

CSCI 491-01

Topics: Internet Programming

Fall 2008

Transport Layer

Derek Leonard
Hendrix College

October 3, 2008

Original slides copyright © 1996-2007 J.F Kurose and K.W. Ross

Chapter 3: Roadmap

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

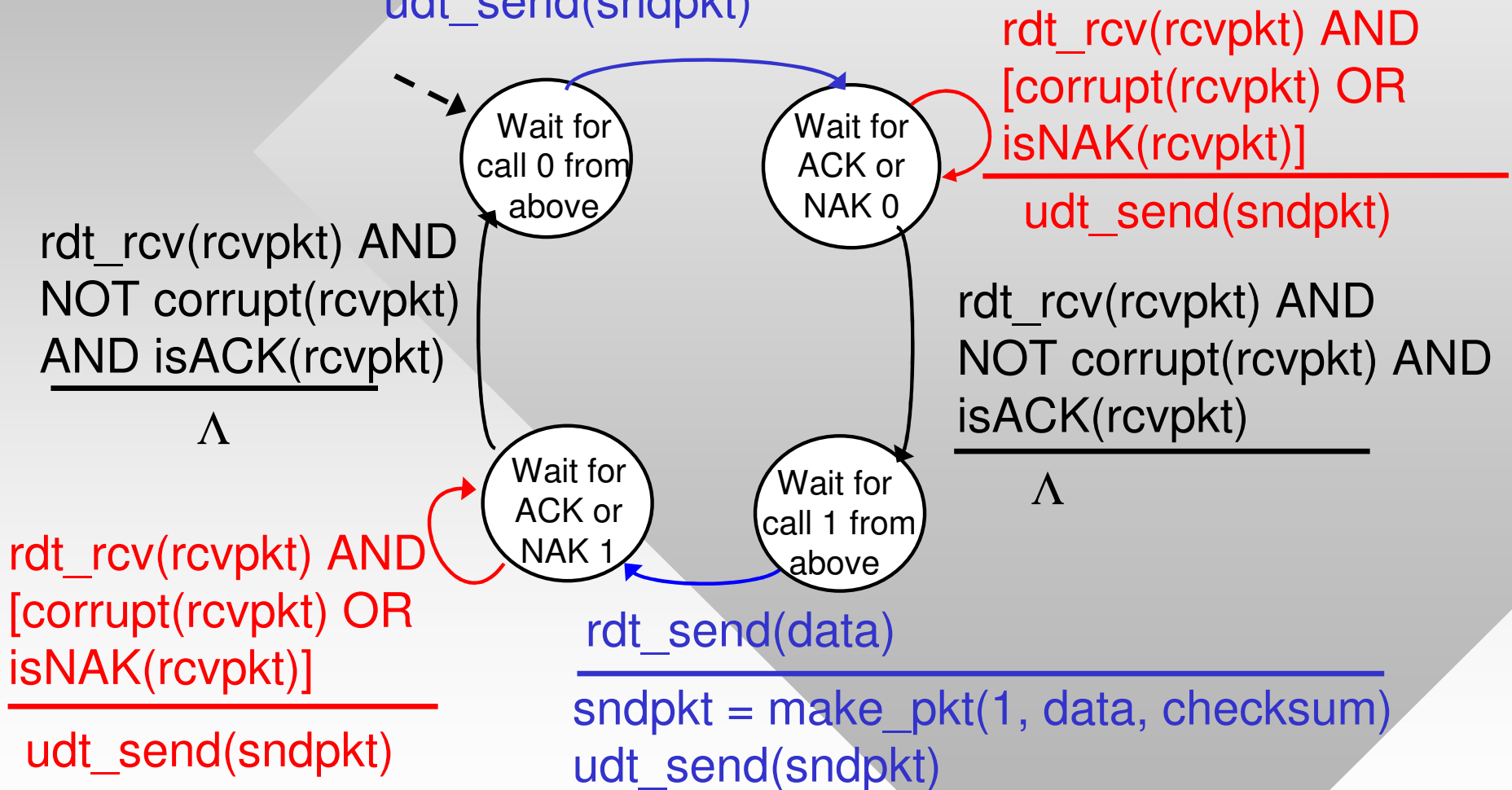
3.7 TCP congestion control

Rdt2.1: Sender, Handles Garbled ACK/NAKs

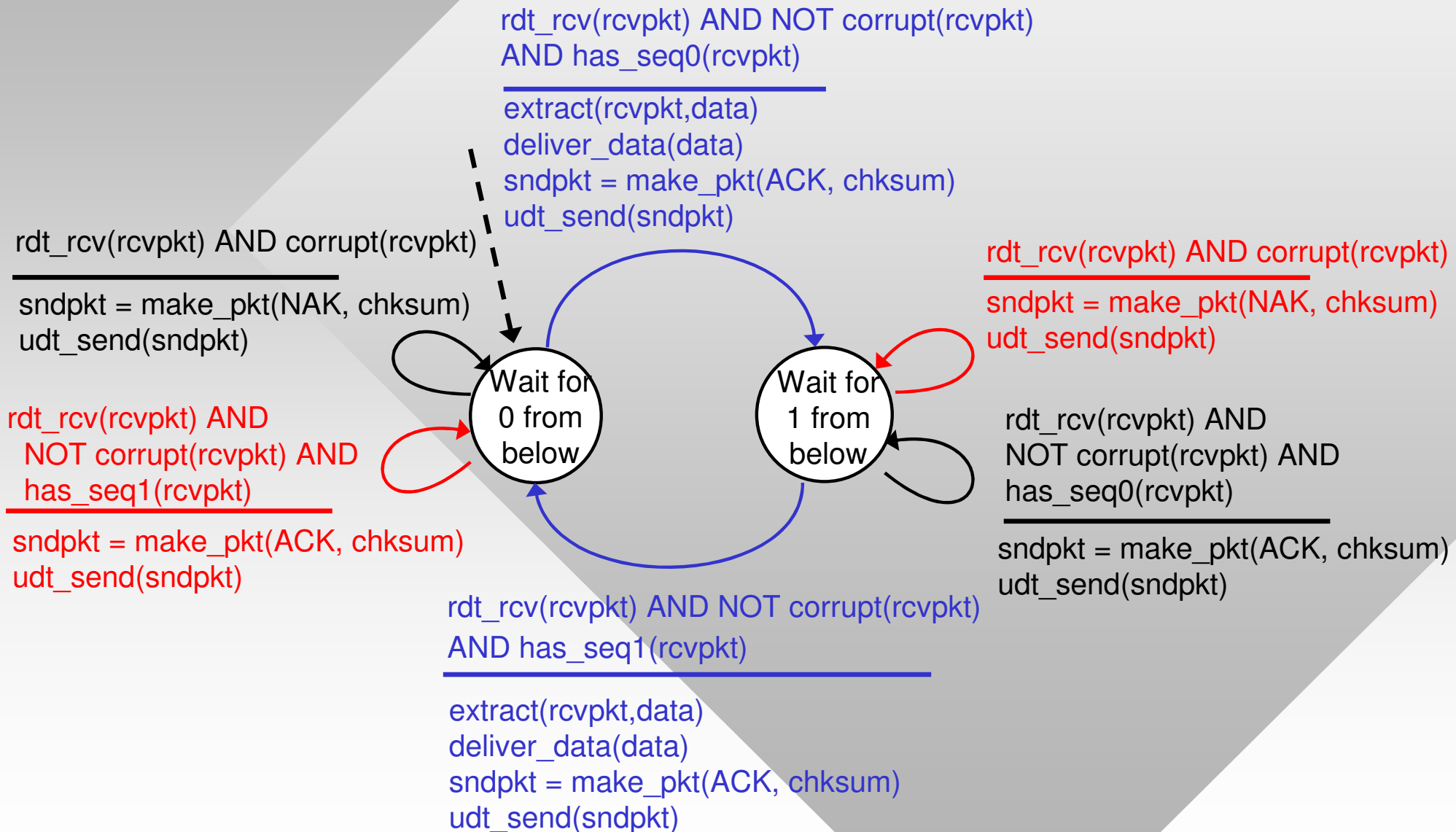
rdt_send(data)

sndpkt = make_pkt(0, data, checksum)

udt_send(sndpkt)



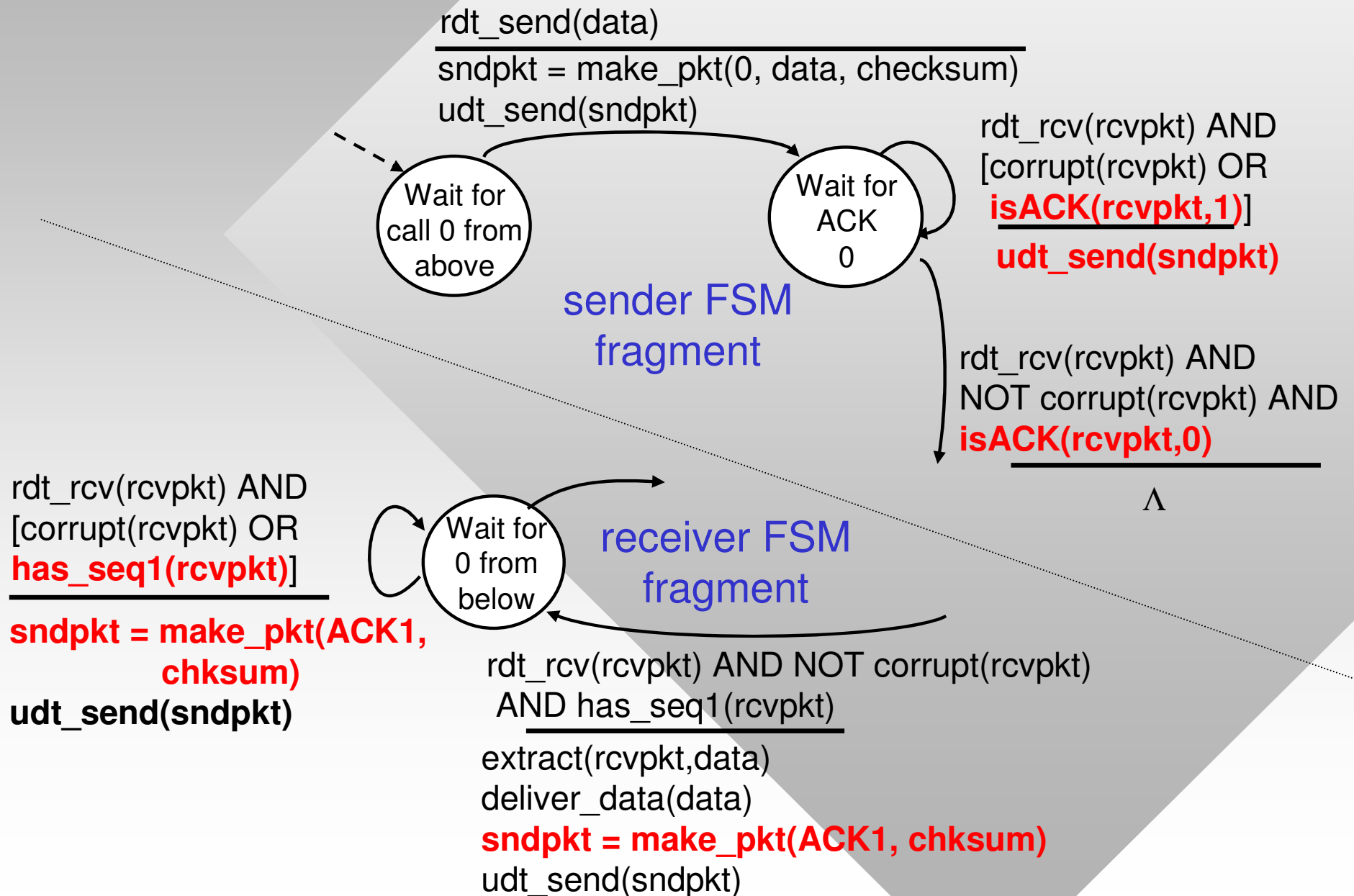
Rdt2.1: Receiver, Handles Garbled ACK/NAKs



Rdt2.2: a NAK-free Protocol

- Same functionality as rdt2.1, using ACKs only
- Instead of NAKs, receiver sends ACK for last packet received OK
 - Receiver must *explicitly* include seq # of pkt being ACKed
- Duplicate ACK at sender results in same action as NAK: *retransmit current pkt*

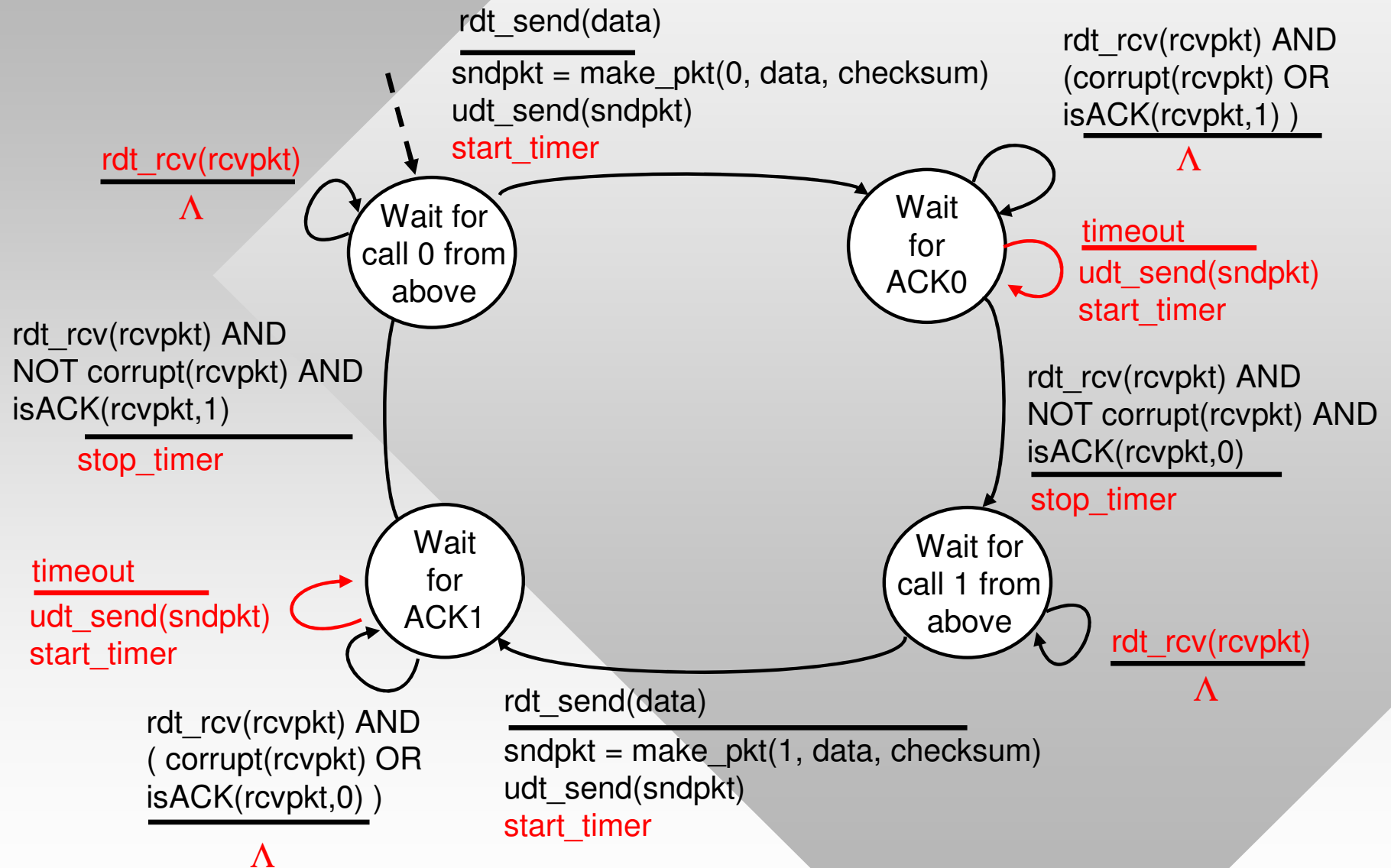
Rdt2.2: Sender, Receiver Fragments



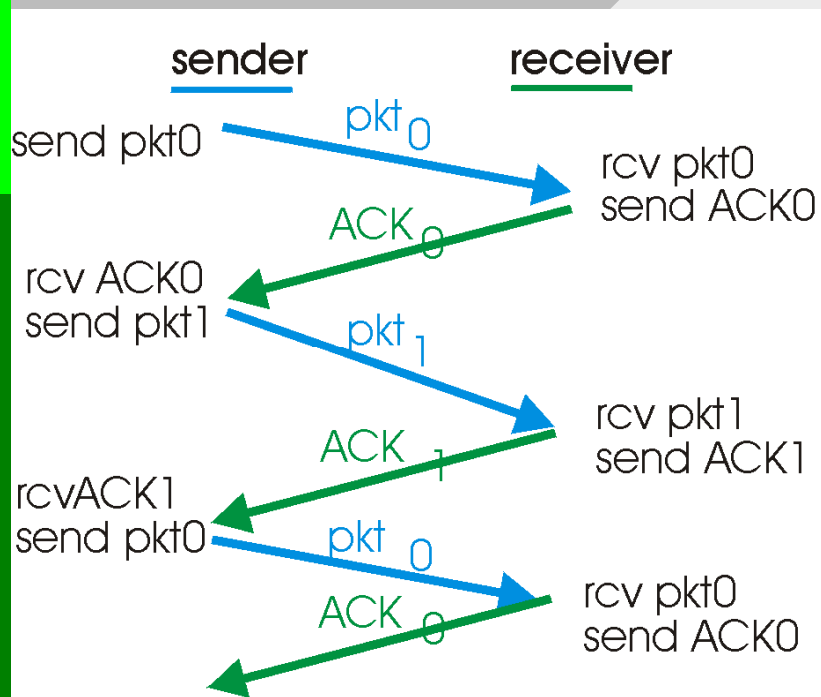
Rdt3.0: Channels With Errors *and* Loss

- New assumption: underlying channel can also lose packets (data or ACKs)
 - Checksum, sequence numbers, ACKs, retransmissions will be of help, but **not enough**
 - **Why not?**
- Approach: sender waits a “reasonable” amount of time for ACK
- Retransmits if no ACK received in this time
 - Sender requires a timer
 - If packet (or ACK) is just delayed (not lost):
 - Retransmission will be duplicate, but the use of seq. #'s already handles this
 - Receiver must specify seq # of packet being ACKed

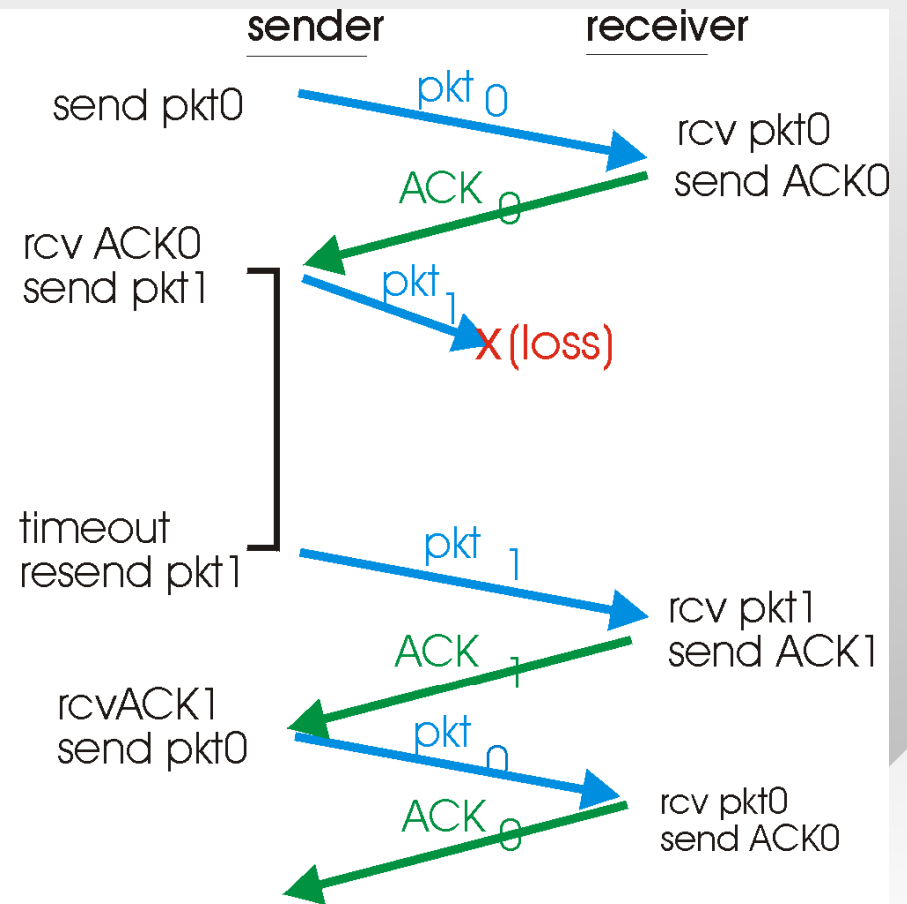
Rdt3.0 Sender



Rdt3.0 in Action

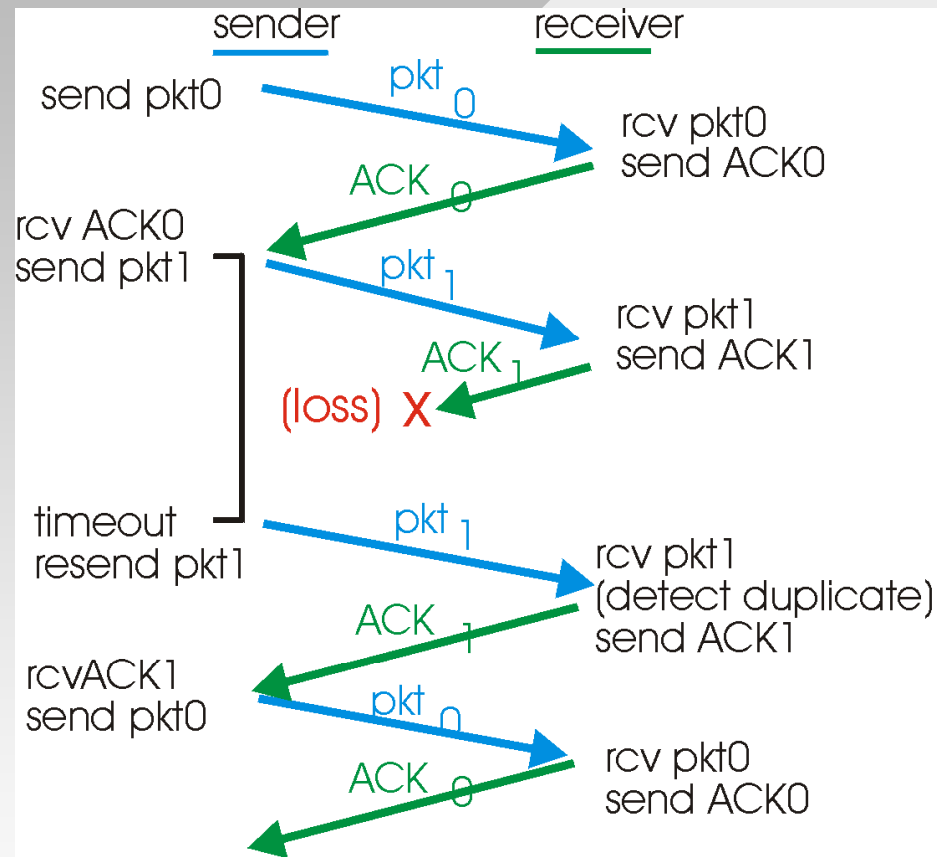


(a) operation with no loss

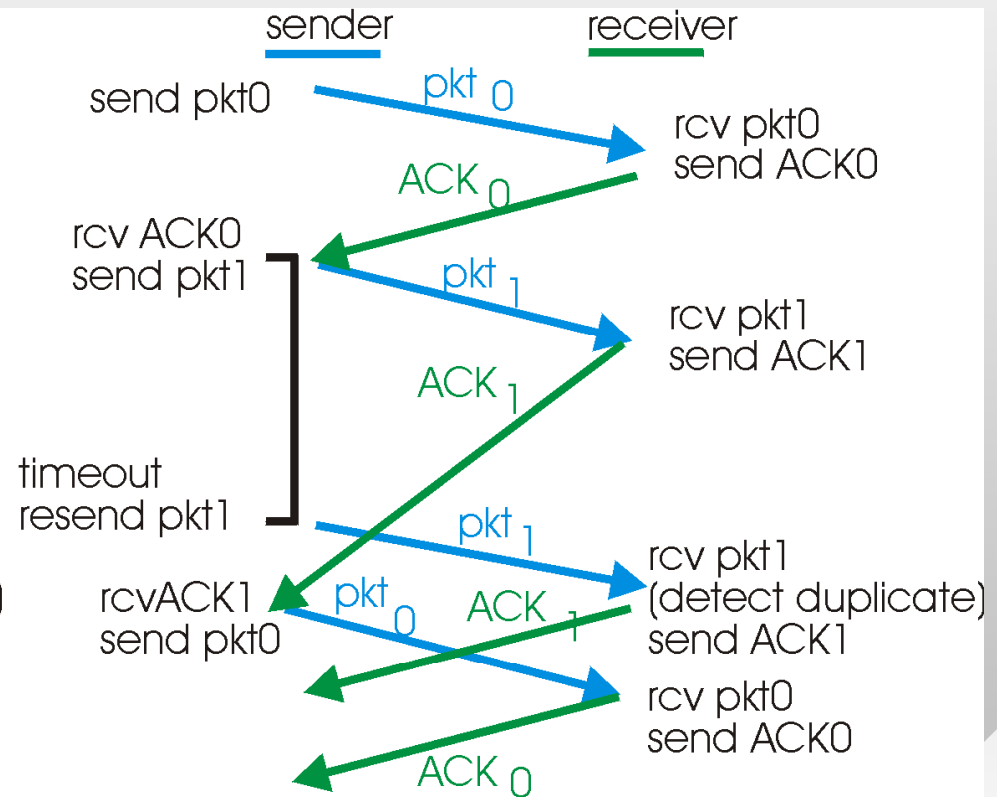


(b) lost packet

Rdt3.0 in Action



(c) lost ACK



(d) premature timeout

Performance of Rdt3.0

Notation: KB = kilobyte;
kb/s = kilobits/sec
gb/s = gigabits/sec

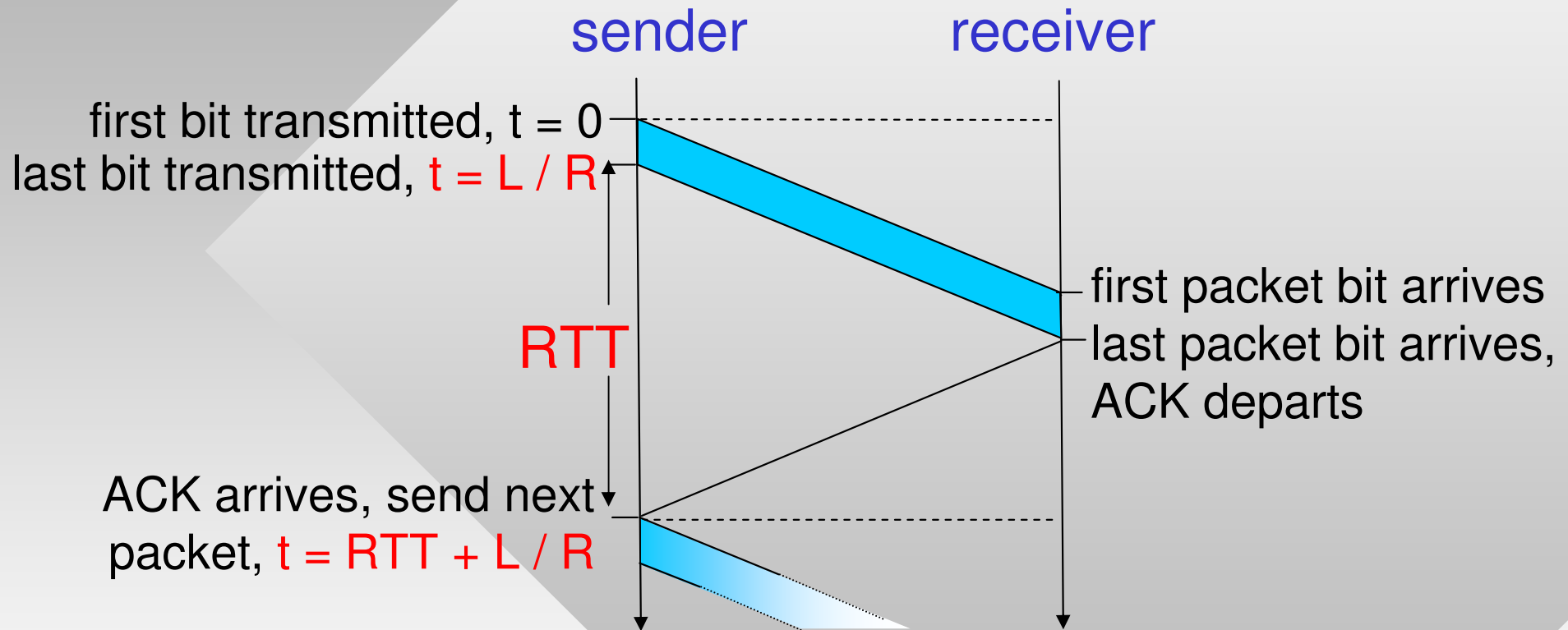
- Rdt 3.0 works, but performance is low
- Example: 1 Gbps link, 15 ms end-to-end propagation delay, 1 KB packets, no loss or corruption:

$$T_{\text{transmit}} = \frac{L \text{ (packet length in bits)}}{R \text{ (transmission rate, bps)}} = \frac{8 \text{ kbits/pkt}}{10^9 \text{ bits/sec}} = 8 \text{ microsec}$$

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

- The server spends 0.008 ms being busy and 30 ms being idle, thus its **utilization is only 0.027%**
- 1-KB pkt every 30 ms → 264 kbits/s thruput over 1 gb/s link
- Network protocol limits use of physical resources!

Rdt3.0: Stop-and-wait Operation



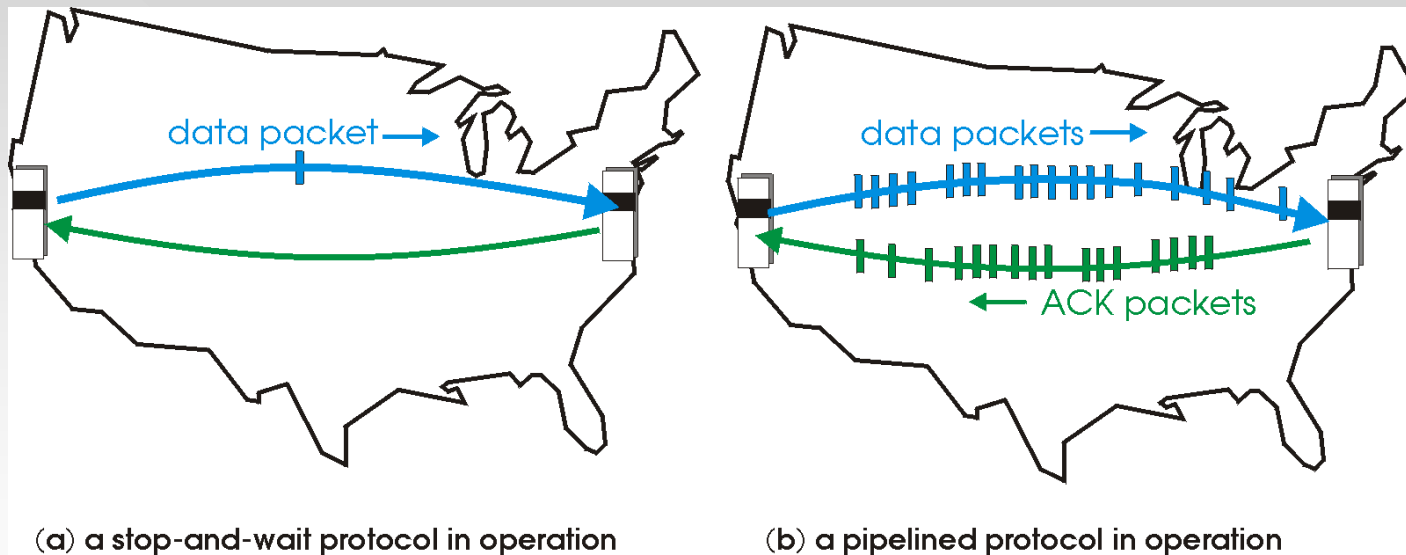
$$U_{\text{sender}} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

Performance of Rdt3.0

- Next assume that 10% of the packets are corrupted or lost and that the timeout value is 1 second
- 90% of packets take $(RTT + L/R) \sim 30$ ms to complete, while 10% require $[\text{timeout} + RTT + 2L/R]$
 - Thus, every 100 packets take $2.7 + 10.3 = 13$ seconds to transmit ($90 \cdot 0.03 + 10 \cdot 1.03 = 13$ seconds)
 - What's the average rate?
- First compute how many packets per second leave the sender (100 pkts in 13 seconds):
 - 130 ms per packet, which results in average rate 7.7 pkts/s
 - In other words, 61.5 kb/s since each packet is 8,000 bits
- This is much worse than 33 pkts/s under no loss

Pipelined Protocols

- **Pipelining**: sender allows multiple, “in-flight”, yet-to-be-acknowledged pkts
 - Range of sequence numbers must be increased
 - Buffering at sender and/or receiver



- Two generic forms of pipelined protocols: *Go-Back-N* and *Selective Repeat*