The Transport Layer: Reliability, Ordering, and Flow Control

CS 352, Lecture 8, Spring 2020

http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Course announcements

- Mid-term 1 Friday in class
  - Closed book, calculators OK, no cell phones
- Review has been released under Sakai resources section
  - Separate PDFs with questions and answers
  - Review worked in recitation
  - Mid-term may contain a few more questions, more challenging questions
- Learning Assistant (LA) program: looking for 352 for next year

Review of concepts

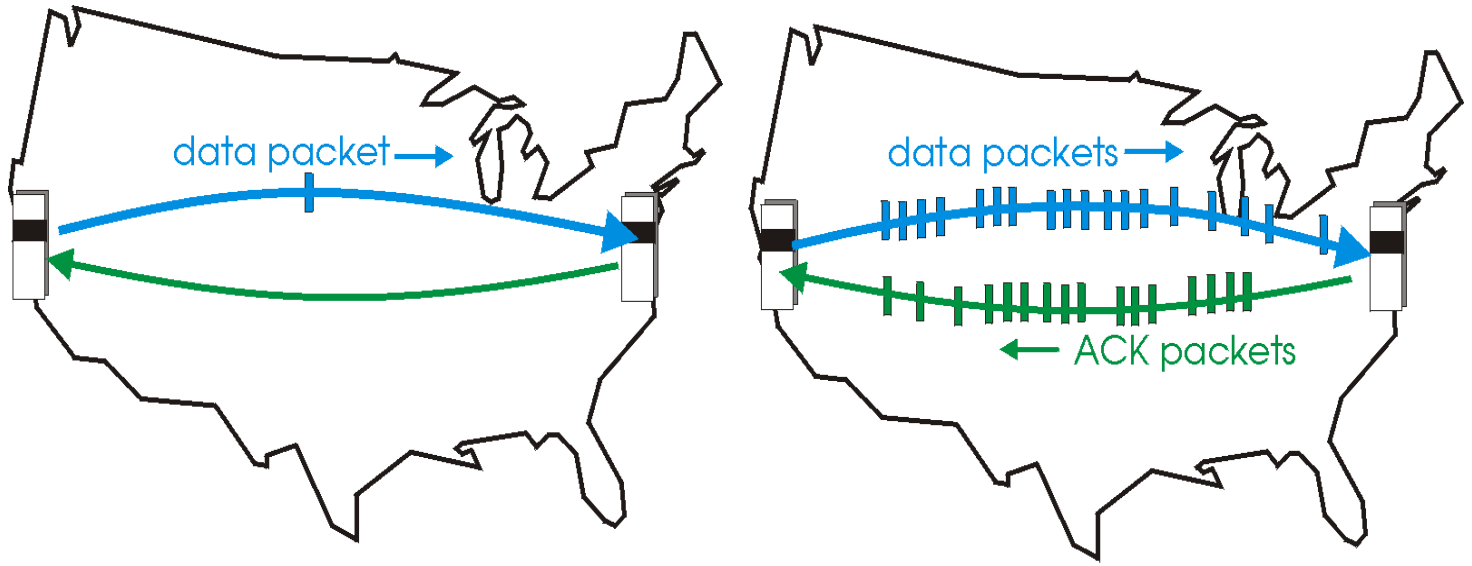
- Transport layer: User Datagram Protocol
  - Error detection using checksum
- Reliable data delivery: stop and wait
  - ACKnowledgments (ACKs)
  - Retransmission TimeOut (RTO)
  - Sequence numbers
  - Main problem: low data rate/throughput
- Reliable data delivery with pipelined data transmission
  - Key idea: increase the number of in-flight packets
  - Why does throughput increase?

Pipelined Reliability

TCP is a pipelined transmission protocol

Sender allows multiple, “in-flight”, yet-to-be-acknowledged packets

A few packets on the way while, concurrently, new packets are transmitted

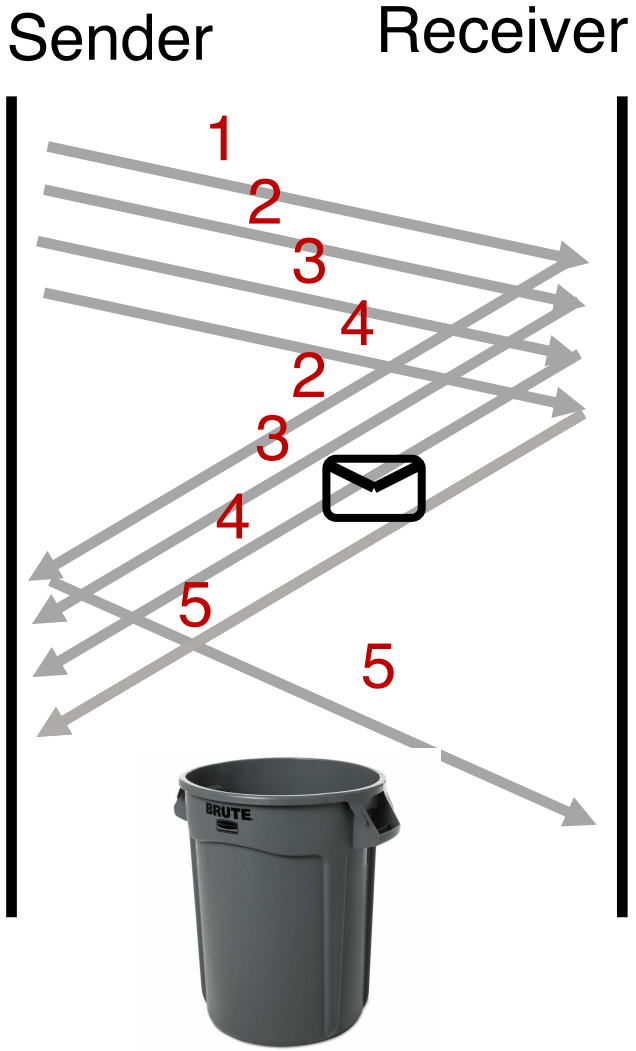


(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

What if some packets/ACKs dropped?

- Sequence numbers help associate an ACK with its packet
  - Note: In TCP, every byte has a sequence #
  - We will often simplify our examples by assuming each packet has a sequence #
- In TCP, the ACK contains the sequence number of the next byte expected
  - Note: example uses packet seq #s
- Q1: If a packet is dropped, should the receiver ACK subsequent packets?
  - Q2: If so, with what sequence number?



Receiver strategies upon packet loss

ACK subsequent pkts?

No

Go-back-N

Yes

Selective Repeat

What seq # on ACK?

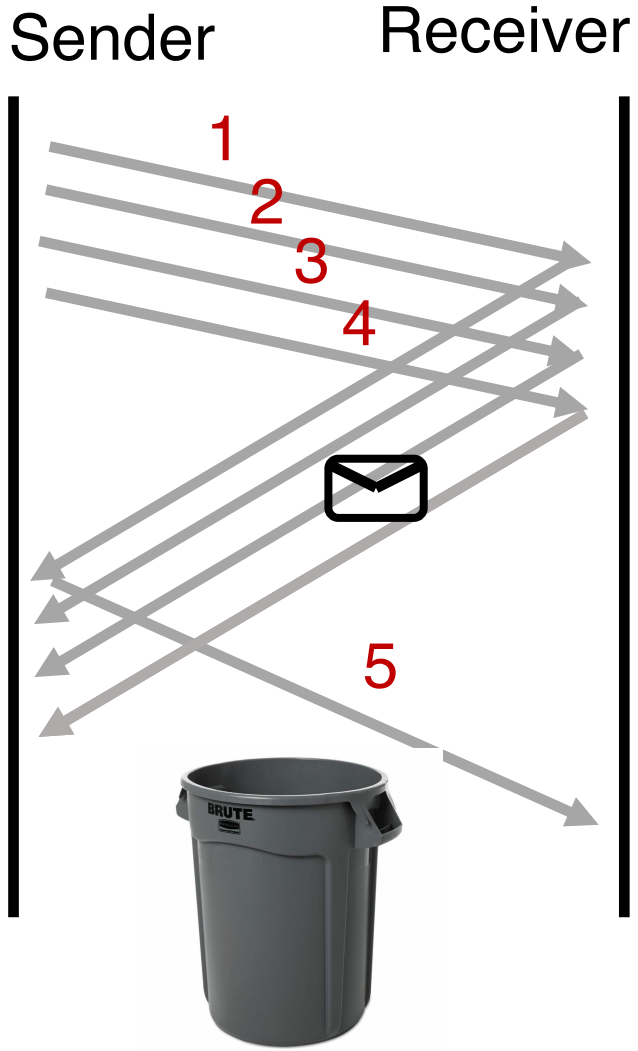
Last pkt in order

Cumulative ACK

TCP's default

Seq # ranges received so far

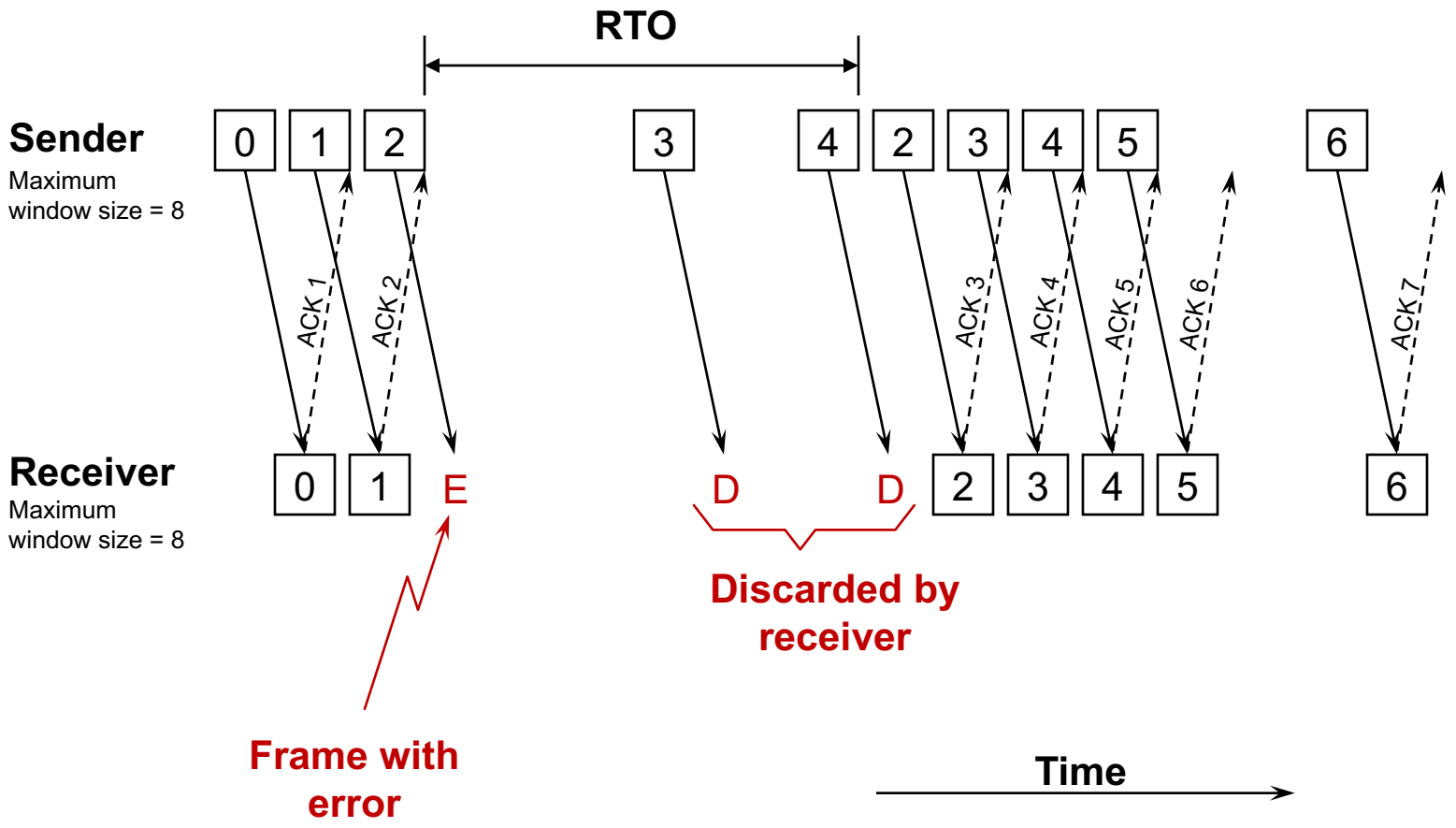
Selective ACK



Sliding Window with Go Back N

- When the receiver notices a missing or erroneous frame:
- It simply discards all frames with greater sequence numbers
  - The receiver will send no ACK
- The sender will eventually time out and retransmit all the frames in its sending window

Go back N


RTO
Sender
Maximum
window size = 8
Receiver
Maximum
window size = 8
0
1
2
3
4
2
3
4
5
6
ACK 1
ACK 2
ACK 3
ACK 4
ACK 5
ACK 6
ACK 7
0
1
E
D
D
2
3
4
5
6
Discarded by
receiver
Frame with
error
Time

Go back N

Go Back N can recover from erroneous or missing frames

But…

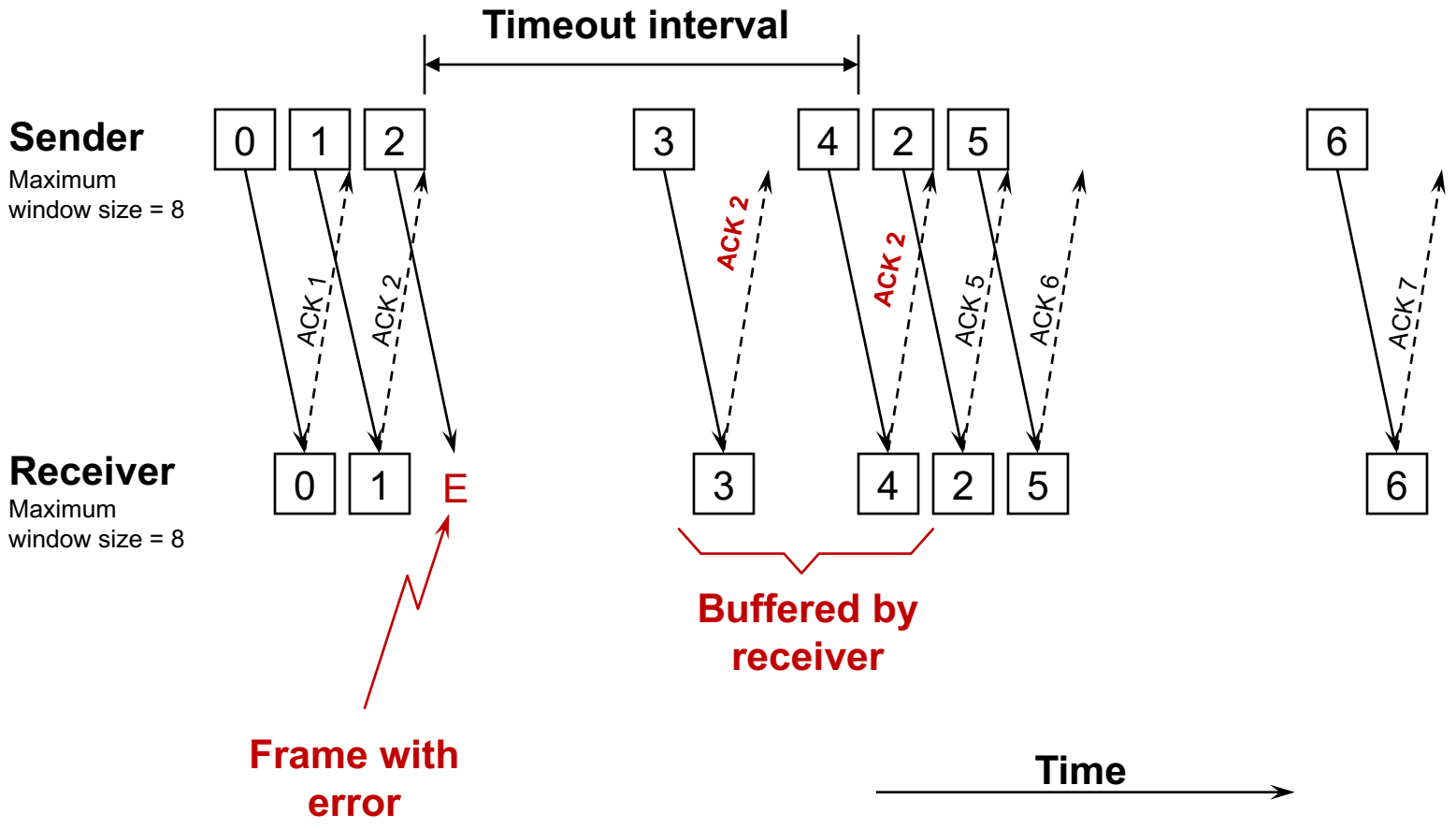
It is wasteful. If there are errors, the sender will spend time retransmitting frames the receiver has already seen

Selective repeat with cumulative ACK

Idea: sender should only retransmit dropped/corrupted segments.

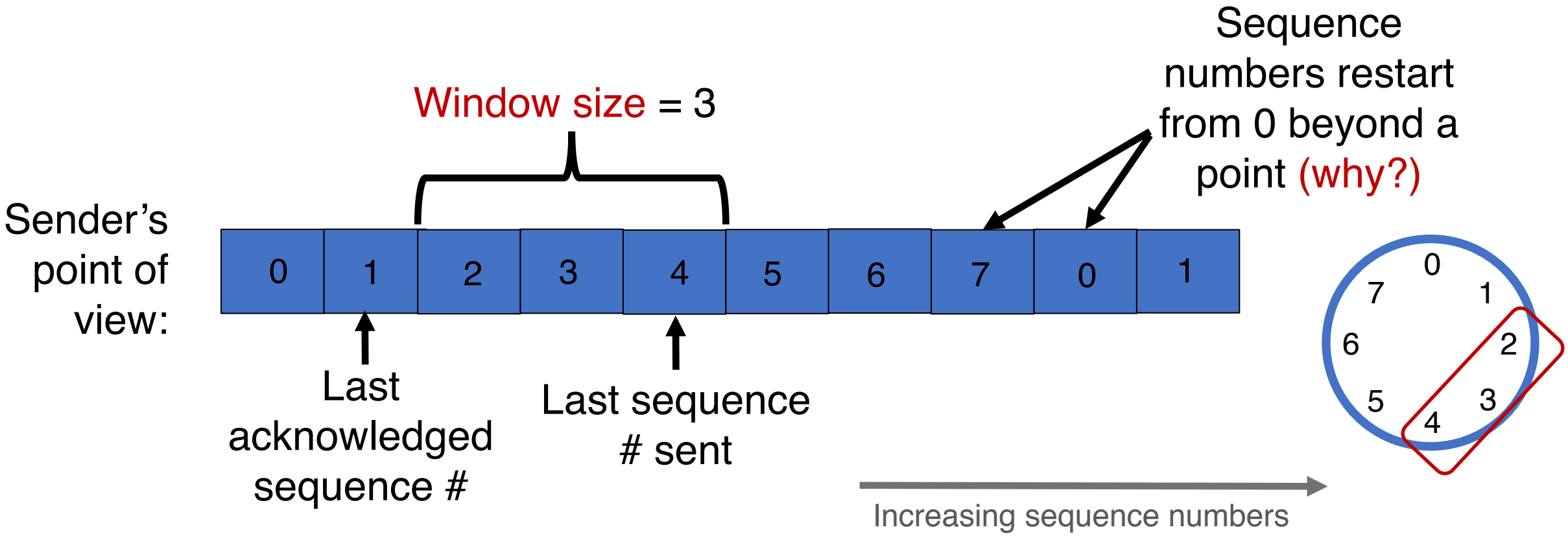
- The receiver stores all the correct frames that arrive following the bad one. (Note that the receiver requires a memory buffer for each sequence number in its receiver window.)
- When the receiver notices a skipped sequence number, it keeps acknowledging the last good sequence number,. i.e., cumulative ACK
- When the sender times out waiting for an acknowledgement, it just retransmits the one unacknowledged frame, not all its successors.
- Note that timeout applies independently for each sequence #

Selective repeat with cumulative ACK



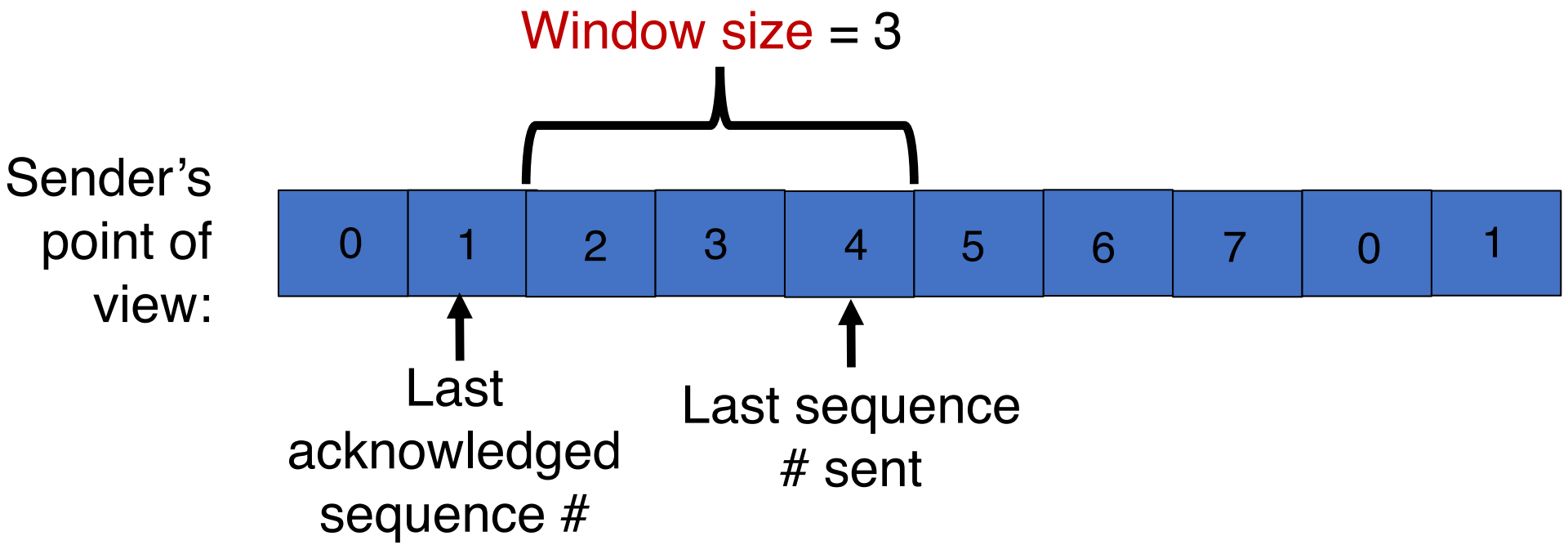
Window

- Window size: The amount of in-flight data (unACKed)
- Window: Sequence numbers of in-flight data



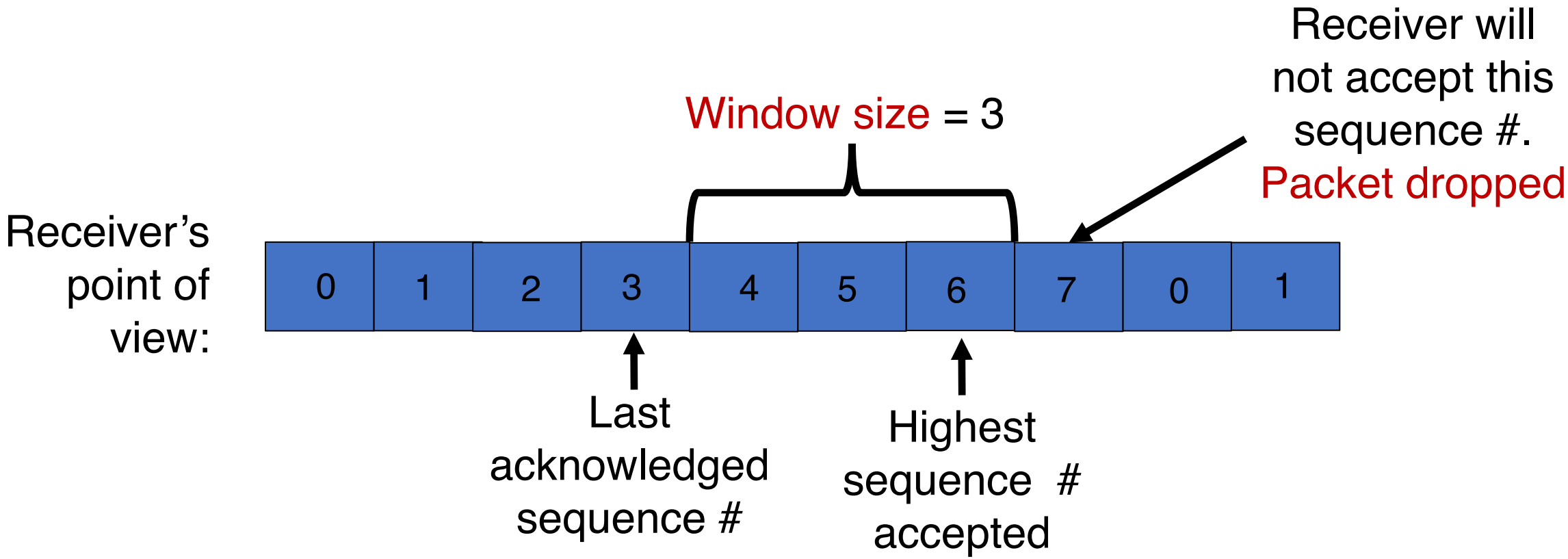
Sliding window

- Suppose sequence number 2 is acknowledged by the receiver
  - Sender can transmit sequence # 5
  - The window “slides” forward



Corresponding window on receiver side

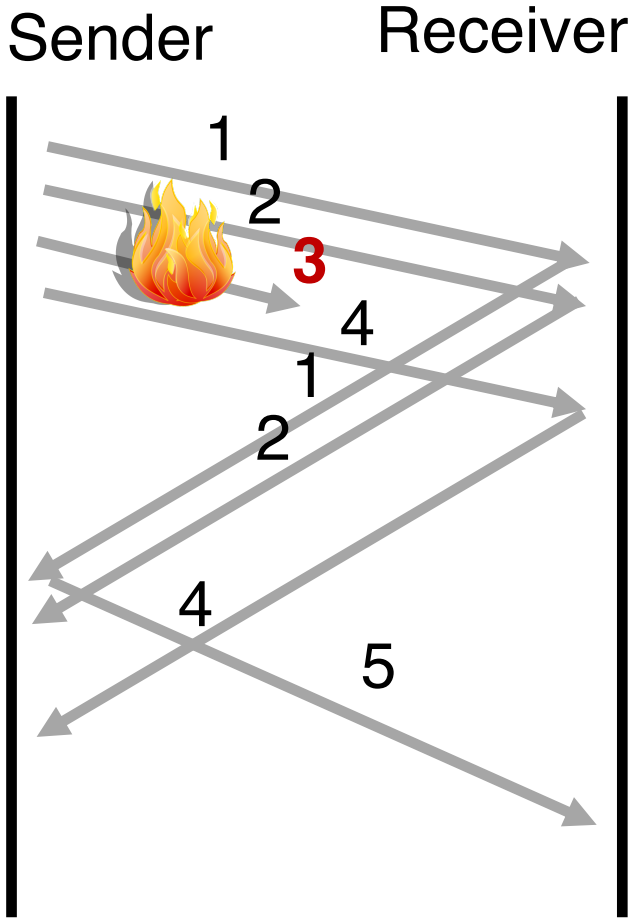
- Receiver only accepts sequence #s as allowed by the current receiver window



Ordered Delivery

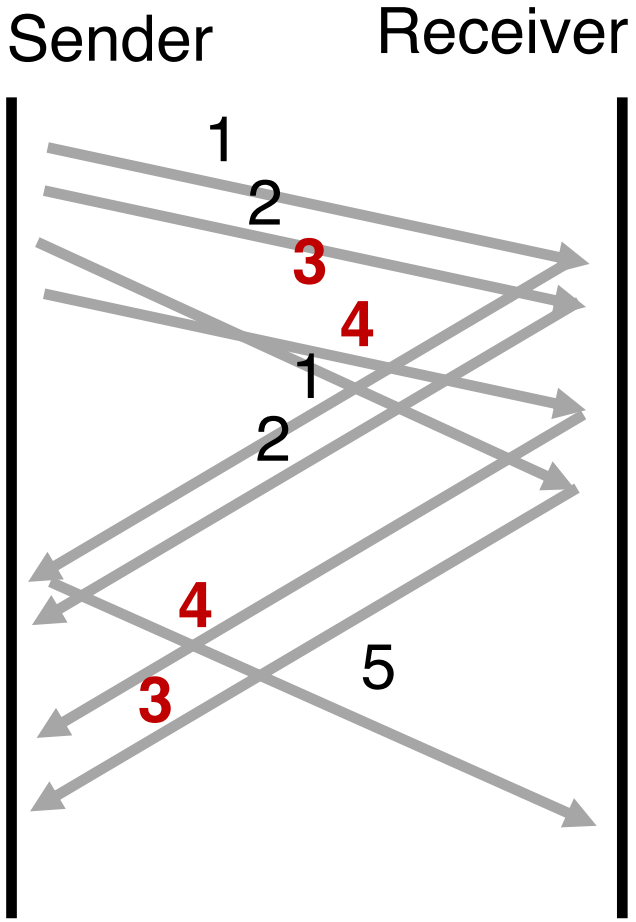
Reordering packets at the receiver side

- Let's suppose receiver gets packets 1, 2, and 4, but not 3 (dropped)
- Suppose you're trying to download a Word document containing a report
- What would happen if transport at the receiver directly presents packets 1, 2, and 4 to the Word application?



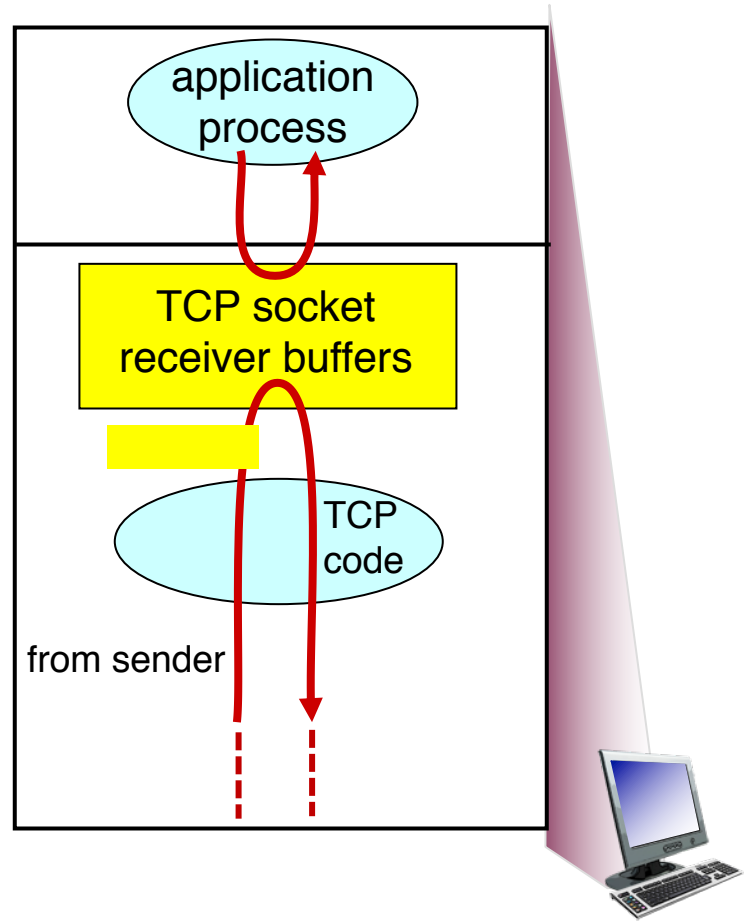
Reordering at the receiver side

- Reordering can also happen due to packets taking different paths through a network
- Receiver needs a general strategy to ensure that data is presented to the application in the same order that the sender side pushed it
- This is accomplished using a memory buffer at the receiver (also called receiver socket buffer)
  - We've already seen the use of memory buffer to have the sender avoid duplicate transmissions



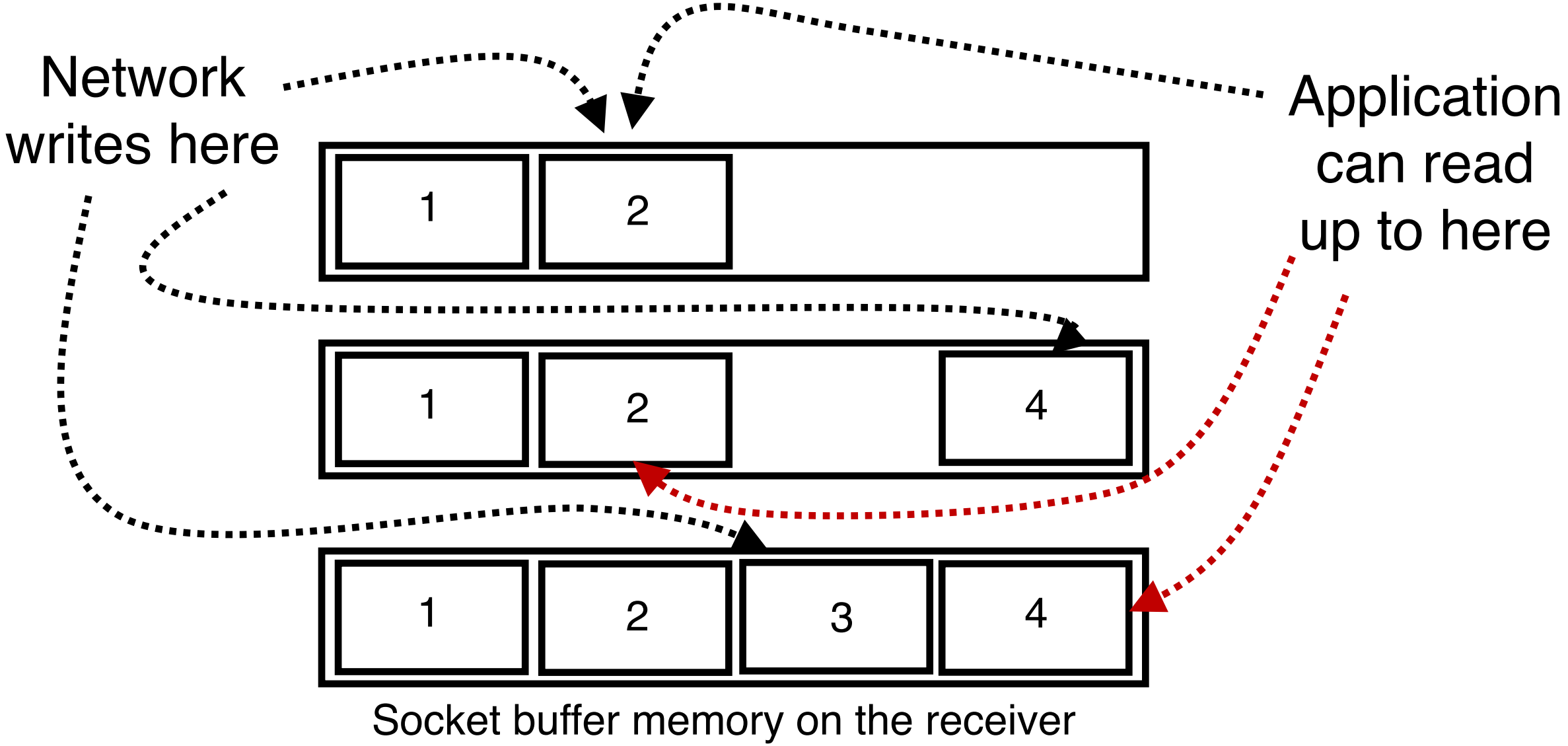
Interaction between apps and TCP

- An app with a TCP socket reads from the TCP receive socket buffer
  - e.g., when you do `data = sock.recv(10)`
- TCP receiver software only releases this data to the application if the data is in order relative to all other data already read by the application



receiver protocol stack

Ordering at the receiver side



Ordering at the receiver side

- The TCP software at the receiver uses socket buffers to hold packets until they can be read by an application in order

- This process is known as TCP reassembly

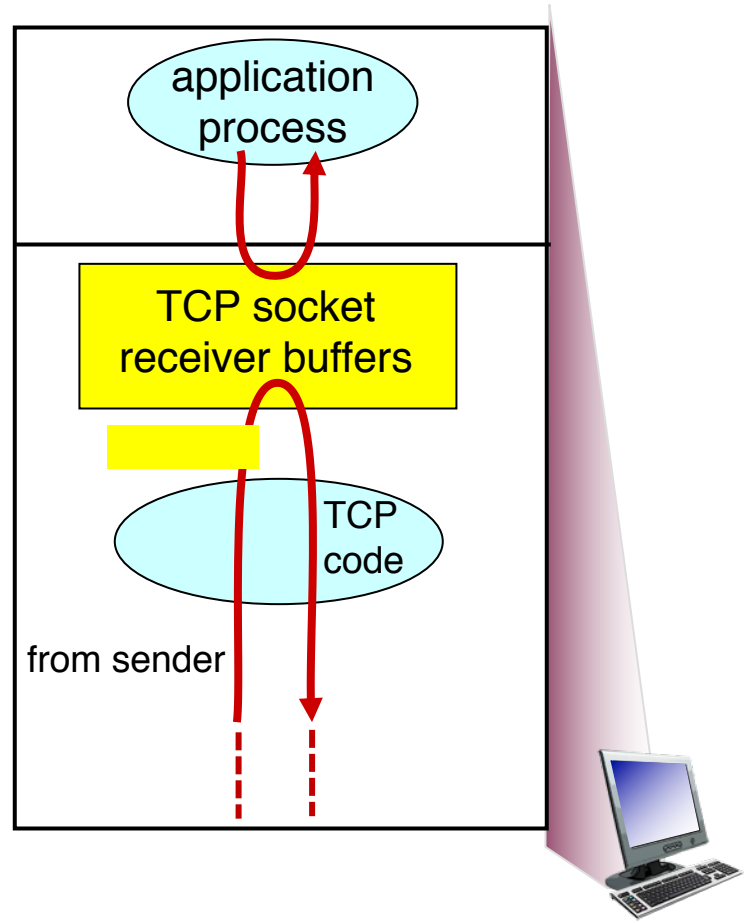
Implications of ordered delivery

- Packets cannot be delivered to the application if there is an in-order packet missing from the receiver's buffer
  - The receiver can only buffer so much out-of-order data
  - Subsequent out-of-order packets dropped
- It doesn't matter that the packets successfully arrive at the receiver from the sender over the network
- TCP application throughput will suffer if there is too much packet "reordering" in the network

Flow control

Implications of buffering at the receiver

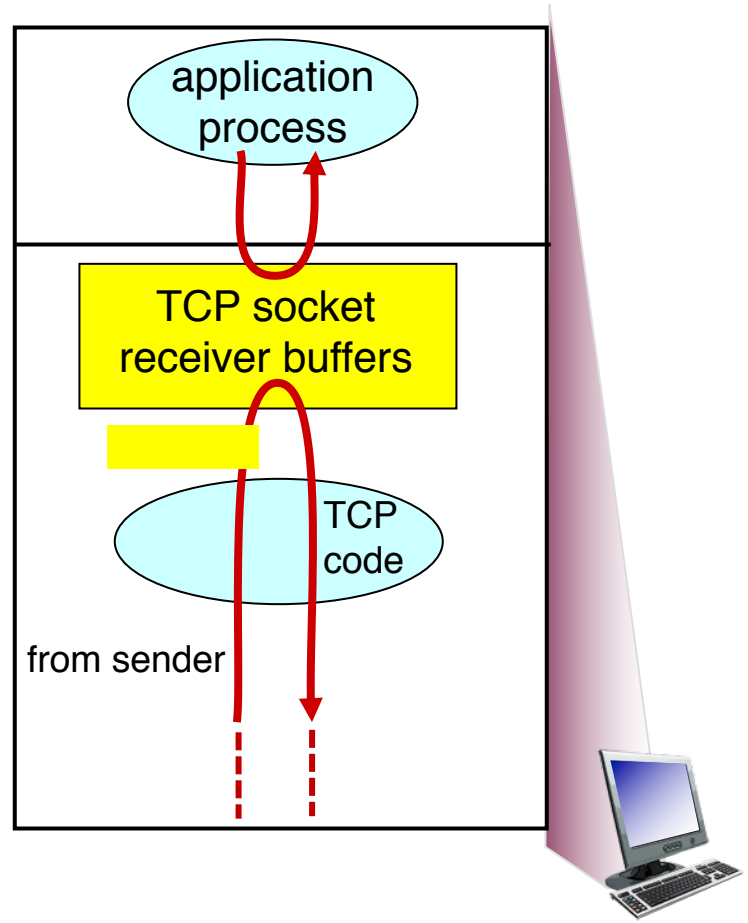
- Applications may read data slower than the sender is pushing data in
  - e.g., what if you never called `recv()`?
- Hence, the permissible window size may vary over time.



receiver protocol stack

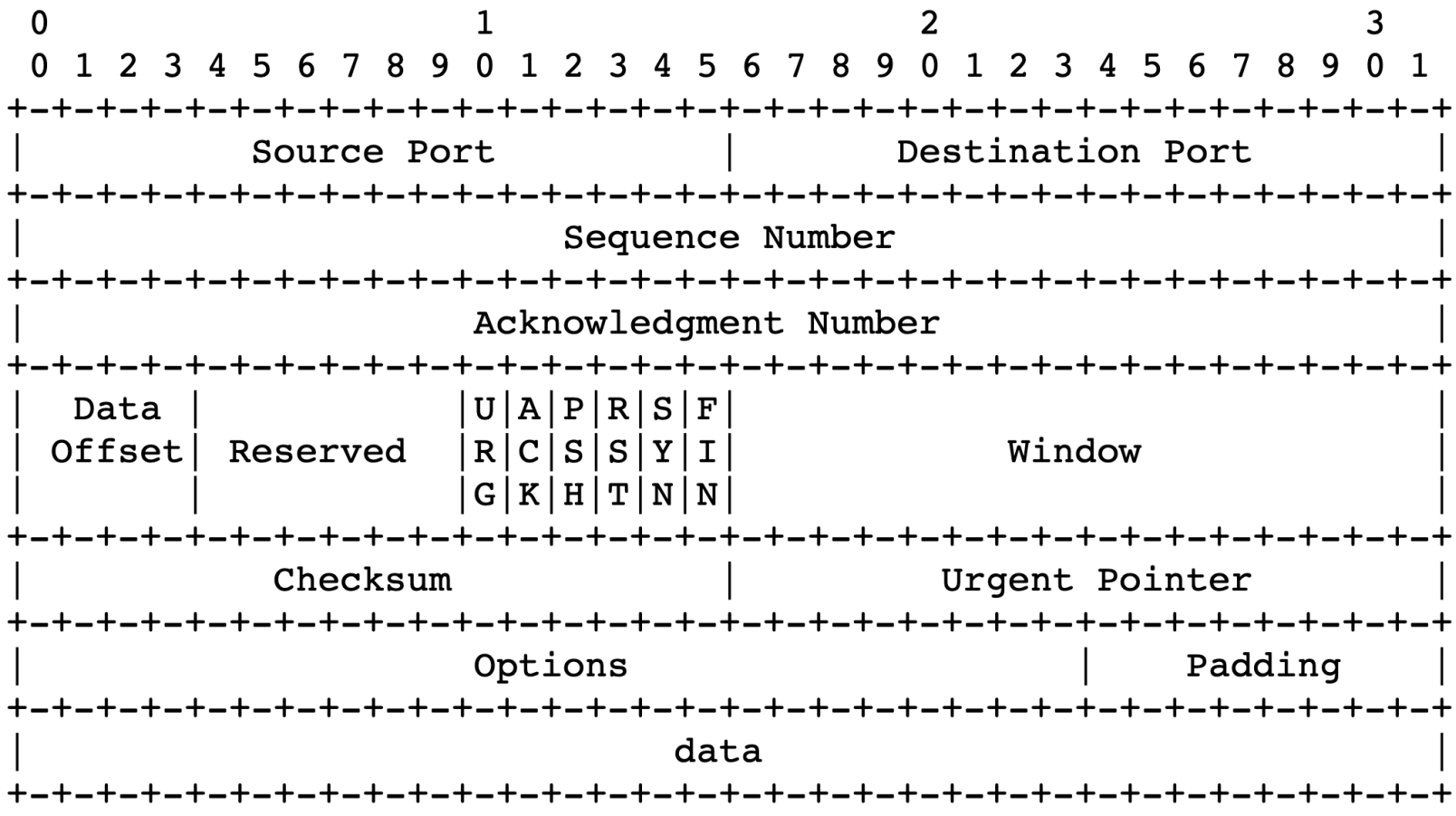
Implications of buffering at the receiver

- A TCP sender can only send as much as the free receiver buffer space available, before packets are dropped at the receiver
- This number is called the receiver window size or advertised window size
- TCP is said to implement flow control
- Sender's window size is bounded by the advertised window size.



receiver protocol stack

TCP headers



TCP Header Format

Note that one tick mark represents one bit position.

Sizing the receiver socket buffer

- For each socket, there is a default size for the memory allocated to the receiving socket buffer
  - Unimaginatively called the receiver socket buffer size
- If this number is too small, sender can't keep too many packets in flight ➔ lower throughput
  - If too large, too much memory consumed
- How big should the receiver socket buffer be?