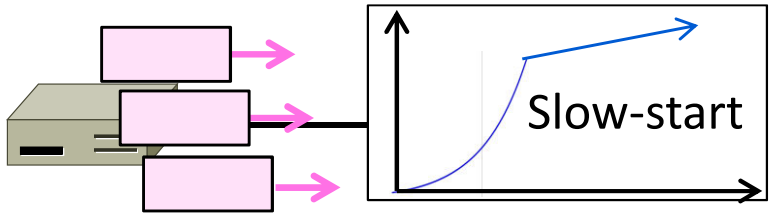
Computer Networks

TCP Slow Start (§6.5.10)

David Wetherall (djw@uw.edu)
Professor of Computer Science & Engineering
UNIVERSITY of WASHINGTON

Topic

- How TCP implements AIMD, part 1
  - "Slow start" is a component of the AI portion of AIMD



Recall

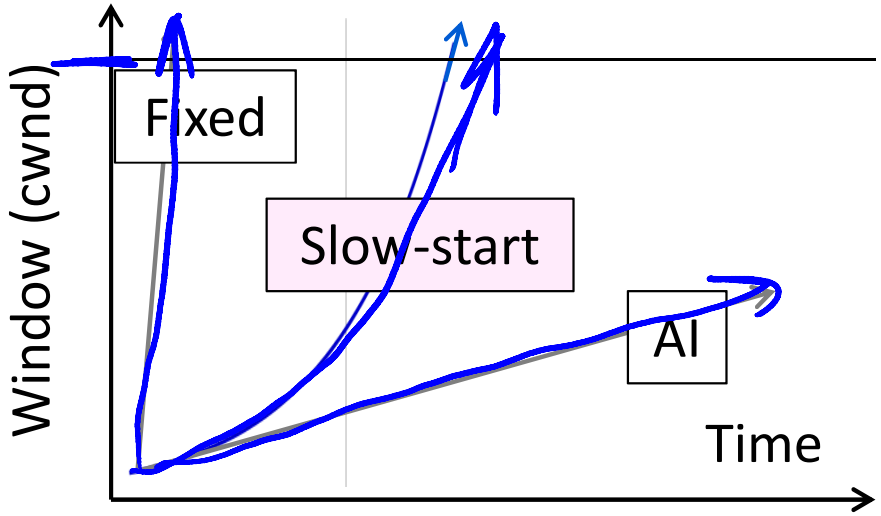
- We want TCP to follow an AIMD control law for a good allocation
- Sender uses a congestion window or cwnd to set its rate (≈cwnd/RTT)
- Sender uses packet loss as the network congestion signal
- Need TCP to work across a very large range of rates and RTTs

TCP Startup Problem

- We want to quickly near the right rate, $cwnd_{IDEAL}$, but it varies greatly
  - Fixed sliding window doesn't adapt and is rough on the network (loss!)
  - AI with small bursts adapts cwnd gently to the network, but might take a long time to become efficient

Slow-Start Solution

- Start by doubling cwnd every RTT
  - Exponential growth (1, 2, 4, 8, 16, …)
  - Start slow, quickly reach large values

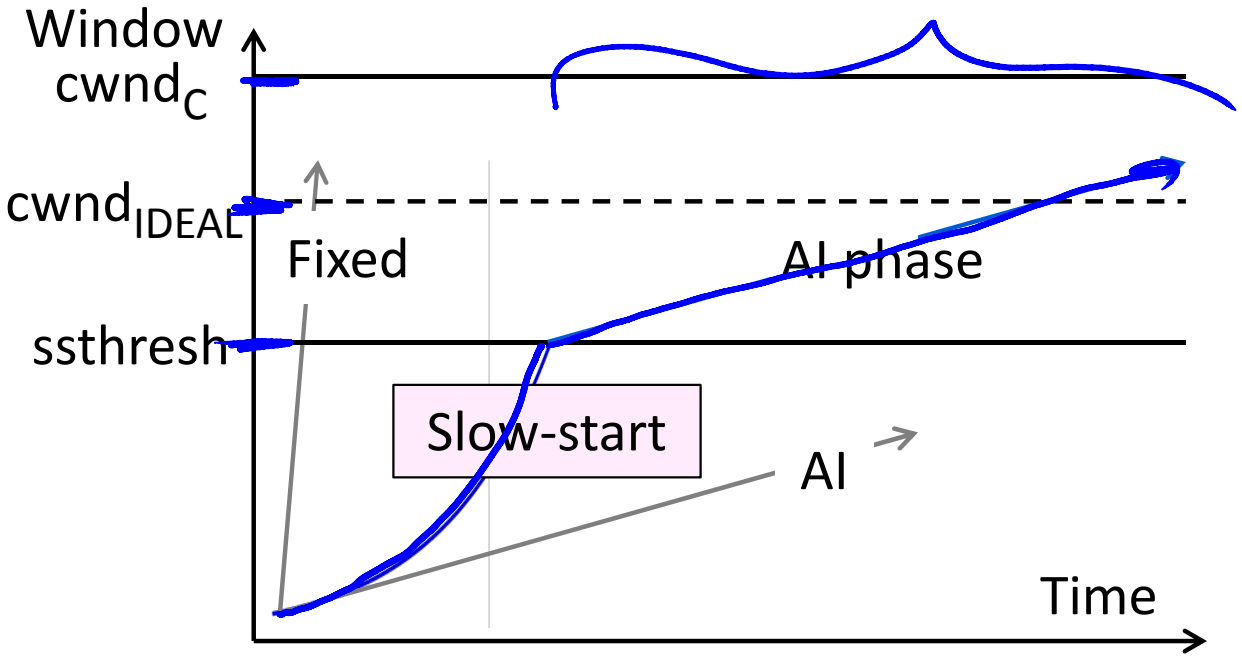


Slow-Start Solution (2)

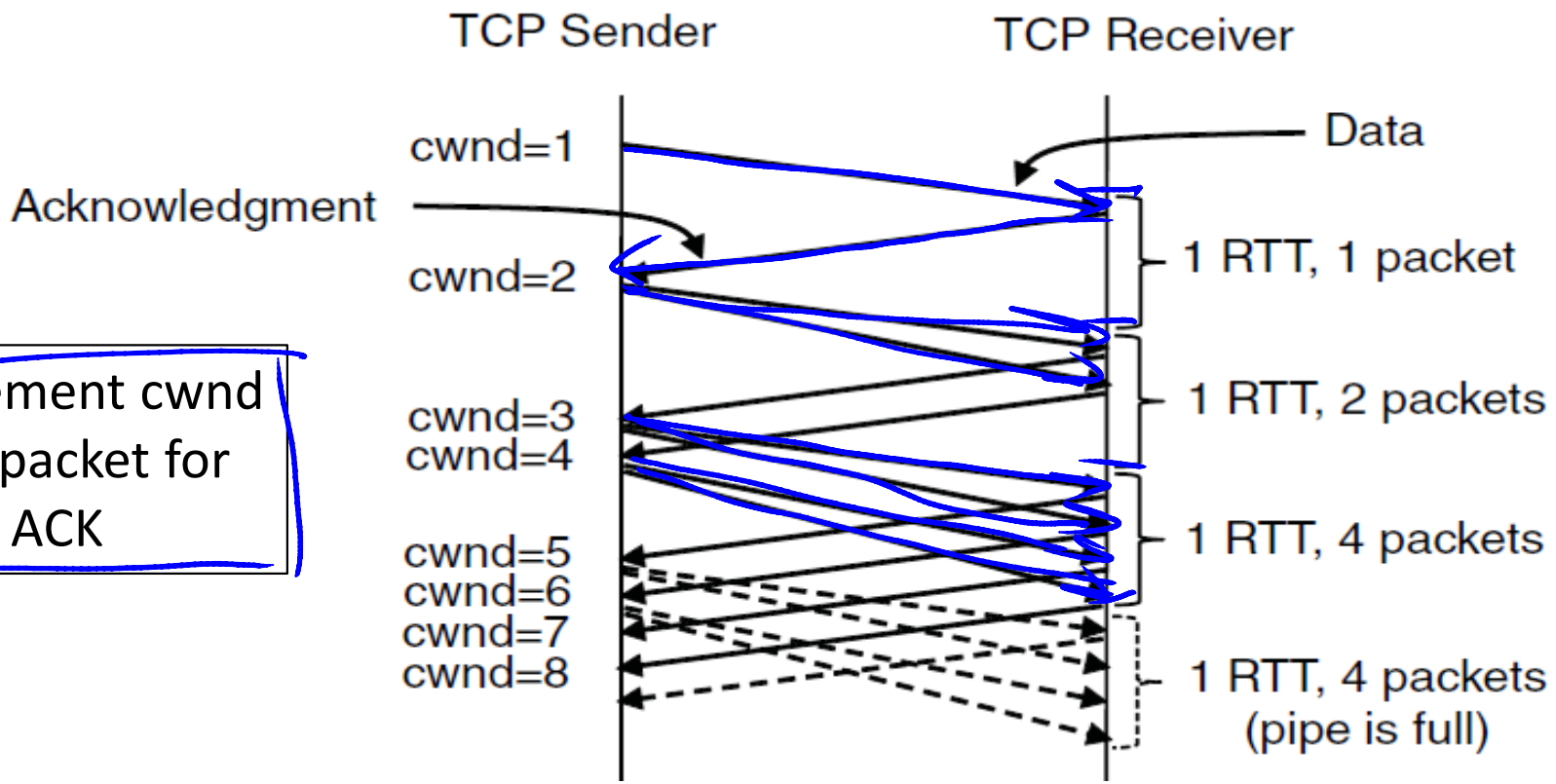
- Eventually packet loss will occur when the network is congested
  - Loss timeout tells us cwnd is too large
  - Next time, switch to AI beforehand
    - Slowly adapt cwnd near right value
- In terms of cwnd:
  - Expect loss for $cwnd_C \approx 2BD+queue$
  - Use ssthresh = $cwnd_C/2$ to switch to AI

Slow-Start Solution (3)

- Combined behavior, after first time
  - Most time spend near right value

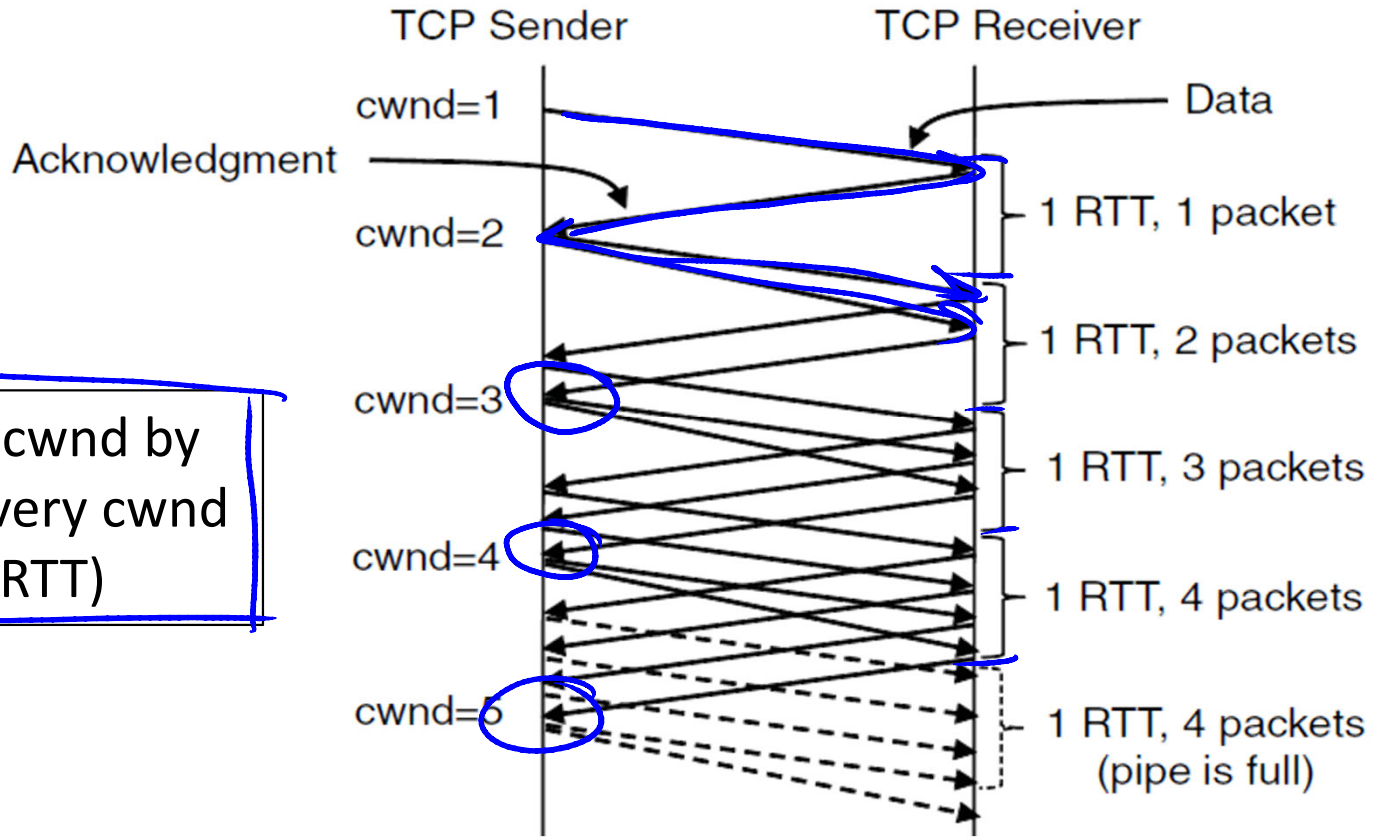


Slow-Start (Doubling) Timeline



Increment cwnd by 1 packet for each ACK

Additive Increase Timeline



Increment cwnd by
1 packet every cwnd
ACKs (or 1 RTT)

TCP Tahoe (Implementation)

- Initial slow-start (doubling) phase
  - Start with cwnd = 1 (or small value)
  - cwnd += 1 packet per ACK
- Later Additive Increase phase
  - cwnd += 1/cwnd packets per ACK
  - Roughly adds 1 packet per RTT
- Switching threshold (initially infinity)
  - Switch to AI when cwnd > ssthresh
  - Set ssthresh = cwnd/2 after loss
  - Begin with slow-start after timeout

Timeout Misfortunes

- Why do a slow-start after timeout?
  - Instead of MD cwnd (for AIMD)
- Timeouts are sufficiently long that the ACK clock will have run down
  - Slow-start ramps up the ACK clock
- We need to detect loss before a timeout to get to full AIMD
  - Done in TCP Reno (next time)

END