CS162
Operating Systems and Systems Programming
Lecture 22

Networking III

April 22, 2010

Ion Stoica
http://inst.eecs.berkeley.edu/~cs162

Review

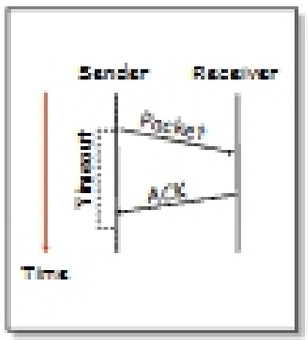
- Link (datalink) layer: Broadcast network; frames sent by one host reaches every other host in same network
  - Multi-access protocol
  - (didn't go over) construct frames, error detection and correction, flow control, …
- Network layer: stitch together multiple link layer networks
  - Deliver a packet to specified network destination
  - (didn't go over) segmentation/reassemble, packet scheduling, buffer management
- Transport layer
  - Multiplexing/demultiplexing (two lectures ago)
  - Flow & congestion control, in-order delivery, reliability (today)

Transport Protocol

- Flow control keeps *one fast sender* from *overwhelming a slow receiver*
- Congestion control keeps a *set of senders* from *overloading the network*
- Reliability makes sure the *receiver got all packets* sent by *sender*
- In-order delivery makes sure the *receiver delivers the packet to application in same order sender* sent them
- Two protocols:
  - Stop-and-Wait
  - Window based

Automatic Repeat reQuest (ARQ)

- Automatic Repeat Request
  - Receiver sends acknowledgment (ACK) when it receives packet
  - Sender waits for ACK and *times out* if does not arrive within some time period
- Simplest ARQ protocol
  - Stop and Wait
  - Send a packet, stop and wait until ACK arrives

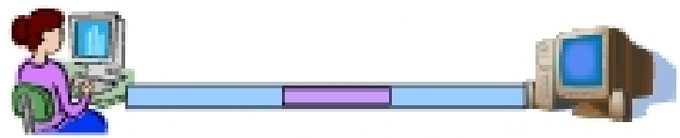


Stop-and-Wait Properties

- Flow control: yes
  - Receiver can implicitly slow down sender by acking a packet only if it has room for at lest another packet
  - Assumption: timeout doesn't trigger before receiving ack
- Congestion control: yes
  - Sender sends a new packet only after previous one made it
  - If network is congested packet or ack is lost → sender doesn't send new data
- Reliability: yes
  - If a packet is lost, sender timeouts and resends the packet
- In-order delivery: yes
  - Receiver doesn't get next packet before receiving (and acking) previous one
- So what's the problem with Stop-and-Wait? Efficiency!

How Fast Can Stop-and-Wait Go?

- Suppose we're sending from UCB to New York:
  - Bandwidth = 1 Mbps (megabits/sec)
  - RTT = 100 msec
  - Maximum packet size a.k.a. Maximum Transmission Unit (MTU) = 1500 B = 12,000 b
  - No other load on the path and no packet loss
- What (approximately) is the fastest we can transmit using Stop-and-Wait?
  - Answer: 12,000b/0.1s = 120 kbps
- How about if Bandwidth = 1 Gbps?



Administrivia

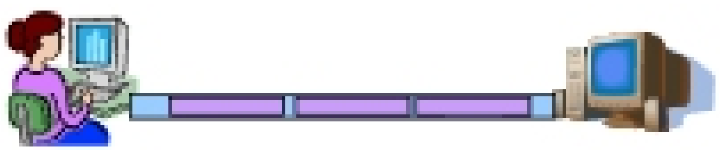
- Keys to access AWS will be sent today
- Last two lectures on security
- Final Exam
  - Friday, May 14, 7:00PM-10:00PM
  - All material from the course
    - With slightly more focus on second half - but you are still responsible for all the material
  - Two sheets of notes, both sides

Sliding Window

- Idea: allow multiple packets in-flight
  - "In-flight" = un-acked packets
- Window size (W): number of packets the sender can send without receiving an ack
  - E.g., after receiving ack for all packet before and including K, send packets K+1, K+2, …, K+W+1
  - Stop-and-wait: particular case of sliding window, W=1
- Receiver tells sender W
  - W cannot be larger than receiver's buffer!

Throughput

- Up to W packets (or bytes) per RTT
- Throughput = W/RTT
- How large should be the window to fully utilize a link with bandwidth B?
  - W = Bandwidth x RTT (i.e., "Bandwidth-Delay" or "Delay-Bandwidth" product)

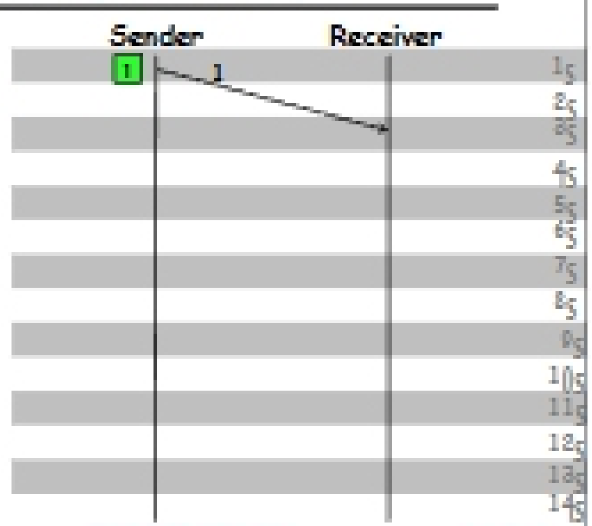


Sliding Window Example (This is NOT TCP !)

- Sender
  - Sending rate = 1 pkt/s
  - Packet size = 1000b
- Receiver:
  - Delivering rate = 0.5 pkt/s
  - Delivers packets in **sequence** to application
  - Acknowledges (acks) each delivered pkt
  - Send negative ack. (nack) if packet lost
- Round-trip time = 4 sec, 2sec each way
- Receiver Window = 4 packets
- Note: max. achievable throughput = 0.5pkt/s = 500b/s

Sliding Window Example

- Sender, at 1s
  - Send 1st pkt



Sliding Window Example

- Sender, at 1s
  - Send 1st pkt
- Receiver, at 3s
  - Get 1st pkt
  - Deliver 1st pkt to appl.
  - Send ack=1 to sender