

CSE/EE 461 – Lecture 17

TCP Congestion Control

David Wetherall
djw@cs.washington.edu

Last Time ...

- The Transport Layer
- Focus
 - How do we allocate bandwidth?
- Topics
 - Congestion
 - Fairness

Application
Presentation
Session
Transport
Network
Data Link
Physical

This Lecture

- The Transport Layer

- Focus

- How does TCP share bandwidth?

- Topics

- Additive Increase/Multiplicative Decrease
- Slow Start
- Fast Recovery

Application
Presentation
Session
Transport
Network
Data Link
Physical

TCP Before Congestion Control

- Just use a fixed size sliding window!

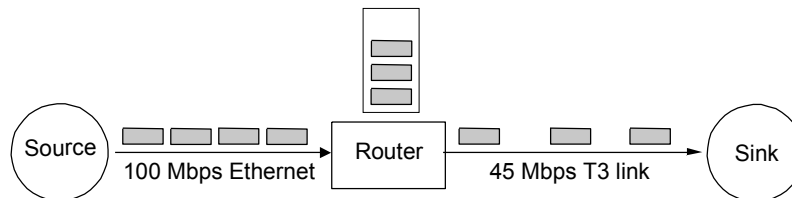
- Will under-utilize the network or cause unnecessary loss

- Congestion control dynamically varies the size of the window to match sending and available bandwidth

- Sliding window uses minimum of cwnd, the congestion window, and the advertised flow control window

- The big question: how do we decide what size the window should be?

TCP Probes the Network

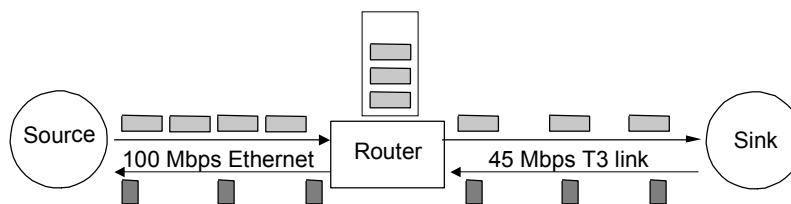


- Each source independently probes the network to determine how much bandwidth is available
 - Changes over time, since everyone does this
- Assume that packet loss implies congestion
 - Since errors are rare; also, requires no support from routers

djw // CSE/EE 461, Winter 2003

L17.5

TCP is “Self-Clocking”



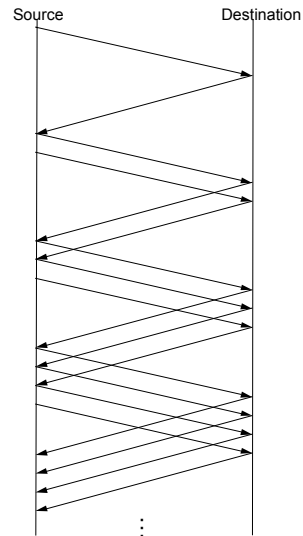
- Neat observation: acks pace transmissions at approximately the bottleneck rate
- So just by sending packets we can discern the “right” sending rate (called the packet-pair technique)

djw // CSE/EE 461, Winter 2003

L17.6

AIMD (Additive Increase/Multiplicative Decrease)

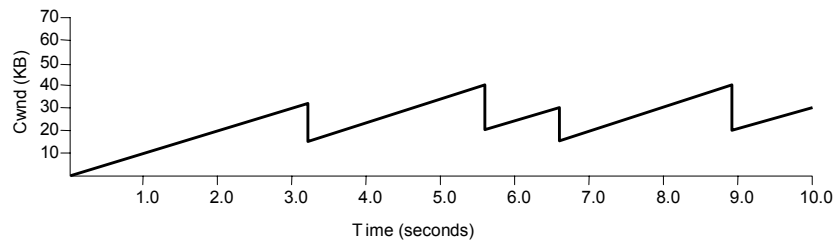
- How to adjust probe rate?
- Increase slowly while we believe there is bandwidth
 - Additive increase per RTT
 - $Cwnd += 1 \text{ packet} / \text{RTT}$
- Decrease quickly when there is loss (went too far!)
 - Multiplicative decrease
 - $Cwnd /= 2$



djw // CSE/EE 461, Winter 2003

L17.7

TCP Sawtooth Pattern

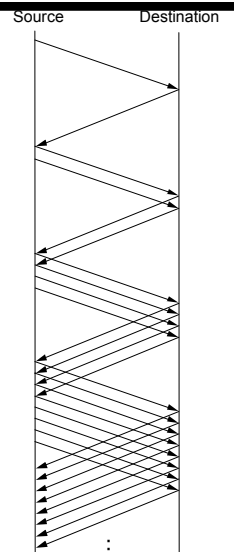


djw // CSE/EE 461, Winter 2003

L17.8

“Slow Start”

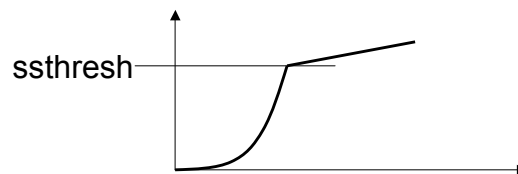
- Q: What is the ideal value of cwnd? How long will AIMD take to get there?
- Use a different strategy to get close to ideal value
 - Double cwnd every RTT
 - $Cwnd *= 2 / RTT$
 - $Cwnd += 1 / \text{packet received}$



djw // CSE/EE 461, Winter 2003

L17.9

Combining Slow Start and AIMD

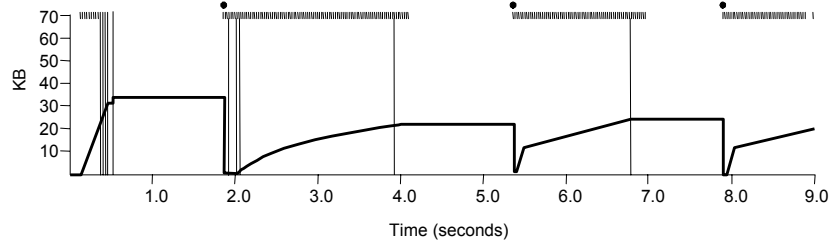


- Slow start is used whenever the connection is not running with packets: initially, and after timeouts
- But we don't want to overshoot our ideal cwnd, so remember the last cwnd that worked with no loss
 - $Ssthresh = cwnd \text{ after } cwnd /= 2 \text{ on loss}$
 - Switch to AIMD once cwnd passes ssthresh

djw // CSE/EE 461, Winter 2003

L17.10

Example (Slow Start +AIMD)

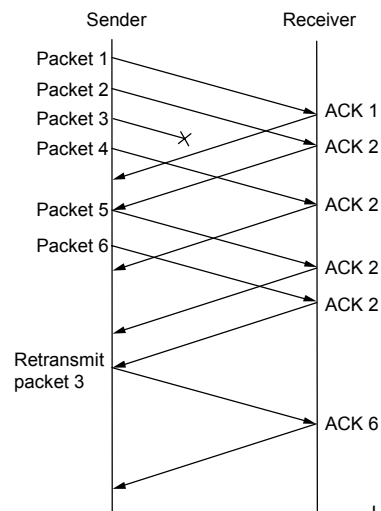


djw // CSE/EE 461, Winter 2003

L17.11

Fast Retransmit

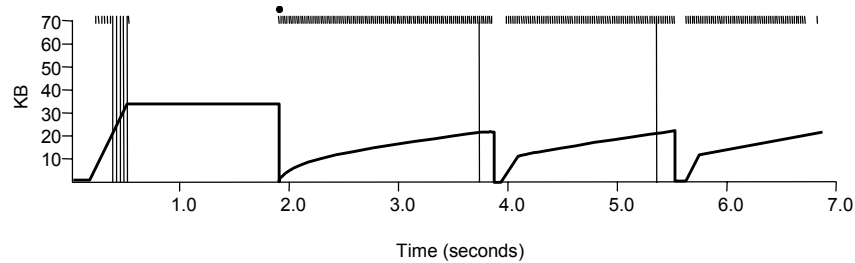
- TCP uses cumulative acks, so duplicate acks start arriving after a packet is lost.
- We can use this fact to infer which packet was lost, instead of waiting for a timeout.
- 3 duplicate acks are used in practice



djw // CSE/EE 461, Winter 2003

L17.12

Example (with Fast Retransmit)



djw // CSE/EE 461, Winter 2003

L17.13

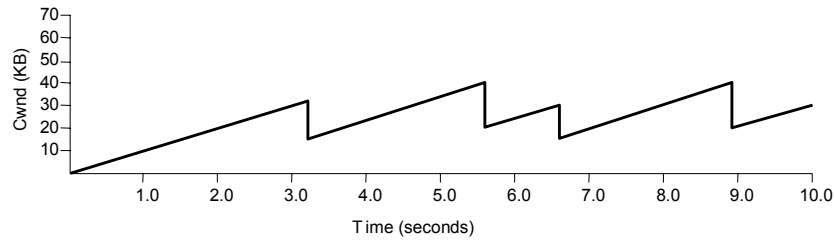
Fast Recovery

- After Fast Retransmit, use further duplicate acks to grow cwnd and clock out new packets, since these acks represent packets that have left the network.
- End result: Can achieve AIMD when there are single packet losses. Only slow start the first time.

djw // CSE/EE 461, Winter 2003

L17.14

Example (with Fast Recovery)



djw // CSE/EE 461, Winter 2003

L17.15

Key Concepts

- TCP probes the network for bandwidth, assuming that loss signals congestion
- The congestion window is managed to be additive increase / multiplicative decrease
 - It took fast retransmit and fast recovery to get there
- Slow start is used to avoid lengthy initial delays
 - Ramp up to near target rate and then switch to AIMD

djw // CSE/EE 461, Winter 2003

L17.16