

Introduction to Computer Networks

TCP Fast Retransmit / Fast Recovery (§6.5.10)



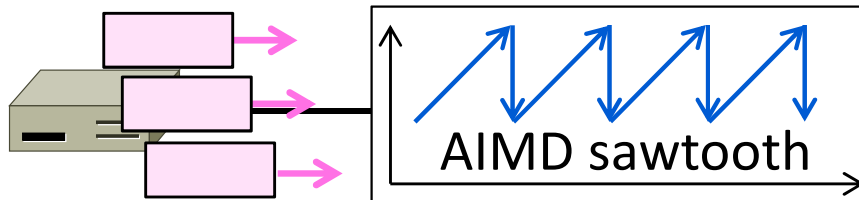
David Wetherall (djw@uw.edu)

Professor of Computer Science & Engineering

UNIVERSITY *of* WASHINGTON

Topic

- How TCP implements AIMD, part 2
 - “Fast retransmit” and “fast recovery” are the MD portion of AIMD



Recall

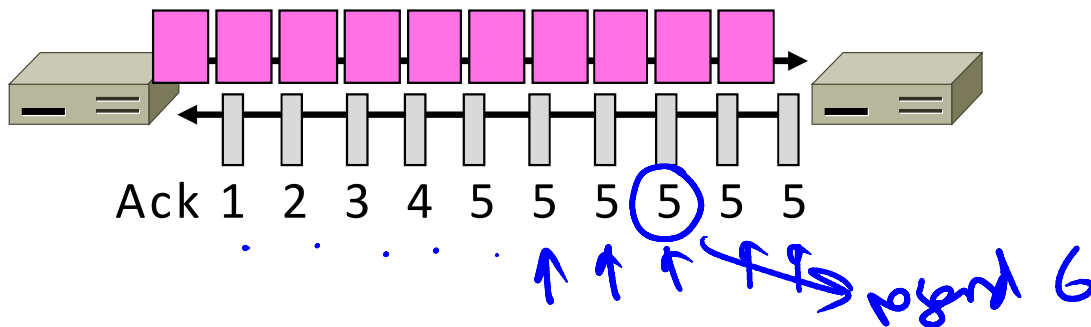
- We want TCP to follow an AIMD control law for a good allocation
- Sender uses a congestion window or cwnd to set its rate ($\approx \text{cwnd}/\text{RTT}$)
- Sender uses slow-start to ramp up the ACK clock, followed by Additive Increase
- But after a timeout, sender slow-starts again with $\text{cwnd}=1$ (as it no ACK clock)

Inferring Loss from ACKs

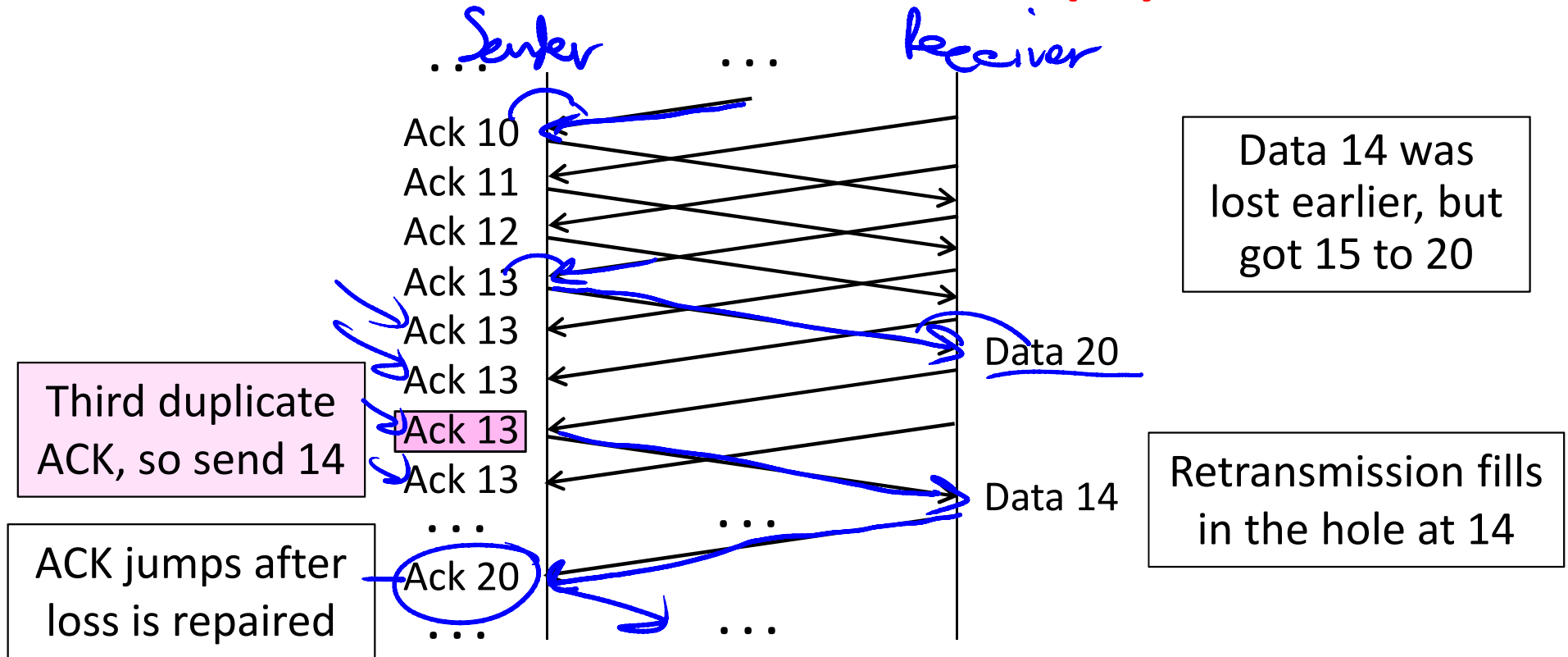
- TCP uses a cumulative ACK
 - Carries highest in-order seq. number
 - Normally a steady advance
- Duplicate ACKs give us hints about what data hasn't arrived
 - Tell us some new data did arrive, but it was not next segment
 - Thus the next segment may be lost

Fast Retransmit

- Treat three duplicate ACKs as a loss
 - Retransmit next expected segment
 - Some repetition allows for reordering, but still detects loss quickly



Fast Retransmit (2)



Fast Retransmit (3)

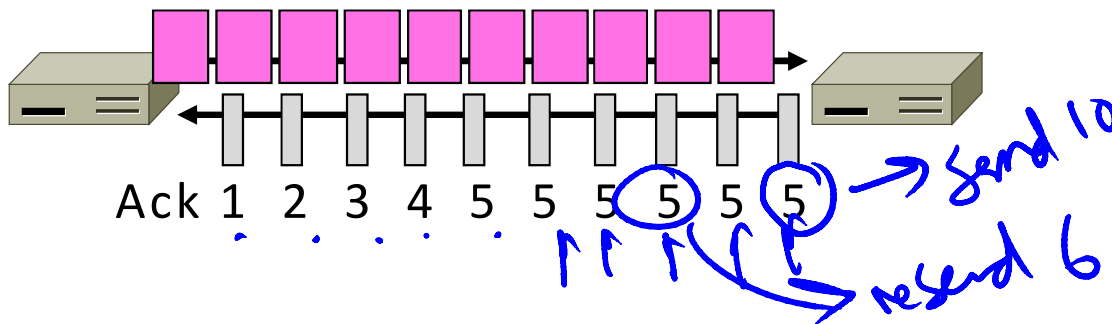
- It can repair single segment loss quickly, typically before a timeout
- However, we have quiet time at the sender/receiver while waiting for the ACK to jump
- And we still need to MD cwnd ...

Inferring Non-Loss from ACKs

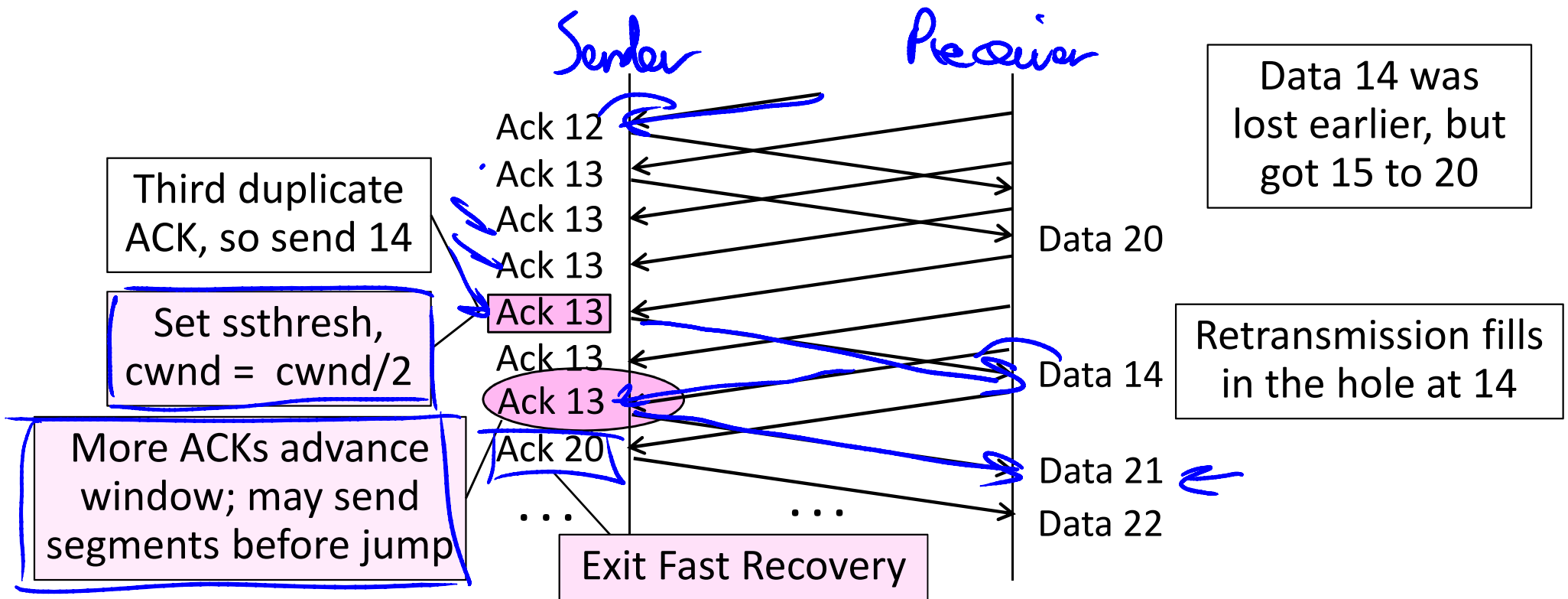
- Duplicate ACKs also give us hints about what data has arrived
 - ➔ Each new duplicate ACK means that some new segment has arrived
 - ➔ It will be the segments after the loss
 - ➔ Thus advancing the sliding window will not increase the number of segments stored in the network

Fast Recovery

- First fast retransmit, and MD cwnd
- Then pretend further duplicate ACKs are the expected ACKs
 - Lets new segments be sent for ACKs
 - Reconcile views when the ACK jumps



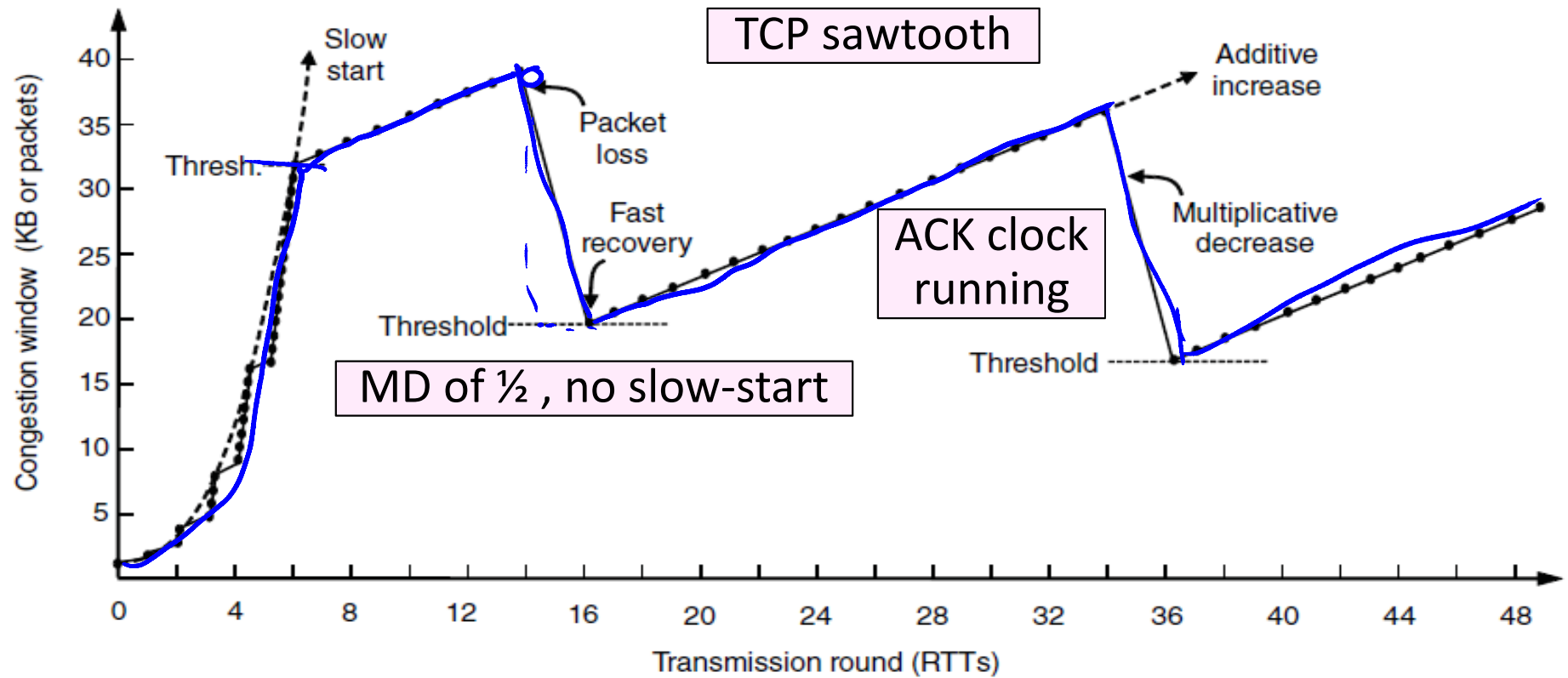
Fast Recovery (2)



Fast Recovery (3)

- With fast retransmit, it repairs a single segment loss quickly and keeps the ACK clock running
- This allows us to realize AIMD
 - No timeouts or slow-start after loss, just continue with a smaller cwnd
- TCP Reno combines slow-start, fast retransmit and fast recovery
 - Multiplicative Decrease is $\frac{1}{2}$

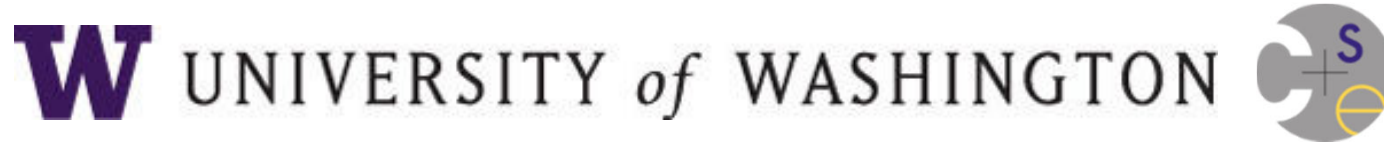
TCP Reno



TCP Reno, NewReno, and SACK

- Reno can repair one loss per RTT
 - Multiple losses cause a timeout
- NewReno further refines ACK heuristics
 - Repairs multiple losses without timeout
- SACK is a better idea ✓
 - Receiver sends ACK ranges so sender can retransmit without guesswork

END



© 2013 D. Wetherall

Slide material from: TANENBAUM, ANDREW S.; WETHERALL, DAVID J., COMPUTER NETWORKS, 5th Edition, © 2011.
Electronically reproduced by permission of Pearson Education, Inc., Upper Saddle River, New Jersey