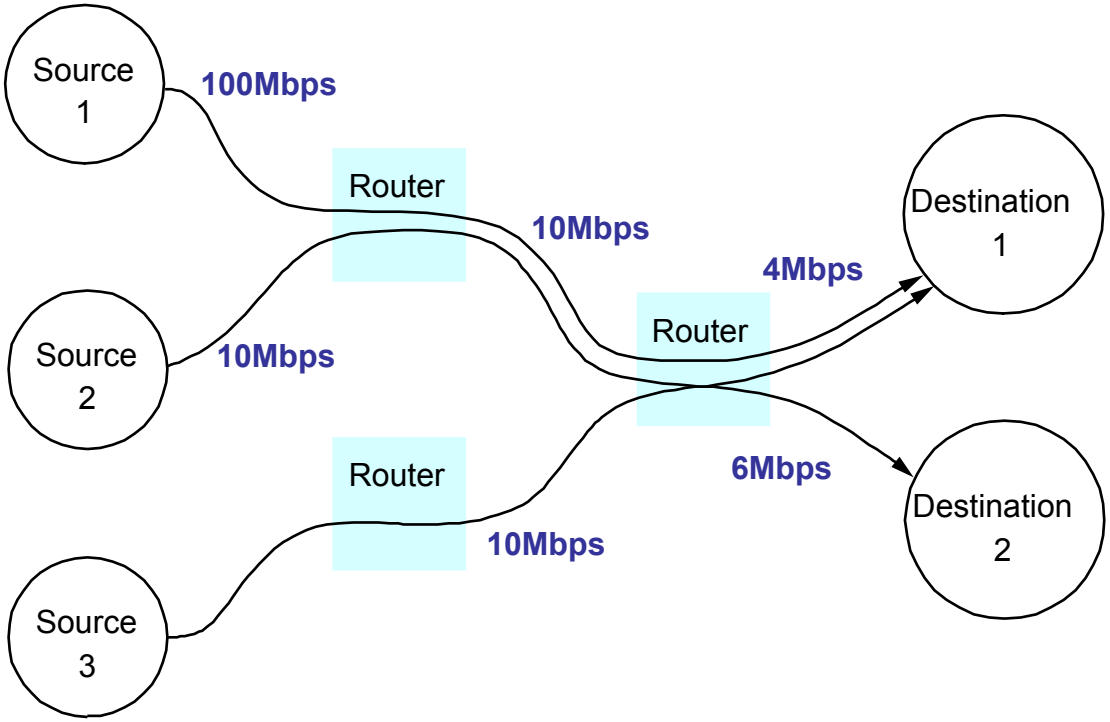
CSE 123: Computer Networks

Alex C. Snoeren

Lecture 21 Overview

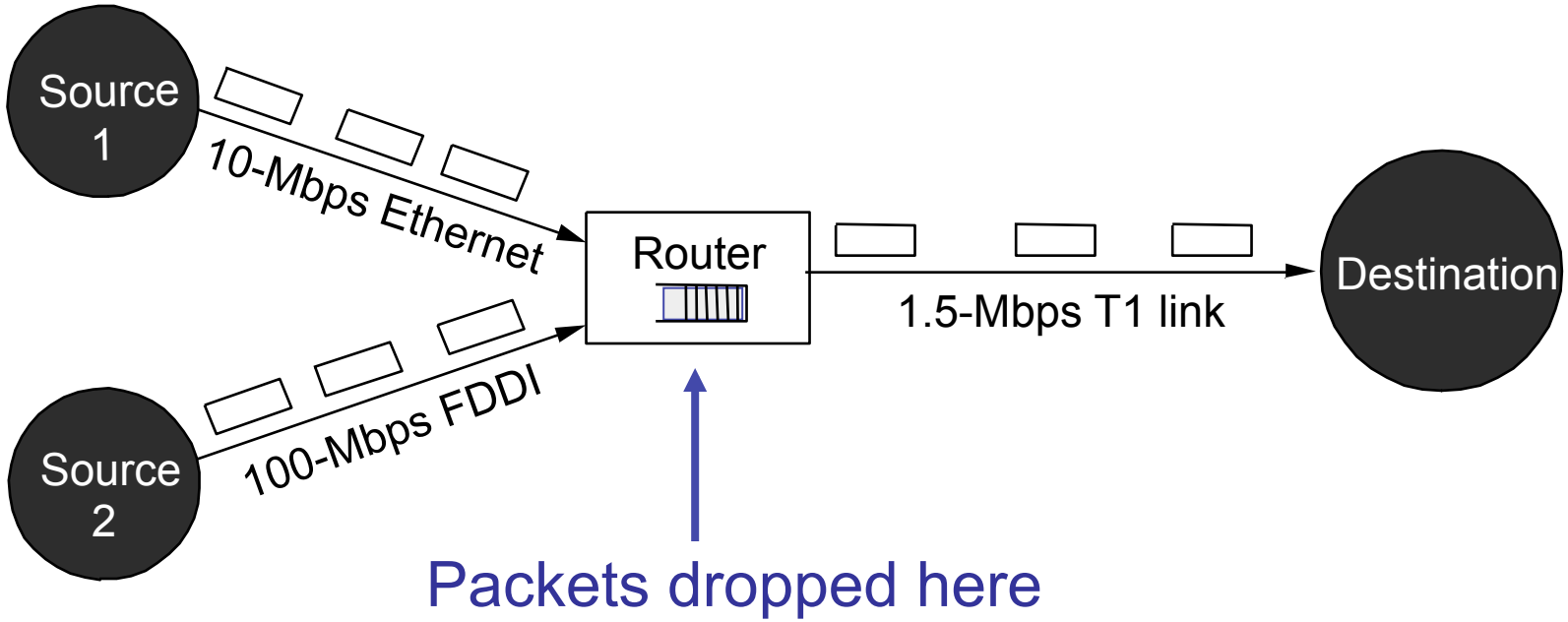
- How fast should a sending host transmit data?
  - Not to fast, not to slow, just right…
- Should not be faster than the sender's share
  - Bandwidth allocation
- Should not be faster than the network can process
  - Congestion control
- Congestion control & bandwidth allocation are separate ideas, but frequently combined

Bandwidth Allocation



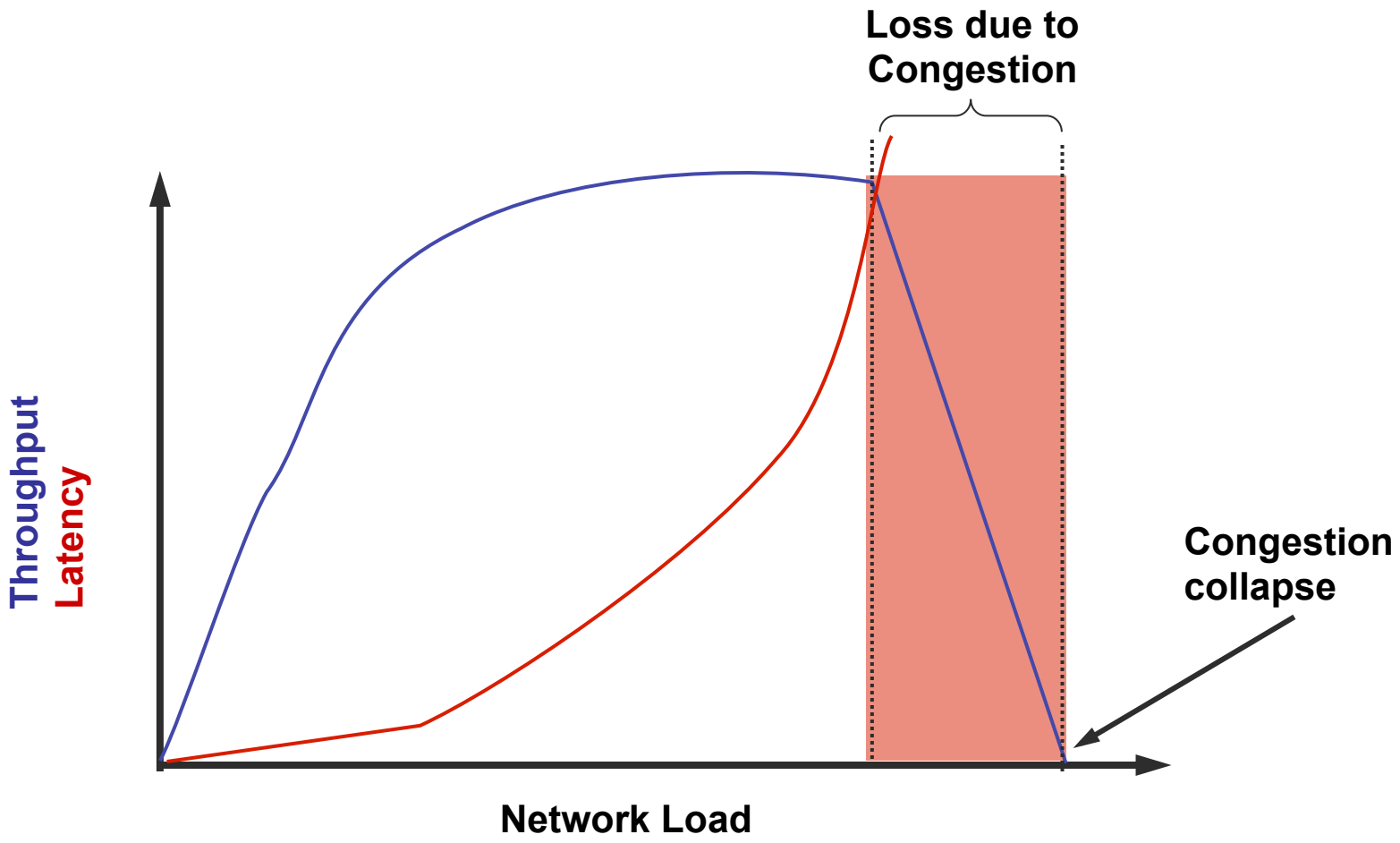
- How much bandwidth should each flow from a source to a destination receive when they compete for resources?
  - What is a "fair" allocation?

Congestion



- Buffer intended to absorb bursts when input rate > output
- But if sending rate is persistently > drain rate, queue builds
- Dropped packets represent wasted work; goodput < throughput

Drop-Tail Queuing



Congestion Collapse

- Rough definition: “When an increase in network load produces a decrease in useful work”

- Why does it happen?
  - Sender sends faster than bottleneck link speed
  - Packets queue until dropped
  - In response to packets being dropped, sender retransmits
  - All hosts repeat in steady state…

Mitigation Options

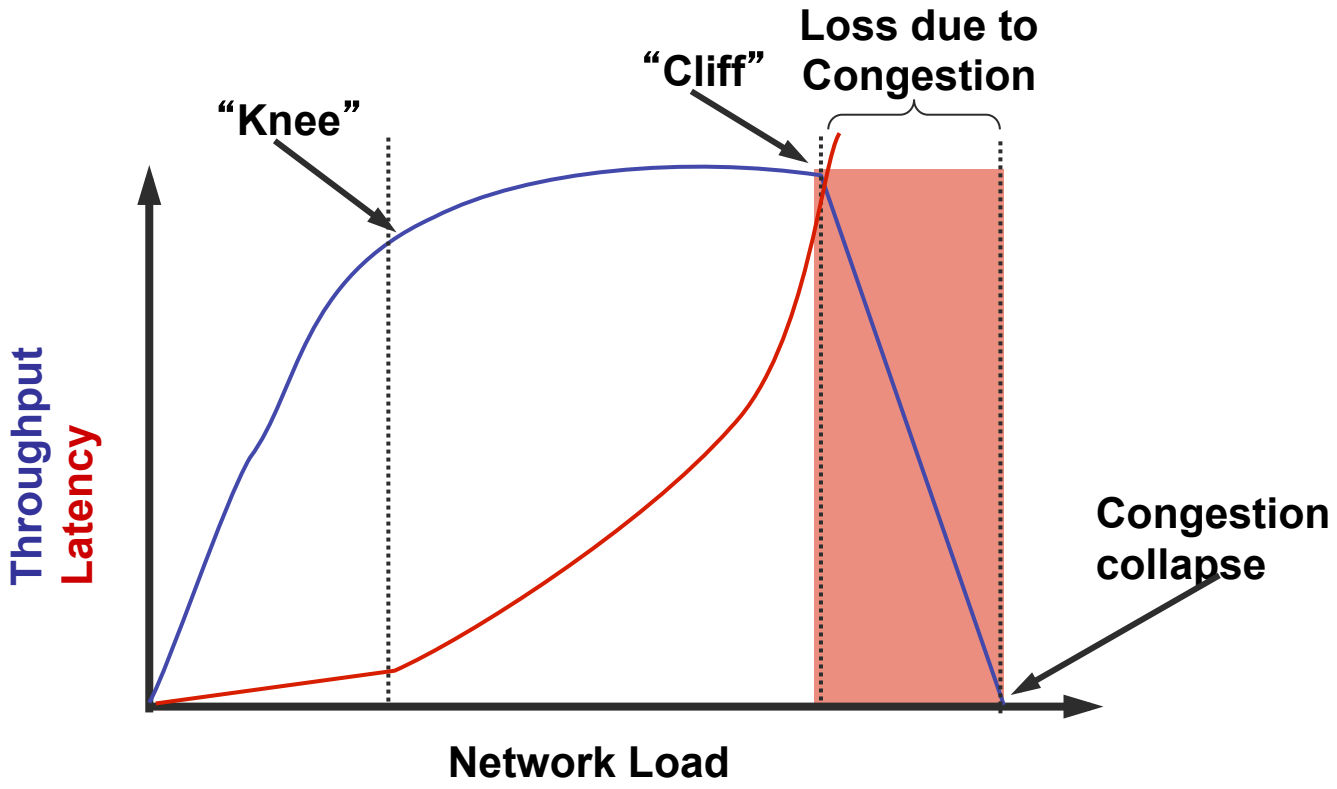
- Increase network resources
  - More buffers for queuing
  - Increase link speed
  - Pros/Cons of these approaches?
- Reduce network load
  - Send data more slowly
  - How much more slowly?
  - When to slow down?

Designing a Control

- Open loop
  - Explicitly reserve bandwidth in the network in advance
- Closed loop
  - Respond to feedback and adjust bandwidth allocation
- Network-based
  - Network implements and enforces bandwidth allocation
- Host-based
  - Hosts are responsible for controlling their sending rate

Proactive *vs.* Reactive

- Congestion avoidance: try to stay to the left of the knee
- Congestion control: try to stay to the left of the cliff



Challenges to Address

- How to detect congestion?
- How to limit sending data rate?
- How fast to send?

Detecting Congestion

- Explicit congestion signaling
  - Source Quench: ICMP message from router to sender
  - DECBit / Explicit Congestion Notification (ECN):
    - » Router *marks* packet based on queue occupancy (i.e. indication that packet encountered congestion along the way)
    - » Receiver tells sender if queues are getting too full
- Implicit congestion signaling
  - Packet loss
    - » Assume congestion is primary source of packet loss
    - » Lost packets indicate congestion
  - Packet delay
    - » Round-trip time increases as packets queue
    - » Packet inter-arrival time is a function of bottleneck link

Throttling Options

- Window-based (TCP)
  - Constrain number of outstanding packets allowed in network
  - Increase window to send faster; decrease to send slower
  - Pro: Cheap to implement, good failure properties
  - Con: Creates traffic bursts (requires bigger buffers)

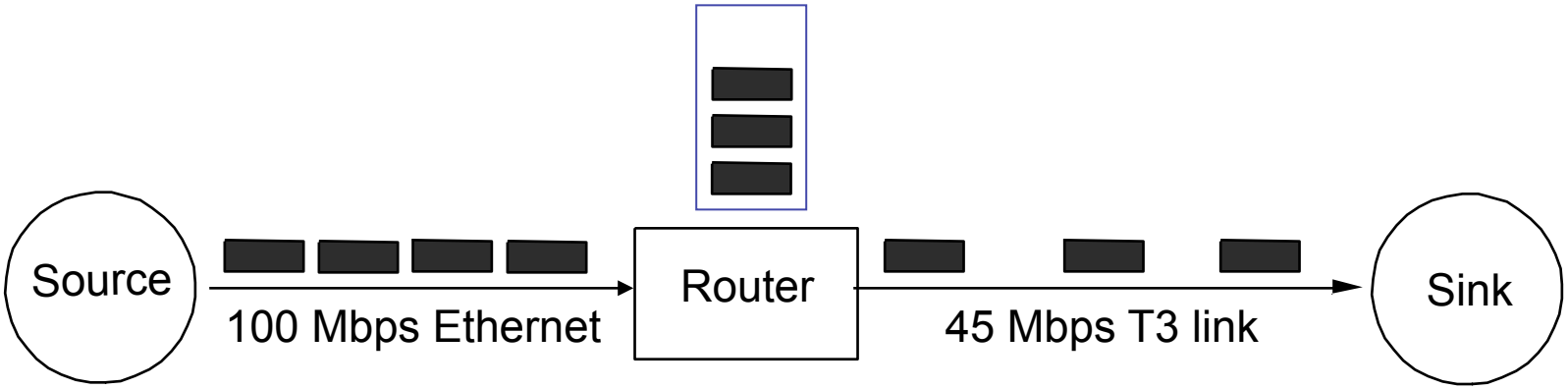
- Rate-based (many streaming media protocols)
  - Two parameters (period, packets)
  - Allow sending of x packets in period y
  - Pro: smooth traffic
  - Con: fine-grained per-connection timers, what if receiver fails?

Choosing a Send Rate

- Ideally: Keep equilibrium at "knee" of power curve
  - Find "knee" somehow
  - Keep number of packets "in flight" the same
  - Don't send a new packet into the network until you know one has left (I.e. by receiving an ACK)
  - What if you guess wrong, or if bandwidth availability changes?

- Compromise: adaptive approximation
  - If congestion signaled, reduce sending rate by *x*
  - If data delivered successfully, increase sending rate by *y*
  - How to relate *x* and *y*? Most choices don't converge…

TCP's Probing Approach



- Each source independently probes the network to determine how much bandwidth is available
  - Changes over time, since everyone does this
- Assume that packet loss implies congestion
  - Since errors are rare; also, requires no support from routers

Basic TCP Algorithm

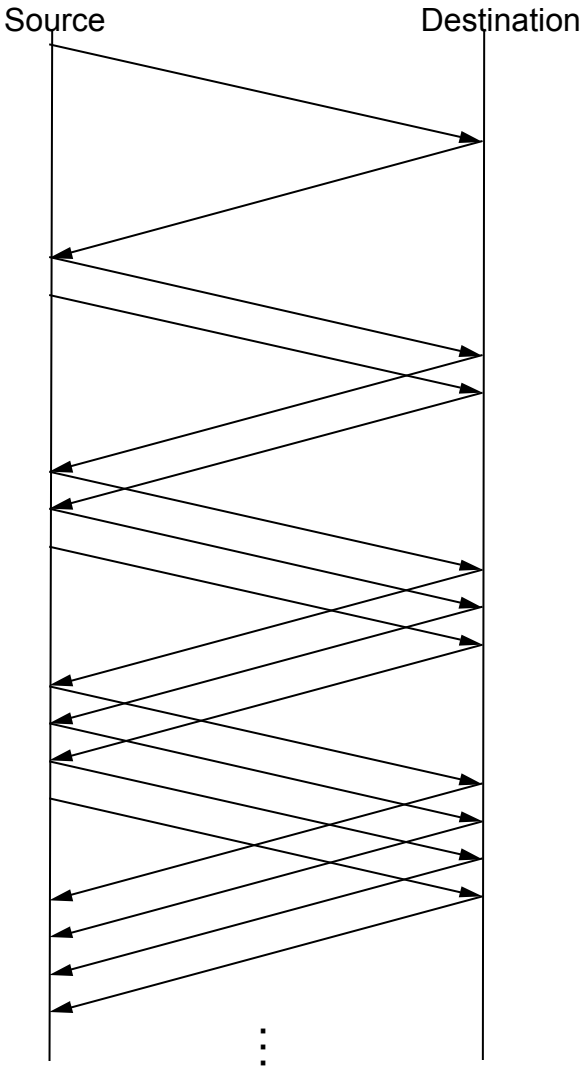
- Window-based congestion control
  - Unified congestion control and flow control mechanism
  - *rwin*: advertised flow control window from receiver
  - *cwnd*: congestion control window
    - » Estimate of how much outstanding data network can deliver in a round-trip time
  - Sender can only send MIN(*rwin,cwnd*) at any time
- Idea: decrease cwnd when congestion is encountered; increase cwnd otherwise
  - Question: how much to adjust?

Congestion Avoidance

- Goal: Adapt to changes in available bandwidth
- Additive Increase, Multiplicative Decrease (AIMD)
  - Increase sending rate by a constant (e.g. MSS)
  - Decrease sending rate by a linear factor (e.g. divide by 2)
- Rough intuition for why this works
  - Let L_i be queue length at time i
  - In steady state: $L_i = N$, where N is a constant
  - During congestion, $L_i = N + yL_{i-1}$, where $y > 0$
  - Consequence: queue size increases multiplicatively
    - » Must reduce sending rate multiplicatively as well

AIMD

- Increase slowly while we believe there is bandwidth
  - Additive increase per RTT
  - *cwnd* += 1 full packet / RTT
- Decrease quickly when there is loss (went too far!)
  - Multiplicative decrease
  - *cwnd* /= 2

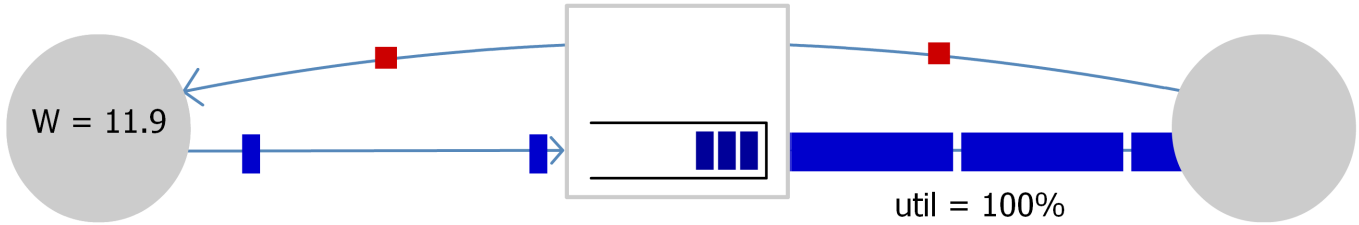


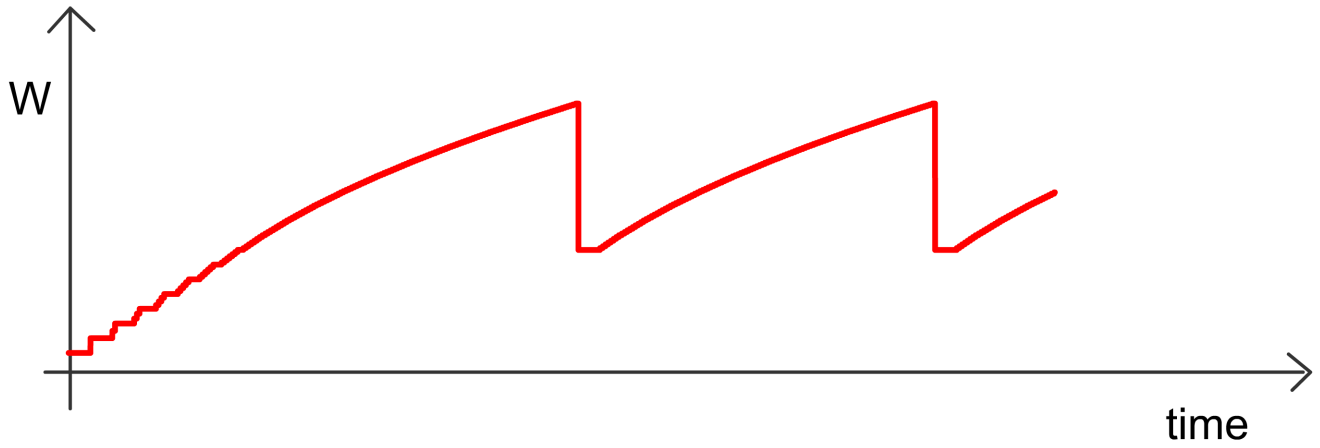
TCP Congestion Control

Only W packets
may be outstanding

Rule for adjusting congestion window (W)

- If an ACK is received: W ← W+1/W
- If a packet is lost: W ← W/2





For next time…

- Finish reading P&D 6.3-4, Sidebar on NAT on p.335
- Keep going on Project 2…