

---

# CSE331: Introduction to Networks and Security

Lecture 10  
Fall 2006



# Announcements

---

- HW 1 Due on Friday
- Midterm I will be held next Friday, Oct. 6th.
  - Will cover all course material up to next Weds.



# Recap: Reliable Transmission

---

- Sliding window algorithm
  - Sequence numbers
  - Sliding window size
- TCP - Transmission Control Protocol

# Window Sizes

---

- If  $RTT \times \text{Bandwidth}$  product is known then  $SWS = RTT \times \text{Bandwidth} / \text{Framesize}$
- Receive window size:
  - 1 = no buffering of out-of-order frames
  - $RWS = SWS$  buffers as many as can be in flight
  - Note that  $RWS > SWS$  is not sensible

# Finite Sequence Numbers

---

- Recall that for Stop-and-Wait we needed two sequence numbers.
- How many do we need for Sliding Window?
- Suppose  $SWS=RWS$ 
  - How many sequence numbers should there be?
  - Is  $SWS + 1$  sufficient?

# Sufficient MaxSeqNum

---

- Frame  $i$ 's sequence num is  $i \% \text{MaxSeqNum}$
- Assuming  $\text{SWS} = \text{RWS}$
- $\text{SWS} < (\text{MaxSeqNum} + 1)/2$
- Why?
  - Consider case where all the ACKS are lost.
  - Suppose  $\text{SWS} = \text{RWS} = 3$
  - $\text{MaxSeqNum} = 5$  (sequence numbers = 0,1,2,3,4) is insufficient



# Roles of Sliding Window Algorithm

- Reliable delivery
  - It provides an efficient retransmission protocol for dealing with errors
- In-order delivery
  - The receiver buffers frames and delivers them in sequence number order
- Flow control
  - It sends ACKs back to give hints to sender
  - More sophisticated version could give # of frames the receiver has room for—throttles the sender.

# Sliding window in practice

---

- TCP (Transmission Control Protocol)
  - Transportation layer protocol
  - Uses sliding window algorithm
  - More complex because it's used in an Internetwork – not over a direct link
  
  - Bandwidth x delay not known
  - Dynamically changes timeouts
  - Larger buffers for in-order delivery

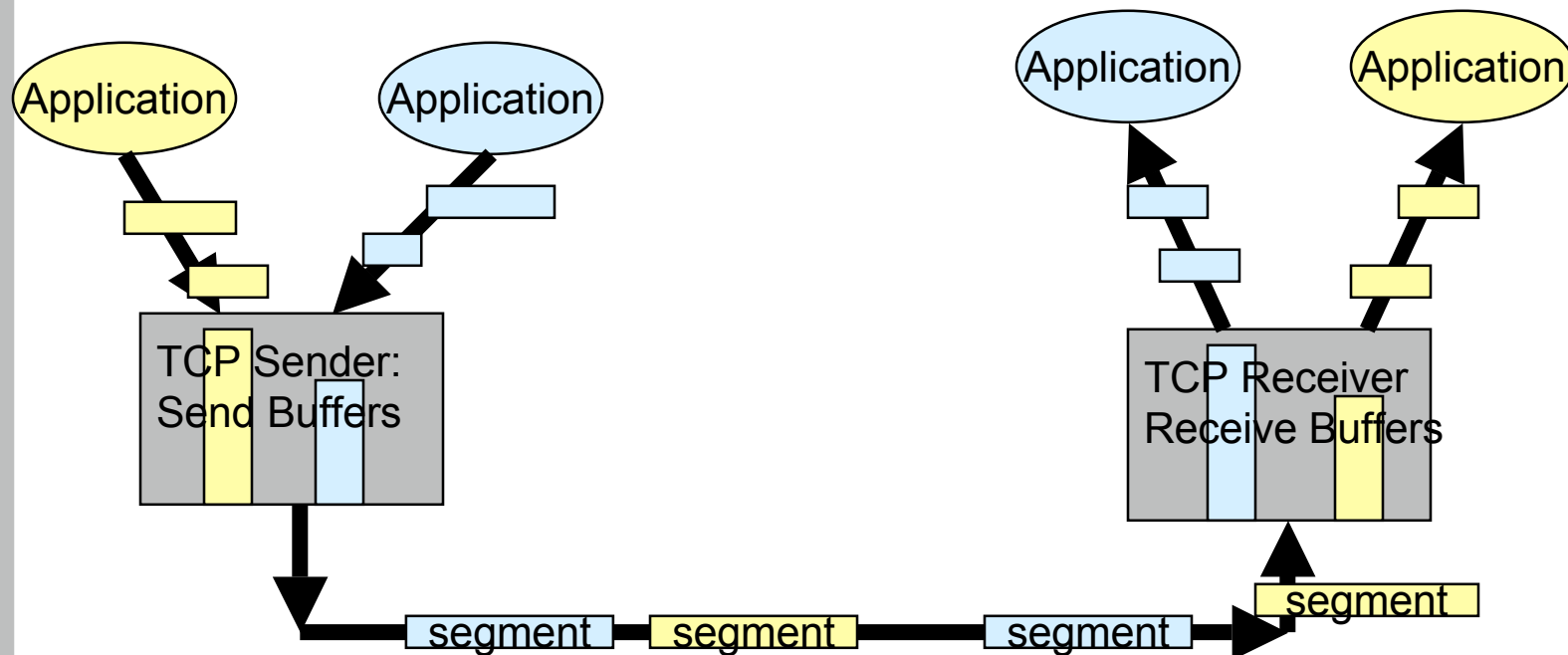


# Transmission Control Protocol (TCP)

- Most widely used protocol for reliable byte streams
  - Reliable, in-order delivery of a stream of bytes
  - Full duplex: pair of streams, one in each direction
  - Flow and congestion control mechanisms
  - Supports ports
- Built on top of IP (hence TCP/IP)

# TCP End-to-End Model

- Buffering corrects errors but may introduce delays



# Using Ports

---

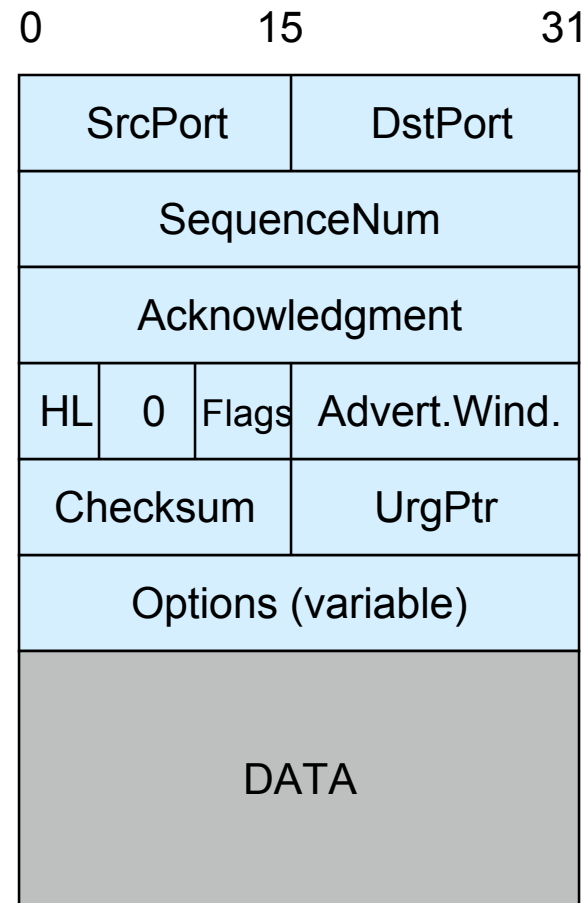
- Client contacts Server at a *well-known port*
  - DNS: port 53
  - POP3: port 110
  - Unix talk : port 517
  - In unix, ports are listed in /etc/services
- Sometimes Client and Server agree on a different port for subsequent communication
- Ports are an abstraction
  - Implemented differently on different OS's
  - Typically a message queue

# Packet Format

- Flags

- SYN
- FIN
- RESET
- PUSH
- URG
- ACK

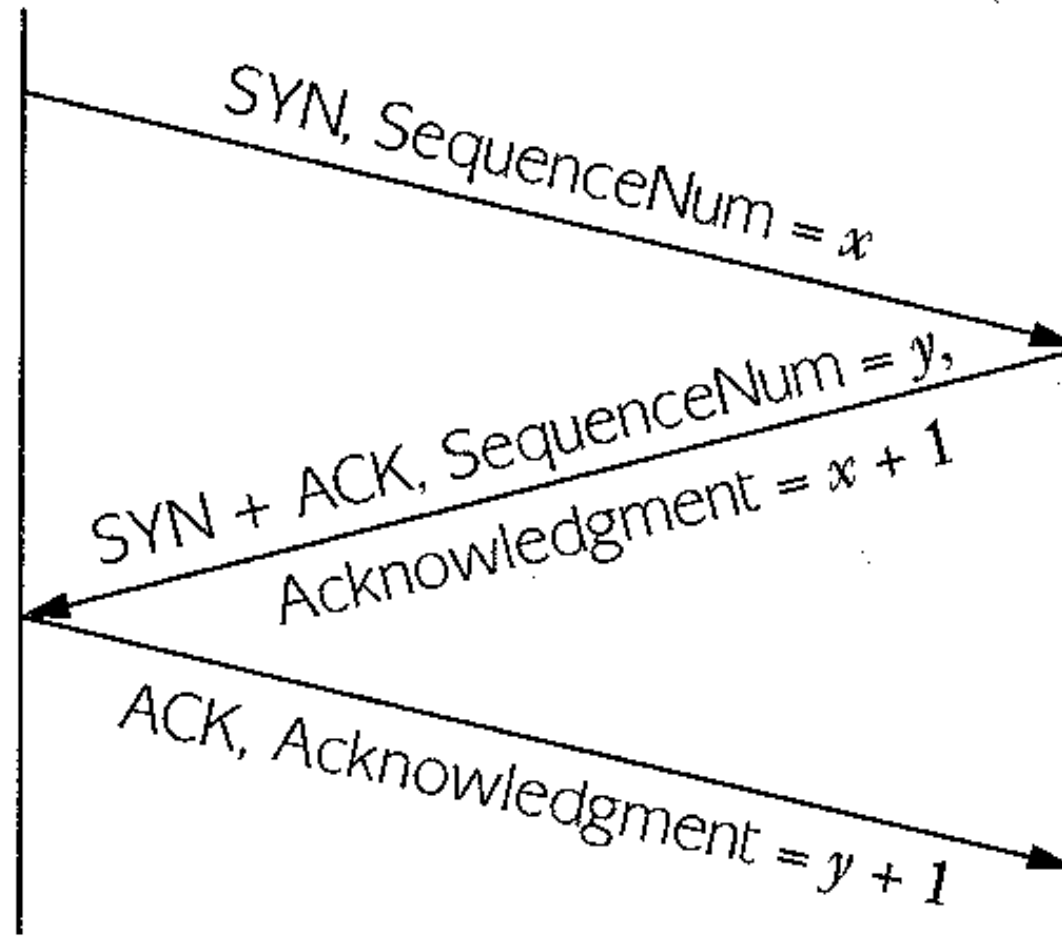
- Fields



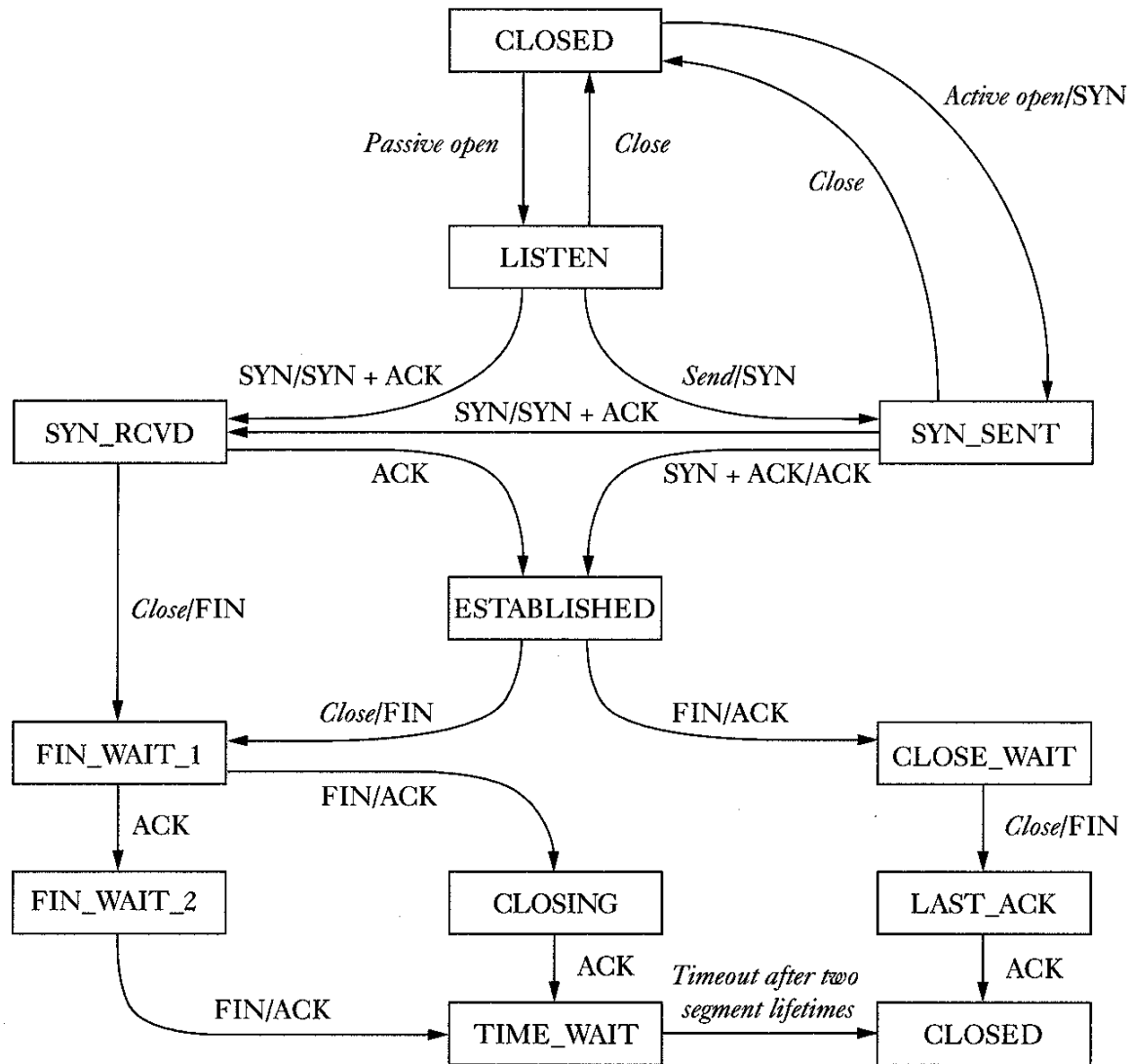
# Three-Way Handshake

Active participant  
(client)

Passive participant  
(server)



# TCP State Transitions



# TCP Receiver

---

- Maintains a buffer from which application reads
- Advertises  $N < \text{buffer size}$  as the window size for sliding window
- Responds with Acknowledge and AdvertisedWindow on each send; updates byte counts when data O.K.
- Application blocked until read() O.K.

# TCP Sender

---

- Maintains a buffer; sending application is blocked until room in the buffer for its write
- Holds data until acknowledged by receiver as *successfully received*
- Implements window expansion and contraction
  - Dynamically adjust the sliding window size
  - *flow* and *congestion* control



# TCP Flow & Congestion Control

---

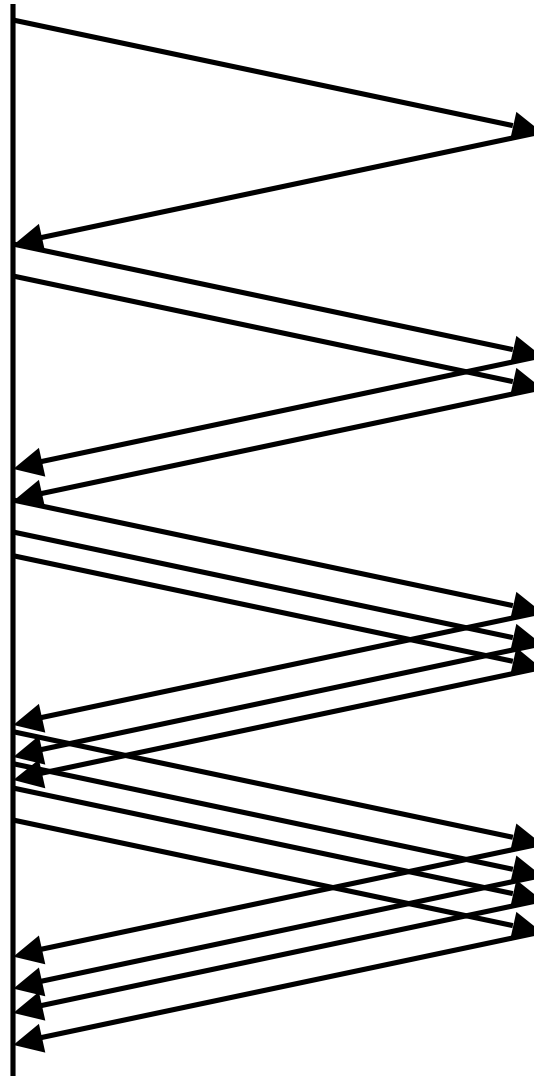
- Flow vs. Congestion Control
  - Flow control protects the recipient from being overwhelmed.
  - Congestion control protects the network from being overwhelmed.
- TCP Congestion Control
  - Additive Increase / Multiplicative Decrease
  - Slow Start
  - Fast Retransmit and Fast Recovery

# Increase and Decrease

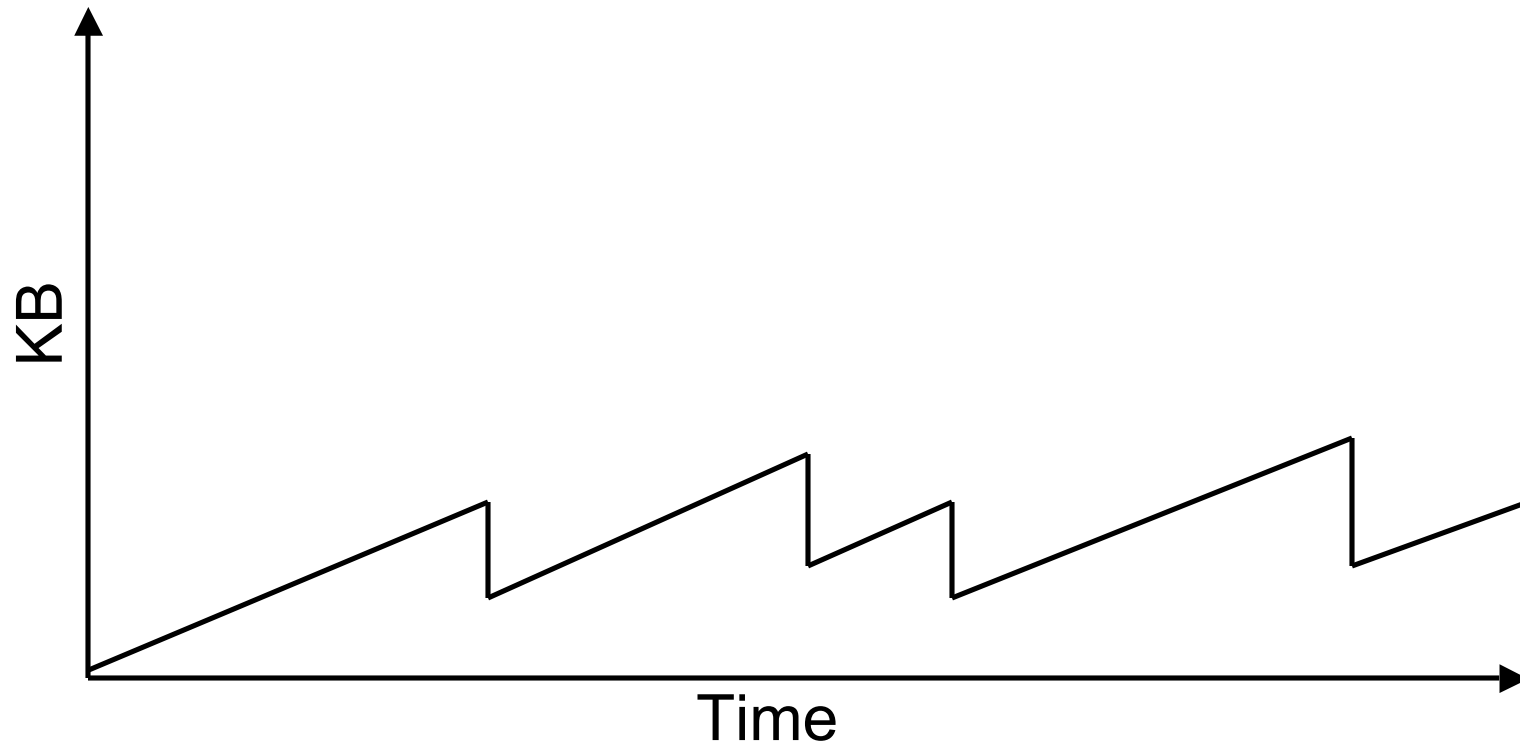
---

- A value CongestionWindow is used to control the number of unacknowledged transmissions.
- This value is increased linearly until timeouts for ACKs are missed.
- When timeouts occur, CongestionWindow is decreased by half to reduce the pressure on the network quickly.
- The strategy is called “additive increase / multiplicative decrease”.

# Additive Increase



# TCP Sawtooth Pattern



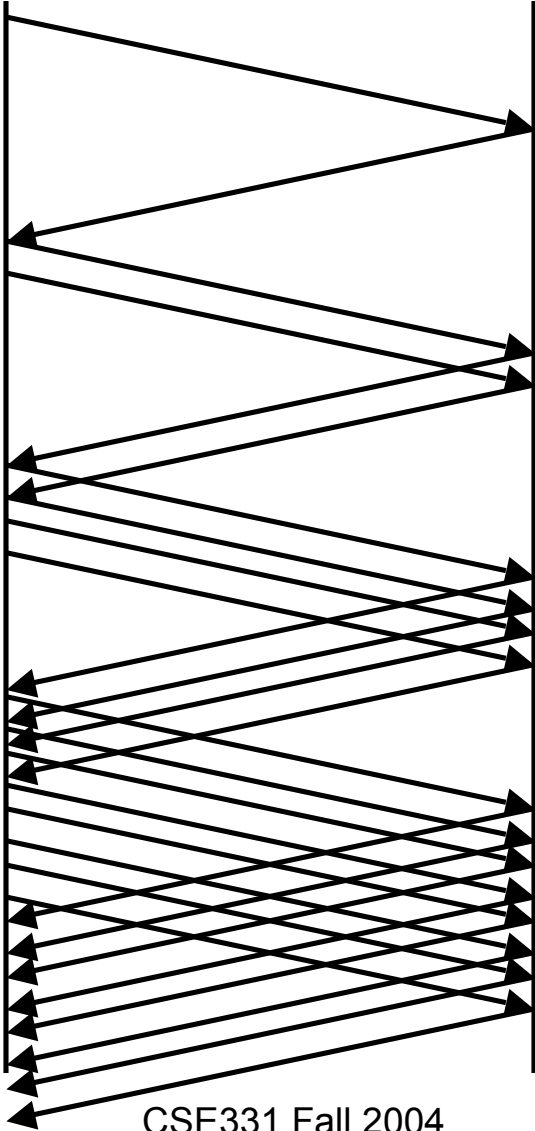


# Slow Start

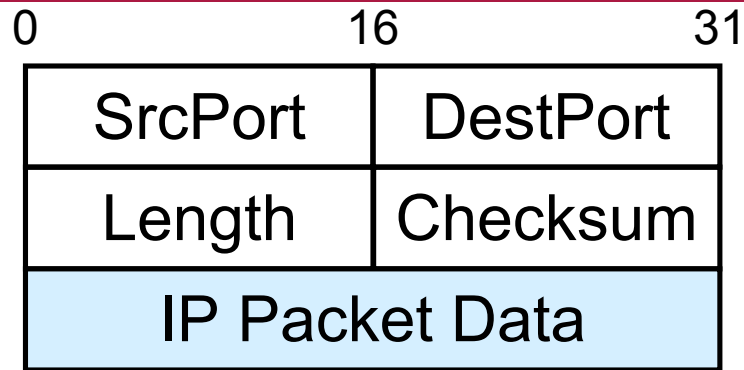
---

- Sending the entire window immediately could cause a traffic jam in the network.
- Begin “slowly” by setting the congestion window to one packet.
- When acknowledgements arrive, double the congestion window.
- Continue until ACKs do not arrive or flow control dominates.

# Slow Start



# User Datagram Protocol (UDP)



- Simplest transport-layer protocol
- Just exposes IP packet functionality to application level
- *Ports* identify sending/receiving process
  - Demultiplexing information
  - (port, host) pair identifies a network process

# UDP End-to-End Model

- Multiplexing/Demultiplexing with Port number

