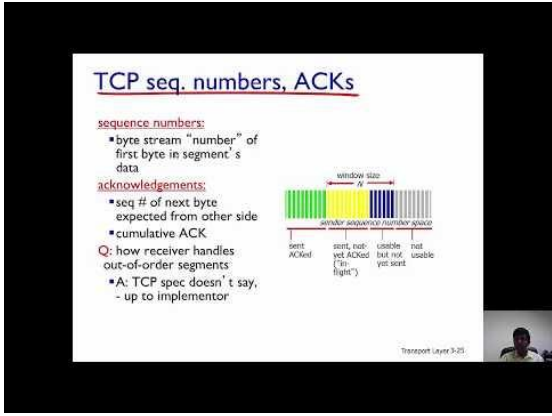
* alumni: “I really liked your courses, especially now at workplace. … A lot of details I cannot remember, but those examples live with me forever. ...”; me: “glad to know. examples are actually more important---they inspire the details. ...”

Computer Networks

TCP Flow Control

Jianping Pan
Fall 2022



* single-user washroom vs csc360?

Our course reps

- Thanks to our diverse student volunteers!
  - B01: Addie Finke <addiefinke>
  - B02: Preet Toor <preett23>
  - B03: Victor Kamel <vkamel>
  - B04: Nubia De La Torre <nubiadelatorre>
  - B05: Samantha Carter <abound>
  - B06: Serena Wollersheim <serenawollersheim>

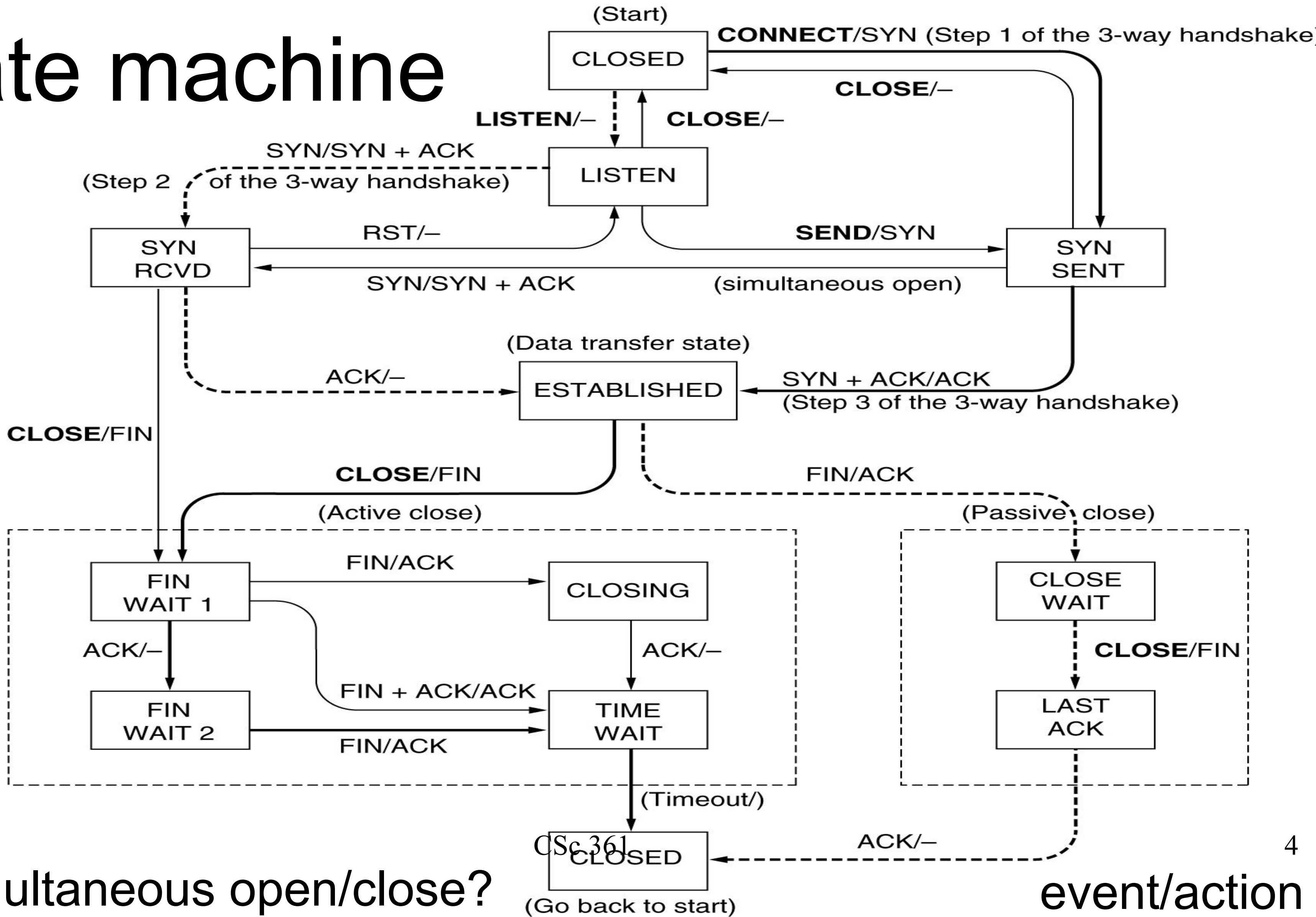
- AAA: Aggregate, Amplify and Anonymize
  - we will e-meet them next Monday
  - we do welcome student feedback directly too

* the teaching team is NOT on discord; please use brightspace for Q&A and online participation; how about uvic teams?

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

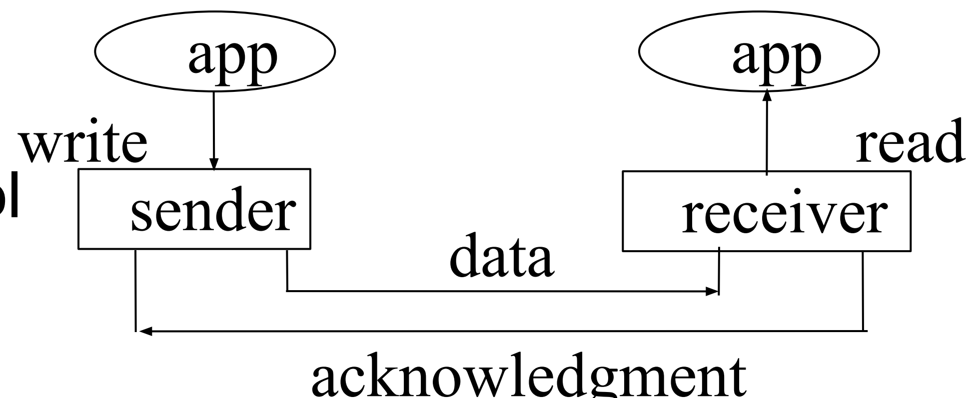


Q: simultaneous open/close?

event/action

Data transfer

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



- Before connection release

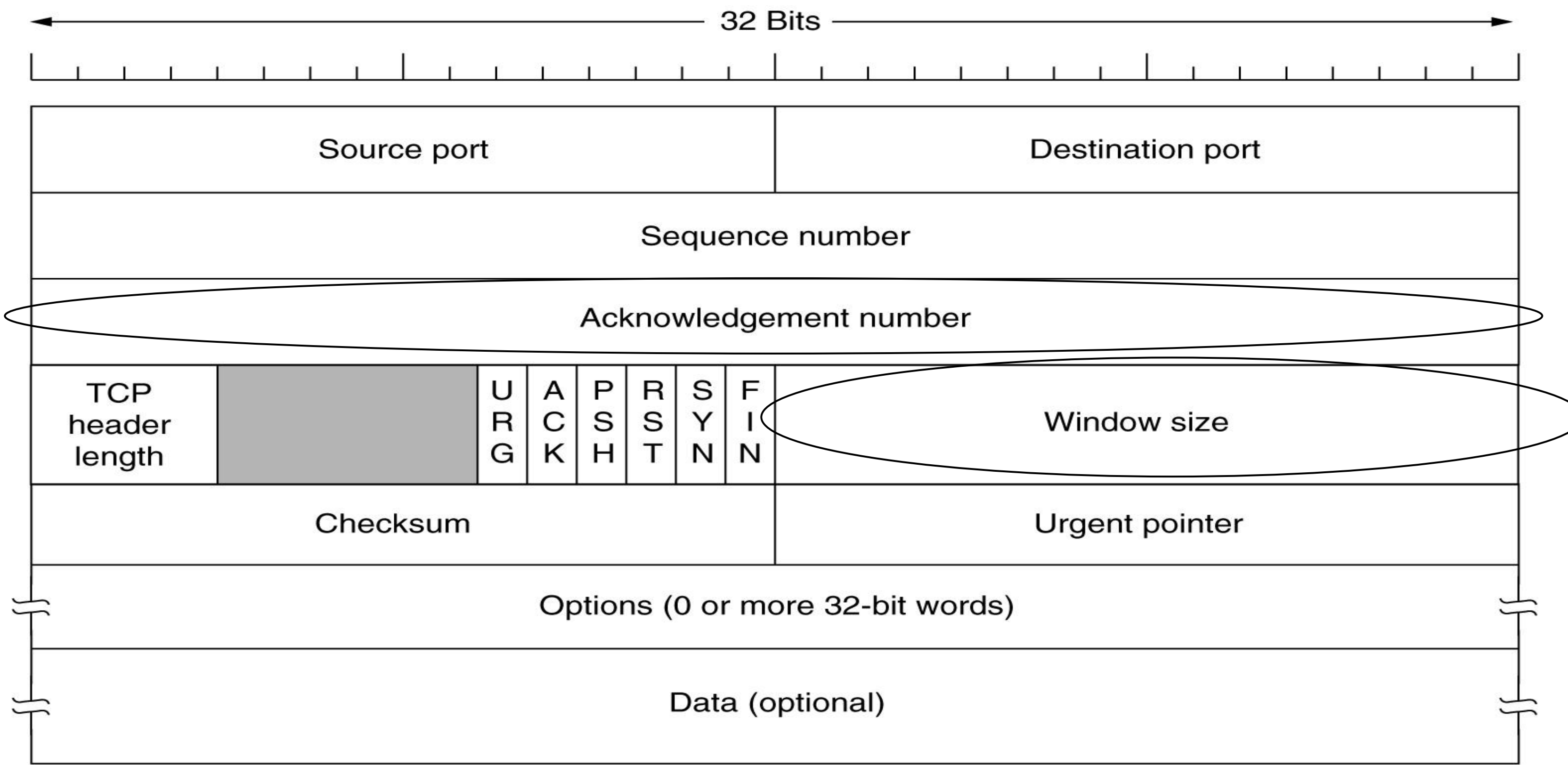
Q: why SYN, FIN also take seqno?

TCP flow control

- Problem (why to do it?)
  - a fast sender to overflow a slow receiver
    - the receiver has no buffer to hold incoming packets
- Approach (how to do it?)
  - 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! (what to do?)

Q: byte vs packet sequence/window?

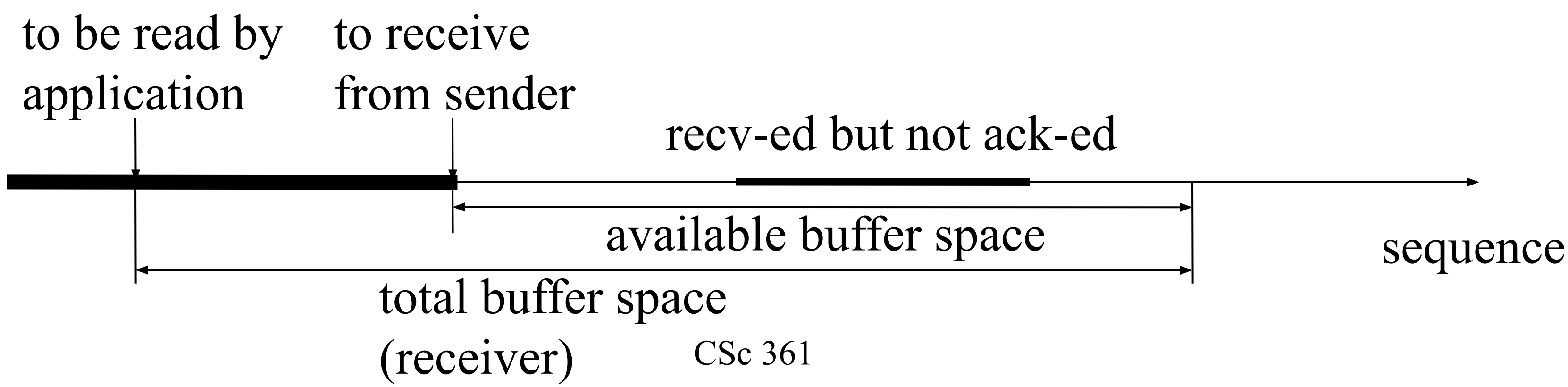
TCP packet header



Q: which packet has no ack no?

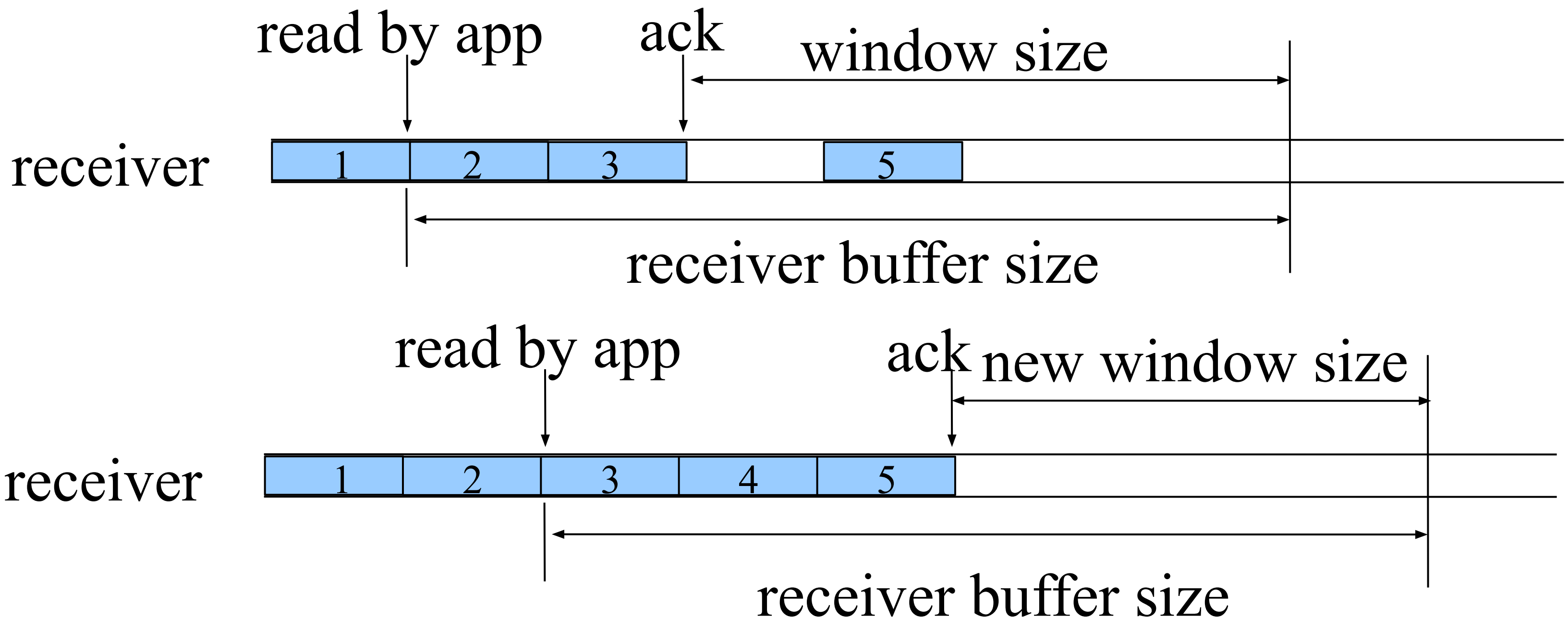
TCP receiver’s view

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



* “iperf -s **-w 1448** -l 724”?

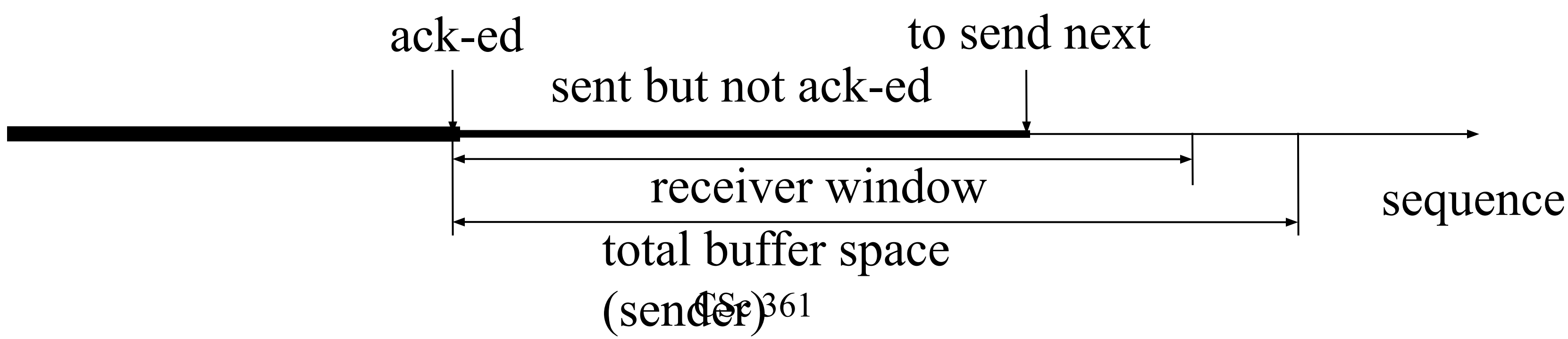
Receiver: sliding window



Q: events?

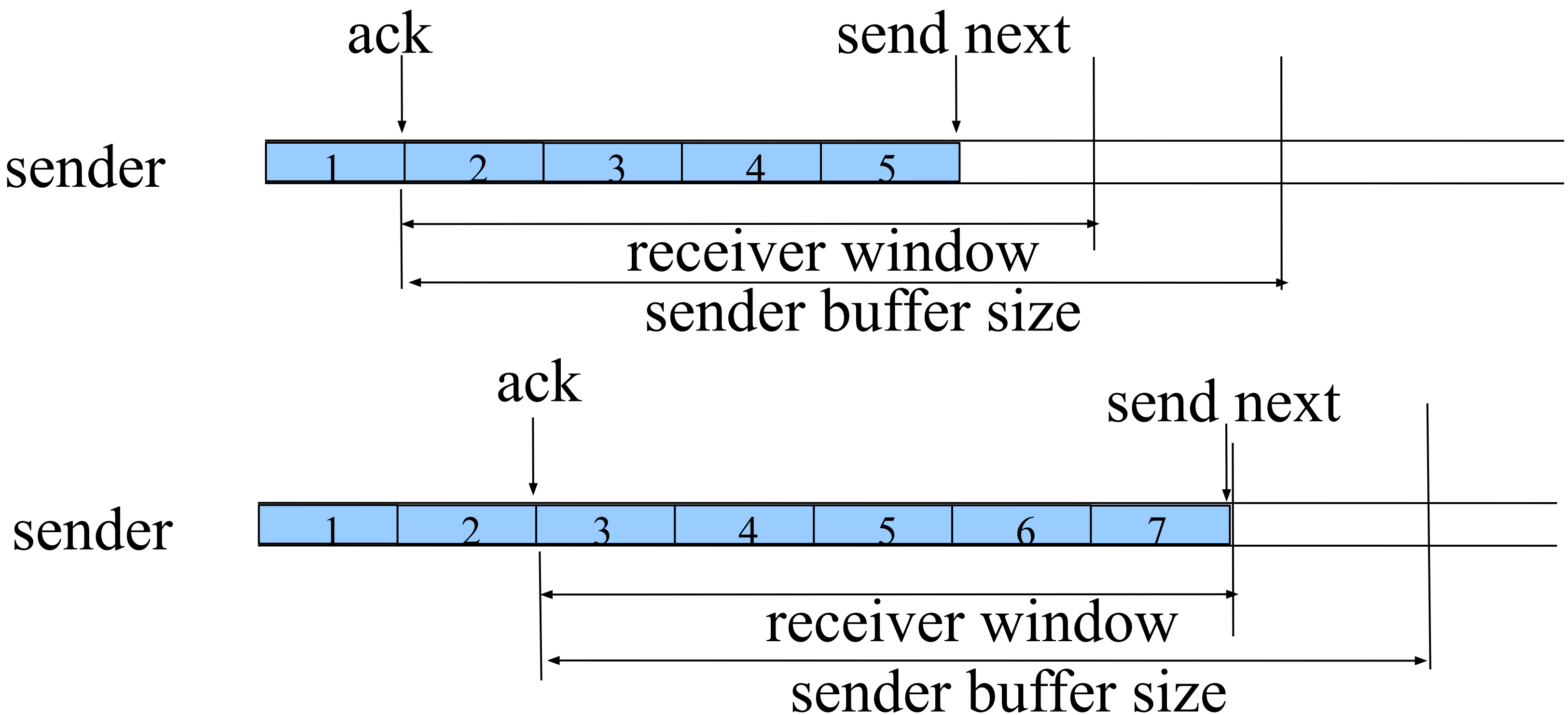
TCP sender's view

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



* “iperf -c h2 **-l 724** -n 5792”?

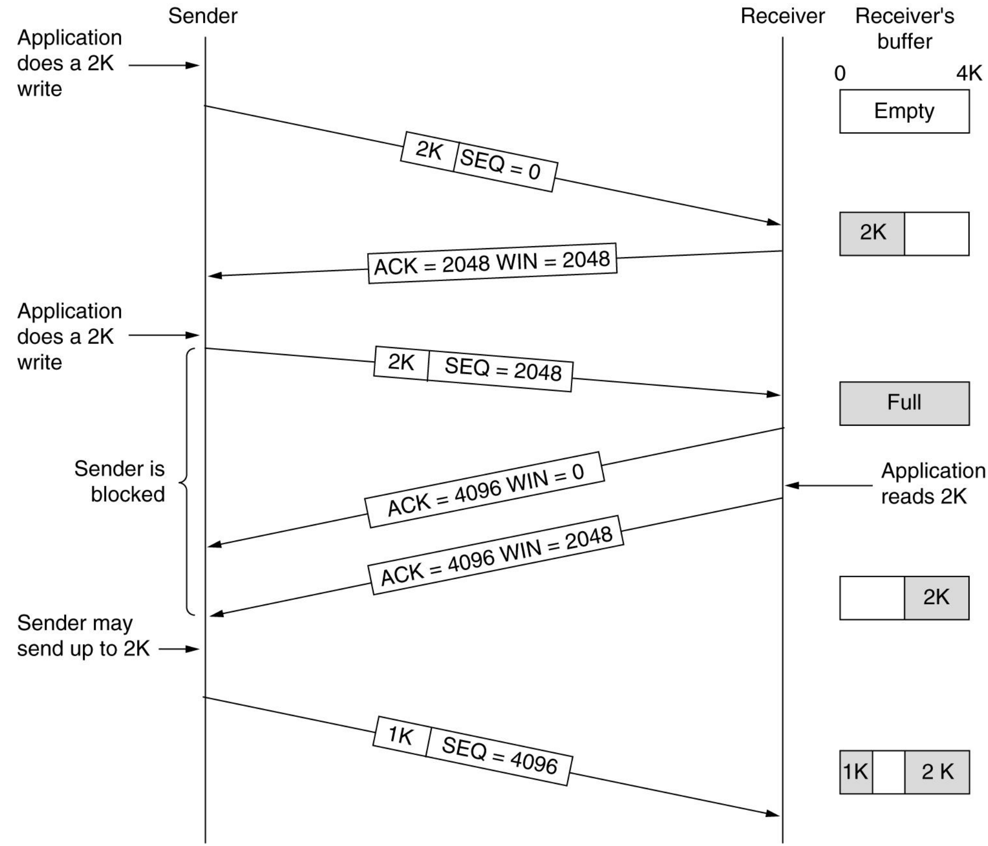
Sender: sliding window



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



Catch up: Urgent pointer

- TCP urgent pointer (16-bit)
  - offset of the LAST byte for urgent data
    - not (LAST+1) per RFC 1122: Host requirements
  - from the current sequence number!
  - for out-of-band (OOB) control information*
    - e.g., interrupt an ongoing file transfer
  - Socket interface
    - send(s, buf, len, MSG_OOB);
  - receiver should process the urgent data immediately

* can be overridden by MSG_OOBINLINE

Q: URG vs PSH?

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
- 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: TCP header length?

* type-in **nc** vs input redirect?

Q: when Nagle's not preferred?

Nagle's algorithm

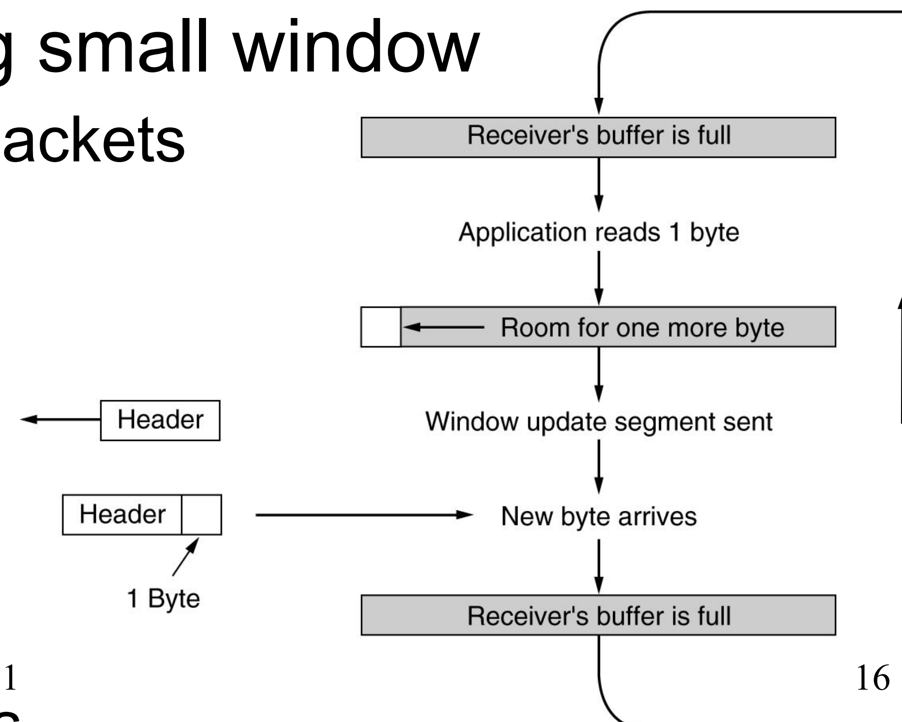
- Goal
  - try to send big packets
  - to lower packet header overhead
- When Nagle's algorithm is not beneficial
  - e.g., mouse movement in X-window
    - mouse pointer stalls and jumps due to delayed update
  - also, interaction with delayed acknowledgment
  - to disable Nagle's algorithm through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

* "nc **-i 1**" or set nodelay socket or systemwise?

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

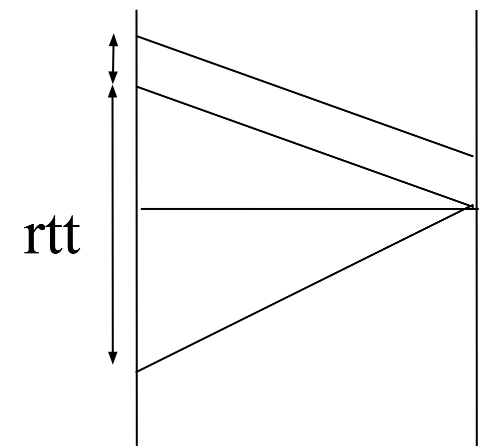


Between sender and receiver

- Sending small packets is 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

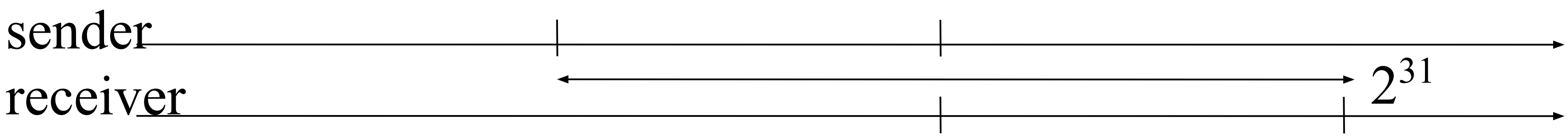
- Window space (16-bit)
  - maximum window size 2^{16}-1: ~64K bytes!
  - achievable throughput limit: ~ win/rtt
  - how to keep the "pipe" full?
- TCP over "long-fat" networks (LFN)
  - long: large round-trip time
  - fat: high bandwidth
  - low utilization due to window limit



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}



* how did we squeeze the space for TCP header length?

Q: why not shift more?

More TCP options

- TLV-like options
  - option-kind: 1-byte
  - option-length: 1-byte, for the entire option
  - option-data:variable length
- E.g., Maximum Segment Size (MSS)
  - exchanged during connection establishment
  - default: 536 bytes
- E.g., Selective Acknowledgment (SACK)
  - stay tuned: “TCP congestion control”
- Zero-padding to keep 32-bit alignment

Q: big vs small segment?

This lecture

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

* how's your RDP design for connection management and flow control? attend Friday's tutorial 1:30pm victoria time too!

Next lecture

- TCP error control (why-how-what: wow!)
  - read KR4: Computer Networking
    - Chapter 3, all sections required this month