

CSCI 466: Networks

Pipelining

Reese Pearsall
Fall 2023

Announcements

PA 2 Posted. Due Wednesday October 18th

OSI Model

Application Layer

Presentation Layer *

Session Layer *

Transport Layer

Network Layer

Data Link Layer

Physical Layer

Application Layer

Messages from Network Applications

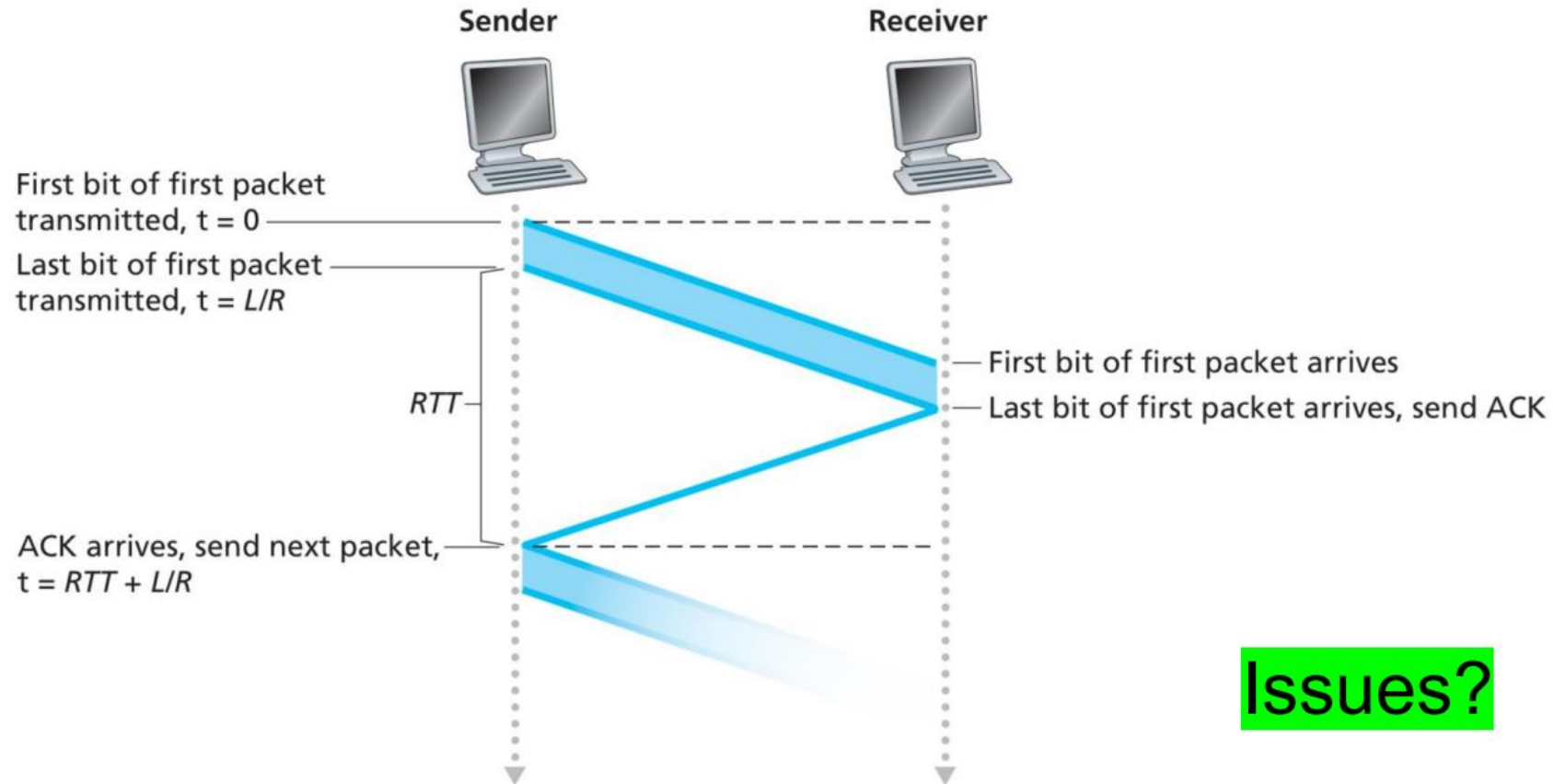


Physical Layer

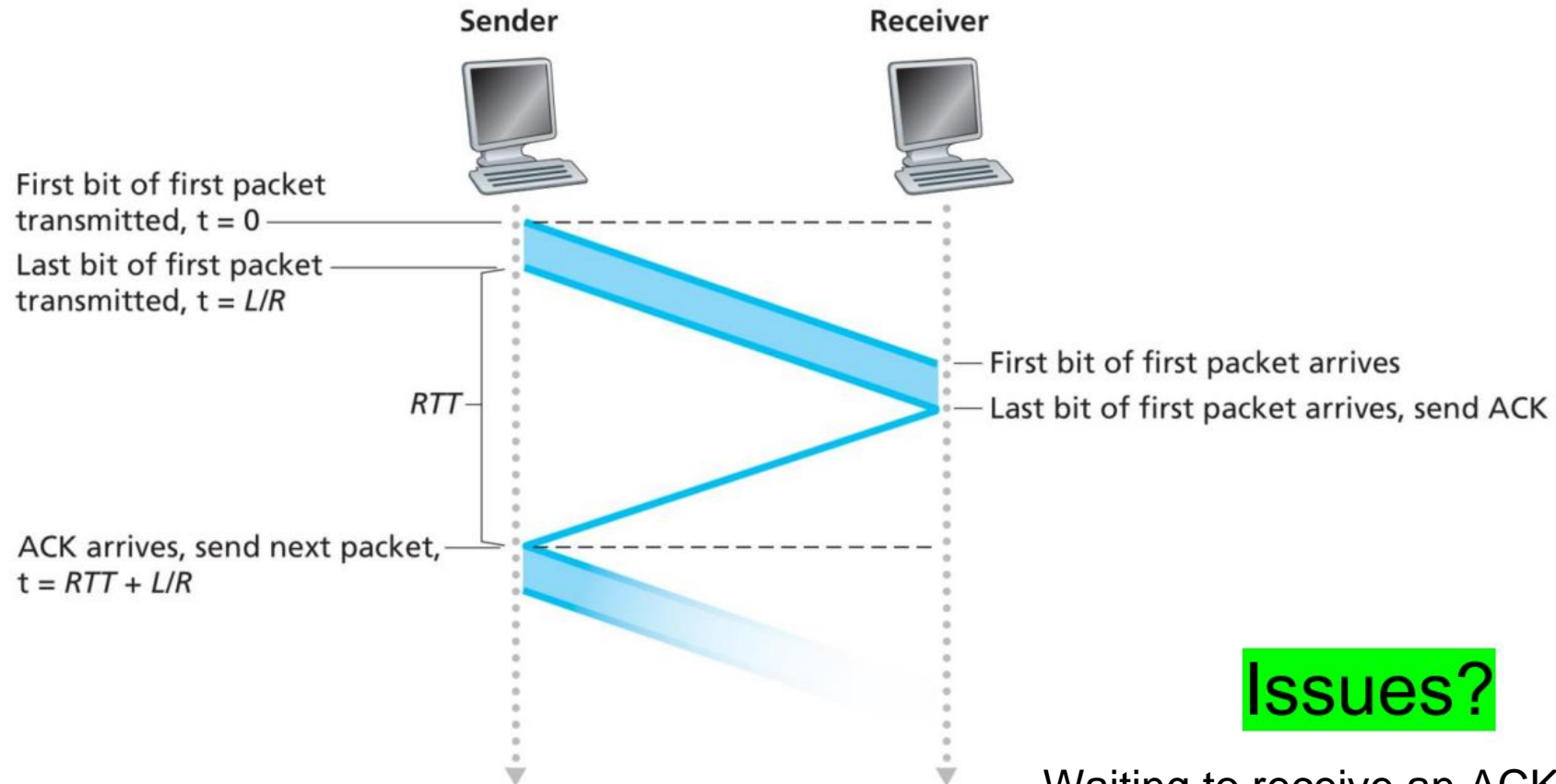
Bits being transmitted over some medium

**In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used*

So far, our reliable data transfer protocols have been **stop-and-wait**



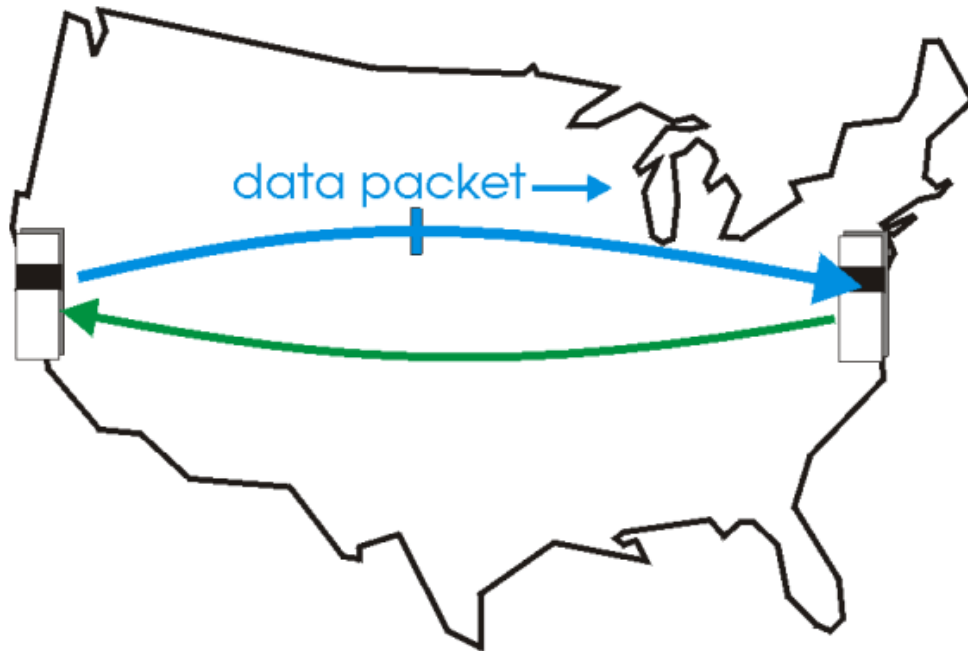
So far, our reliable data transfer protocols have been **stop-and-wait**



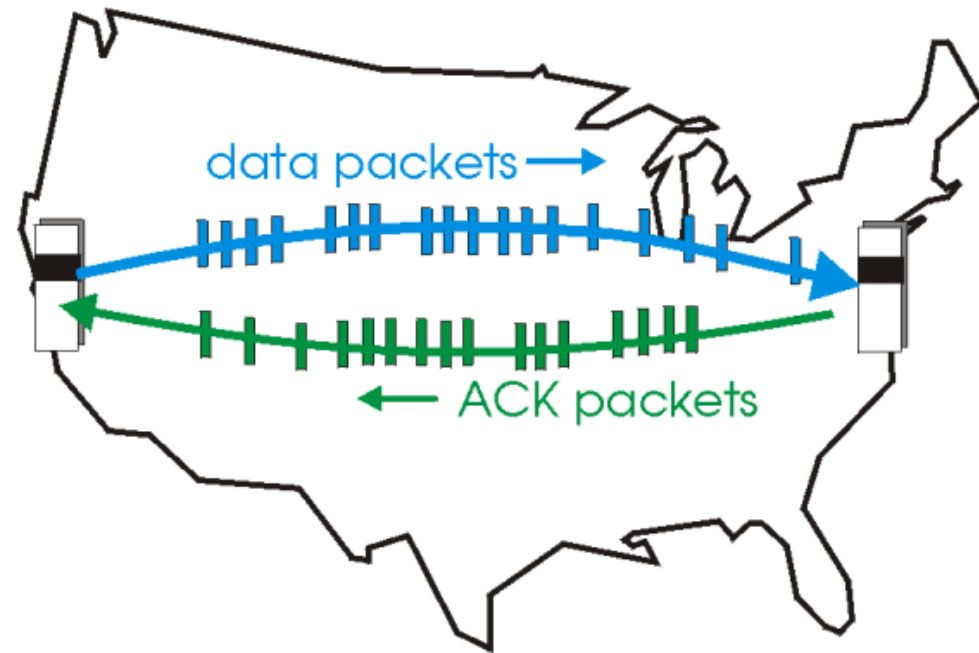
Issues?

Waiting to receive an ACK is inefficient
We spend a *lot* of time waiting

Pipelining: sender allows multiple, “in-flight”, yet-to-be acknowledged packets

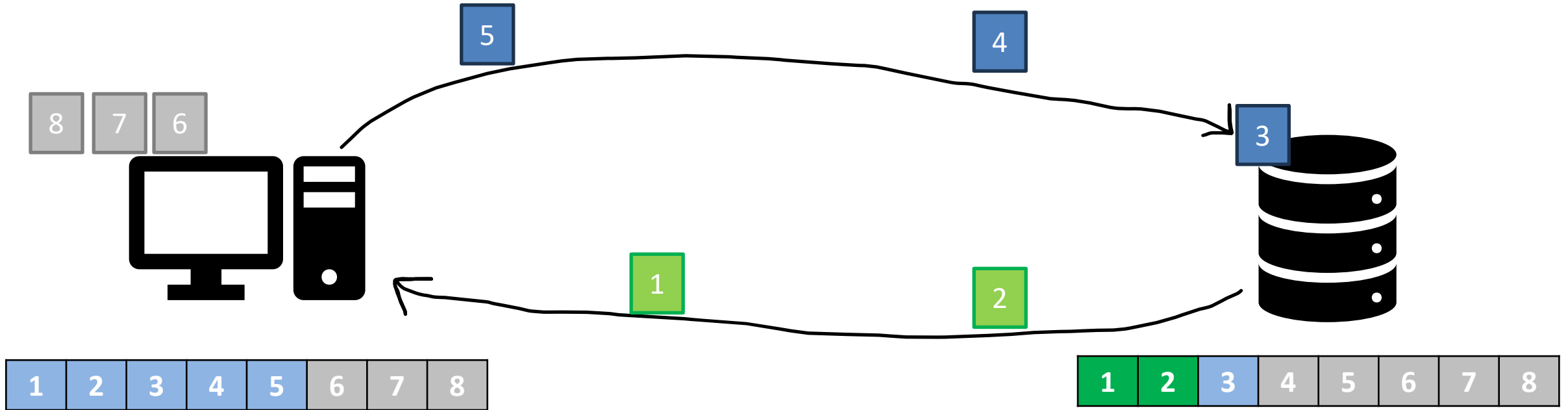


(a) a stop-and-wait protocol in operation



(b) a pipelined protocol in operation

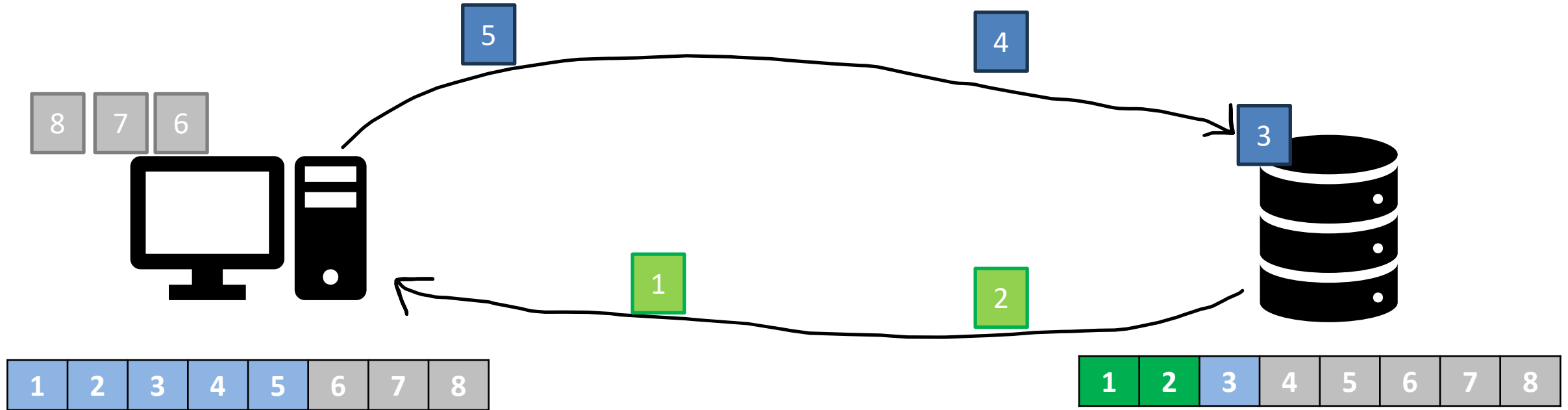
Pipelining: sender allows multiple, “in-flight”, yet-to-be acknowledged packets



Consequences:

- Need to use a wider range of sequence numbers

Pipelining: sender allows multiple, “in-flight”, yet-to-be acknowledged packets

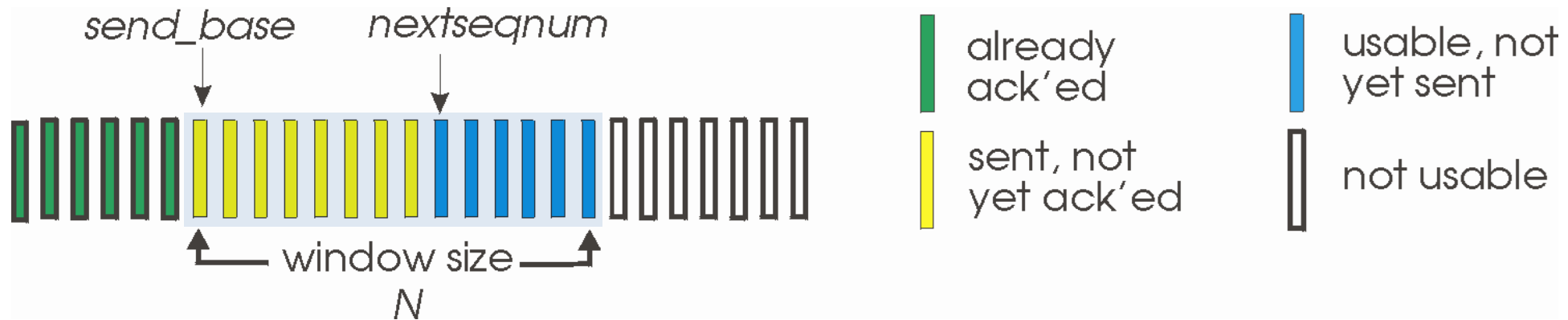


Consequences:

- Need to use a wider range of sequence numbers
- Sender and receiver may need to **buffer** more than one packet

Go-Back-N: sender

- sender: “window” of up to N , consecutive transmitted but unACKed pkts
 - k -bit seq # in pkt header

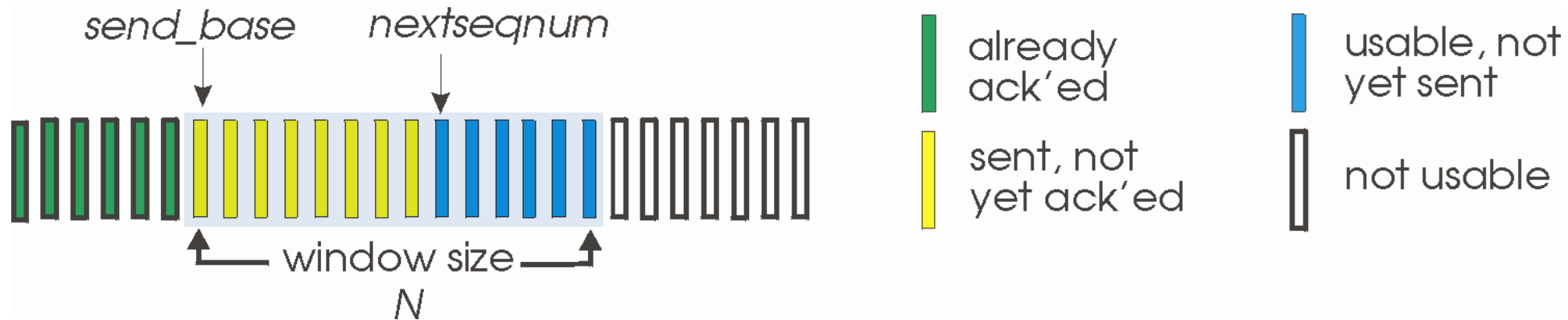


- ***cumulative ACK***: $ACK(n)$: ACKs all packets up to, including seq # n
 - on receiving $ACK(n)$: move window forward to begin at $n+1$
- timer for oldest in-flight packet
- ***timeout(n)***: retransmit packet n and all higher seq # packets in window

Go-Back-N: sender

(Why not have an infinite window size?)

- sender: “window” of up to N , consecutive transmitted but unACKed pkts
 - k -bit seq # in pkt header

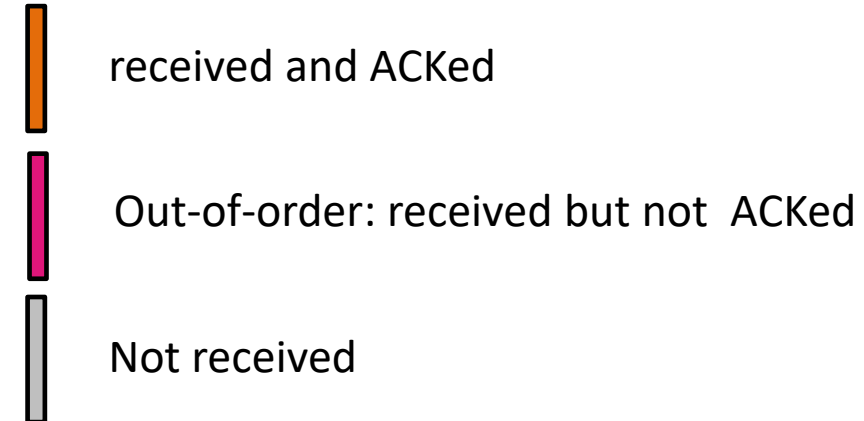
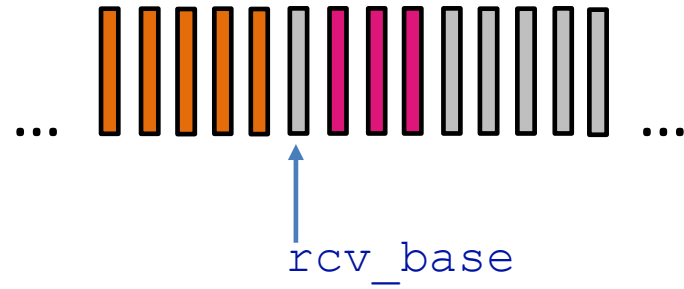


- **cumulative ACK:** $ACK(n)$: ACKs all packets up to, including seq # n
 - on receiving $ACK(n)$: move window forward to begin at $n+1$
- timer for oldest in-flight packet
- **timeout(n):** retransmit packet n and all higher seq # packets in window

Go-Back-N: receiver

- ACK-only: always send ACK for correctly-received packet so far, with highest *in-order* seq #
 - may generate duplicate ACKs
 - need only remember `rcv_base`
- on receipt of out-of-order packet:
 - can discard (don't buffer) or buffer: an implementation decision
 - re-ACK pkt with highest in-order seq #

Receiver view of sequence number space:



sender window (N=4)

0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8

sender

send pkt0
send pkt1
send pkt2
send pkt3
(wait)

rcv ack0, send pkt4
rcv ack1, send pkt5

ignore duplicate ACK



pkt 2 timeout

send pkt2
send pkt3
send pkt4
send pkt5

receiver

receive pkt0, send ack0
receive pkt1, send ack1

receive pkt3, discard,
(re)send ack1

receive pkt4, discard,
(re)send ack1

receive pkt5, discard,
(re)send ack1

rcv pkt2, deliver, send ack2
rcv pkt3, deliver, send ack3
rcv pkt4, deliver, send ack4
rcv pkt5, deliver, send ack5

https://www2.tkn.tu-berlin.de/teaching/rn/animations/gbn_sr/

Selective repeat: the approach

- *pipelining*: multiple packets in flight
- *receiver individually ACKs* all correctly received packets
 - buffers packets, as needed, for in-order delivery to upper layer
- sender:
 - maintains (conceptually) a timer for each unACKed pkt
 - timeout: retransmits single unACKed packet associated with timeout
 - maintains (conceptually) “window” over *N* consecutive seq #s
 - limits pipelined, “in flight” packets to be within this window

https://www2.tkn.tu-berlin.de/teaching/rn/animations/gbn_sr/

Selective repeat: sender and receiver

sender

data from above:

- if next available seq # in window, send packet

timeout(n):

- resend packet n , restart timer

ACK(n) in [sendbase, sendbase+N-1]:

- mark packet n as received
- if n smallest unACKed packet, advance window base to next unACKed seq #

receiver

packet n in [rcvbase, rcvbase+N-1]

- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order packets), advance window to next not-yet-received packet

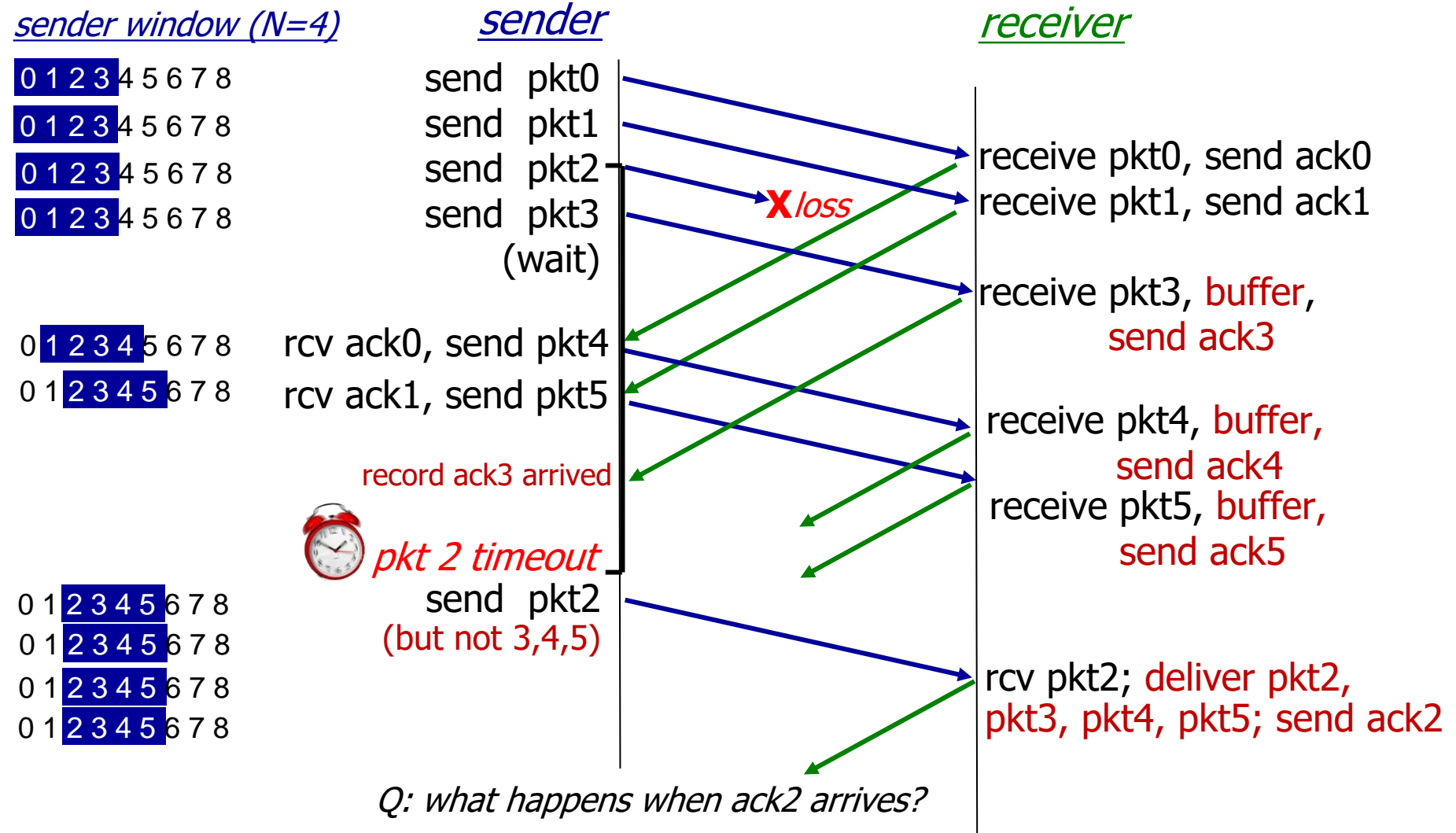
packet n in [rcvbase-N, rcvbase-1]

- ACK(n)

otherwise:

- ignore

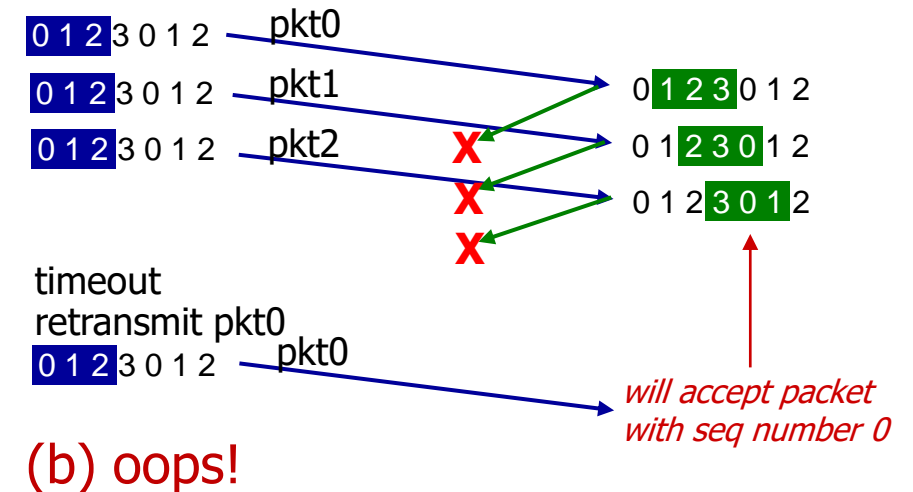
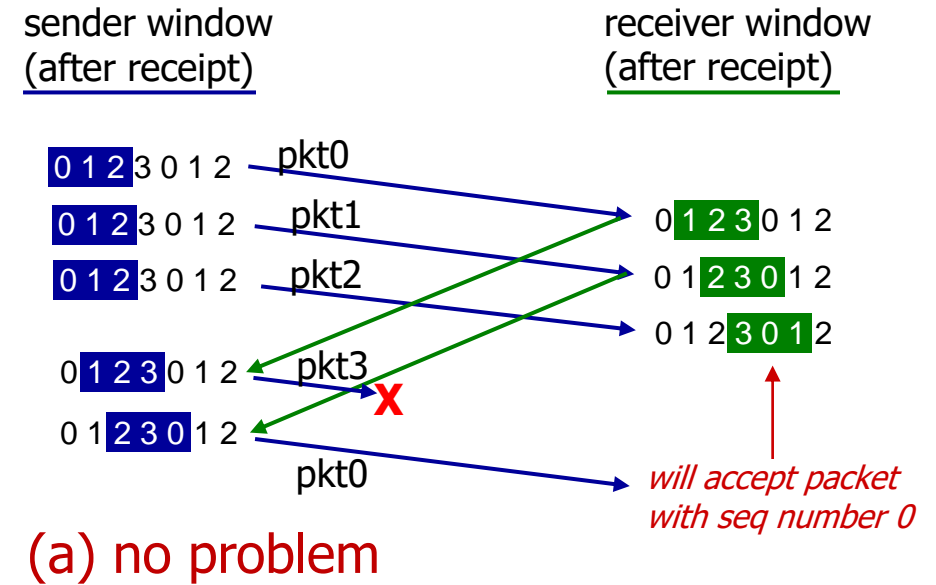
Selective Repeat in action



Selective repeat: a dilemma!

example:

- seq #s: 0, 1, 2, 3 (base 4 counting)
- window size=3

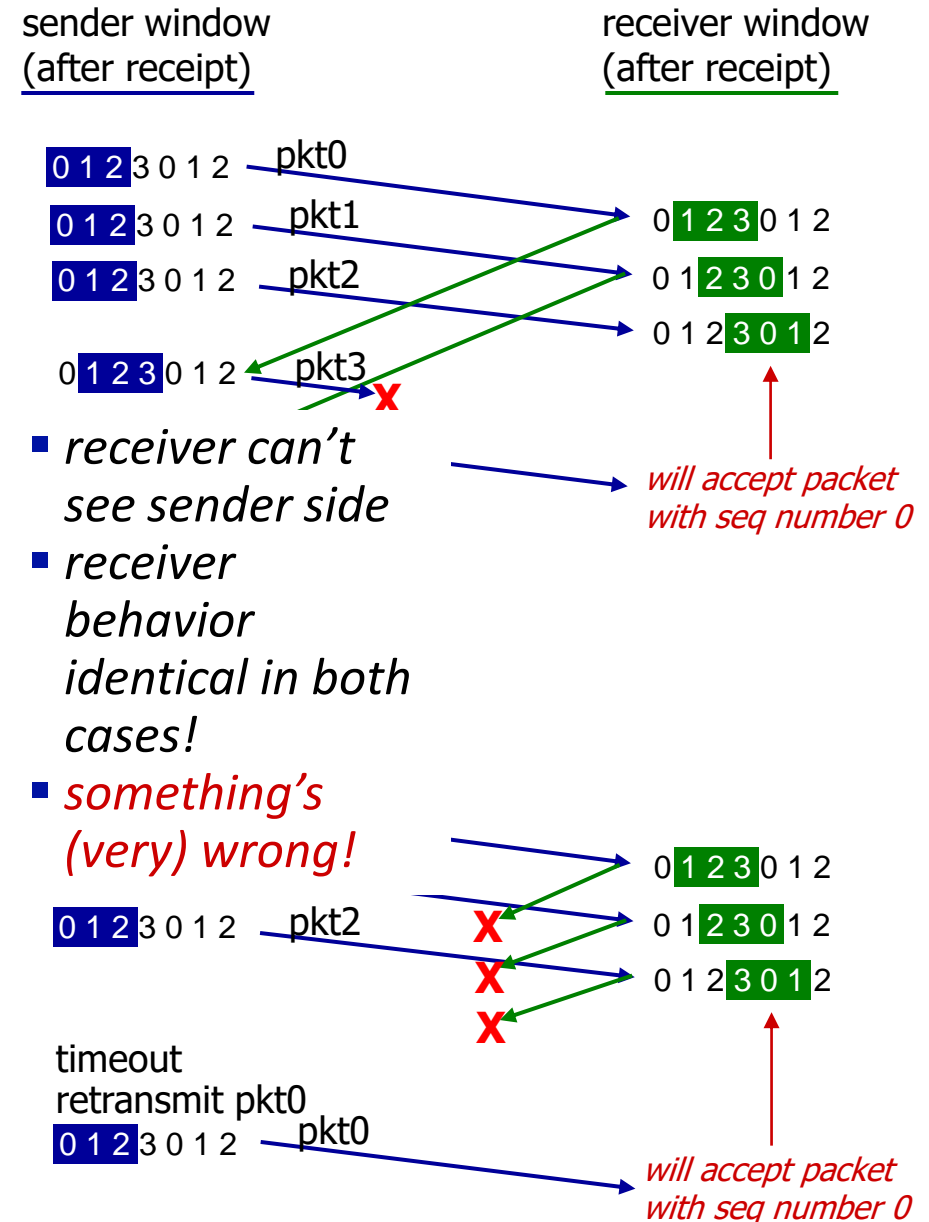


Selective repeat: a dilemma!

example:

- seq #s: 0, 1, 2, 3 (base 4 counting)
- window size=3

Q: what relationship is needed between sequence # size and window size to avoid problem in scenario (b)?



Transport Layer

RDT Principles

Checksum- Used to detect bit errors in transmitted packets

Checksum- Used to detect bit errors in transmitted packets

Timer- Used to timeout/retransmit a packet, possibly because the packet (or its ACK) was lost within a channel

Checksum- Used to detect bit errors in transmitted packets

Timer- Used to timeout/retransmit a packet, possibly because the packet (or its ACK) was lost within a channel

Sequence Number- Used for sequential numbering of packets of data flowing from sender to receiver. Gaps in sequence number of packets allow the receiver to detect a lost or duplicate packet

Checksum- Used to detect bit errors in transmitted packets

Timer- Used to timeout/retransmit a packet, possibly because the packet (or its ACK) was lost within a channel

Sequence Number- Used for sequential numbering of packets of data flowing from sender to receiver. Gaps in sequence number of packets allow the receiver to detect a lost or duplicate packet

Acknowledgement- Used by the receiver to tell the sender that a packet or set of packets has been received correctly. ACKs will typically carry the sequence # of the packet being acknowledged

Checksum- Used to detect bit errors in transmitted packets

Timer- Used to timeout/retransmit a packet, possibly because the packet (or its ACK) was lost within a channel

Sequence Number- Used for sequential numbering of packets of data flowing from sender to receiver. Gaps in sequence number of packets allow the receiver to detect a lost or duplicate packet

Acknowledgement- Used by the receiver to tell the sender that a packet or set of packets has been received correctly. ACKs will typically carry the sequence # of the packet being acknowledged

Negative Acknowledgement- Used by the receiver to tell the sender that a packet has not been received correctly. Negative acknowledgements will typically carry the sequence number of the packet that was not received correctly

Checksum- Used to detect bit errors in transmitted packets

Timer- Used to timeout/retransmit a packet, possibly because the packet (or its ACK) was lost within a channel

Sequence Number- Used for sequential numbering of packets of data flowing from sender to receiver. Gaps in sequence number of packets allow the receiver to detect a lost or duplicate packet

Acknowledgement- Used by the receiver to tell the sender that a packet or set of packets has been received correctly. ACKs will typically carry the sequence # of the packet being acknowledged

Negative Acknowledgement- Used by the receiver to tell the sender that a packet has not been received correctly. Negative acknowledgements will typically carry the sequence number of the packet that was not received correctly

Window, pipelining- The sender may be restricted to sending only packets with sequence numbers that fall within a given range. By allowing multiple packets to be transmitted but not yet acknowledged, sender utilization can be increased over a stop-and-wait mode of operation