The Transport Layer: Reliability

CS 352, Lecture 7, Spring 2020

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

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Course announcements

- Project 1 released. Need partners?
- Quiz 2 completed, no quiz 3
- Mid-term 1 next Wednesday
  - In class, 1 h 20 m
  - Covers lectures 1 through 8 (coming Friday)
  - Closed book
  - Only calculators allowed. No cell phones
- Pattern: multiple choice, reasoning, problem solving
  - A review will be released later this week

Review of concepts

- Mail access protocols: POP, IMAP, HTTP
- Transport-layer protocols: UDP, TCP
- Support communication between processes
  - *vs.* network layer: endpoints

- Transport guarantees: reliability, ordering, performance
  - You get two of these at most
- Demultiplexing: map packet to app-level socket
- User Datagram Protocol (UDP): thin wrapper around net layer
  - Connectionless, best-effort service, simple
  - Demultiplex using dst IP address, dst port
  - Who uses UDP?

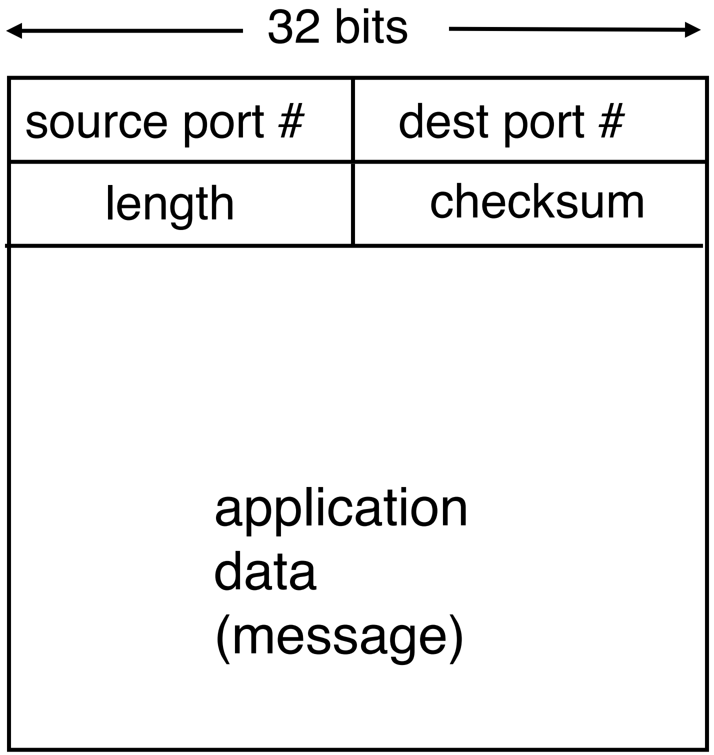
Error Detection

Necessary, but insufficient, for reliability

Data may get corrupted along the way…

- Bits flipped from 0→1 or 1→0
- Packet bits lost
- How to detect errors?

- Idea: compute a function over the data
  - Store the result along with the data
- Function must be easy to compute
- Function & stored data efficient to verify
- Ideas for functions?



UDP segment format

From the UDP specification (RFC 768)

- Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

- The pseudo header conceptually prefixed to the UDP header contains the source address, the destination address, the protocol, and the UDP length.

UDP Checksum

Sender:

- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
- NO - error detected
- YES - no error detected.

UDP checksum example

- Note: when adding numbers, a carryout from the most significant bit needs to be added to the result
- Example: add two 16-bit integers

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

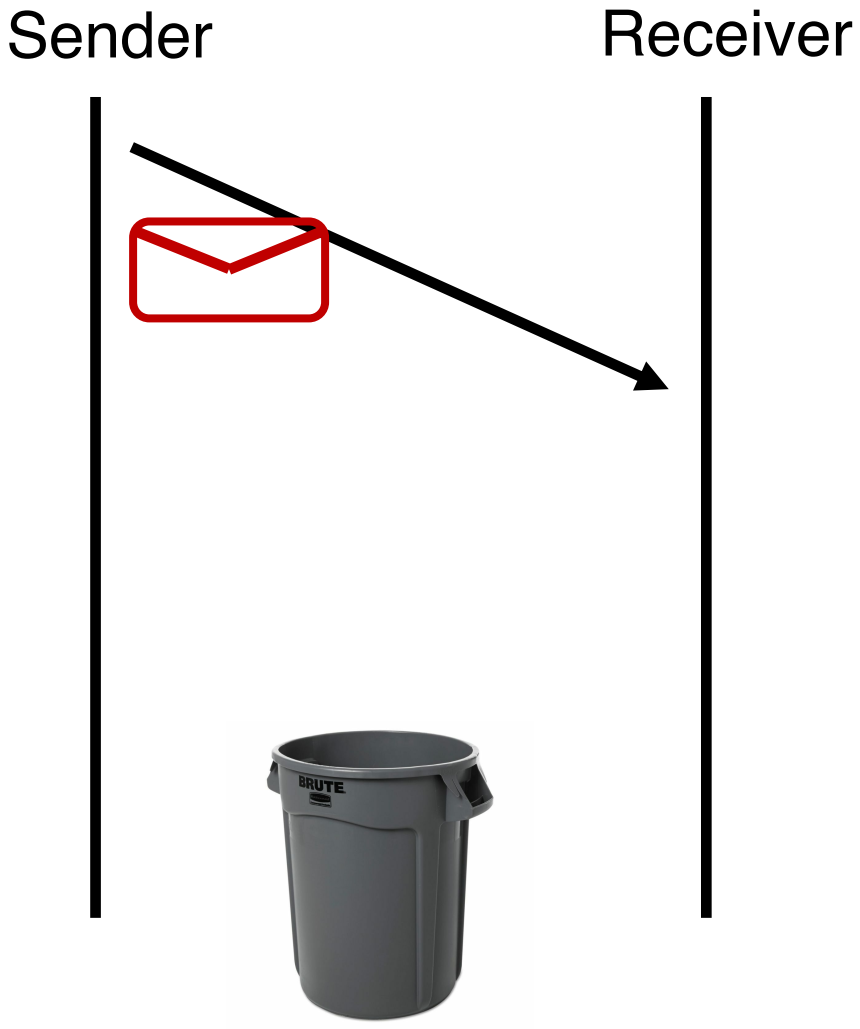
User Datagram Protocol

- A thin shim around best-effort network delivery
  - Lightweight to send one-off request/response messages
  - Lightweight transport for loss-tolerant delay-sensitive applications
- Provides basic multiplexing/demultiplexing for applications
- No reliability, performance, or ordering guarantees
  - Need Transmission Control Protocol (TCP)
- Can do basic error detection (bit flips) using checksums
  - Error detection is a necessary condition for reliability
- But need to do a lot more to achieve reliable data delivery
  - Subject of the rest of this lecture
  - Mechanisms in general; some of them implemented in TCP

Reliable data delivery

Stop and Wait

Packet loss

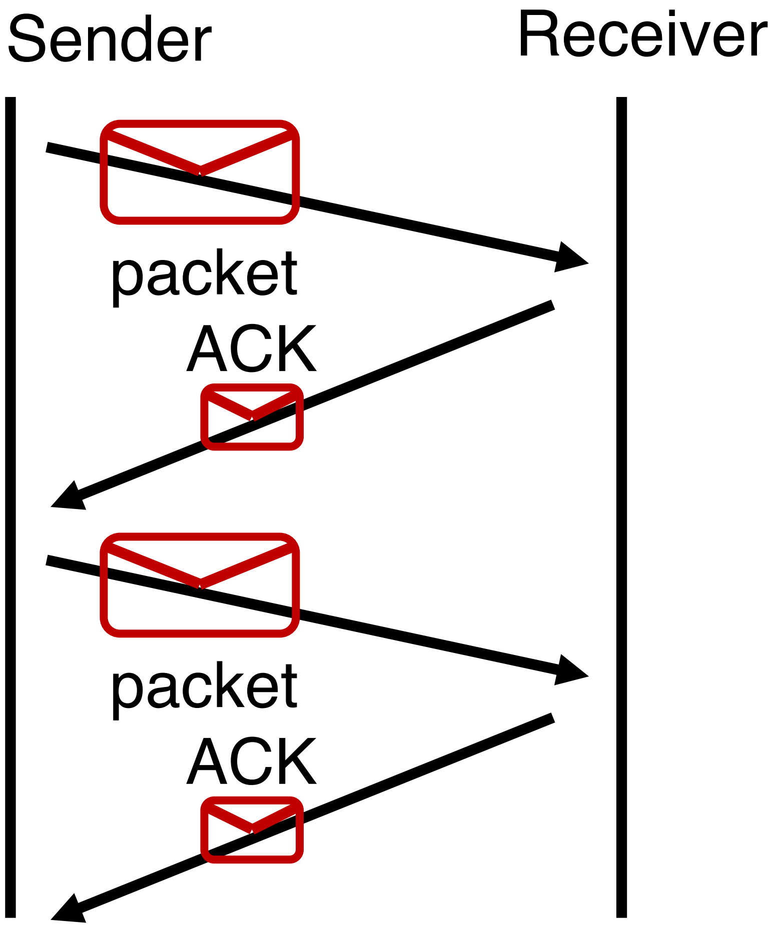


- How might a sender and receiver ensure that data is delivered reliably (despite some packets being lost)?

- TCP uses three mechanisms

Coping with packet loss: (1) ACK

- Key idea: Receiver returns an acknowledgment (ACK) per packet sent
- If sender receives an ACK, it knows that the receiver got the packet.

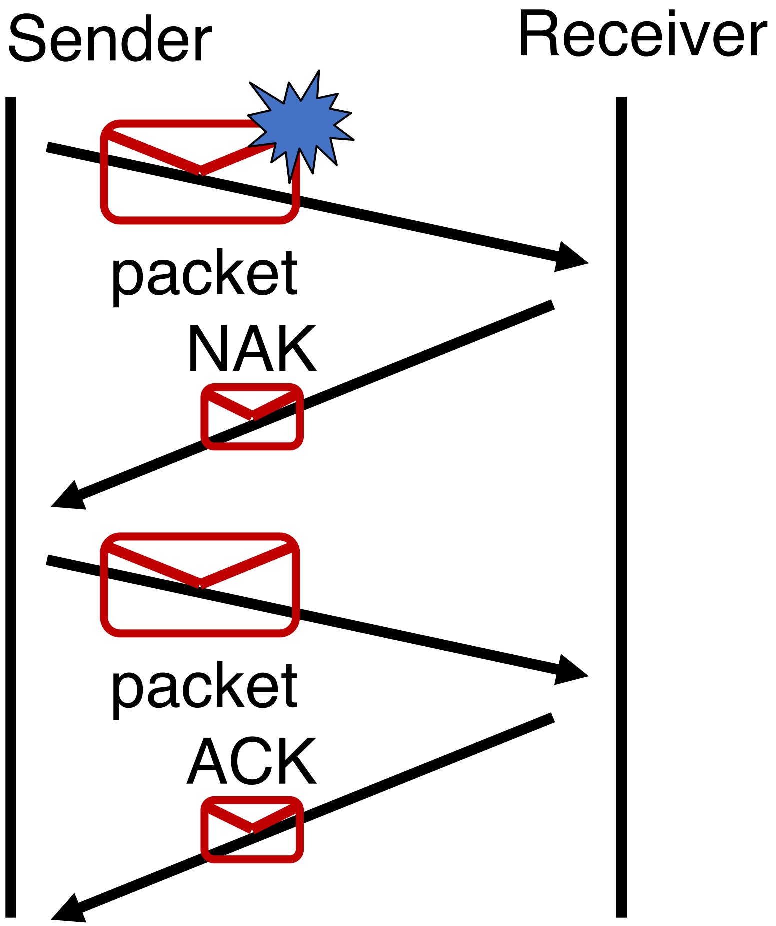


Coping with packet corruption: (1) ACK

- ACKs also work to detect packet corruption on the way to the receiver
  - A receiver could send a negative acknowledgment, or a NAK, if it receives a corrupted packet

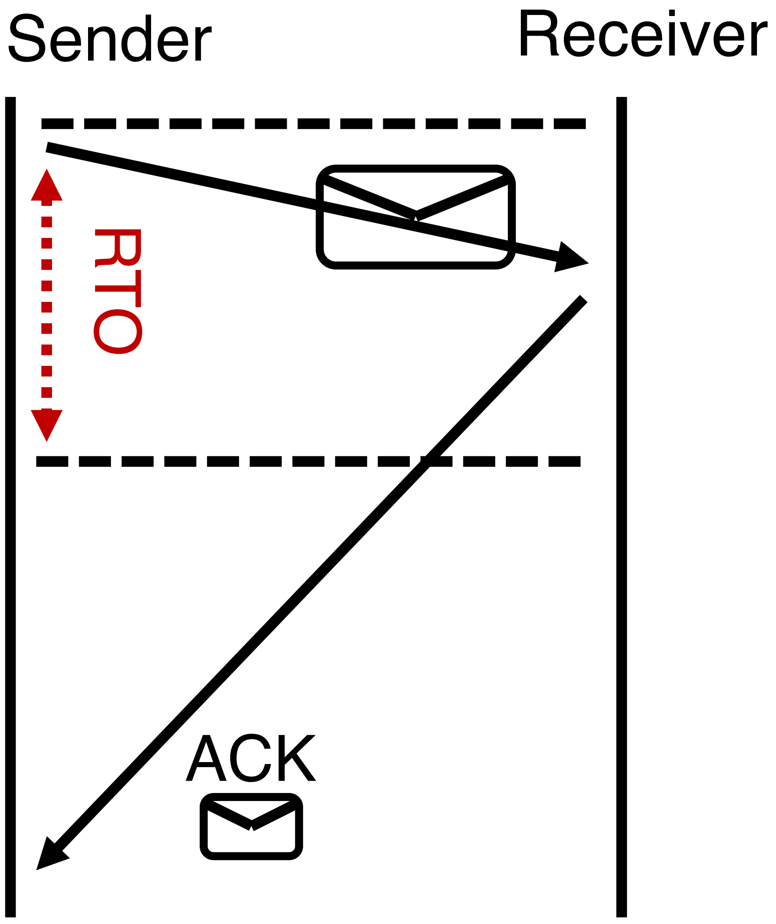
- TCP only uses positive acknowledgments (ACKs).

- What if a packet was lost and ACK never arrives?



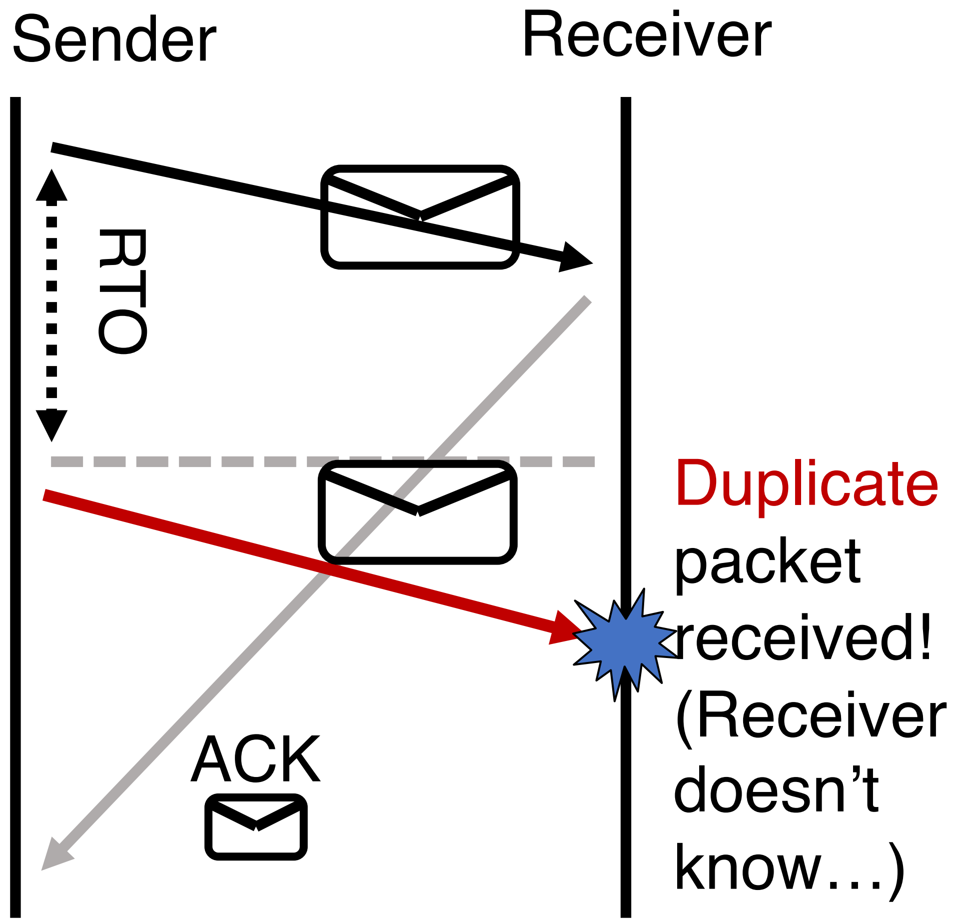
Coping with packet loss: (2) RTO

- Key idea: Wait for a duration of time (called retransmission timeout or RTO) before re-sending the packet
- In TCP, the onus is on the sender to retransmit lost data when ACKs are not received
- Note that retransmission works also if ACKs are lost or delayed



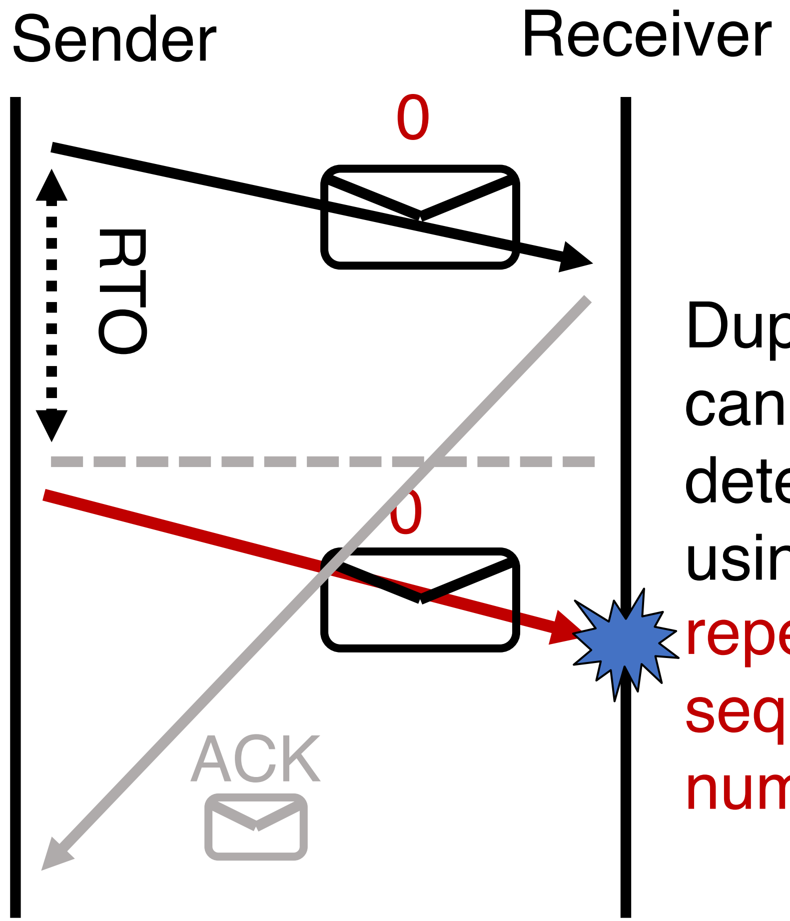
Coping with packet loss

- What if ACKs are delayed?
  - Sender may retransmit the same data
  - Receiver wouldn't know that it just received duplicate data from this retransmitted packet
- Add some identification to each packet to help distinguish between adjacent transmissions
  - This is known as the sequence number



Coping with packet loss: (3) Seq nums

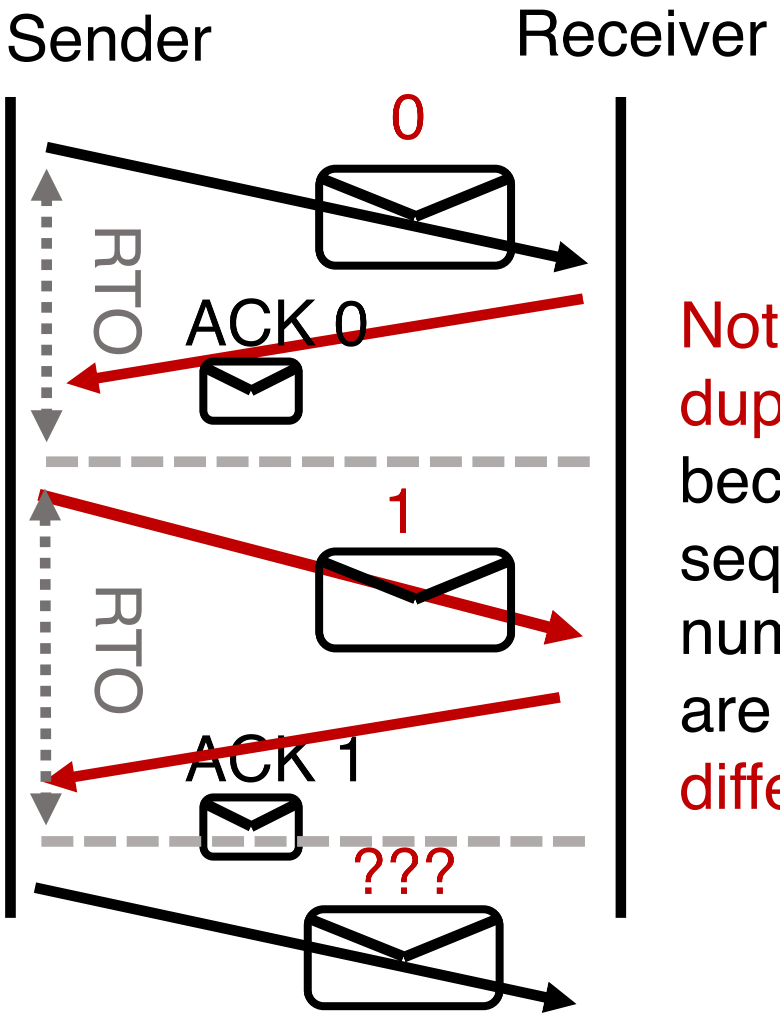
- Bad cases: packet dropped or ACK dropped or ACK delayed beyond RTO
- Sequence numbers of adjacent transmitted packets are exactly the same!
  - Receiver can disambiguate a fresh packet from a retransmission



Duplicate can be detected using repeated sequence number

Coping with packet loss: (3) Seq nums

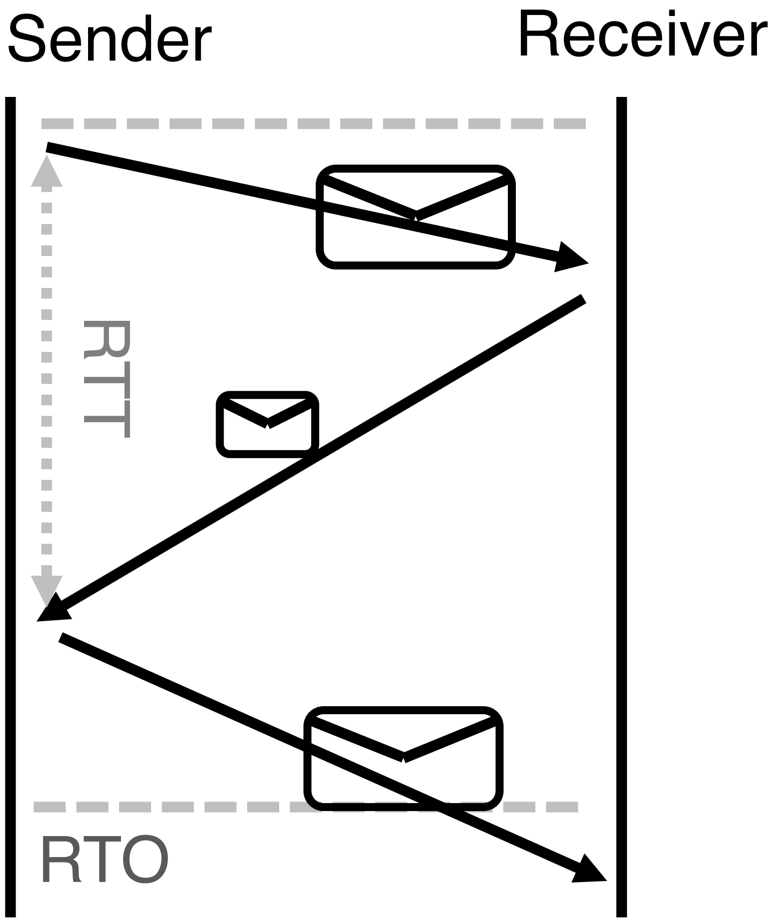
- Good case: packet received and ACK received within the RTO
- Sequence numbers of adjacent transmitted packets are different
- Q1: where are the sequence numbers written to, exactly?
- Q2: what is the seq# of third packet?



Not duplicate, because sequence numbers are different

Stop-and-Wait Reliability

- Scheme so far: sender waits for an ACK/RTO before sending another packet
  - Also called “stop and wait” reliability
- Suppose no packets are dropped
  - Round-trip-time: 100 milliseconds
  - Packet size: 12,000 bits
  - Link rate: 12 Mega bits/s
- At what rate is the sender getting data across to the receiver?



120 Kilo bit/s (1% of link rate)

Reliable data delivery

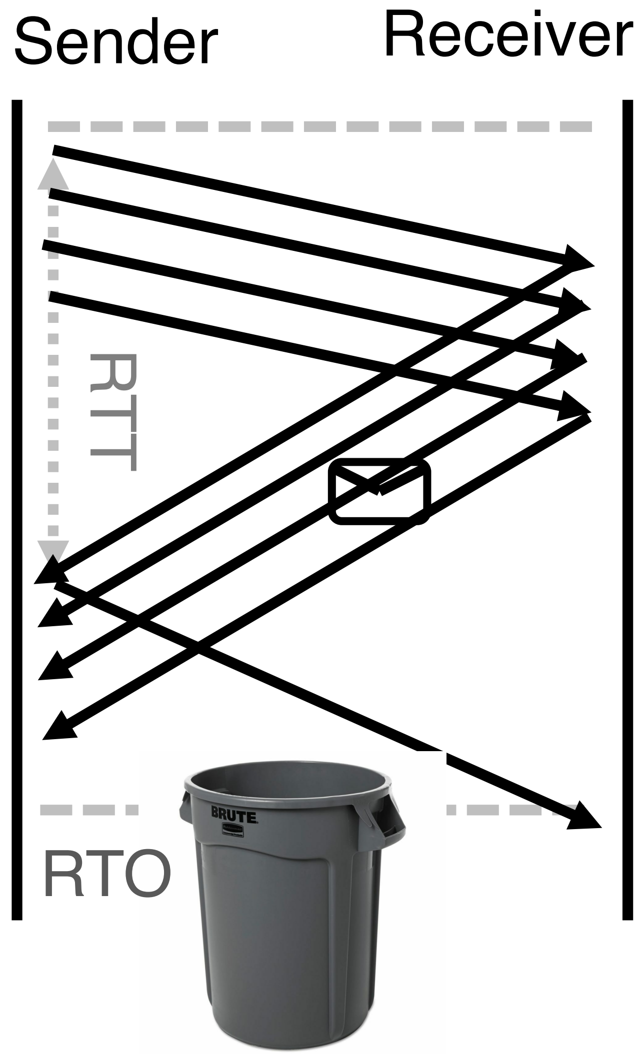
Pipelined Transmission, a.k.a. Sliding Window Protocols

Amount of “in-flight” data

- We term the amount of unACKed data as data “in flight”
- With just one packet in flight, the data rate is limited by the packet delay (RTT) rather than available bandwidth (link rate)
- Idea: Keep many packets in flight
  - Also referred to as pipelined transmission
- More packets in flight improves throughput
  - Throughput is the amount of data delivered per unit time.

Keeping many packets in flight

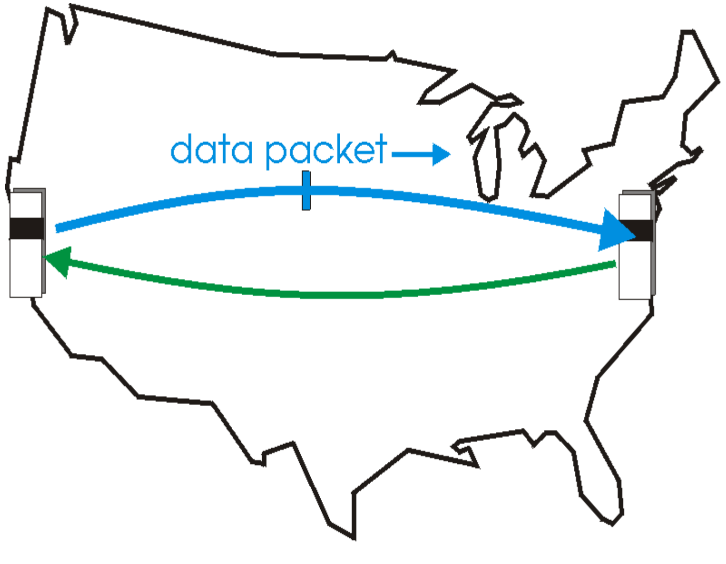
- With link rate 12 Mega bits/s and packet size 12,000 bits, what is the transmission delay of a single packet?
- Remember the earlier throughput of 120 Kbit/s. If there are, say 4 packets in flight, throughput is 480 Kbits/s!
- We just improved the throughput 4 times by keeping 4 packets in flight



TCP is a pipelined transmission protocol

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

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

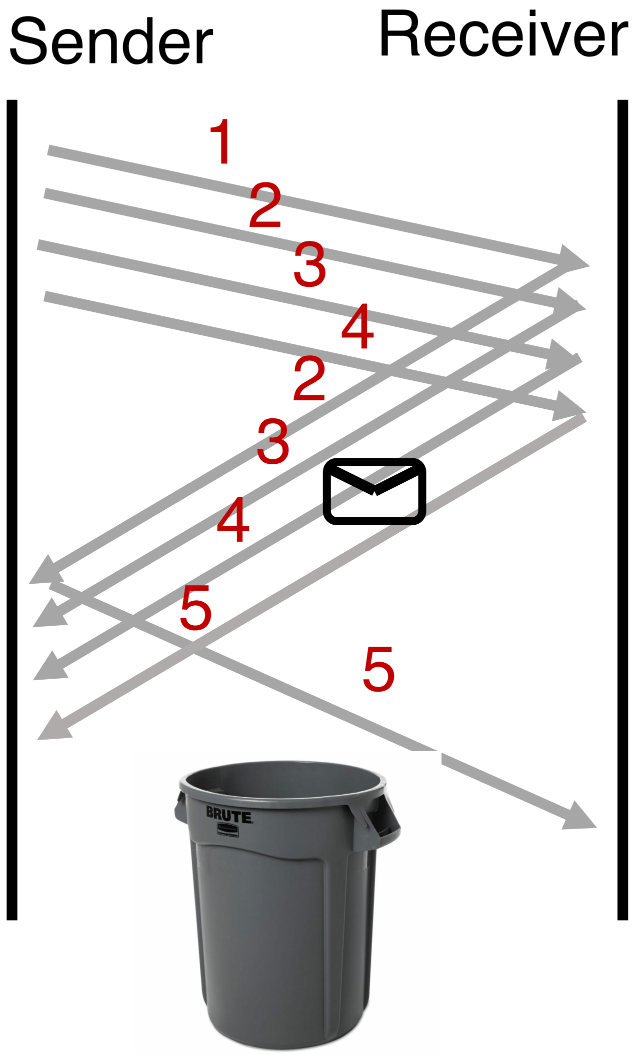


(a) a stop-and-wait protocol in operation

(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?

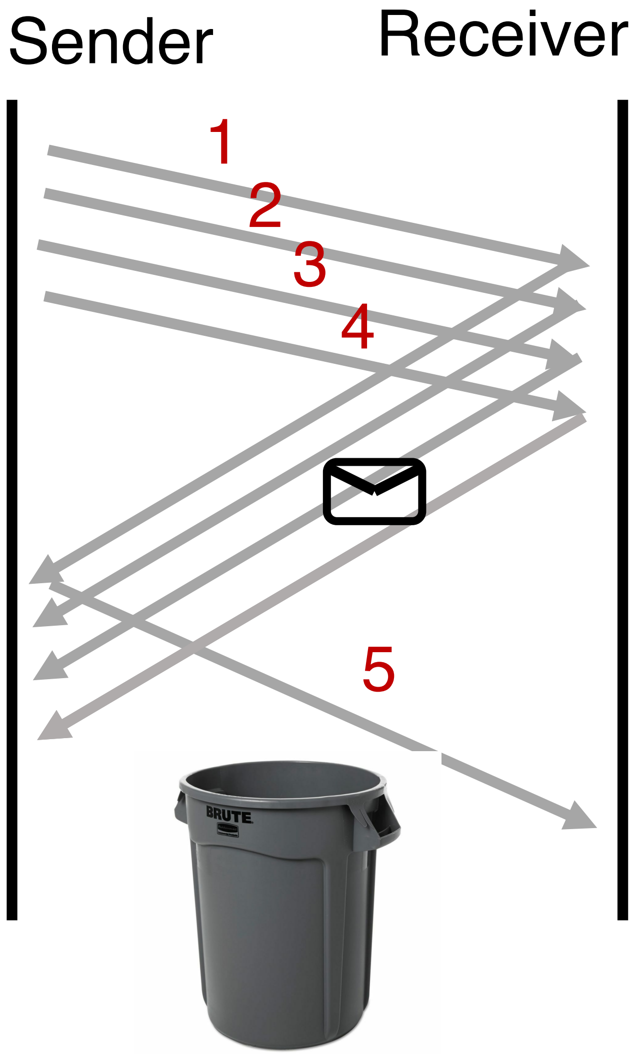
Last pkt in order

Cumulative ACK

TCP's default

Seq # ranges received so far

Selective ACK



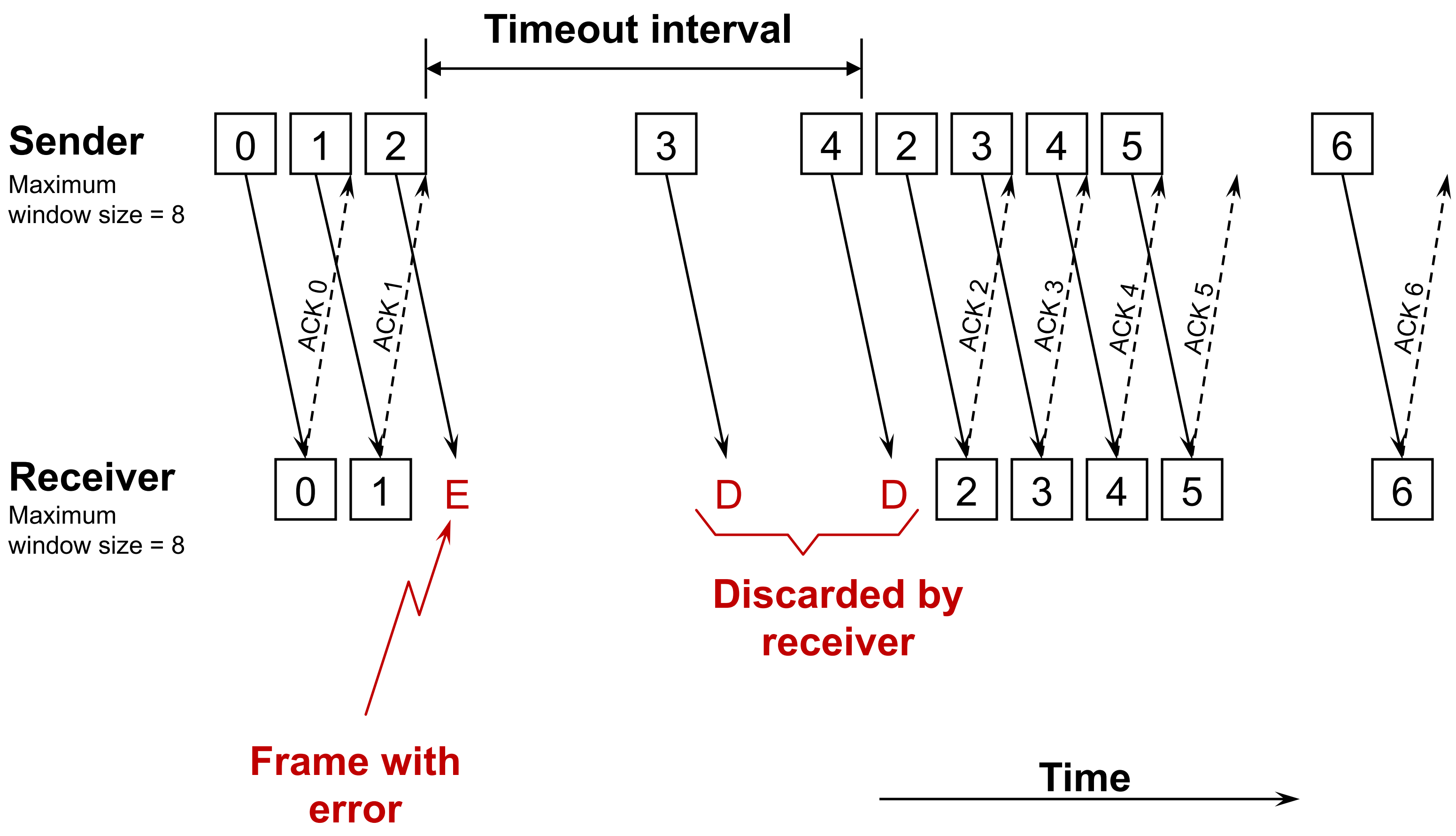
Sliding Window with Go Back N

- 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


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

Go Back N *(cont'd)*

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.

Selective repeat