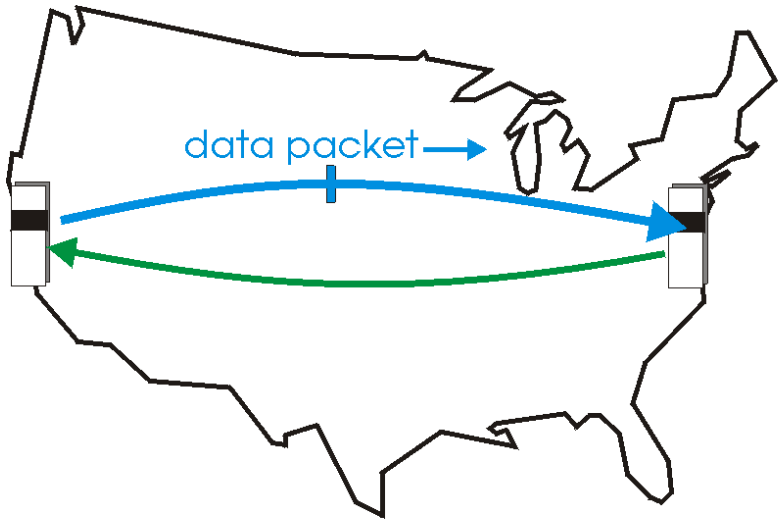
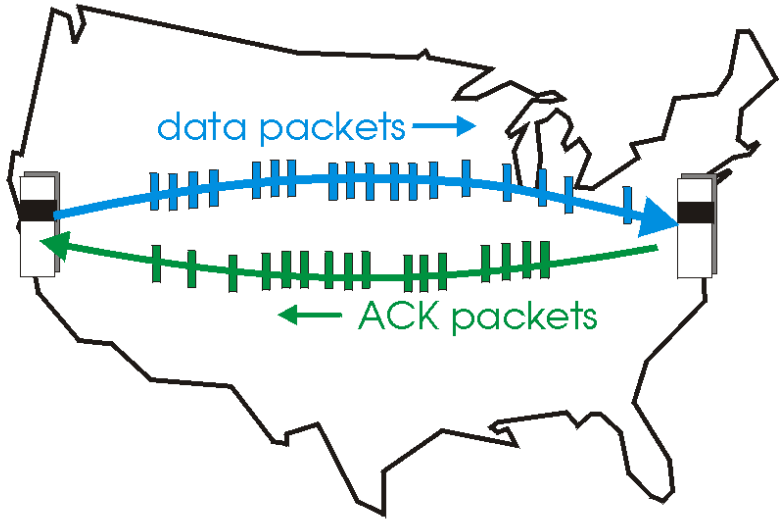
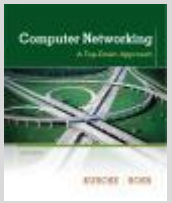
Pipelined reliable transport



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



(b) a pipelined protocol in operation



Chapter 3 outline

3.1 Transport-layer services

3.2 Multiplexing and demultiplexing

3.3 Connectionless transport: UDP

3.4 Principles of reliable data transfer

3.5 Connection-oriented transport: TCP

- Segment structure
- Reliable data transfer
- Flow control
- Connection management

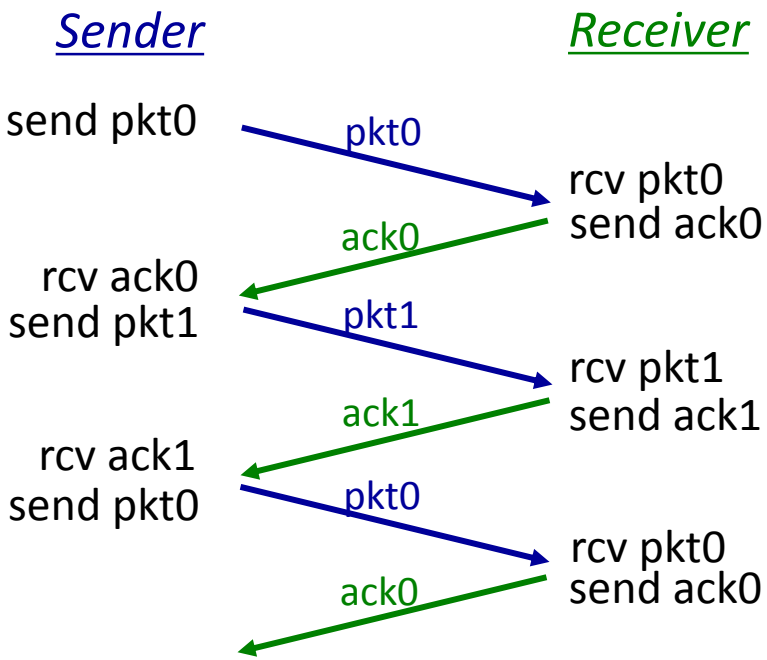
3.6 Principles of congestion control

3.7 TCP congestion control

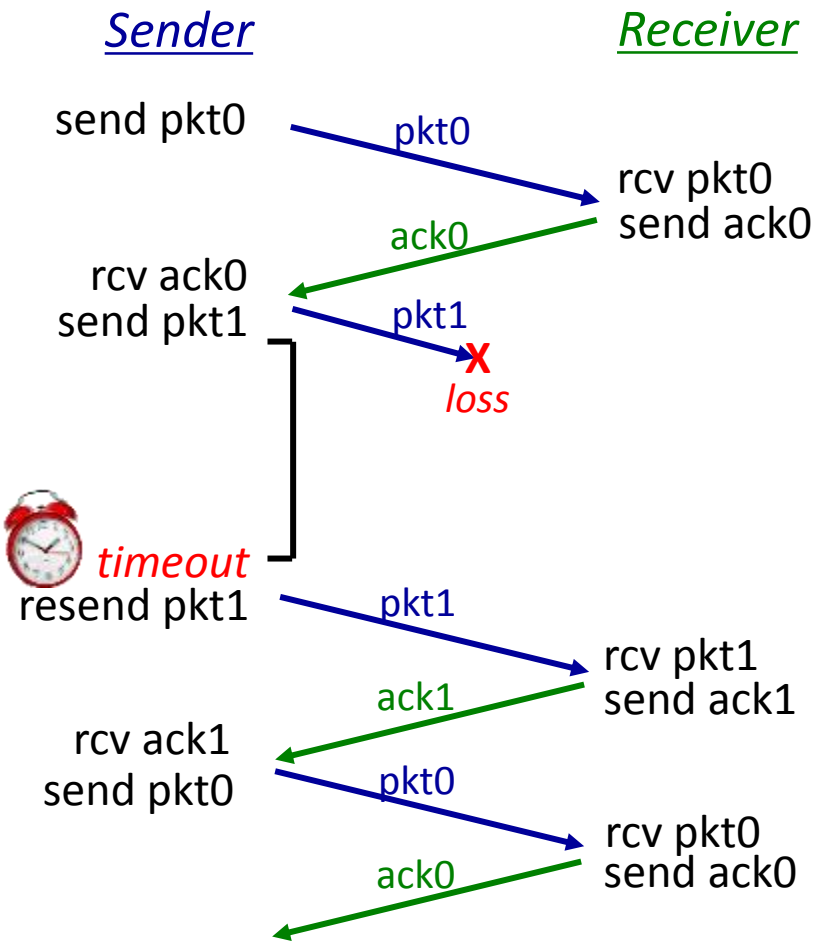
Overview

- Reliable data transfer
  - Real networks: packets get corrupted/lost/delayed
  - Reliability: ACKs, sequences numbers, timers
    - e.g. rdt3.0 using stop-and-wait-protocol
  - Problem: Way to slow for fat long pipes!
- Pipelined reliable data transfer
  - Good-Back-N (GBN) protocol
  - Selective Repeat (SR) protocol

rdt3.0 in action

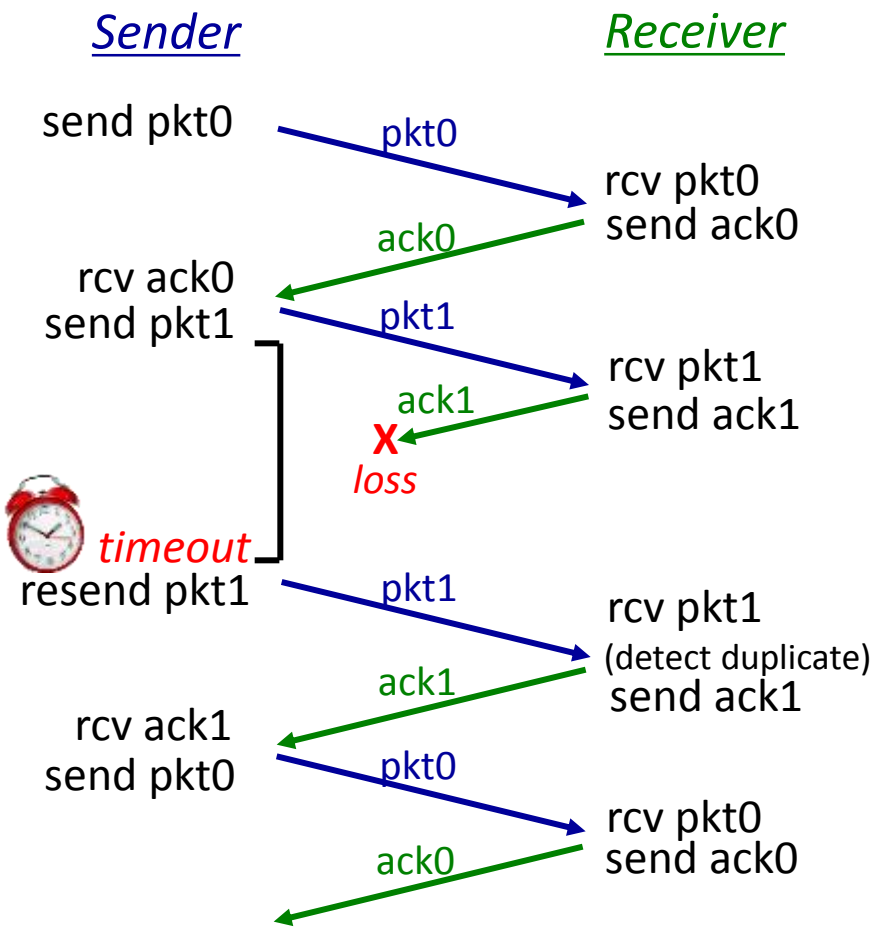


(a) no loss

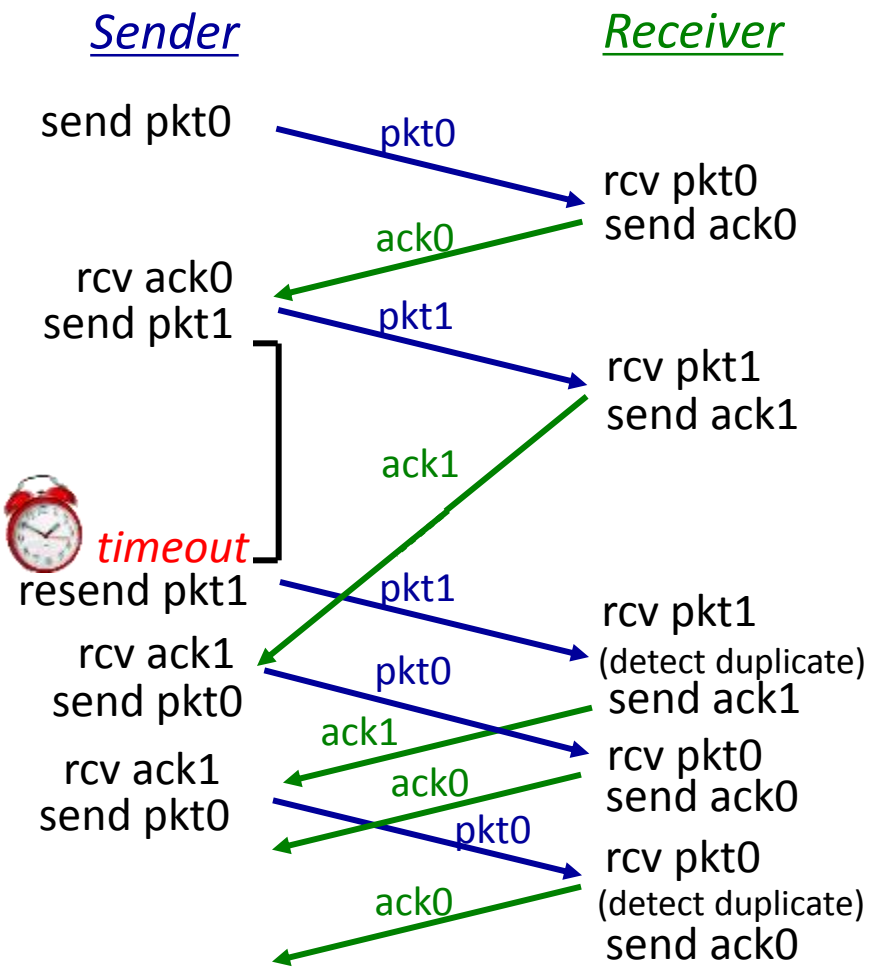


(b) packet loss

rdt3.0 in action

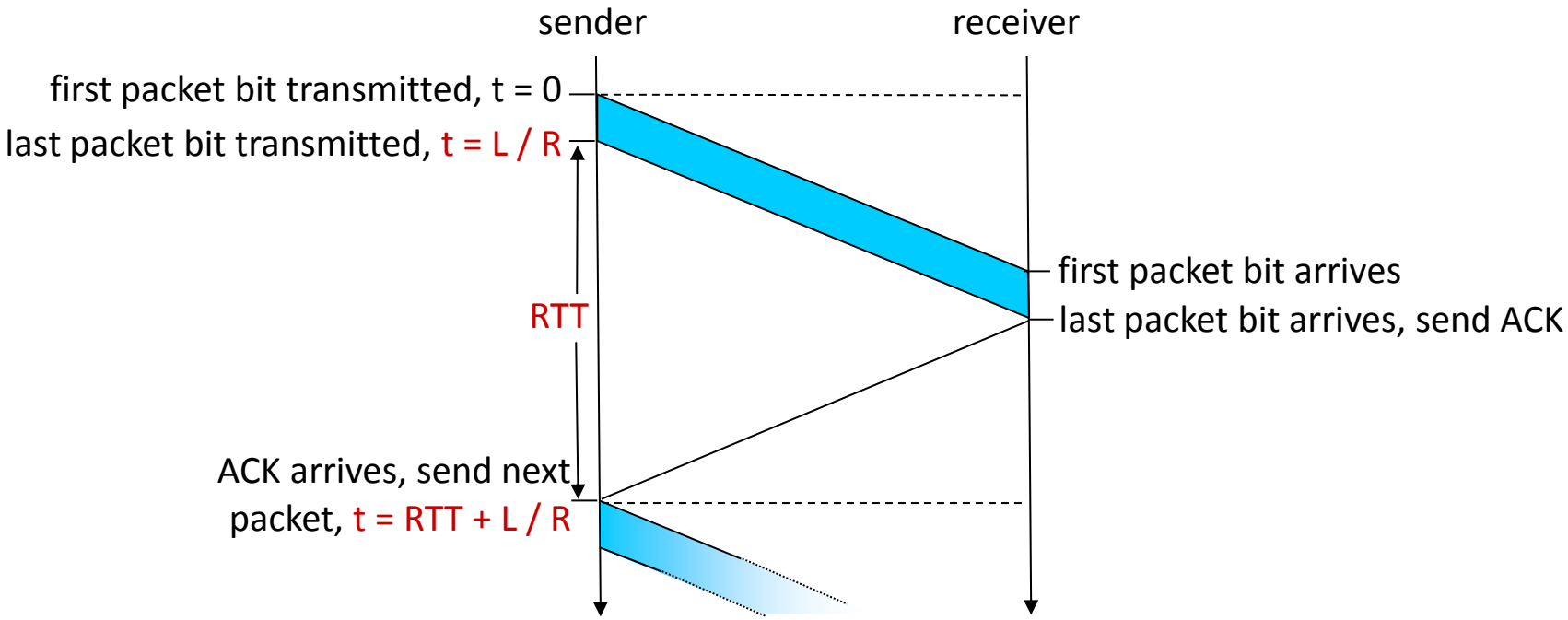


(c) ACK loss



(d) premature timeout / delayed ACK

rdt3.0: stop-and-wait operation

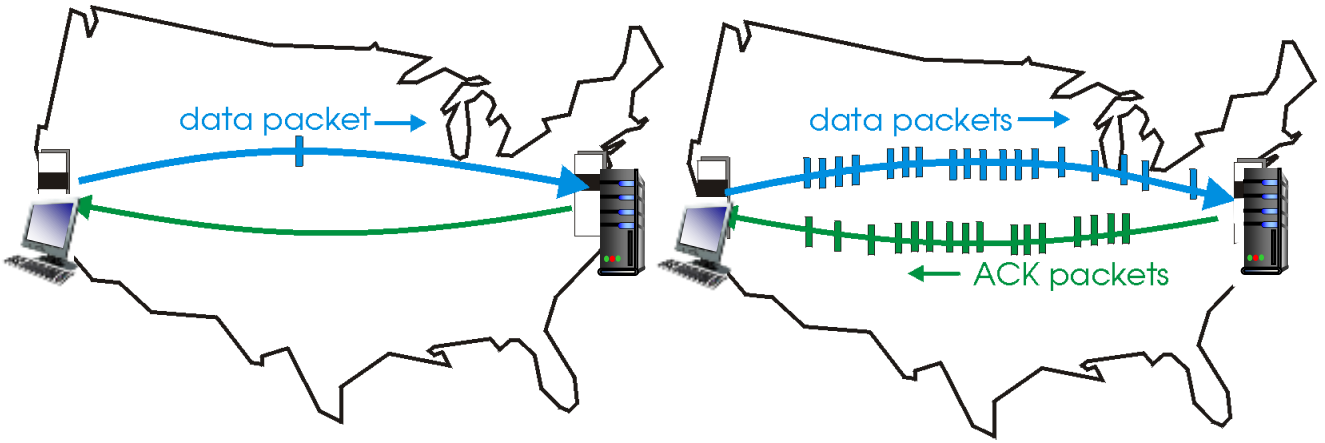


$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

Pipelined protocols

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

- Range of sequence numbers must be increased
- Buffering at sender and/or receiver



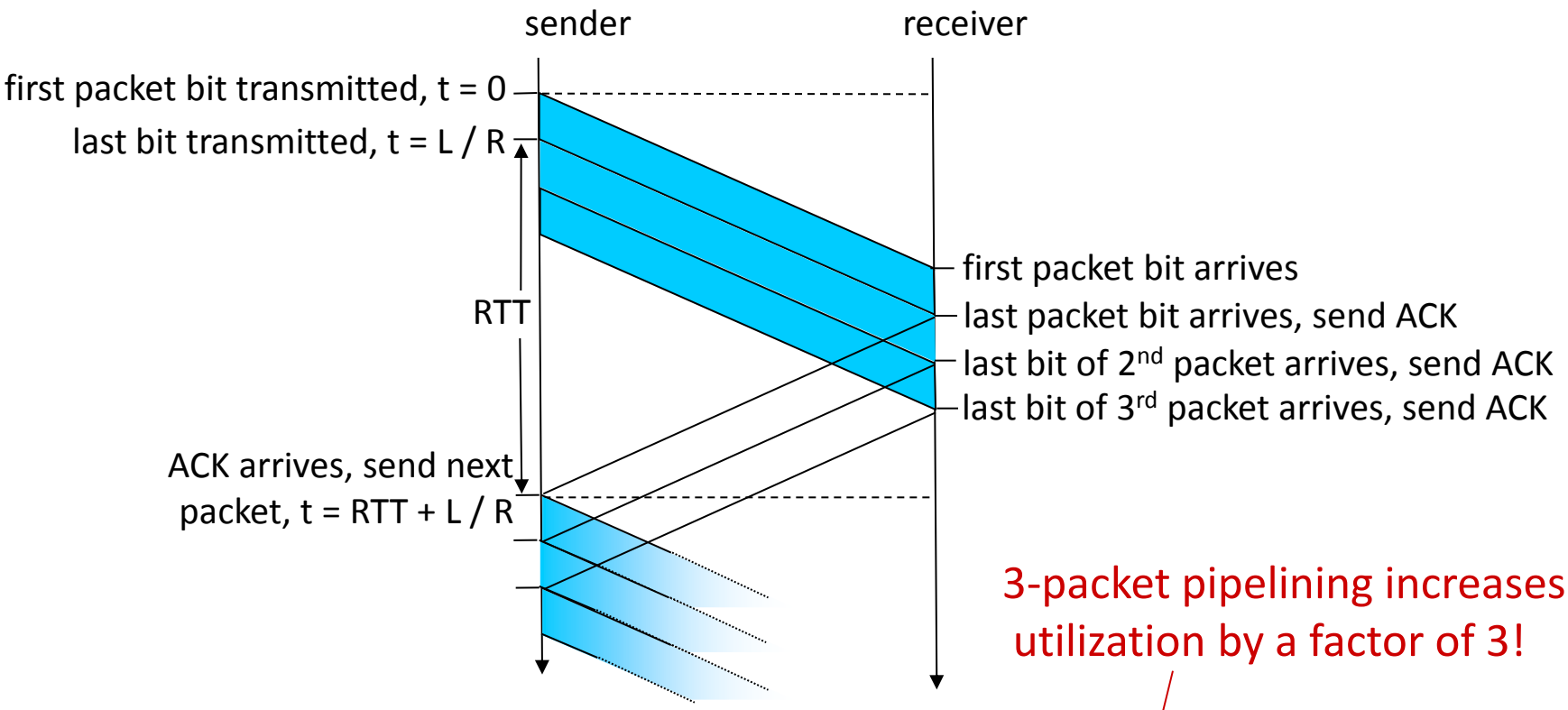
(a) a stop-and-wait protocol in operation

(b) a pipelined protocol in operation

❖ Two generic forms of pipelined protocols:

- *Go-Back-N (GBN)*
- *Selective Repeat (SR)*

Pipelining: increased utilization



3-packet pipelining increases utilization by a factor of 3!

$$U_{sender} = \frac{3L / R}{RTT + L / R} = \frac{.0024}{30.008} = 0.00081$$

Pipelined protocols: overview

Go-Back-N:

- Sender can have up to N unACKed packets in pipeline
- Receiver only sends *cumulative ACK*
  - Doesn't ACK packet if there's a gap
- Sender has timer for oldest unACKed packet
  - When timer expires, retransmit *all* unACKed packets

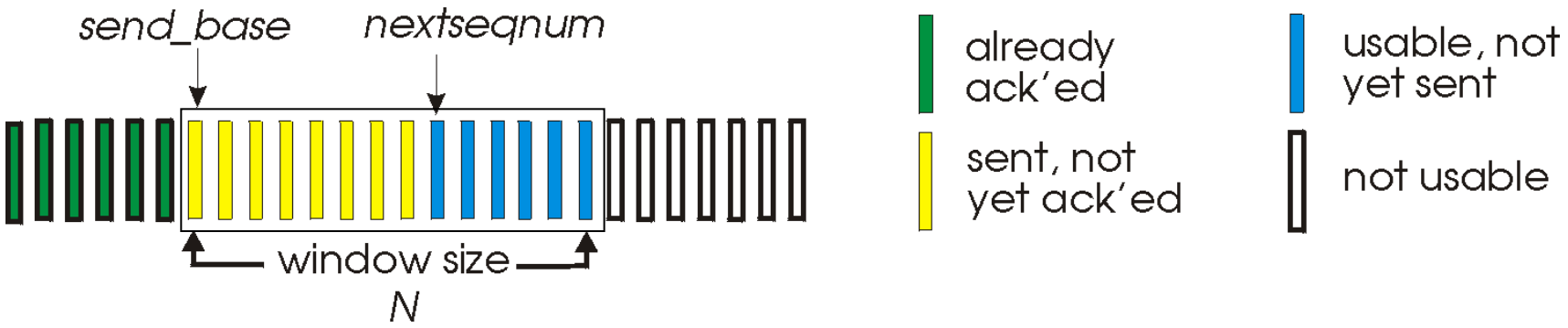
Selective Repeat:

- Sender can have up to N unACKed packets in pipeline
- Receiver sends *individual ACK* for each packet
- Sender maintains timer for each unACKed packet
  - When timer expires, retransmit only that unACKed packet

N = window size
sliding-window protocol

Go-Back-N: sender

- k-bit sequence # in packet header
- Window of up to N, consecutive unACKed packets allowed



- ACK(n): ACKs all packets up to, including sequence # n
  - *Cumulative ACK*
  - May receive duplicate ACKs (see receiver)
- Timer for oldest in-flight packet
- *timeout(n):* retransmit packet n and all higher sequence # packets in window

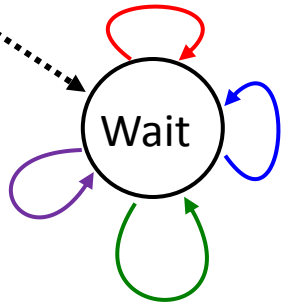
GBN sender

```
rdt_send(data)
if (nextseqnum < base+N) {
   sndpkt[nextseqnum] = make_pkt(nextseqnum,data,chksum)
   udt_send(sndpkt[nextseqnum])
   if (base == nextseqnum)
      start_timer
   nextseqnum++
}
else
   refuse_data(data)
```

Invocation from above. If window is not full, create and send packet. If full, return data to sender.

Λ

base=1
nextseqnum=1



rdt_rcv(rcvpkt) &&
corrupt(rcvpkt)

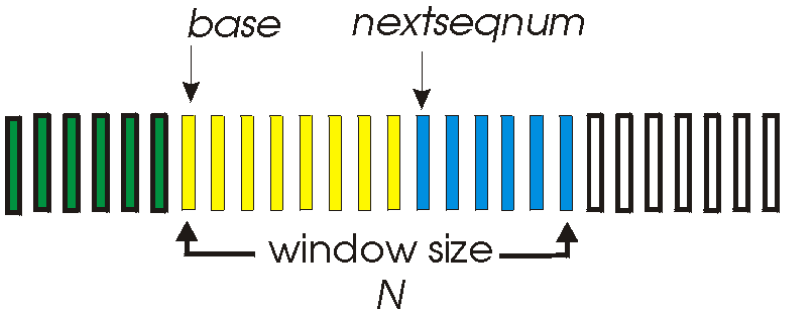
Λ

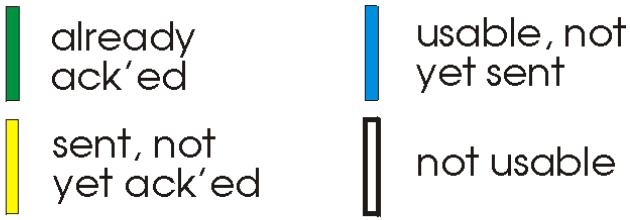
```
timeout
start_timer
udt_send(sndpkt[base])
udt_send(sndpkt[base+1])
…
udt_send(sndpkt[nextseqnum-1])
```

Timeout. Resend all packets that have been previously sent but not ACK'd.

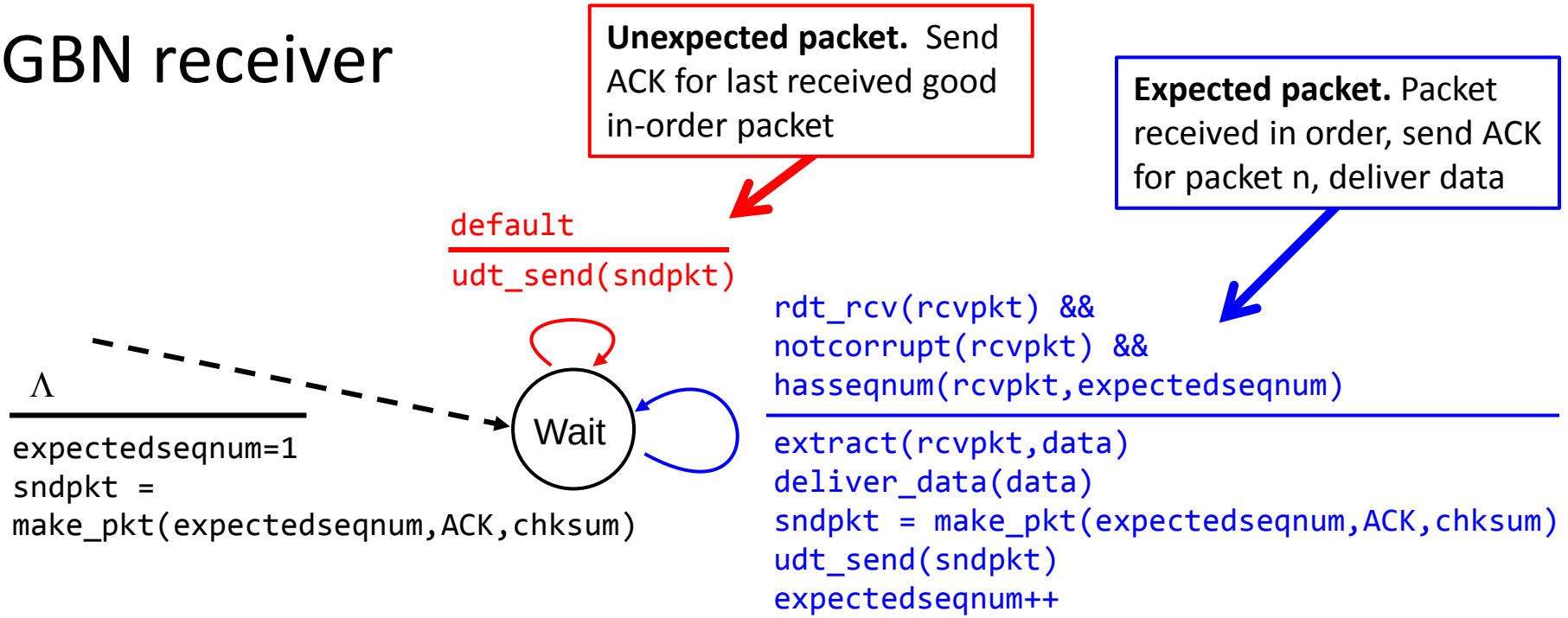
Receipt of ACK. Receipt of packet with seq. # n, all packets up to an including n are good.

```
rdt_rcv(rcvpkt) &&
notcorrupt(rcvpkt)
base = getacknum(rcvpkt) + 1
if (base == nextseqnum) stop_timer
else                    start_timer
```



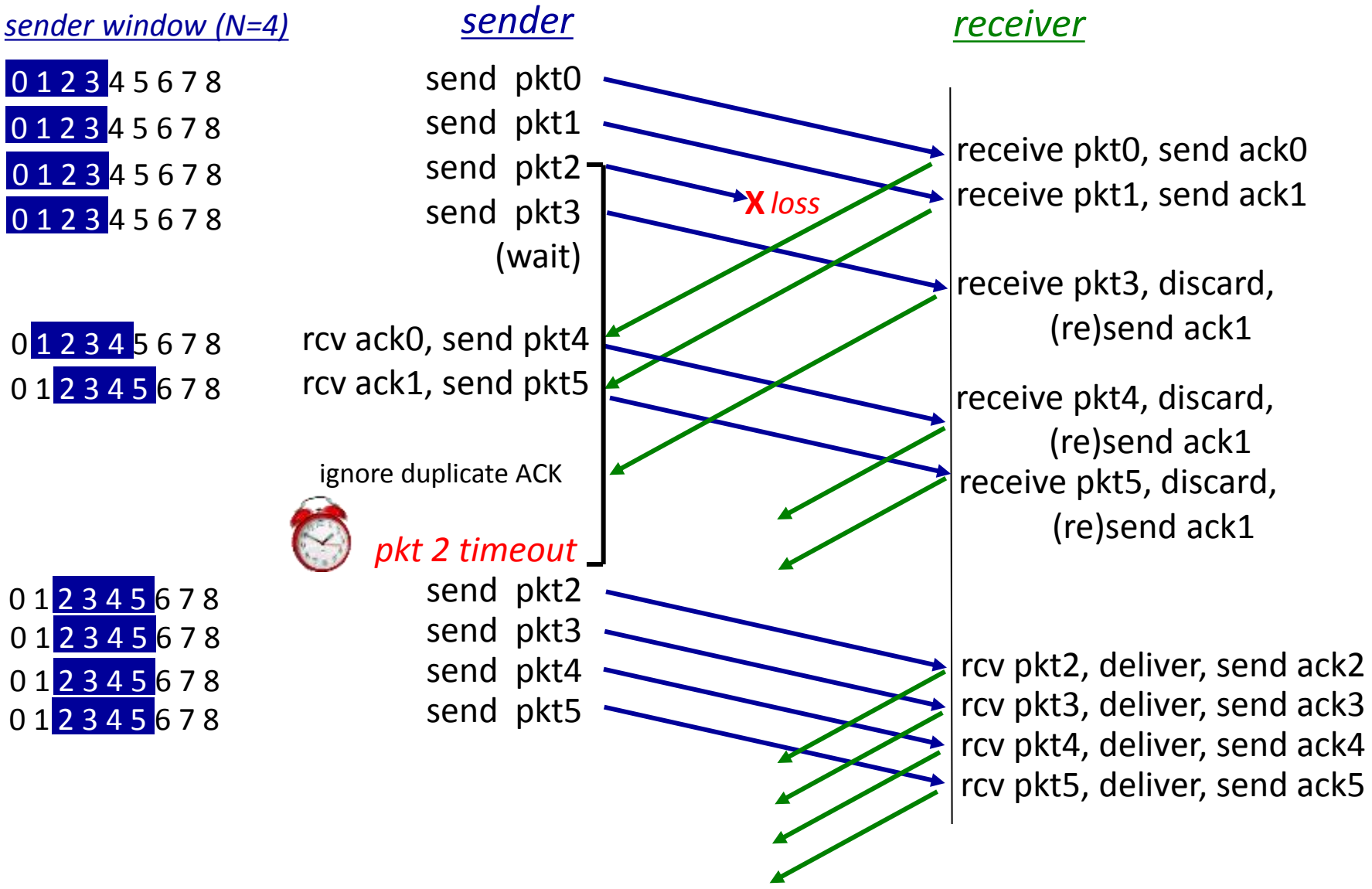


GBN receiver



- ACK-only: always send ACK for correctly-received packet with highest *in-order* sequence #
  - May generate duplicate ACKs
  - Need only remember `expectedseqnum`
- Out-of-order packet:
  - Discard (don't buffer): *No receiver buffering!*
  - Re-ACK packet with highest in-order sequence #

GBN in action

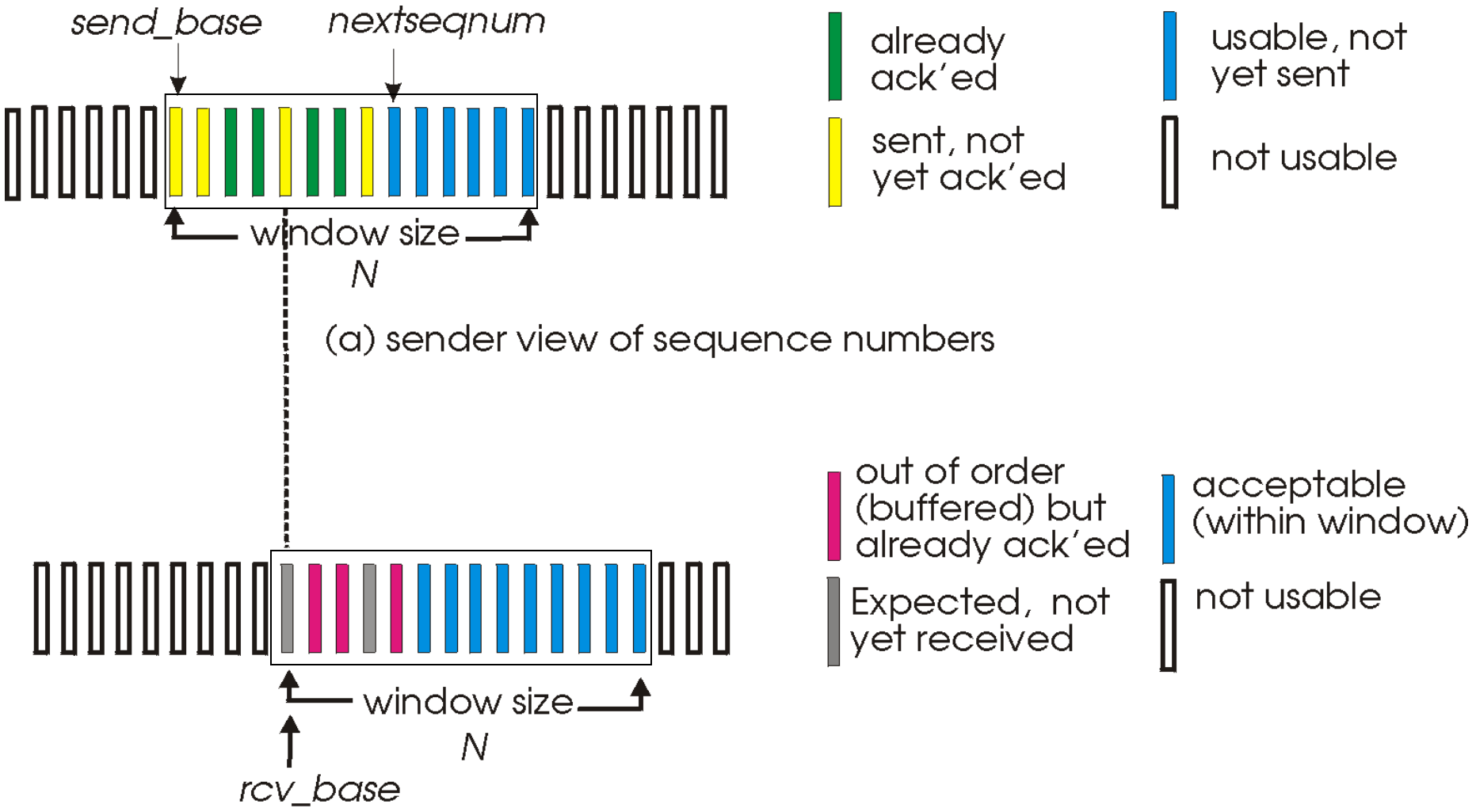


http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/go-back-n/index.html

Selective repeat

- Receiver *individually* acknowledges *all* correctly received packets
  - Buffers packets, as needed, for eventual in-order delivery to upper layer
- Sender only resends packets for which ACK not received
  - Sender has timer for each unACKed packet
- Sender window
  - *N* consecutive sequence #'s
  - Limits sequence #s of sent, unACKed packets

Selective repeat: sender, receiver, windows



Selective repeat

Sender

Data from above:

- If next available sequence # in window, send packet

Timeout(n):

- Resend packet n, restart timer

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

- Mark packet n as received
- If n smallest unACKed packet, advance window base to next unACKed sequence #

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

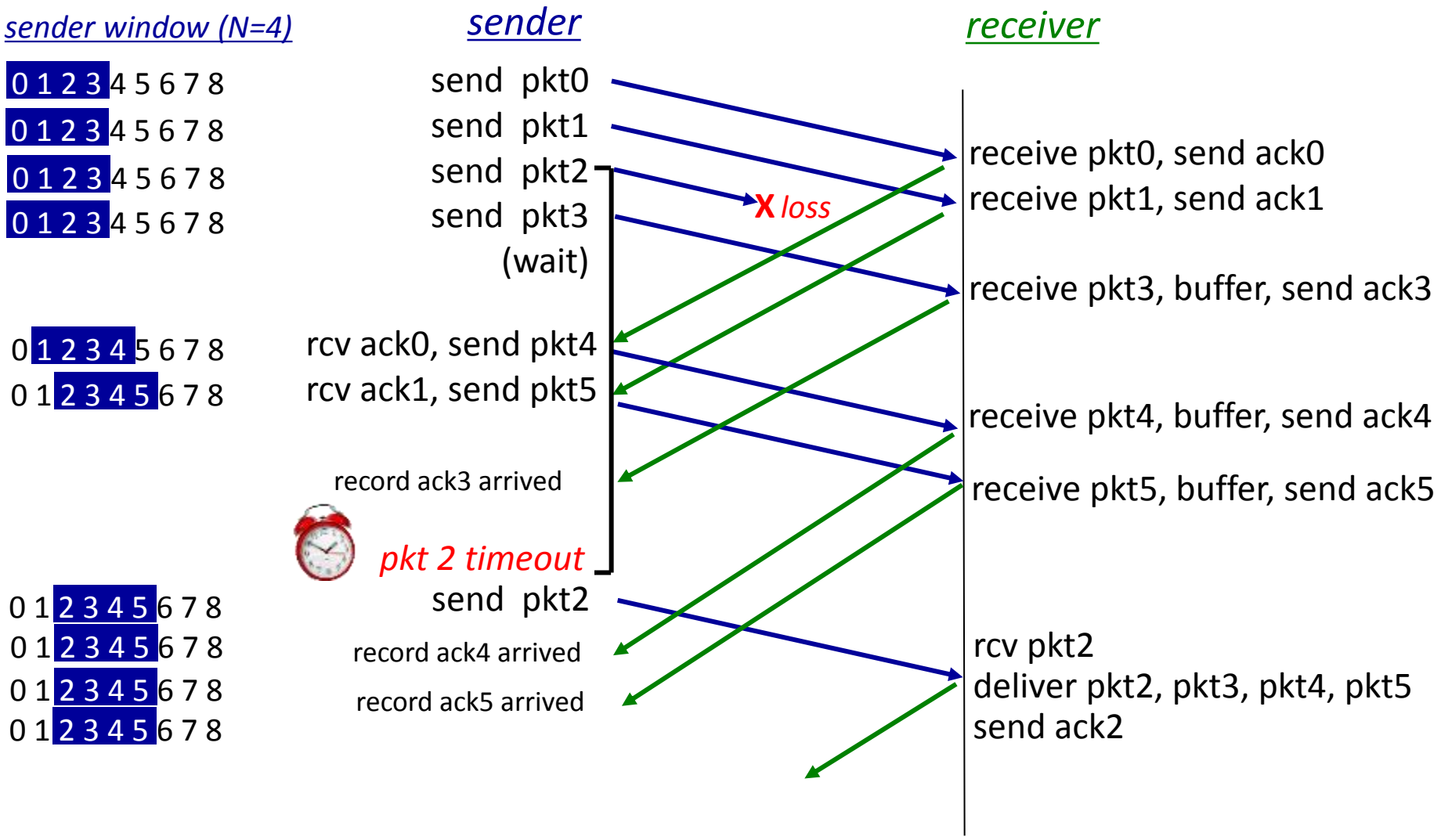
Packet n in [rcvbase-N,rcvbase-1]

- ACK(n)

Otherwise:

- Ignore

Selective repeat in action



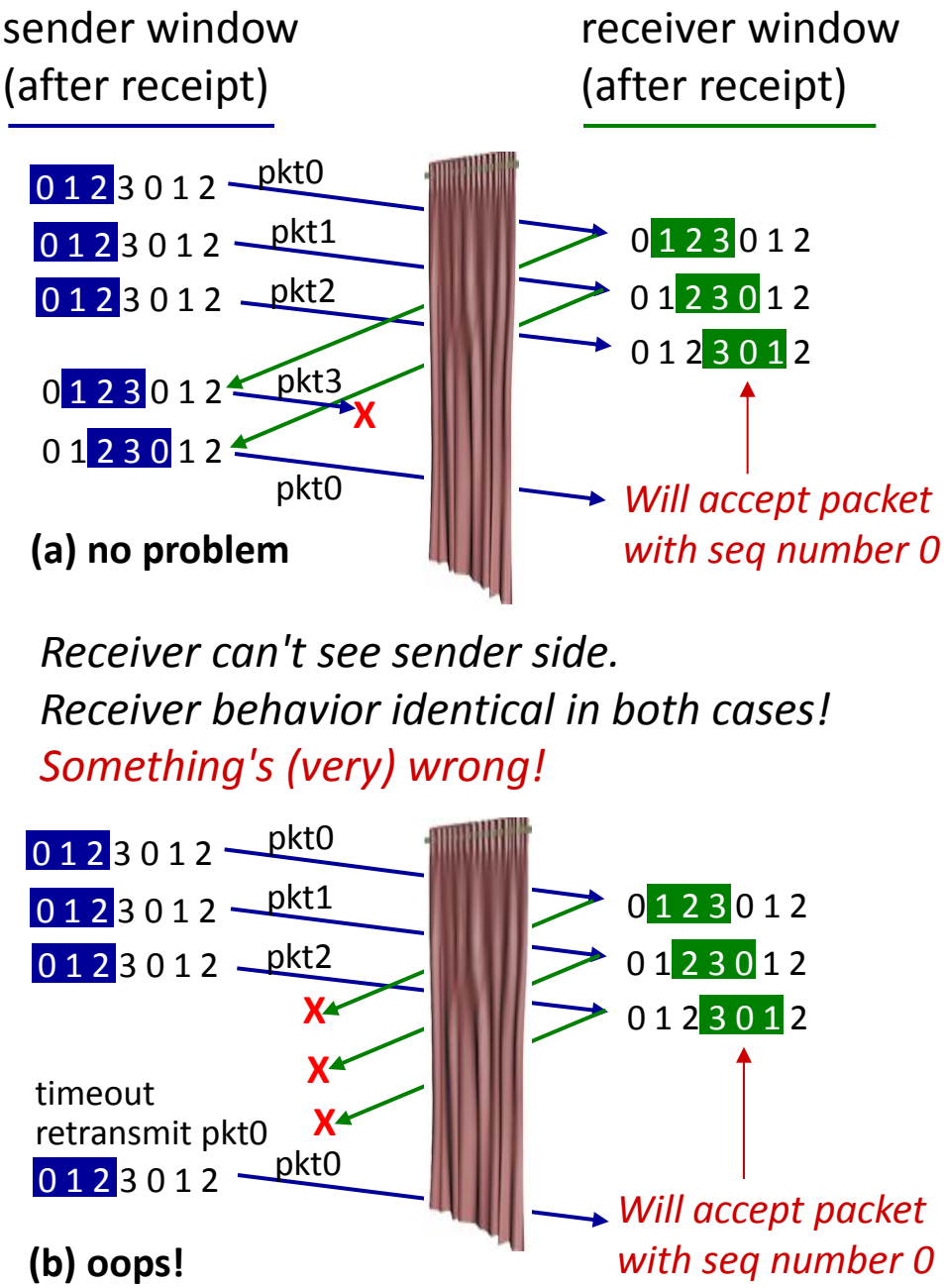
Q: What happens when ack2 arrives?

Selective repeat: dilemma

Example:

- Sequence #'s: 0, 1, 2, 3
- Window size=3
- Receiver sees no difference in two scenarios!
- Duplicate data accepted as new in (b)

Q: What relationship between sequence # size and window size to avoid problem in (b)?



http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/SR/index.html

Reliable data transport mechanisms

Checksum	Detect bit errors in a packet
Timer	Used to retransmit packet should the packet or its ACK never arrive.
Sequence number	Used to resend since packets or ACK of packet may go missing
Acknowledgment (ACK)	Used to tell sender that receiver has gotten certain packet or set of packets. Typically carries the sequence number.
Negative acknowledgment (NACK)	Used to tell sender that receiver got a corrupted packet. Typically carries a sequence number.
Window, pipelining	Sender can only send packets within a certain window of sequence numbers. Increases utilization compared to stop-and-wait protocol.

Summary

- Reliable data transport
  - Handles packet corruption, lose, or delay
  - Tricky but can be done using ACKs, sequence numbers, and timers (e.g. rdt3.0)
- Efficient reliable data transport
  - Even more complex since stop-and-wait too slow
  - Multiple unACKed packets in flight
  - Go-Back-N
  - Selective repeat