ELEN 4017

Network Fundamentals

Lecture 16

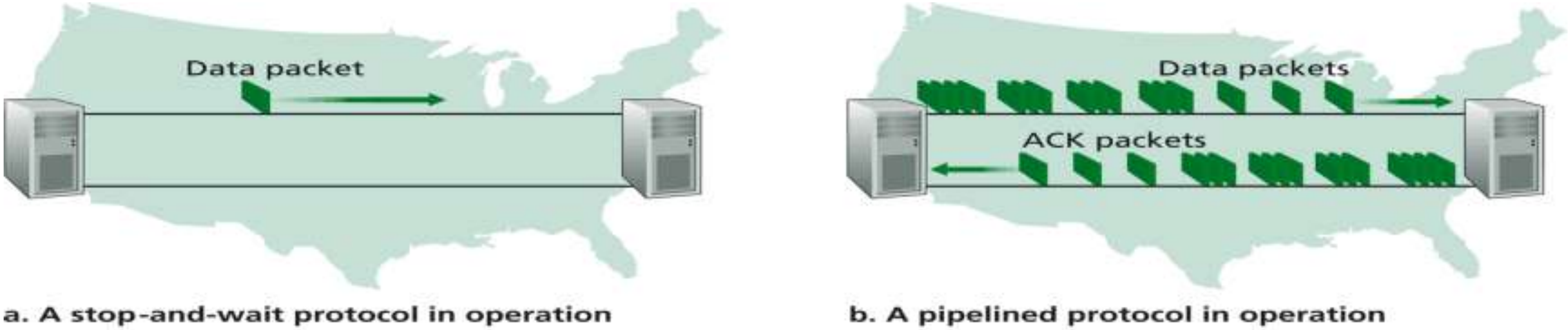
Purpose of lecture

Chapter 3: Transport Layer

- Pipelining
- Go-Back-N protocol

Reliable data transfer

- Last lecture focused on FSM design of stop-and-wait reliable data transfer protocols.
- To improve the throughput we look at pipelining i.e. sending multiple messages back-to-back.



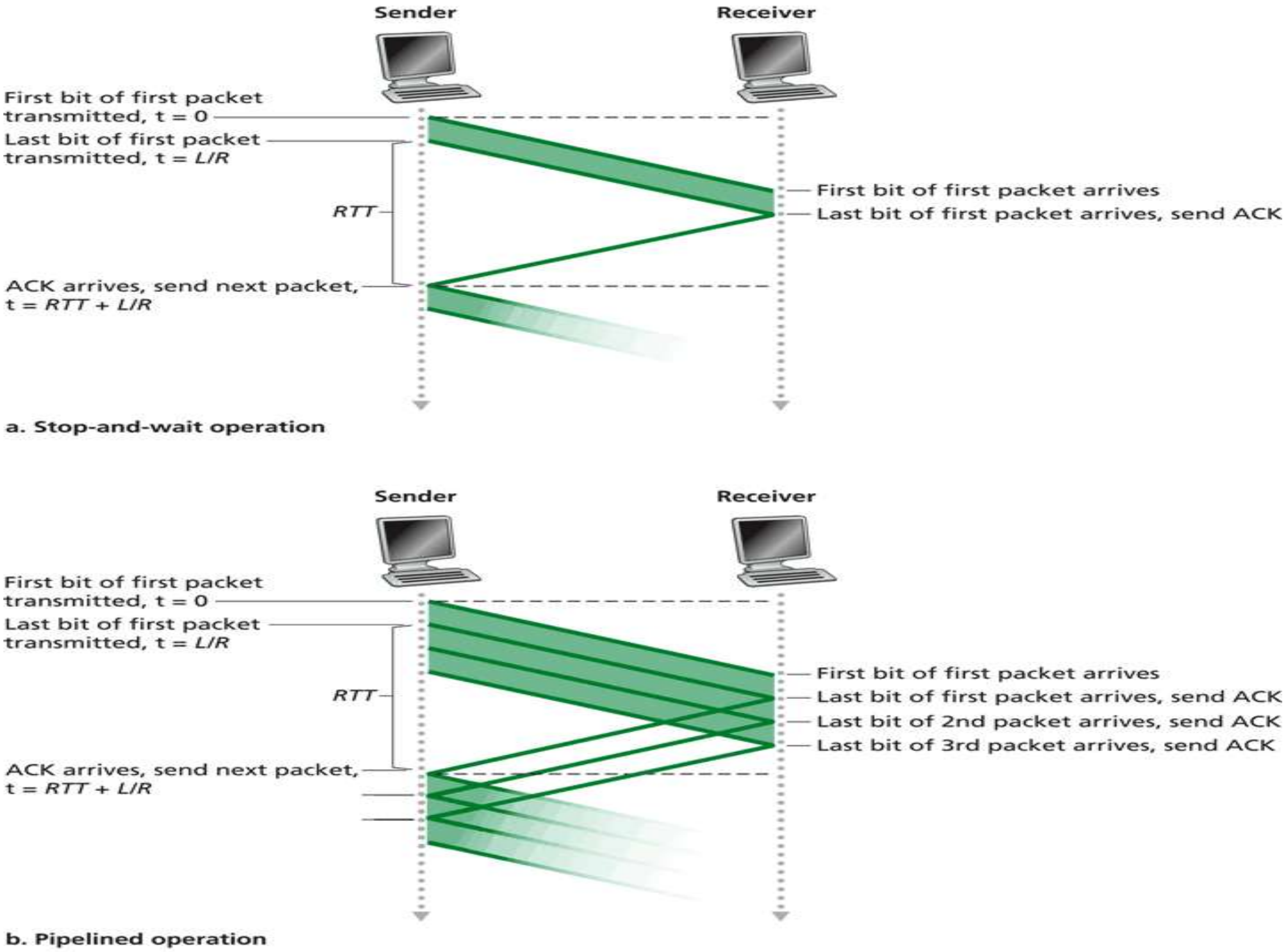
a. A stop-and-wait protocol in operation

b. A pipelined protocol in operation

Figure 3.17 ◆ Stop-and-wait versus pipelined protocol

Pipelining

- Pipelining requires the following enhancements to our protocol
  - Increase range of sequence numbers since each in-transit packet must have a unique sequence number.
  - Sender and receiver must buffer more than 1 packet. Sender must buffer packets which have been sent but have not yet been acknowledged.
  - The number of sequence numbers needed as well as buffer allocation depends on how lost, corrupted and overly delayed packets are handled.
- 2 approaches are considered:
  - Go-Back-N and Selective Repeat


Sender
Receiver
First bit of first packet transmitted, t = 0
Last bit of first packet transmitted, t = L/R
RTT
First bit of first packet arrives
Last bit of first packet arrives, send ACK
ACK arrives, send next packet, t = RTT + L/R
Sender
Receiver
First bit of first packet transmitted, t = 0
Last bit of first packet transmitted, t = L/R
RTT
First bit of first packet arrives
Last bit of first packet arrives, send ACK
Last bit of 2nd packet arrives, send ACK
Last bit of 3rd packet arrives, send ACK
ACK arrives, send next packet, t = RTT + L/R

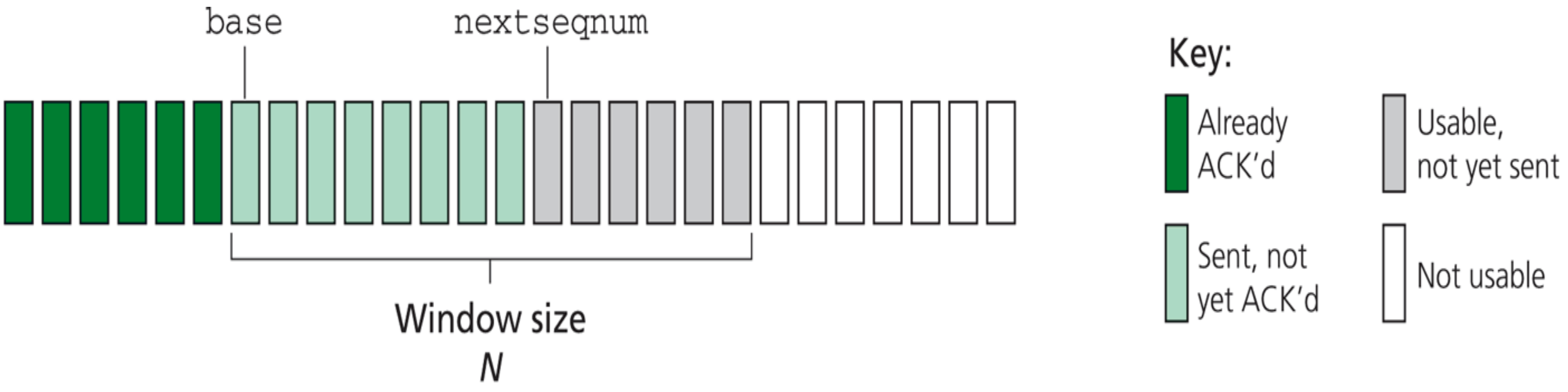
a. Stop-and-wait operation

b. Pipelined operation

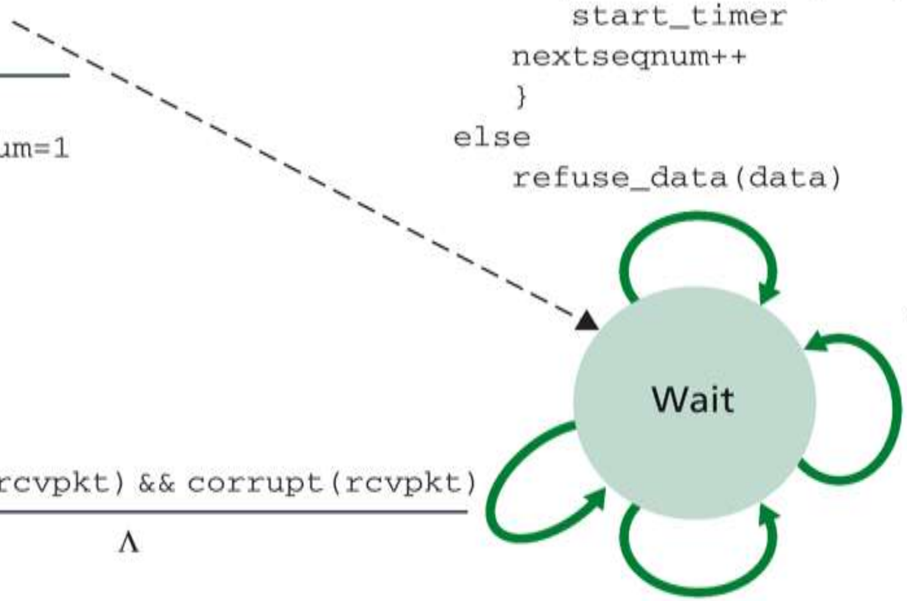
Figure 3.18 ◆ Stop-and-wait and pipelined sending

Go-Back-N (GBN) / Sliding window

- Sender is allowed to transmit multiple packets without waiting for an acknowledgement, but is constrained to have no more than some maximum allowable number, N, of unacknowledged packets in the pipeline.
- Definitions:
  - base = seq no of oldest unacknowledged packet
  - nextseqnum = smallest unused sequence number
- 4 intervals can be defined:
  - [0, base -1] → sent and acknowledged
  - [base, nextseqnum -1] → sent but not acknowledged
  - [nextseqnum, base +N-1 → can be used to sent packets immediately if they arrive from higher layer.
  - [base + N, …] → cannot be used until more acknowledgements are received.



Extended FSM of GBN sender



```
rdt_send(data)

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

```
Λ

base=1
nextseqnum=1
```

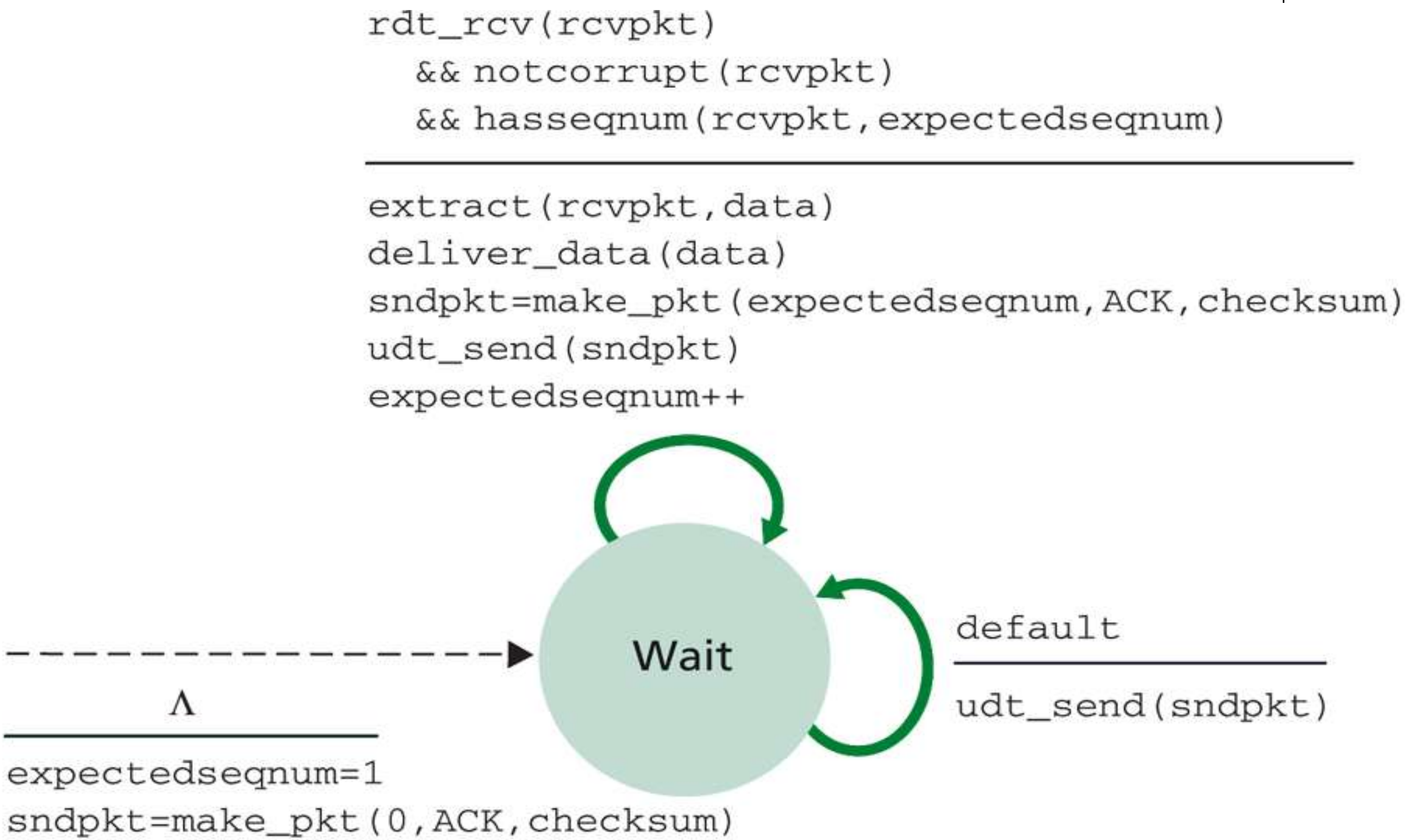
```
timeout

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

```
rdt_rcv(rcvpkt) && corrupt(rcvpkt)

Λ
```

```
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt)

base=getacknum(rcvpkt)+1
If(base==nextseqnum)
    stop_timer
else
    start_timer
```

Whats been extended in FSM ?

Extended FSM of GBN receiver



```
rdt_rcv(rcvpkt)
   && notcorrupt(rcvpkt)
   && hasseqnum(rcvpkt,expectedseqnum)
```

```
extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(expectedseqnum,ACK,checksum)
udt_send(sndpkt)
expectedseqnum++
```

```
default
```

```
udt_send(sndpkt)
```

Λ

```
expectedseqnum=1
sndpkt=make_pkt(0,ACK,checksum)
```

GBN behaviour - sender

- **Invocation from above:**
  - When rdt_send is called from above, it first checks to see if the window is full. If window is full, it blocks the application from sending.
  - On a real system the data might be buffered.
- **Cumulative acknowledgement**
  - An ACK for packet with sequence number **n** is taken as a cumulative acknowledgement of all packets with sequence number up to and including n.
- **Timeout**
  - If a timeout occurs, the sender **resends all packets** that have previously been sent but not yet acknowledged. Hence the name **Go-back-N**.
  - **Timer can be considered as a timer for the oldest unacknowledged packet**. If an ACK is received, but there are still unacknowledged packets, the timer is restarted. If all packets are acknowledged, timer is stopped.

GBN behaviour - receiver

- If a packet with sequence number n is received correctly and is in order (that is, last delivered packet to upper layer has seq no (n-1), the receiver sends an ACK for packet n and delivers data to upper layer.
- In all other cases the receiver **discards** the packet and resends an ACK for most recently received in order packet.
- **Note that packets are delivered one at a time to upper layer** → if packet k has been received and delivered, then all packets with sequence number (k-1) have also been delivered. Thus cumulative acknowledgements are a natural choice for GBN.
- **Could GBN be considered wasteful since it discards packets received out-of-order?**

Ordered delivery

- Packet n is expected but packet n+1 arrives.
- Receiver could buffer this packet, and transmit to upper layer when packet n is received.
- However if packet n is lost, both it and packet n+1 will eventually have to be retransmitted as a result of GBN retransmission rule.
- The advantage of discarding the packet is that is that the receiver does not have to do any buffering.
- Receiver only has to keep track of the next packet needed in the sequence.
- Disadvantage is that on subsequent retransmission, packet n+1 could be lost or garbled, and thus would lead to more retransmissions.

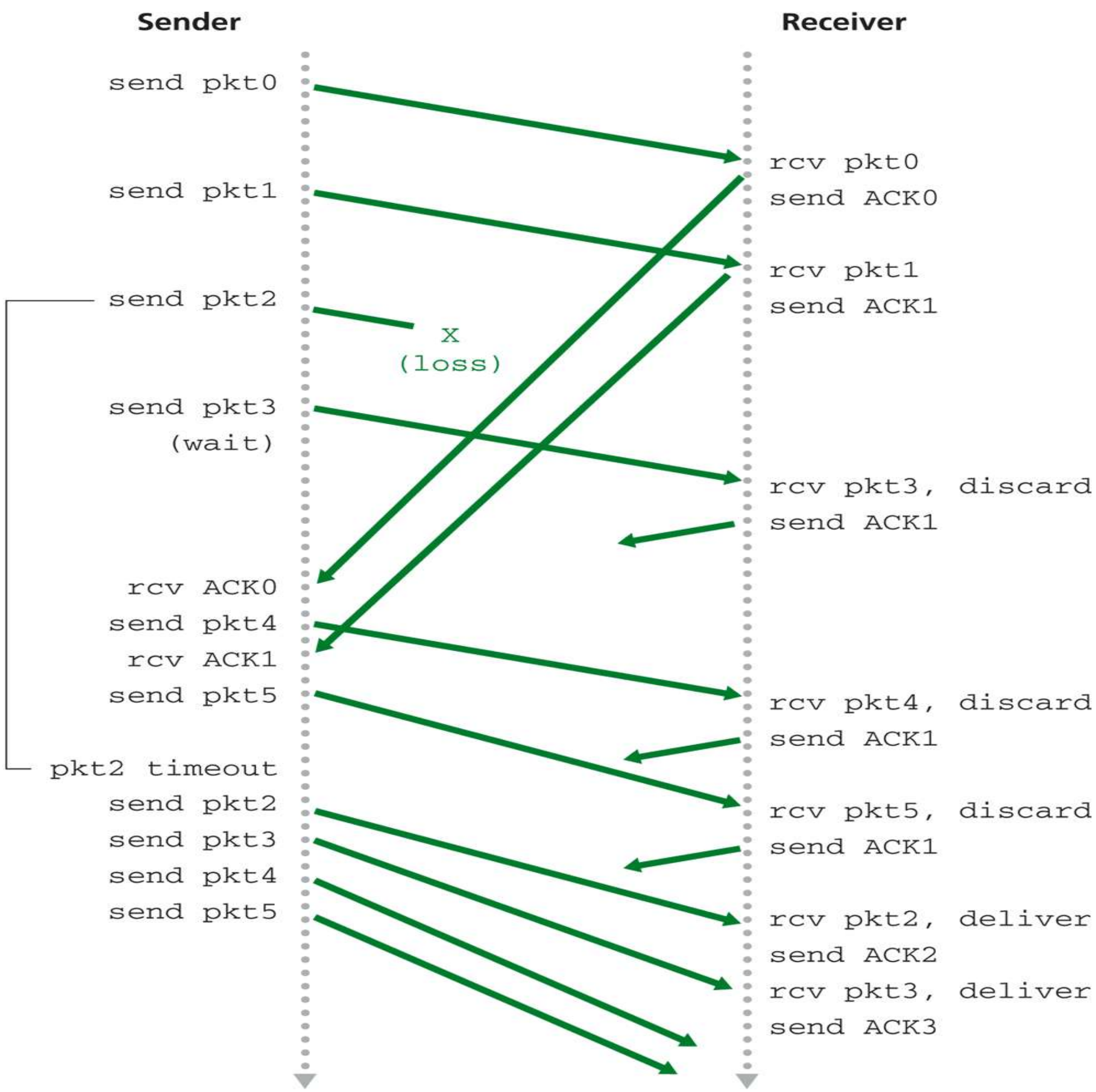


Figure 3.22 ♦ Go-Back-N in operation

Note that the ACK is sent for each packet received.
In what case would an accumulative ACK be useful?

- Applet demo.
- Do a google search for ‘Sliding Window animation’ to find applets on the Internet.
- Experiment by deleting packets, deleting ACKs, etc. to verify the behaviour of GBN.

Criticisms of GBN

- If we have a large window size and a long RTT, many packets can be in the pipeline.
- A single packet error can cause a retransmission of a large number of packets, many unnecessarily.
- As the number of errors increase, the pipeline is actually filled with retransmissions.
- Using the dictation analogy, every time a single word is garbled, the surrounding 1000 words are retransmitted.

Benefits of GBN

- Cumulative acknowledge prevents retransmissions of lost ACKS in some cases.
- Does not require too many resources on the sender (1 timer per window) or receiver (no buffering of out of order packets.