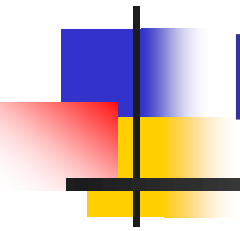
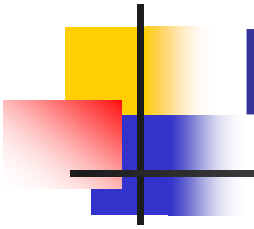


Reliable Data Transfer

Wanjiun Liao

March 30 2018

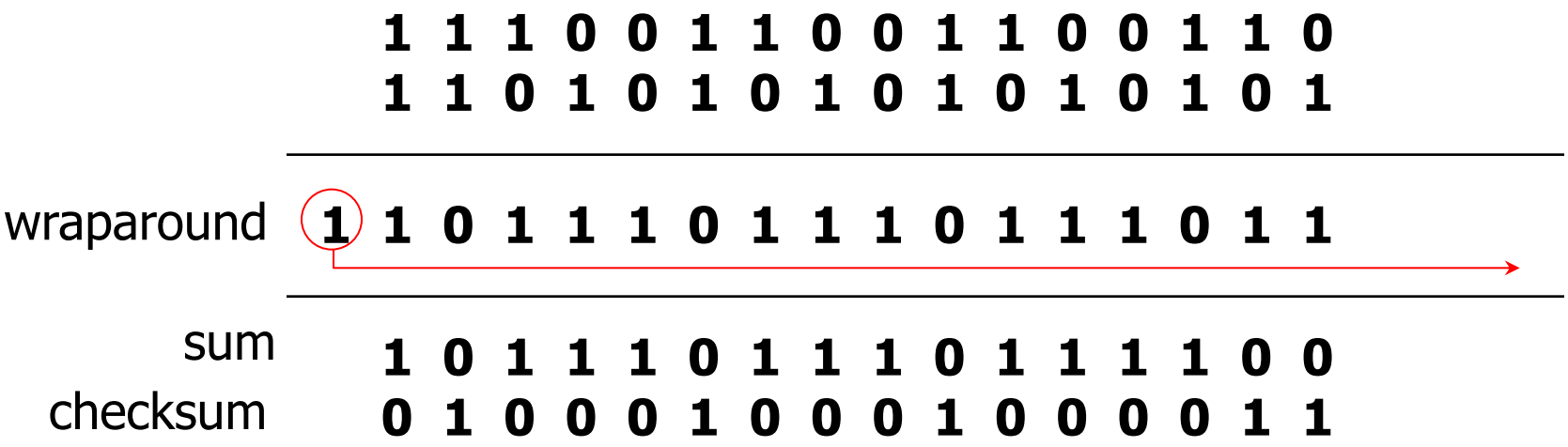


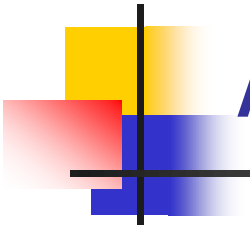
Reliable transfers?

- In sequence, no loss, no error, no duplication etc.
- Acknowledgment + retransmission
  - Two options:
    - Positive ACK (ACK)
    - Negative ACK (NAK)
- Error detection and sequence number
- Timeout

Error Detection

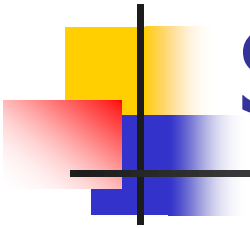
- Internet checksum
  - 1's complement
  - When adding numbers, a carryout from the most significant bit needs to be added to the result





Automatic Repeat reQuest (ARQ)

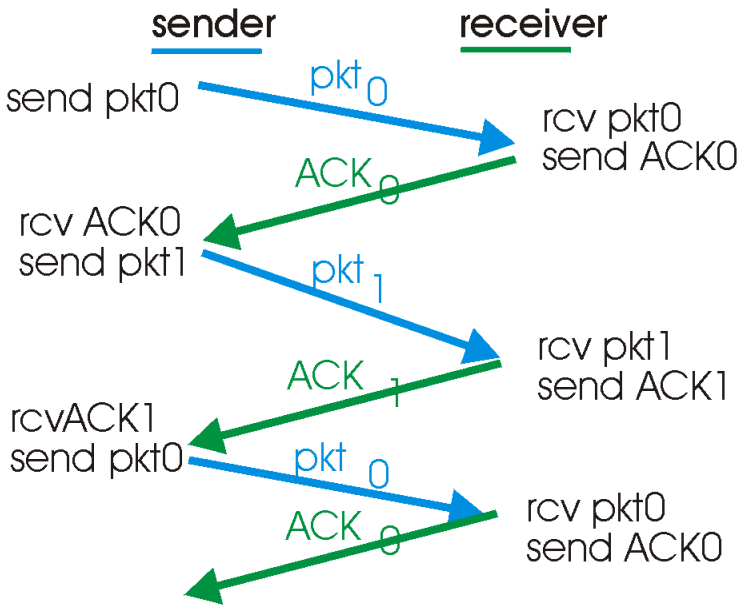
- Alternating-bit protocol
  - Stop and wait (S&W)
- Sliding window protocol
  - Go back N (GBN)
  - Selective repeat (selective retransmission, selective reject) (SR)



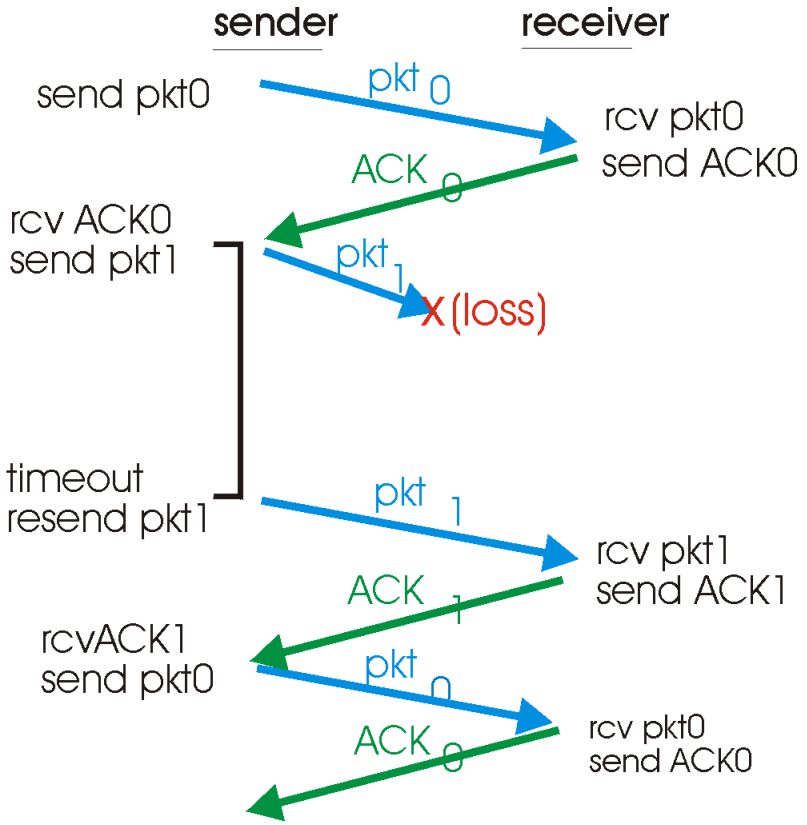
Stop and Wait

- Two questions to answer
  - How?
  - What if a lossy channel with bit errors?
- Sender
  - Transmit single frame and wait for ACK
  - If no ACK within timeout, retransmit
  - How to tell duplicate ACK from the normal one?
- Receiver
  - If OK, send ACK
  - If received frame damaged, discard it
  - How to tell retransmission from new one?
- One-bit sequence number is fine.

S&W Example

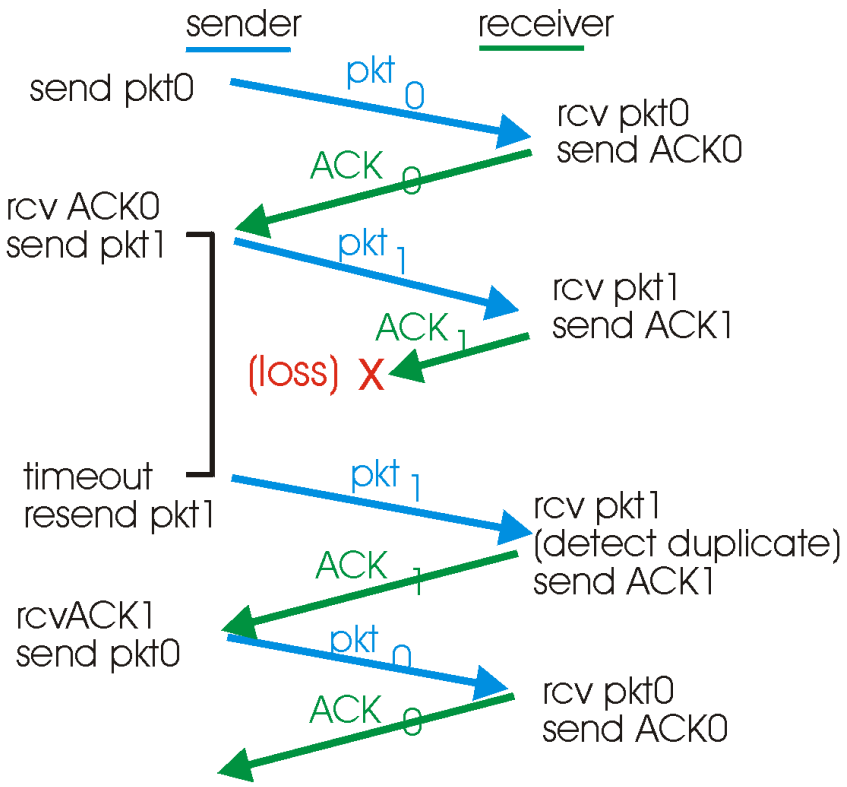


(a) operation with no loss

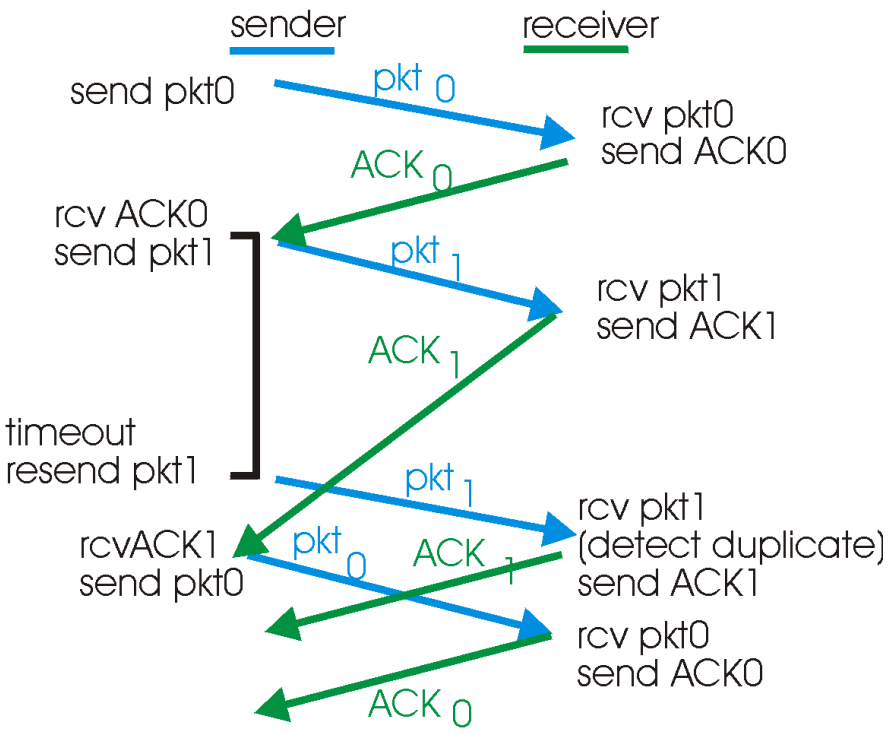


(b) lost packet

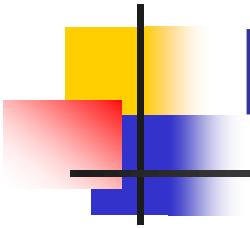
S&W Example (cont.)



(c) lost ACK



(d) premature timeout



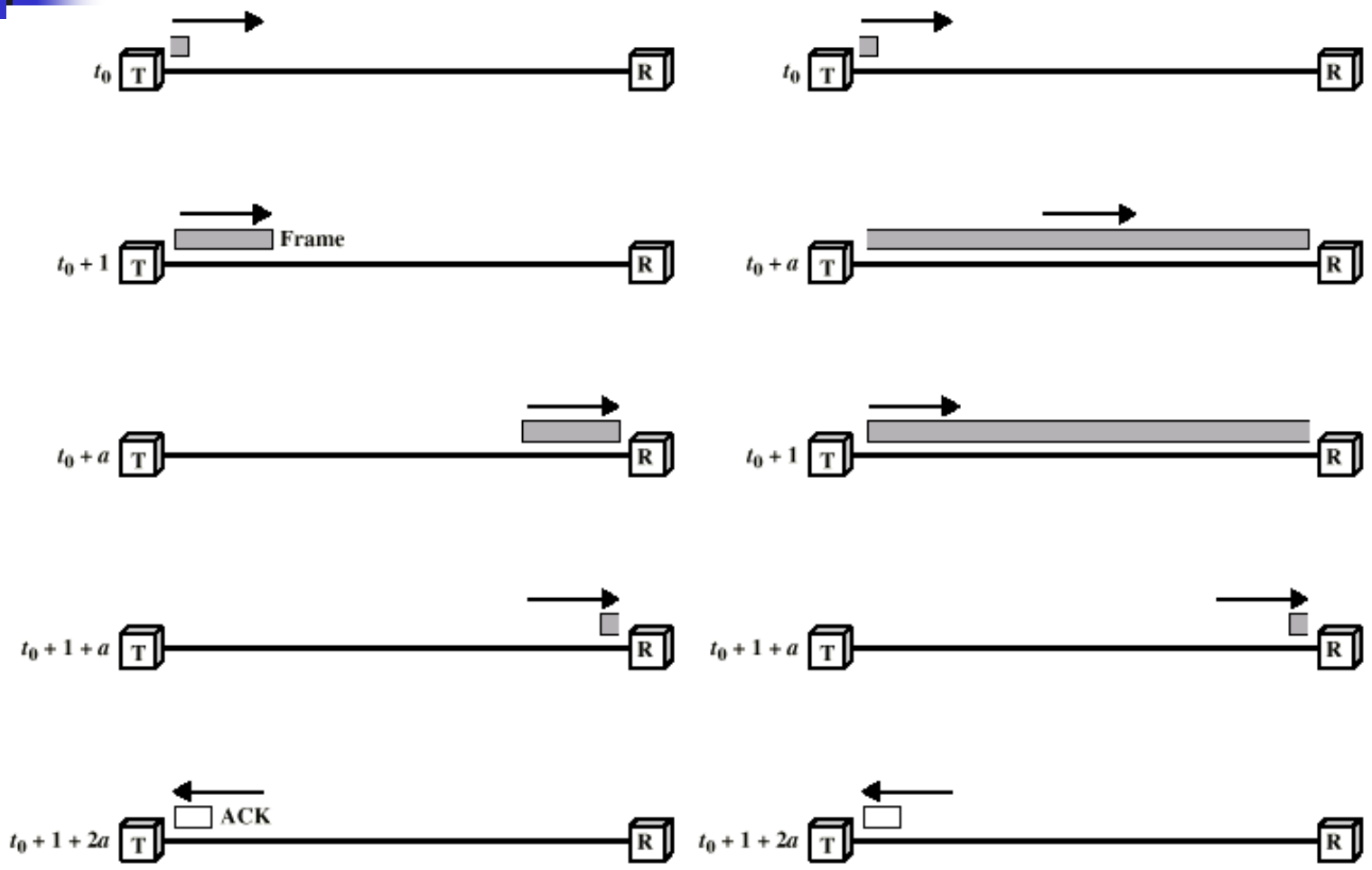
Problem with Stop and Wait

- Work well for a few large frames, but become inadequate, when
  - One small frame at a time
  - If a>1, S&W has inefficient link utilization

    where a = propagation delay/transmission delay

- Solution: pipelining!

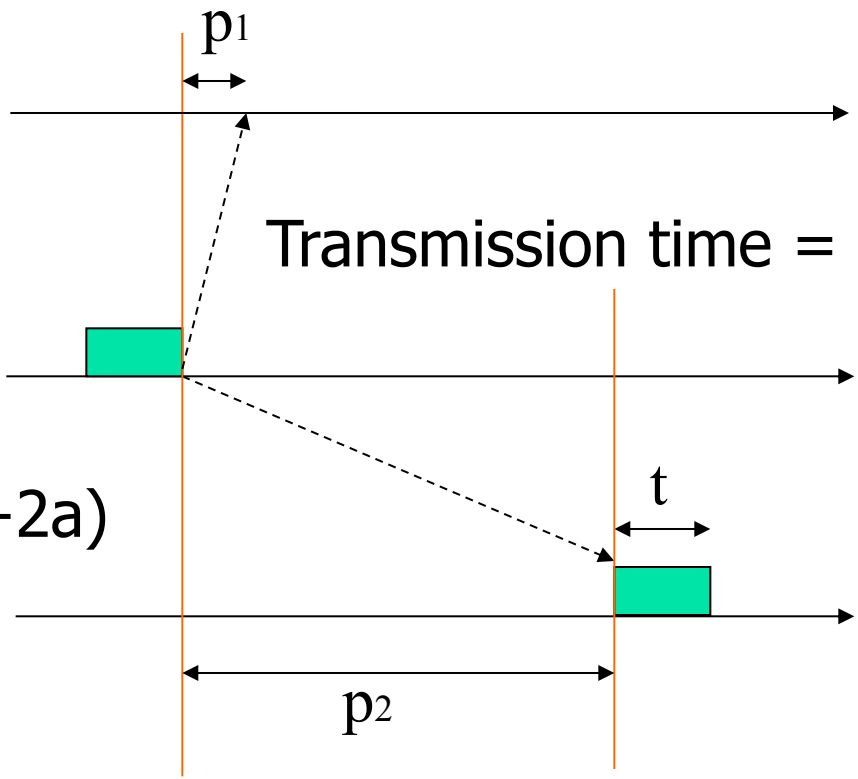
Stop and Wait Link Utilization



a>1

a<1

Stop and Wait Link Utilization



U = 1/(1+2a)

Propagation delay = distance / light speed

S&W Utilization Revisited

- Example-1:
  - (ATM cell) Data: 424 bits
  - Data rate: 155.52 Mbps

    t = 424/(155.52x10^6) = 2.7x10^-6 sec
  - 10^6 meters fiber optics

    p = 10^6/(2x10^8) = 0.5x10^-2 sec

    a = p/t ~ 1850

U = 1/(1+2a) = 0.00027

S&W Utilization Revisited

- Example-2:
  - (Ethernet frame) Data: 1000 bits
  - Data rate: 10 Mbps ~ 1Gbps

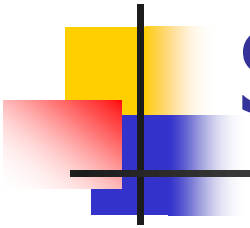
    t = 1000/(10x10^6) = 10^-4 sec
  - 0.1 to 10 km fiber optics

    p = 1000/(2x10^8) = 0.5x10^-5 sec

    a = p/t ~ 0.05

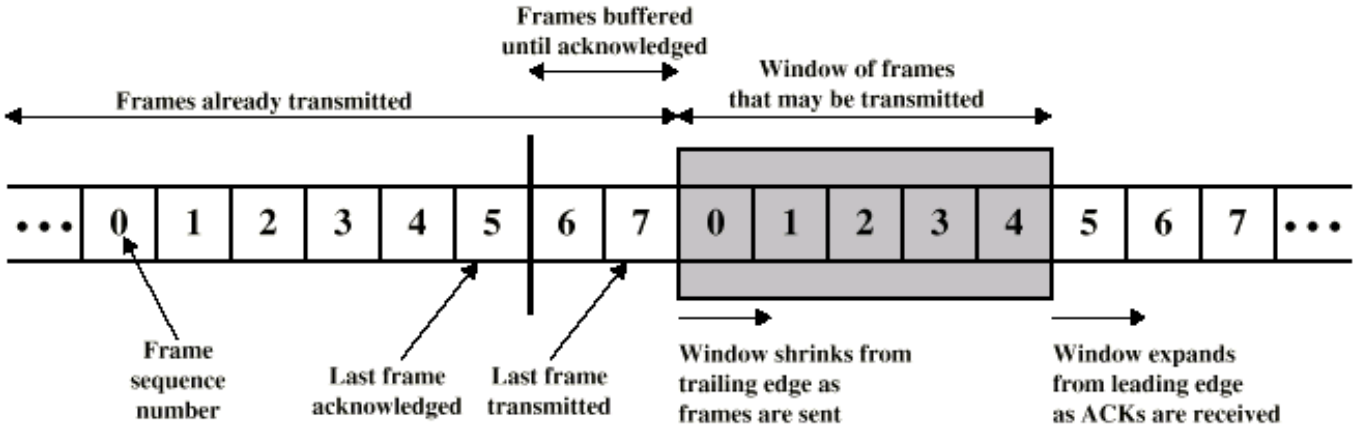
U = 1/(1+2a) = 0.91

Sliding Window Mechanism

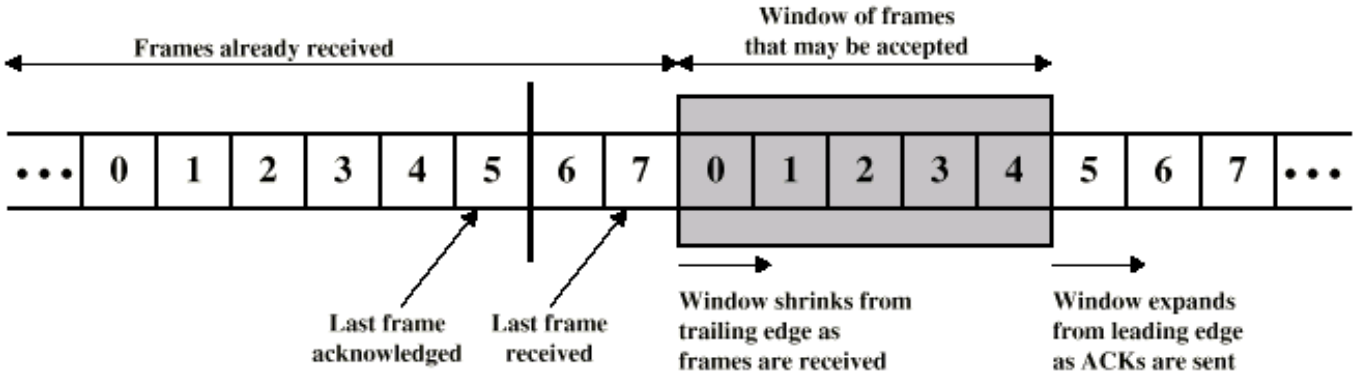


- Allow multiple frames to be in transit → improve S&W's low utilization as a > 1
- Source can send up to W frames without ACK
- Each frame is numbered and buffered at both sides
- Four intervals in a sliding window
  - Base, next_seq, W
- Sender's window vs. receiver window
- Window size vs. sequence number

Sliding Window Diagram

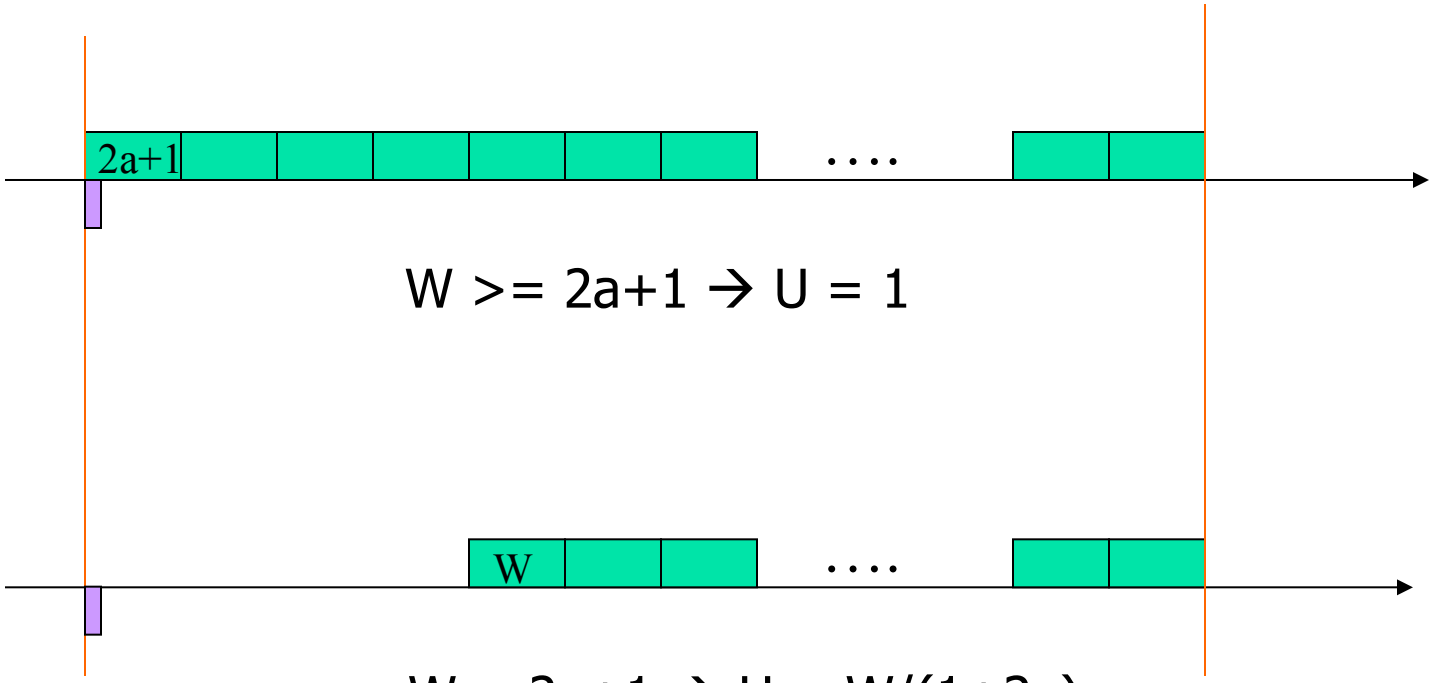


(a) Sender's perspective



(b) Receiver's perspective

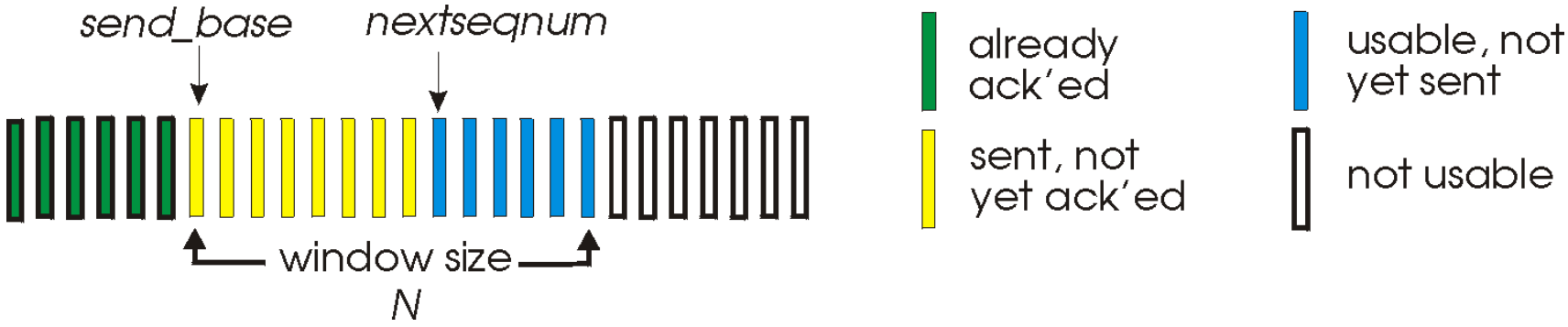
Sliding Window Link Utilization



Go Back N (GBN)

Sender:

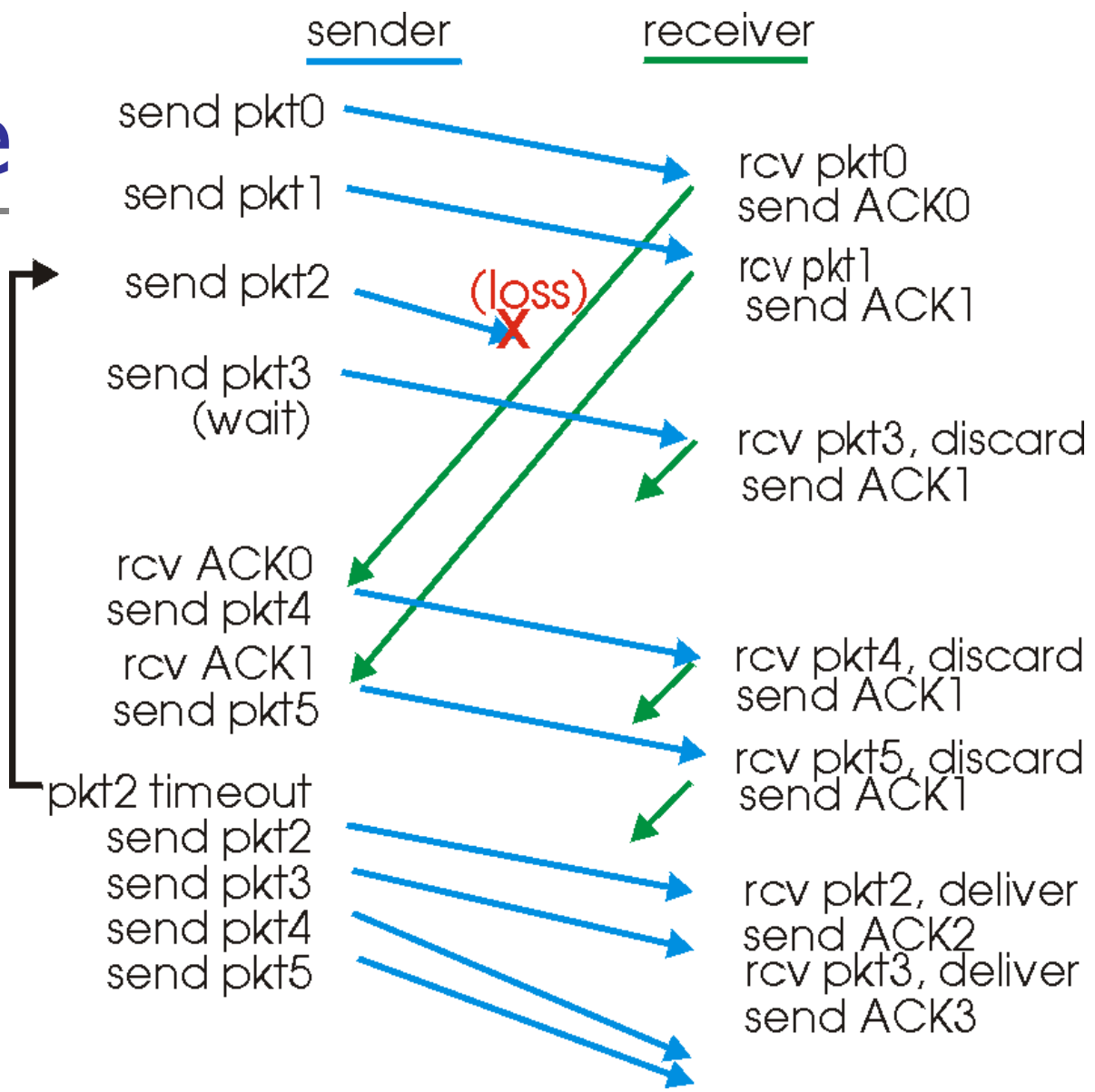
- k-bit seq # in pkt header
- "window" of up to N, consecutive unack'ed pkts allowed



Receiver:

- ACK-only: always send ACK for correctly-received pkt with highest *in-order* seq #
- Discard out-of-order packets -> no buffering

GBN Example


sender
receiver
send pkt0
send pkt1
send pkt2
(loss)
send pkt3
(wait)
rcv ACK0
send pkt4
rcv ACK1
send pkt5
pkt2 timeout
send pkt2
send pkt3
send pkt4
send pkt5
rcv pkt0
send ACK0
rcv pkt1
send ACK1
rcv pkt3, discard
send ACK1
rcv pkt4, discard
send ACK1
rcv pkt5, discard
send ACK1
rcv pkt2, deliver
send ACK2
rcv pkt3, deliver
send ACK3

Selective Repeat (SR)

sender

data from above :

- if next available seq # in window, send pkt

timeout(n):

- resend pkt n, restart timer

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

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

receiver

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

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

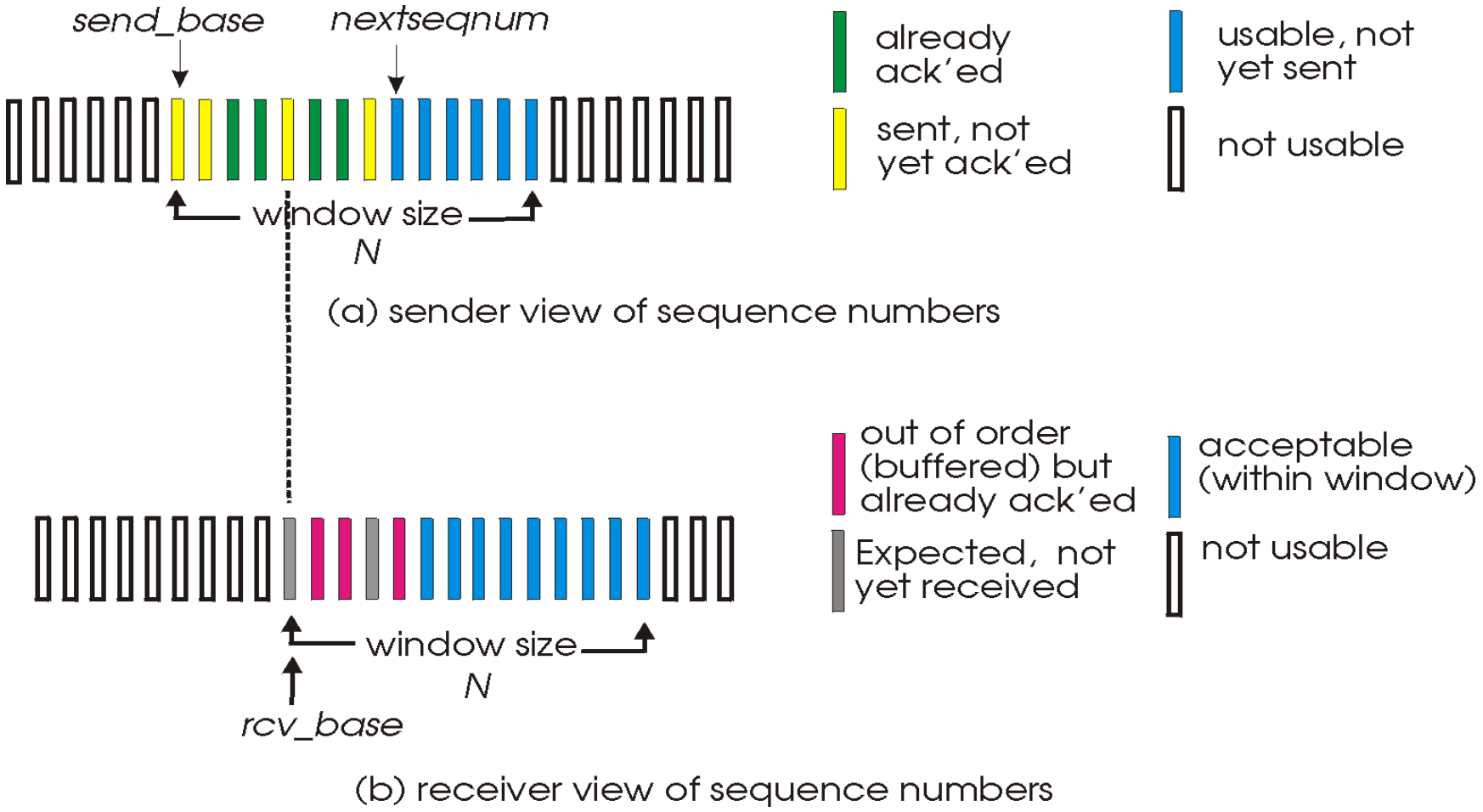
pkt n in [rcvbase-N,rcvbase-1]

- ACK(n)

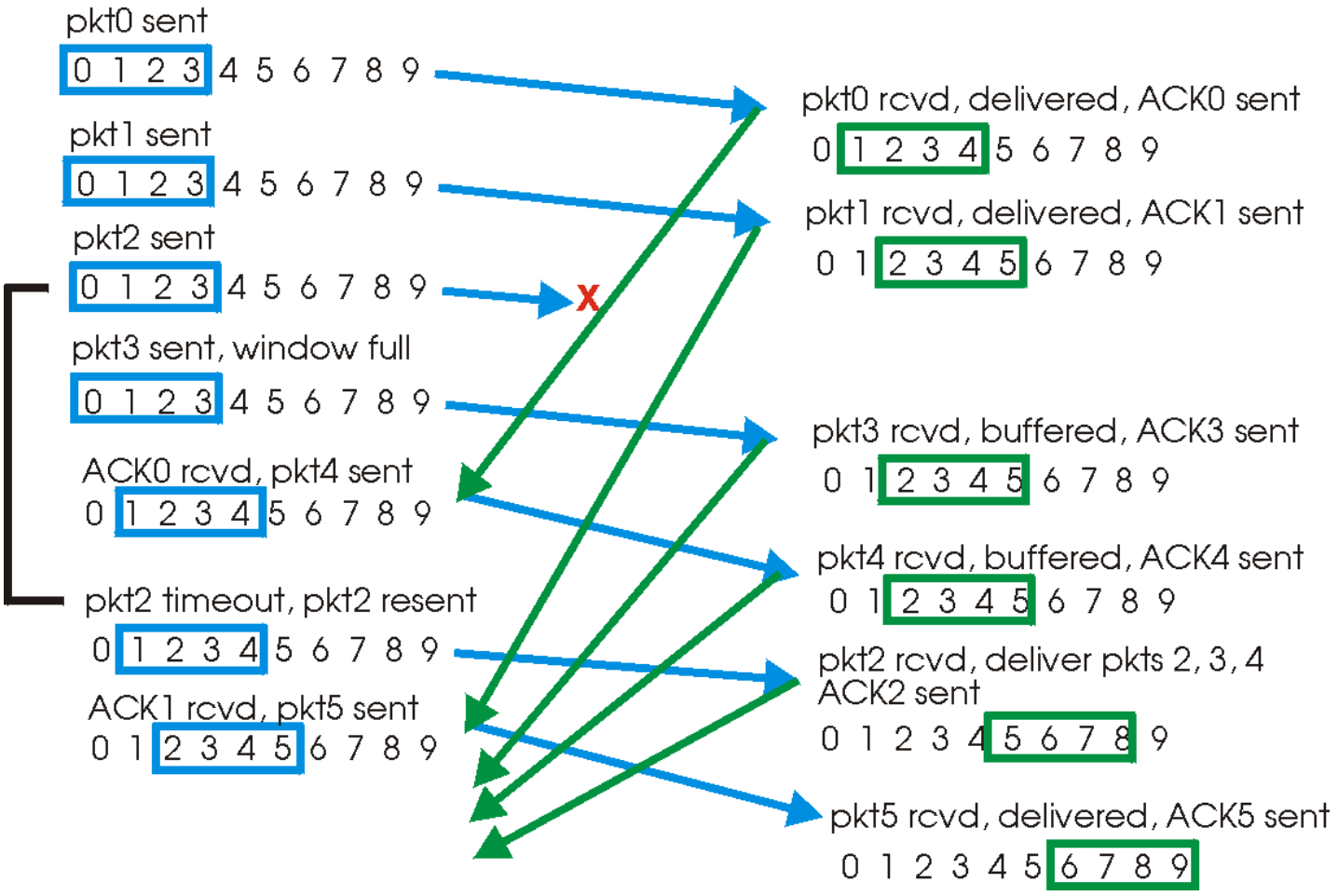
otherwise:

- ignore

SR: Sender and Receiver's Windows



SR Example


pkt0 sent
0 1 2 3 4 5 6 7 8 9
pkt1 sent
0 1 2 3 4 5 6 7 8 9
pkt2 sent
0 1 2 3 4 5 6 7 8 9
X
pkt3 sent, window full
0 1 2 3 4 5 6 7 8 9
ACK0 rcvd, pkt4 sent
0 1 2 3 4 5 6 7 8 9
pkt2 timeout, pkt2 resent
0 1 2 3 4 5 6 7 8 9
ACK1 rcvd, pkt5 sent
0 1 2 3 4 5 6 7 8 9
pkt0 rcvd, delivered, ACK0 sent
0 1 2 3 4 5 6 7 8 9
pkt1 rcvd, delivered, ACK1 sent
0 1 2 3 4 5 6 7 8 9
pkt3 rcvd, buffered, ACK3 sent
0 1 2 3 4 5 6 7 8 9
pkt4 rcvd, buffered, ACK4 sent
0 1 2 3 4 5 6 7 8 9
pkt2 rcvd, deliver pkts 2, 3, 4
ACK2 sent
0 1 2 3 4 5 6 7 8 9
pkt5 rcvd, delivered, ACK5 sent
0 1 2 3 4 5 6 7 8 9

SR Dilemma

Example:

- seq #'s: 0, 1, 2, 3
- window size=3
- receiver sees no difference in two scenarios!
- incorrectly passes duplicate data as new in (a)

Q: what relationship between seq # size and window size?