CSc 450/550
Computer Networks
Flow Control

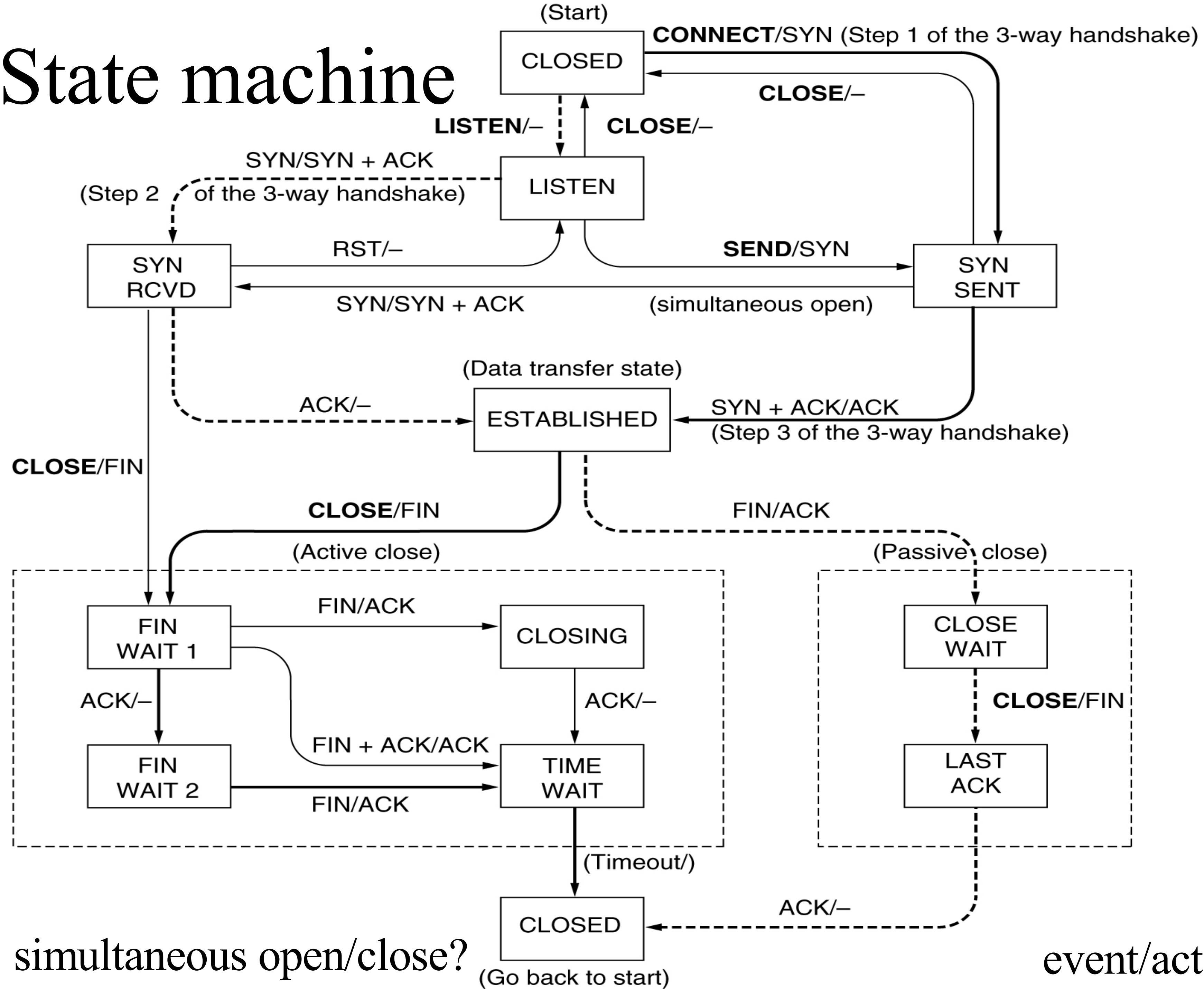
Jianping Pan

Summer 2007

Review: TCP basics

- Services provided by TCP
  - connection-oriented, reliable data transfer
- Services provided by IP
  - connectionless, unreliable packet delivery
- TCP protocol mechanisms: to fill the gap
  - last lecture: TCP connection management
    - connection establishment and release
  - flow, error and congestion control

State machine

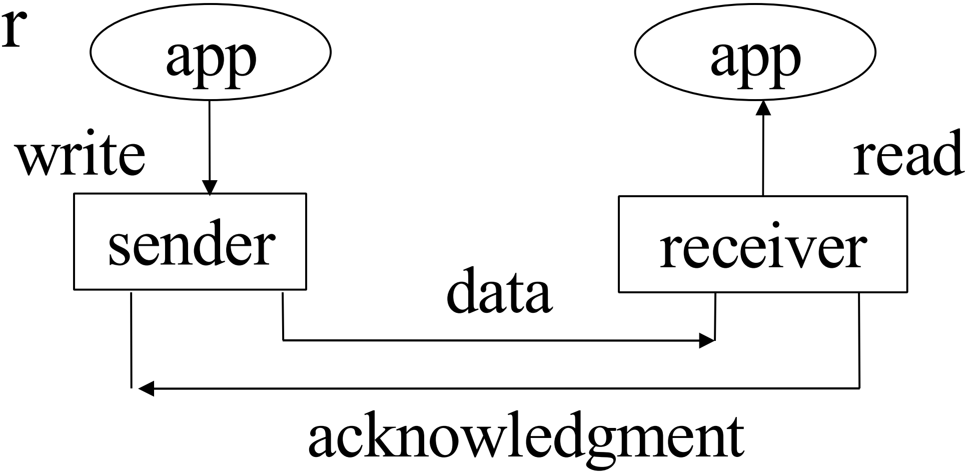


Q: simultaneous open/close?

event/action

Data transfer

- After connection establishment
- Data transfer: bidirectional in TCP
  - – reliable data transfer
    - **flow control**
    - error control
    - congestion control



- Before connection release

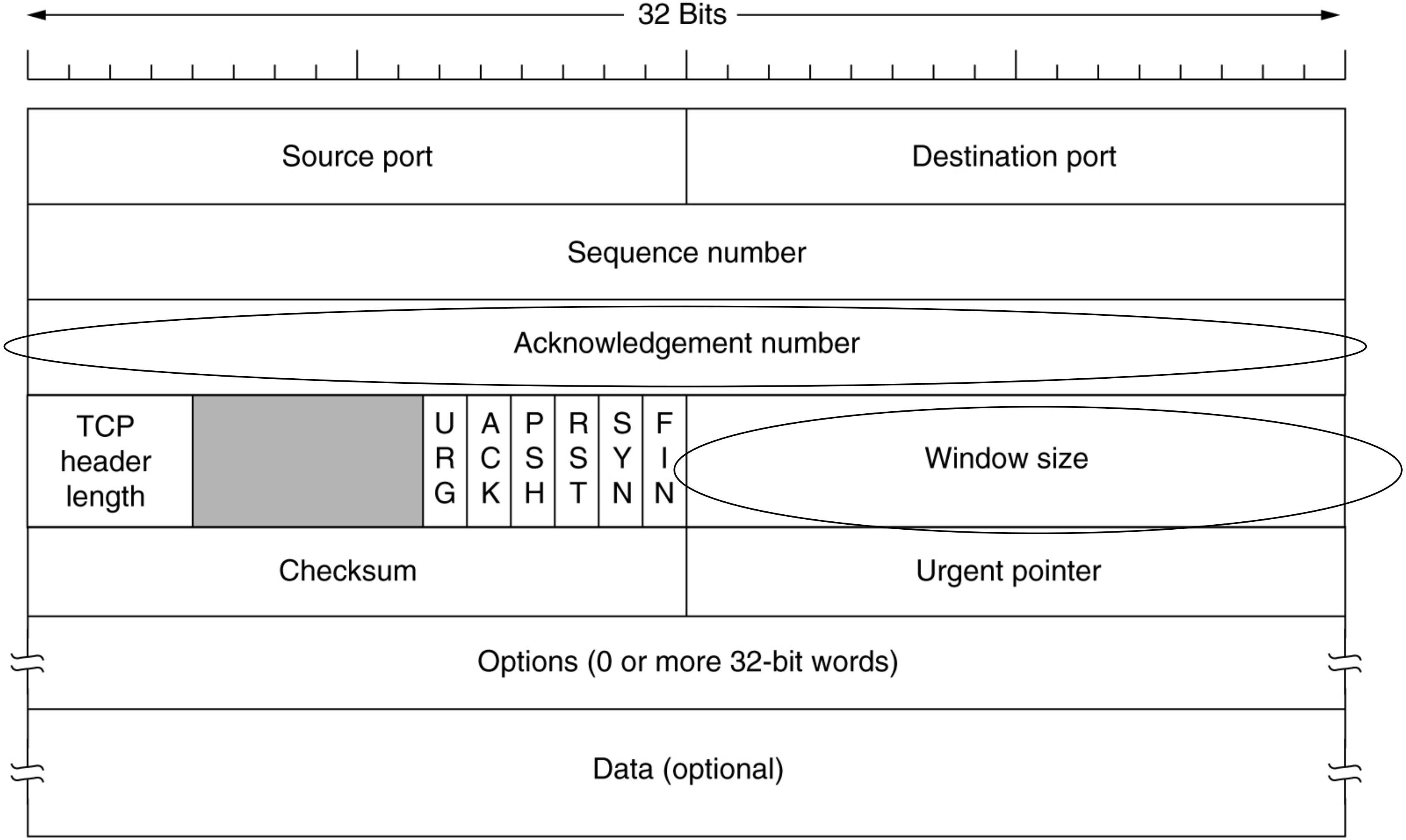
Q: why SYN, FIN also take seqno?

TCP flow control

- Problem
  - a fast sender to overflow a slow receiver
    - the receiver has no buffer to hold incoming packets
- Approach
  - let the receiver tell the sender how much to send
    - window-based: the available space at the receiver
    - or, rate-based: the sending rate allowed, e.g., ATM
  - TCP: receiver window size (16-bit)
    - advertised window size in bytes!

Q: byte vs packet sequence/window?

TCP packet header



32 Bits

Source port

Destination port

Sequence number

Acknowledgement number

TCP header length

URG

ACK

PSH

RST

SYN

FIN

Window size

Checksum

Urgent pointer

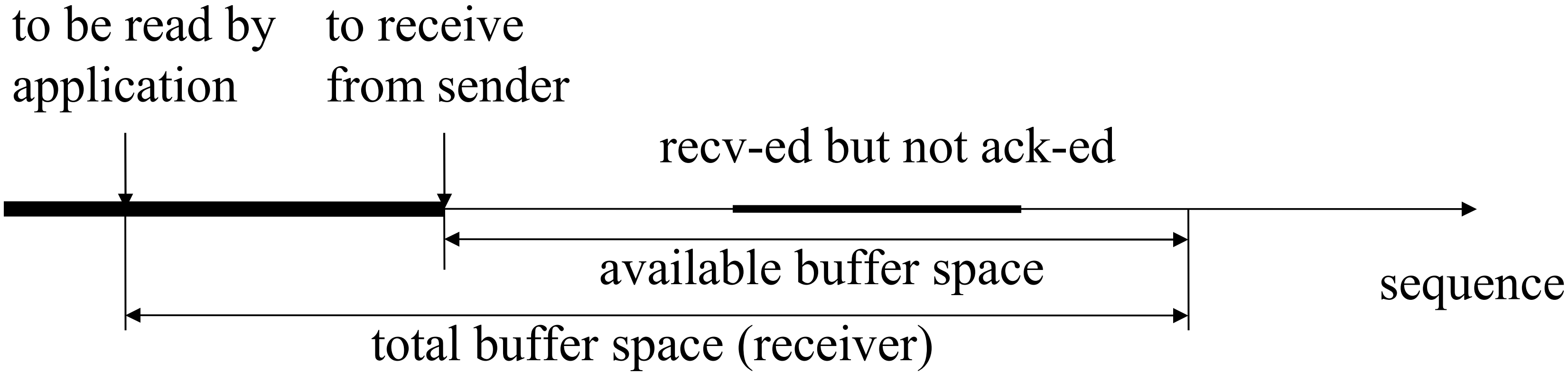
Options (0 or more 32-bit words)

Data (optional)

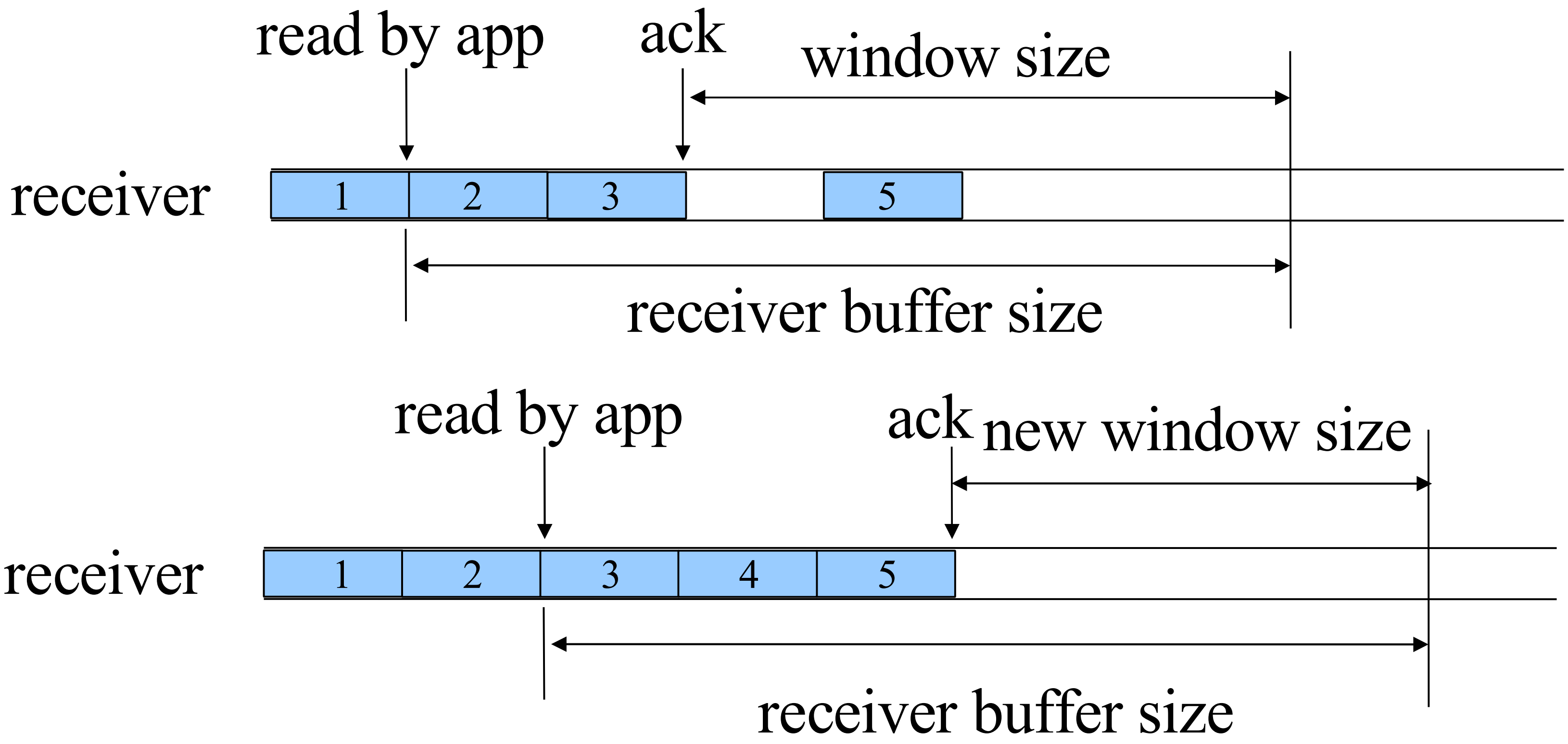
Q: which packet has no ack no?

TCP receiver's view

- Sequence space
  - acknowledgment number
    - the next ***continuous*** byte to receive from the sender
  - receiver window
    - available buffer space at receiver



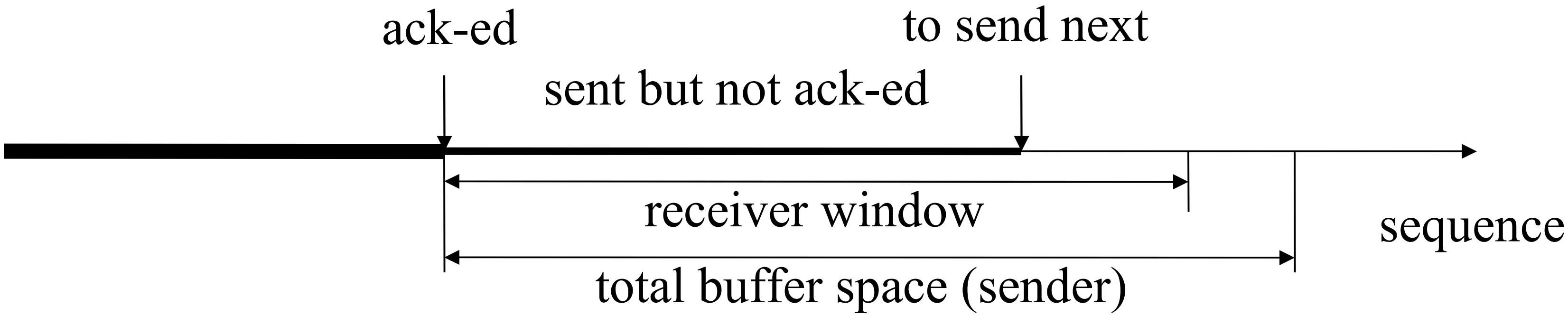
Receiver: sliding window



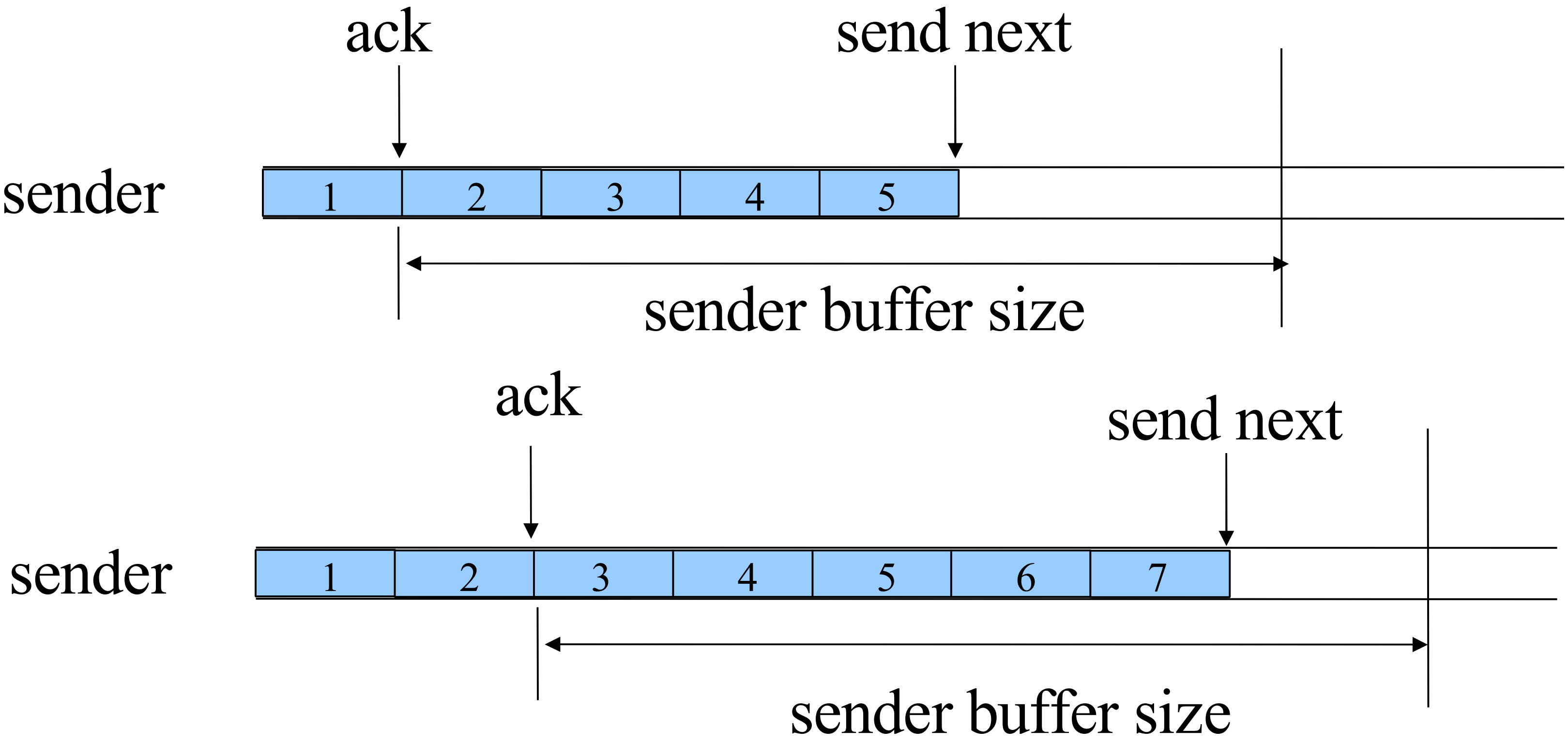
Q: events?

TCP sender's view

- Sequence space
  - sequence number
    - the first byte sequence in the payload
  - sender window
    - min {receiver window, total buffer space}



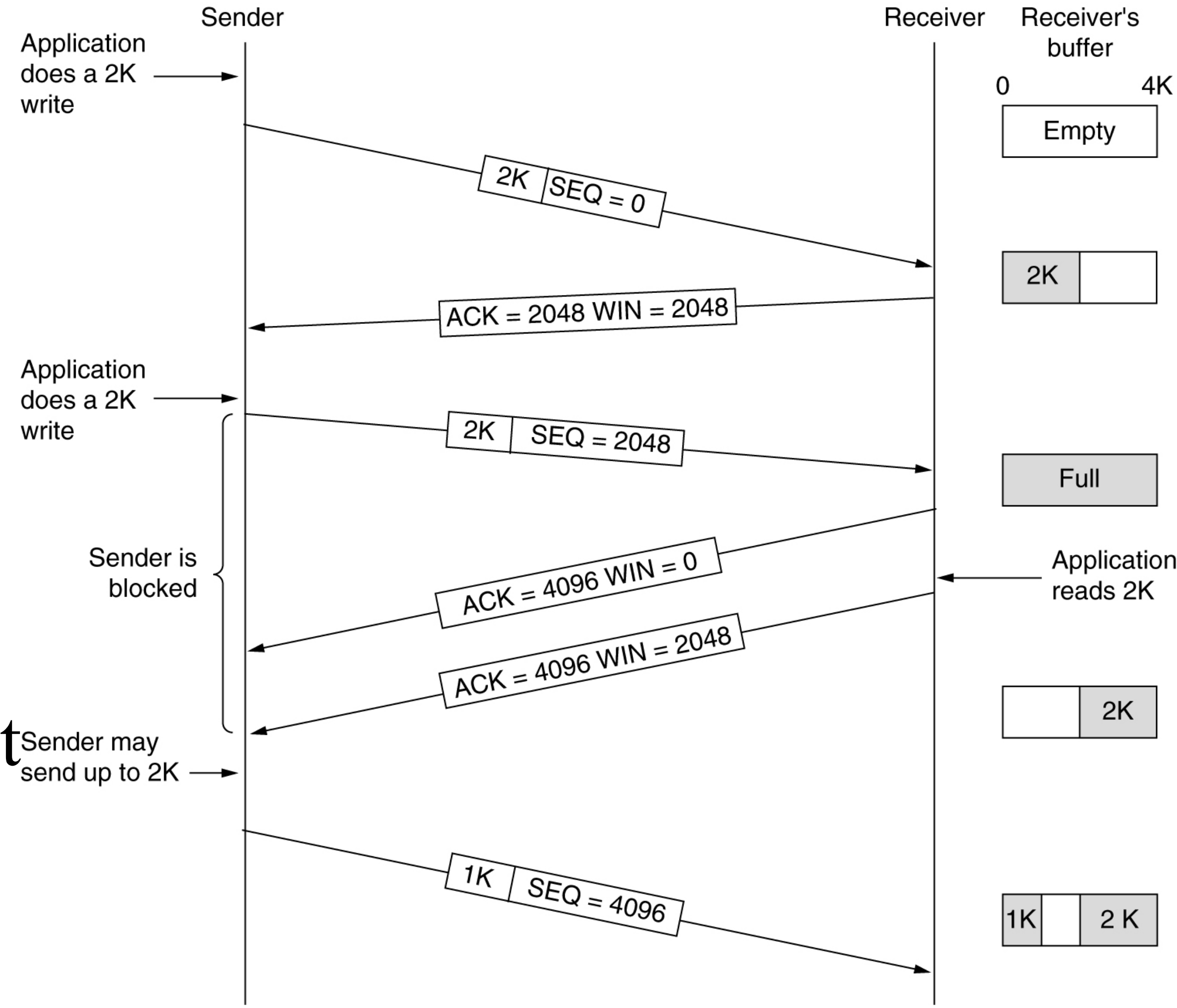
Sender: sliding window



Q: events?

Sliding window-based flow control

- Window control
  - sliding window
    - acknowledgment
  - variable window
    - window size
- When win=0
  - no data can be sent
  - exception
    - urgent data
    - window probes to avoid deadlock



Sender: small packet problem

- Problem
  - application keeps writing data byte-by-byte
  - TCP sends many small data packets
    - also trigger many acknowledgment packets
  - high overhead

Q: TCP header length?

- John Nagle's algorithm
  - send the first byte and wait for acknowledgment
    - or send when an MSS worth of data accumulated
  - send the rest bytes accumulated so far

Q: when Nagle's not preferred?

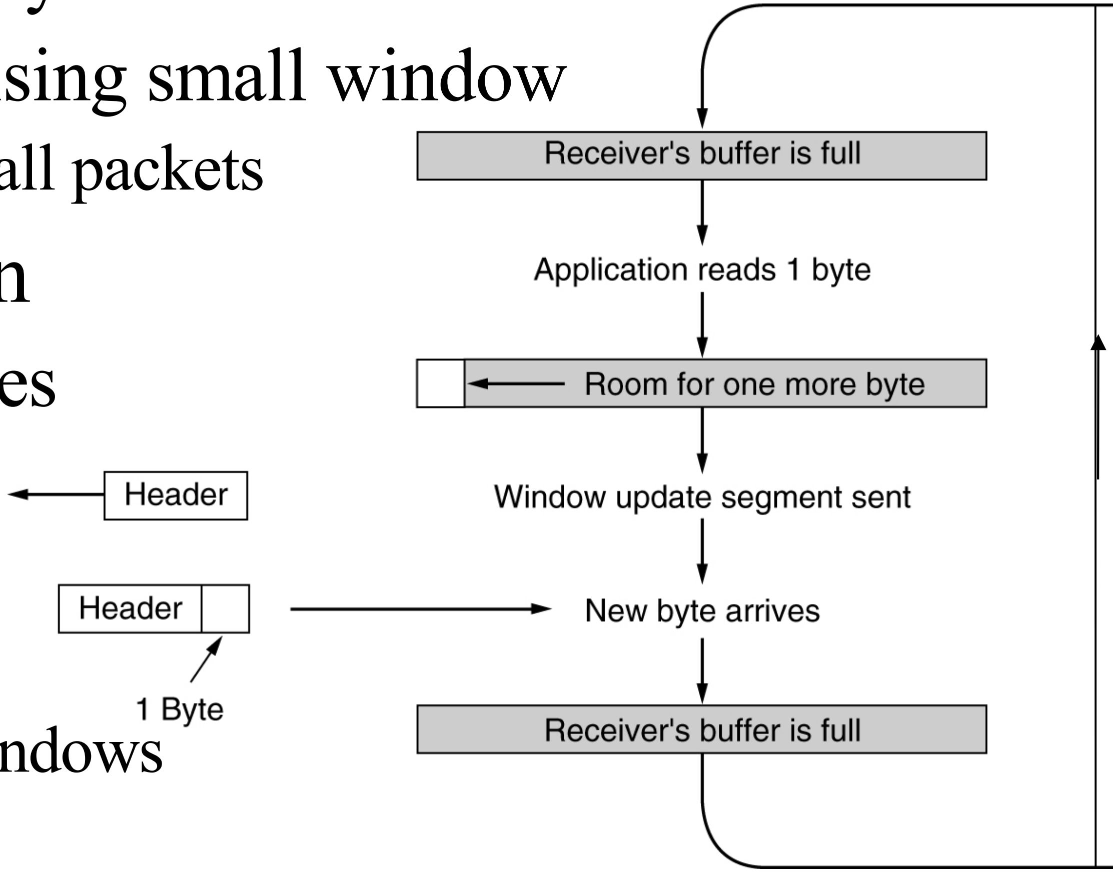
Nagle's algorithm

- Goal
  - try to send big packets
  - to lower packet header overhead
- When Nagle's algorithm is not beneficial
  - e.g., mouse movement in X-window
    - mouse pointer stalls and jumps due to delayed update
  - also, interaction with delayed acknowledgment
  - to disable Nagle's algorithm through socket API
    - setsockopt(..., ..., TCP_NODELAY, ..., ...);

Q: why delay acknowledgment?

Receiver: small packet problem

- Problem
  - silly window syndrome: application keeps reading data byte-by-byte
  - receiver keeps advertising small window
    - sender has to send small packets
- David Clark's solution
  - receiver only advertises
    - at least one MSS, or
    - half window size
  - goal
    - try to advertise big windows

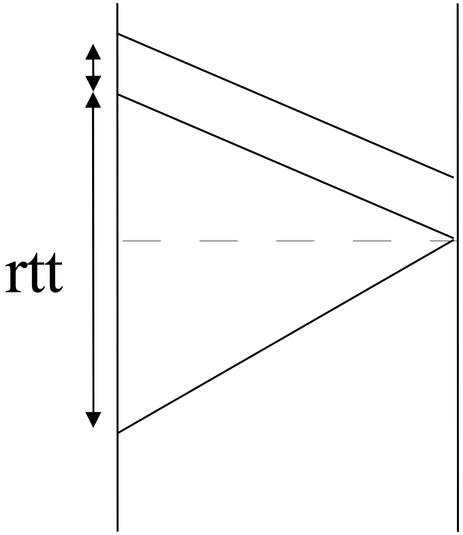


Between sender and receiver

- Sending small packets are bad
  - application always gives small write/read
- Sender's approach: Nagle's algorithm
  - try to wait until a big packet can be sent
- Receiver's approach: Clark's solution
  - try to wait until a big window can be advertised
  - delayed acknowledgment
    - piggyback acknowledgment packets with data packets
- Trade-off: extra delay

TCP window space

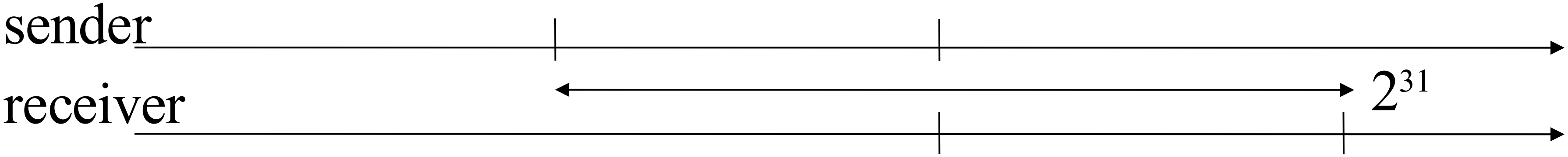
- Window space (16-bit)
  - maximum window size 2^{16}-1: ~64K bytes!
  - achievable throughput limit: ~ win/rtt
  - keep the "pipe" full
- TCP over "long-fat" networks (LFN)
  - long: large round-trip time
  - fat: high bandwidth
  - low utilization due to window limit



elephant network

TCP large window

- Extension: TCP large window
  - TCP window scale option
  - left shift up to 14 bit
    - i.e., maximum window size 2^{30}-1: 1GB
- TCP sequence number space (32-bit)
  - new data: within 2^{31} from left window edge
  - 2 * maximum window size <= 2^{31}



Q: why not shift more?

This lecture

- TCP flow control
  - purpose
  - mechanism
    - sliding variable window: seqno, ackno, win
- Explore further
  - TCP large window, PAWS with timestamp
    - RFC1323: TCP extensions for high performance
  - in tcpdump (or Ethereal)
    - time sip:spt > dip:dpt: P **144**:192 (48) **ack** 321 **win** 16022

Next lectures

- TCP error control
- TCP congestion control