

Computer Networks

COL 334/672

Reliability in TCP

Slides adapted from KR

Sem 1, 2025-26

Recap

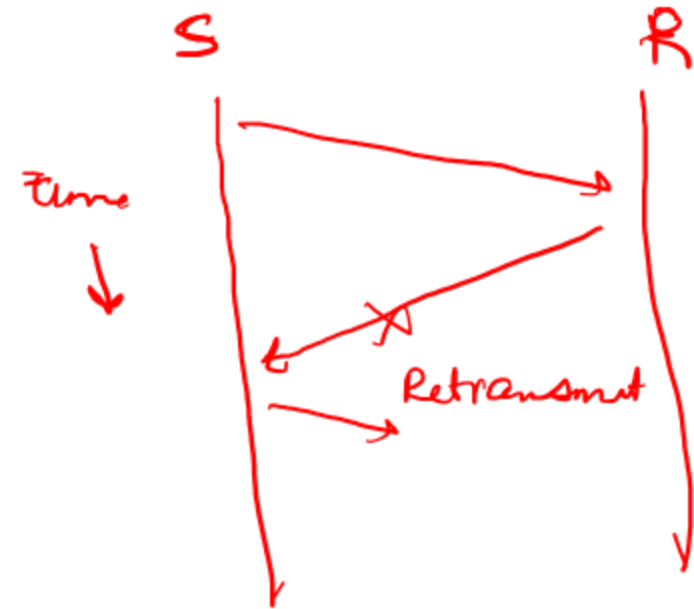
Reliability in Transport layer

①. Forward Error correction (Don't use it)

② Automatic Repeat Request (ARQ)

↳ Stop & wait

↳ Pipelining or Sliding window protocols



ACK every packet independently
or ACK only in-order packet

→ ① Go BACK N
② Selective Repeat

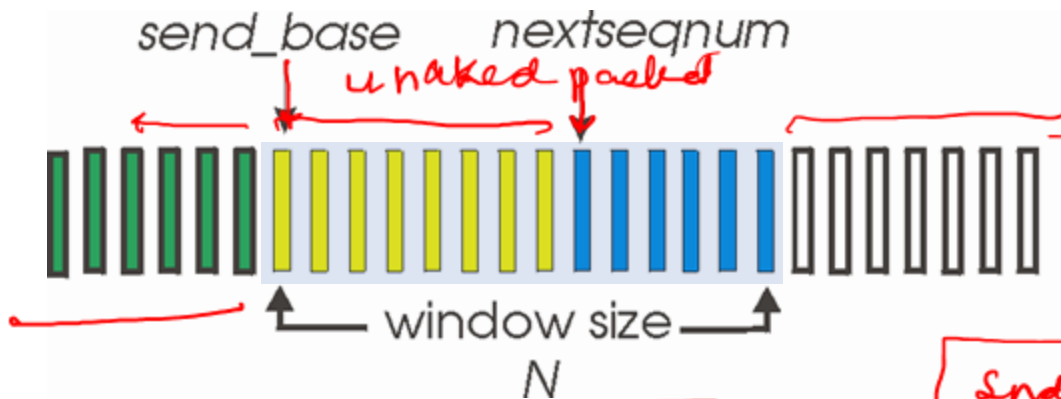
Go-Back-N: sender

$$2^k \geq N$$

- SWS of N packets
- k -bit seq # in pkt header

$ACK(n)$

all pkts before n have
been received



$$\text{cur seq} \leq \text{snd_base} + N$$

$\text{snd_base} > \text{Ack}(n)$
old Ack, discard

if $ACK(n) > \text{snd_base}$
 $\text{snd_base} = n$

- **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

$$ACK(n) = n$$

$$ACK(n) = n+1$$

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:
 - discard (don't buffer) the packet
 - re-ACK pkt with highest in-order seq #

Go-Back-N in action

maintain a receive buffer

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

LIMITATION

wasted bandwidth in case of out-of-order delivery

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

h/w or logical → timer

Selective repeat: sender and receiver

Event-driven program
↳ I/O interrupts / timer interrupts

cur-seq # \leq snd-base + N

100 101

sender

→ data from above:

- if next available seq # in window, send packet (start a timer for the pkt)

→ 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 #

set sendbase to the next unacked pkt

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

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

record ack3 arrived



pkt 2 timeout
 send pkt2
 (but not 3,4,5)

Q: what happens when ack2 arrives?

receiver

receive pkt0, send ack0
 receive pkt1, send ack1

receive pkt3, buffer,
send ack3 rcw base = 2

receive pkt4, buffer,
send ack4
 receive pkt5, buffer,
send ack5

rcv pkt2; deliver pkt2,
pkt3, pkt4, pkt5; send ack2

rcw base = 6

X/loss

Claim:



CLAIM

$$\text{snd base} \geq \text{rcv base} - N$$

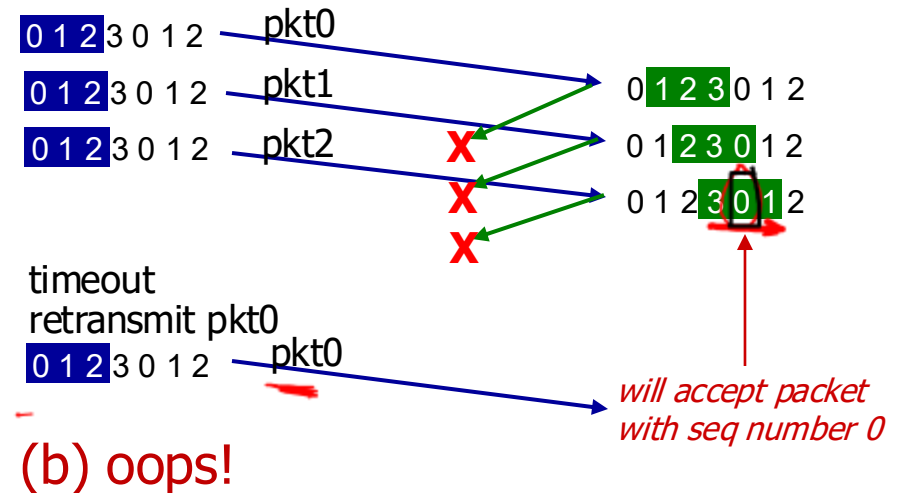
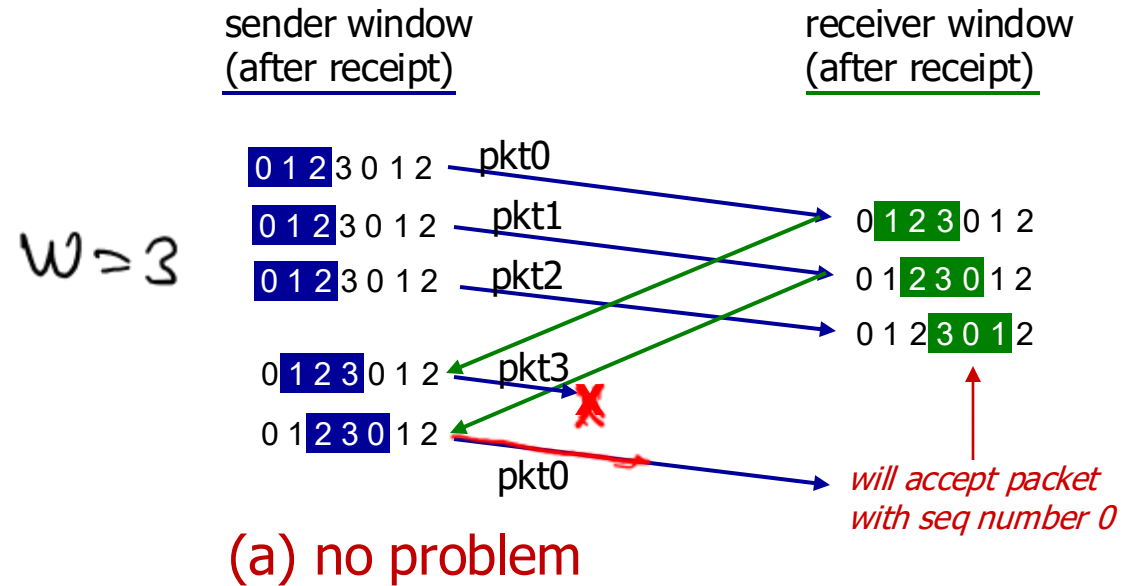
E.g. if $N = 10$, $\text{rcv base} = 100$

there is no way possible that $\text{snd base} = 90$,
else sender would not have sent the packet
with seq# 100

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

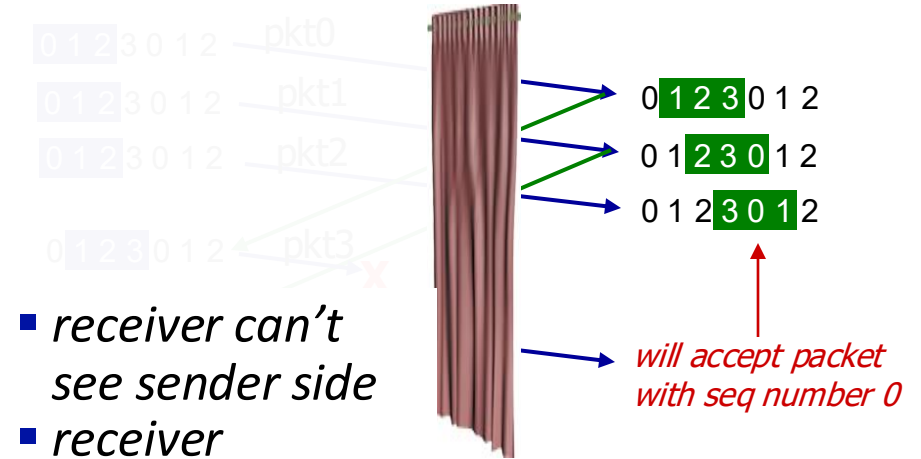
Q: what relationship is needed
between sequence # size and
window size to avoid problem
in scenario (b)?

Maximum discrepancy = $2N$

$$2N \leq 2^k$$
$$N \leq 2^{k-1}$$

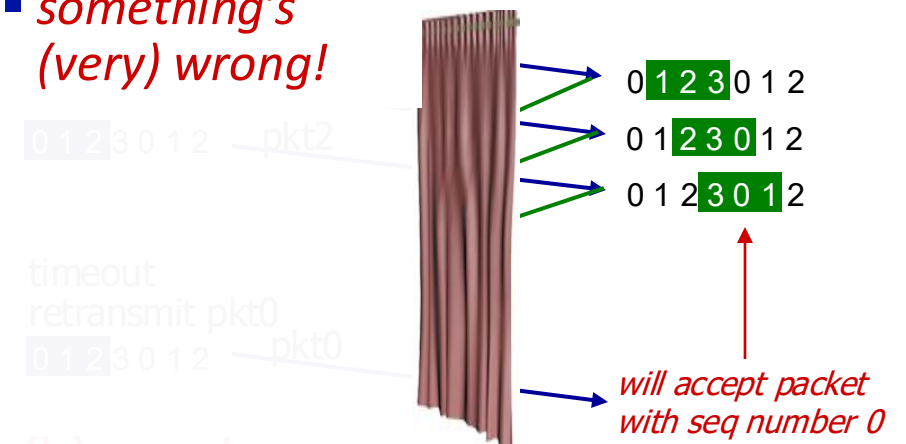
sender window
(after receipt)

receiver window
(after receipt)



- receiver can't see sender side
- receiver behavior identical in both cases!

■ *something's (very) wrong!*



(b) oops!

How does TCP implement reliability?

FACTS ABOUT TCP

①. TCP is a connection-oriented protocol

↳ Establishment & Termination protocols

②. Byte stream protocol

↳ seq #s are bytes ; not packets

③. Bidirectional data transfer

↳ Seq #s needed for both sender & receiver

↳ Ack #s needed for both sender & receiver

