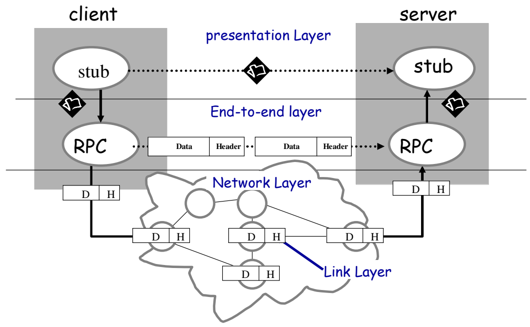
L13: Sharing in network systems

6.033 Spring 2007

http://web.mit.edu/6.033

Slides from many folks

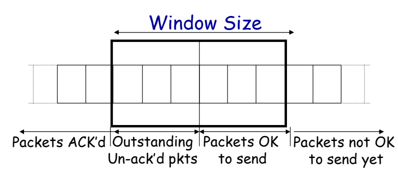
Where is sharing happening?



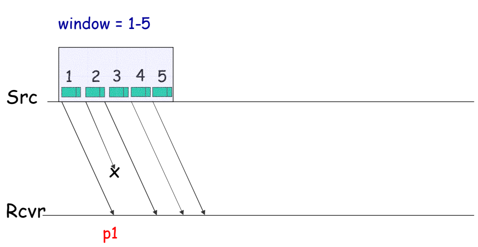
This Lecture

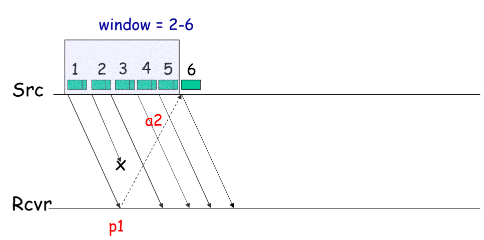
- Problems:
  - Sharing server
  - Sharing network
- Solution:
  - Set the window size carefully
  - Sharing server: flow control
  - Sharing the network: congestin control

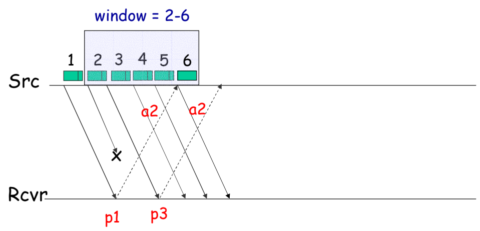
Sliding Window

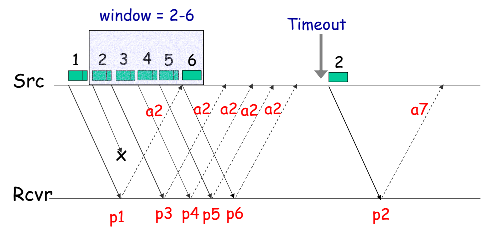


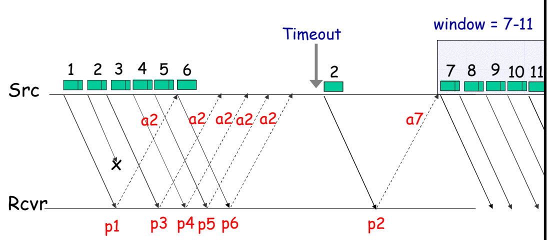
- The window advances/slides upon the arrival of an ack
- The sender sends only packets in the window
- Receiver usually sends *cumulative acks*
  - i.e., receiver acks the next expected in-order packet











In this example, the receiver sent cumulative acks, but the same behavior happens if the receiver acks the received sequence number

What is the right window size?

- The window limits how fast the sender sends
- Two mechanisms control the window:
  - Flow control
  - Congestion control

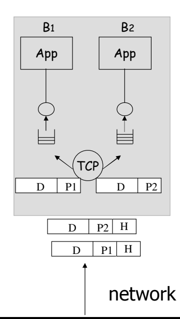
Flow Control

- The receiver may be slow in processing the packets à receiver is a bottleneck
- To prevent the sender form overwhelming the receiver, the receiver tells the sender the maximum number of packets it can buffer fwnd
- Sender sets W ≤ fwnd

How to set fwnd?

Multiple applications run on the same machine but use different ports

- Fwnd = B x RTT
  - Size of queue substitute for B
- Adapts to
  - RTT changes
  - B changes
- "self-pacing"



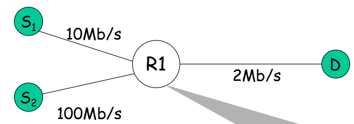
Sharing the network

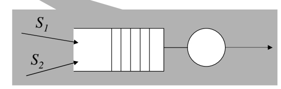
How do you manage the resources in a huge system like the Internet, where users with different interests share the same resources?

Difficult because of:

- Size
  - Millions of users, links, routers
- Heterogeneity
  - bandwidth: 9.6Kb/s (then modem, now cellular), 10 Tb/s
  - latency: 50us (LAN), 133ms (wired), 1s (satellite), 260s (Mars)

Congestion

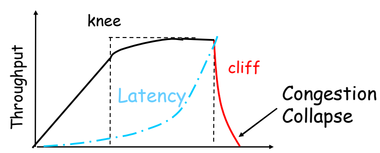




- Sources share links, and buffer space
- Why a problem?
  - Sources are unaware of current state of resource
  - Sources are unaware of each other
- Manifestations:
  - Lost packets (buffer overflow at routers)
  - Long delays (queuing in router buffers)
  - Long delays may lead to retransmissions, which lead to more packets….

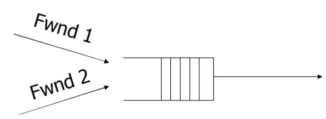
Danger: Congestion Collapse

Increase in input traffic leads to decrease in useful work



- Causes of Congestion Collapse
  - Retransmissions introduce duplicate packets
  - Duplicate packets consume resources wasting link capacity

Example: old TCP implementations



- Long haul network (i.e., large RTT)
- Router drops some of TCP 2's fwnd packets
  - Each discard packet will result in timeout
- At timeout TCP 2 resends complete window
  - Cumulative ACK, timeouts fire off at "same" time
- Blizzard of retransmissions can result in congestion collapse
  - Insufficiently adaptive timeout algorithm made things worse

What can be done in general?

- Avoid congestion:
  - Increase network resources
    - But demands will increase too!
  - Admission Control & Scheduling
    - Used in telephone networks
    - Hard in the Internet because can't model traffic well
  - Perhaps combined with Pricing
    - senders pay more in times of congestion
- Congestion control:
  - Ask the sources to slow down; But how?
    - How do the sources learn of congestion?
    - What is the correct window?
    - How to adapt the window as the level of congestion changes?

How do senders learn of congestion?

Potential options:

- Router sends a Source Quench to the sender
- Router flags the packets indicating congestion
- Router drops packets when congestion occurs
  - Sender learns about the drop because it notices the lack of ack
  - Sender adjusts window

Case study: current TCP

- Define a congestion control window cwnd
- Sender's window is set to W = min (fwnd, cwnd)
- Simple heuristic to find cwnd:
  - Sender increases its cwnd slowly until it sees a drop
  - Upon a drop, sender decreases its cwnd quickly to react to congestion
  - Sender increases again slowly
- No changes to protocol necessary!

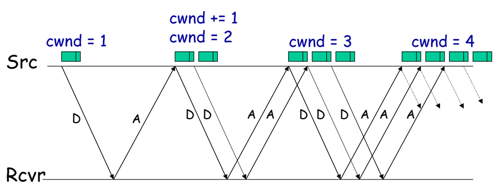
TCP Increase/decrease algorithm

- AIMD:
  - Additive Increase Multiplicative Decrease
- Every RTT:

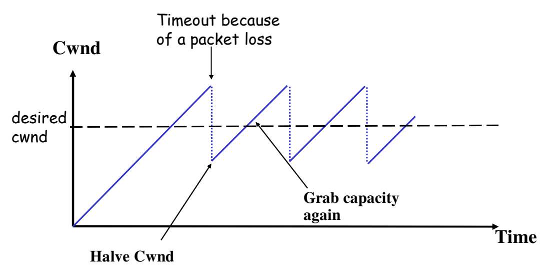
  No drop: cwnd = cwnd + 1

  drop: cwnd = cwnd /2

Additive Increase



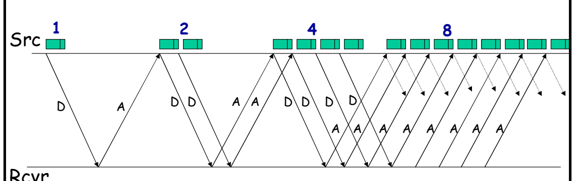
TCP AIMD



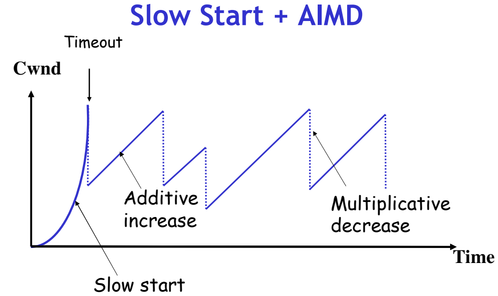
Need the queue to absorb these saw-tooth oscillations

TCP "Slow Start"

- How to set the initial cwnd?
- At the beginning of a connection, increase exponentially
  - Every RTT, double cwnd



Slow Start + AIMD



Fairness?

- No!
  - Applications don't have to use TCP
  - Use multiple TCP connections

Summary

- Controlling complexity in network systems
  - Layering
  - Interesting division of labors based on E2E principle
  - Case study: Internet
- Interesting problems and techniques
  - Packets
  - Protocols
  - …
- Client-server implementation

- Next: Application-level reliability and security