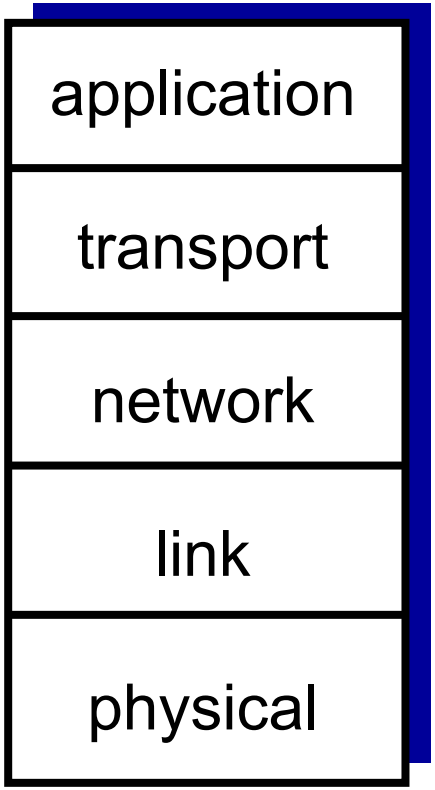
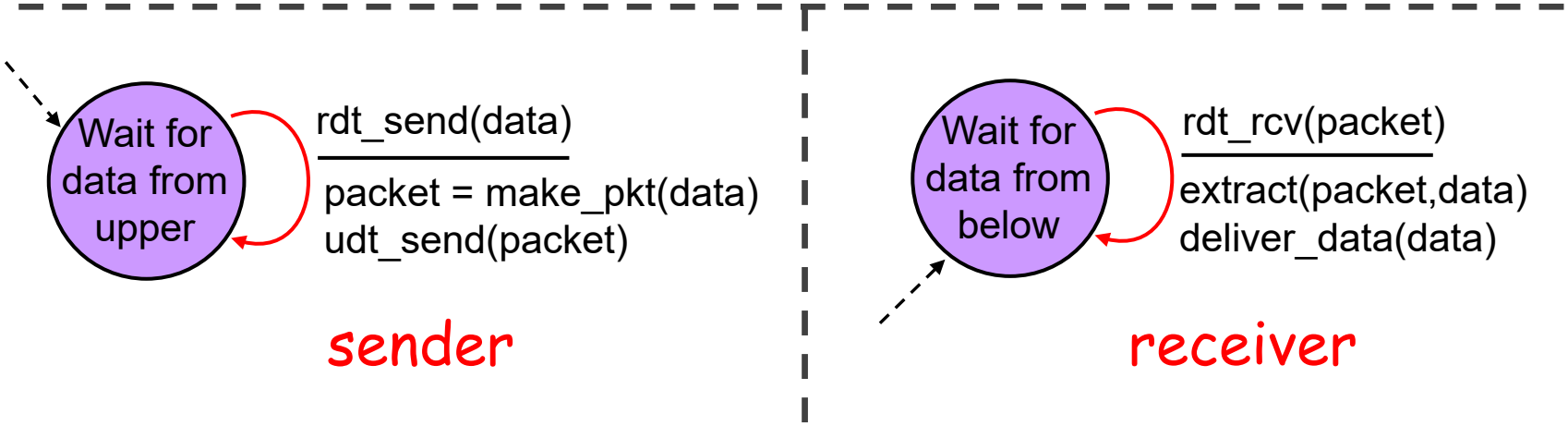
Designing Reliable Protocols

- Network layer service is unreliable.
  - Packets may be lost or corrupted during transmission.
- A reliable transport protocol should
  - ensure that packets are received by receiver in good order.
- Scenario: one sender, one receiver
  - Sender sends data packets to receiver
  - Receiver feeds back to sender



rdt 1.0: Reliable Channel

- Assume underlying channel is perfectly reliable.
- No error checking is needed.
  - Sender sends data into underlying (perfect) channel
  - Receiver reads data from underlying (perfect) channel



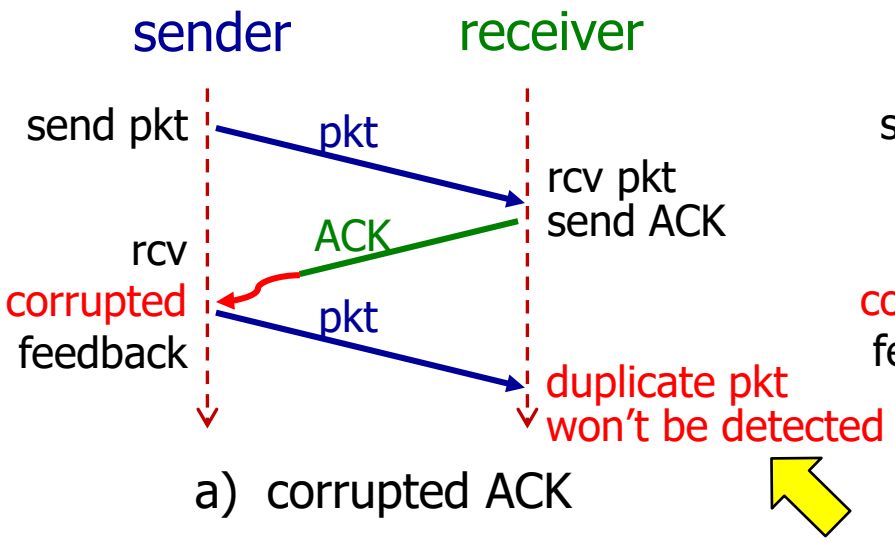
rdt 2.0: Channel with *Bit Errors*

- Assumption:
  - underlying channel may flip bits in packets
  - other than that, the channel is perfect

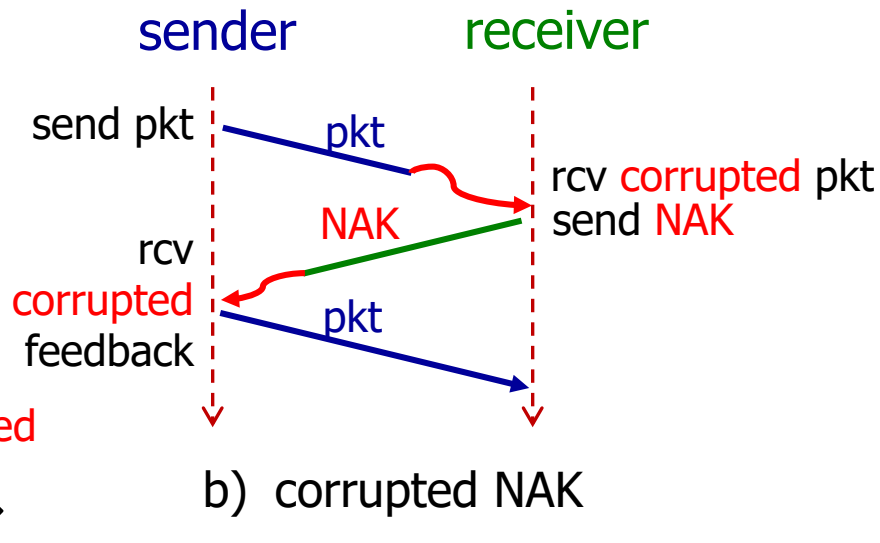
- Receiver may use ***checksum*** to detect bit errors.
- Receiver will feed back to sender:
  - *Acknowledgements (ACKs):* receiver explicitly tells sender that packet received is OK.
  - *Negative acknowledgements (NAKs):* receiver explicitly tells sender that packet has errors.
  - Sender retransmits packet on receipt of NAK.

rdt 2.0 has a Fatal Flaw!

- What happens if ACK/NAK is corrupted?
  - Sender doesn't know what happened at receiver!
- So what should the sender do?
  - Sender should retransmit when receives garbled ACK or NAK.
  - Issue: receiver may receive duplicate packet.



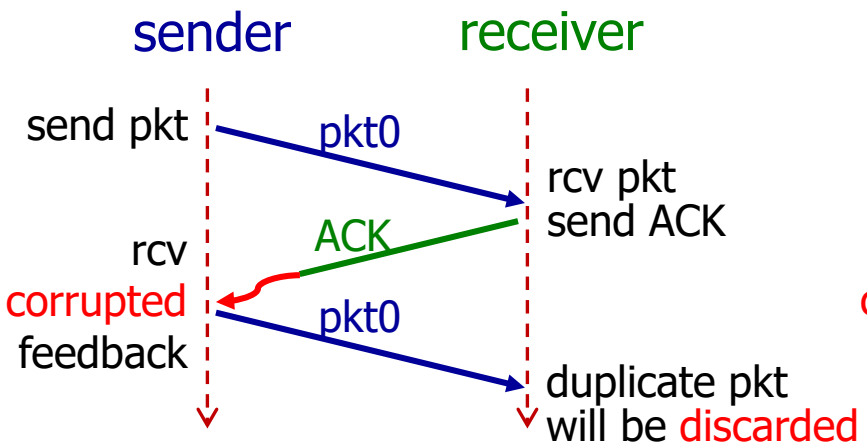
a) corrupted ACK



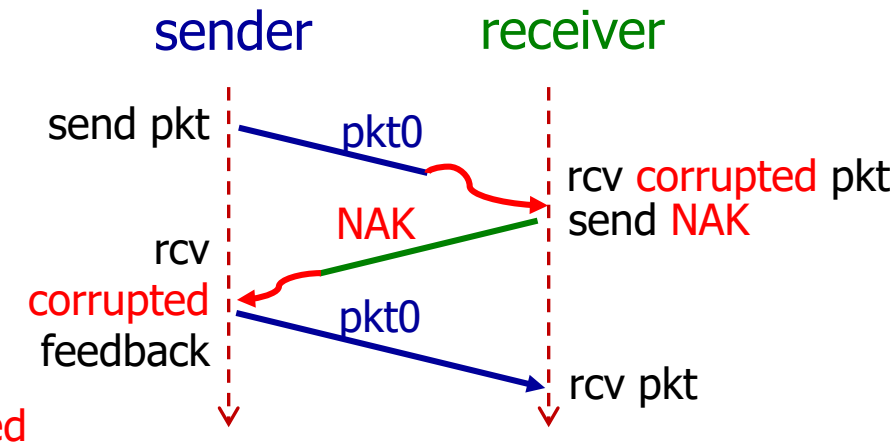
b) corrupted NAK

rdt 2.1: rdt 2.0 + Packet Seq.

- To handle duplicates:
  - Sender retransmits current packet if ACK/NAK is garbled.
  - Sender adds *sequence number* to each packet.
  - Receiver discards (doesn't deliver up) duplicate packet.
- This gives rise to protocol rdt 2.1.



a) corrupted ACK



b) corrupted NAK

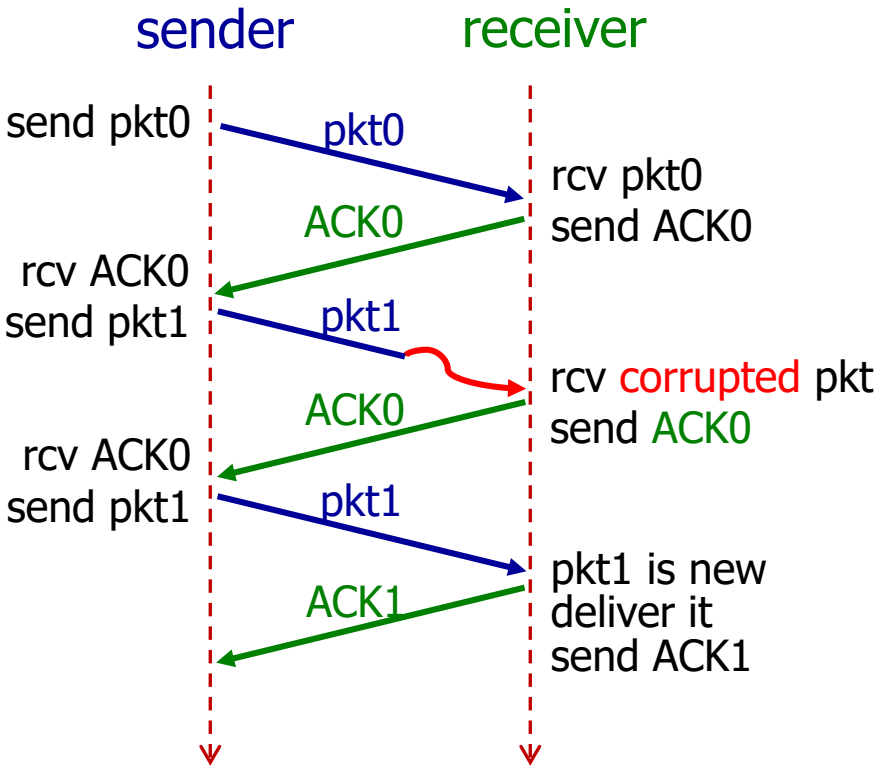
rdt 2.2: a NAK-free Protocol

- Same assumption and functionality as rdt 2.1, but use ACKs only.
- Instead of sending NAK, receiver sends ACK for the last packet received OK.
  - Now receiver must *explicitly* include seq. # of the packet being ACKed.
- Duplicate ACKs at sender results in same action as NAK: *retransmit current pkt*.

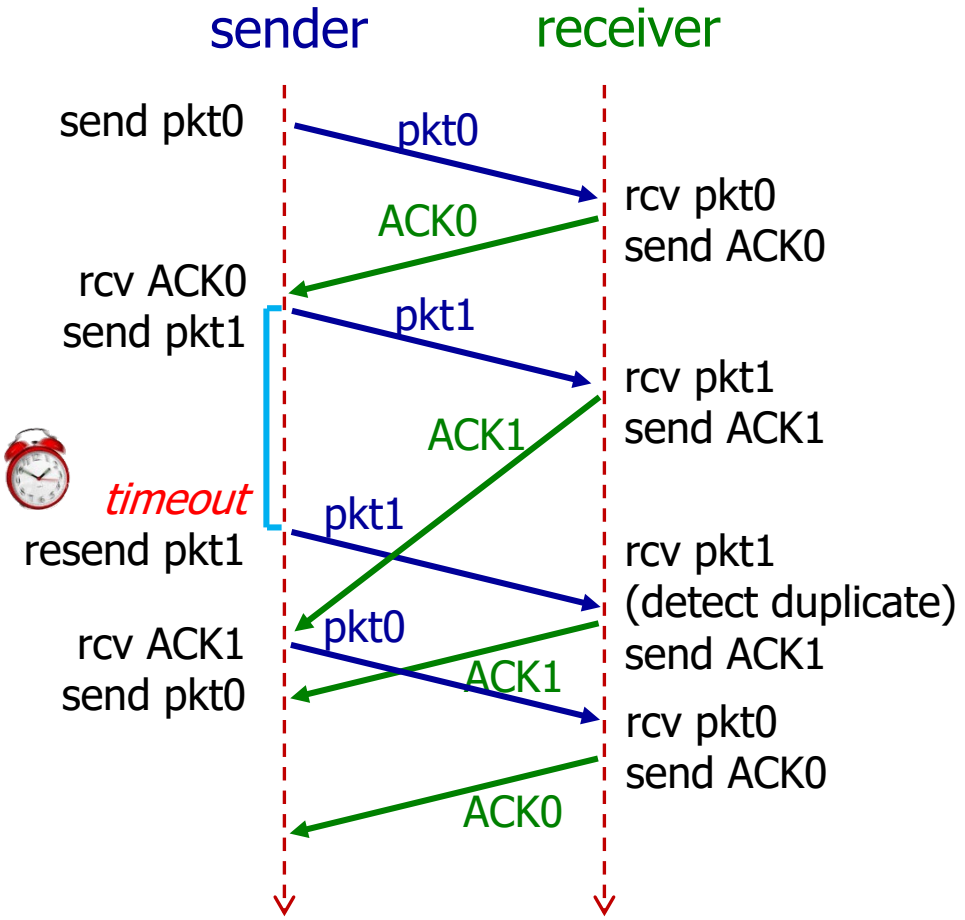
rdt 3.0: Channel with *Errors* & *Loss*

- Assumption: underlying channel
  - may flip bits in packets
  - may lose packets
  - may incur arbitrarily long packet delay
  - but won't re-order packets
- To handle packet loss:
  - Sender waits "reasonable" amount of time for ACK.
  - Sender retransmits if no ACK is received till *timeout*.
- Sender relies on timeout/retransmission to deal with both packet corruption and packet loss.

rdt 2.2 vs. rdt 3.0



b) rdt 2.2: resend due to duplicate ACK0



d) 3.0 no resend on duplicate ACK1

PREVIOUS LECTURE

Stop-and-wait Protocols

rdt Version	Scenario	Features Used
1.0	no error	nothing
2.1	data packet Bit Error feedback packet Bit Error	checksum, ACK/NAK, sequence number
2.2	data packet Bit Error feedback packet Bit Error	checksum, ACK, sequence number
3.0	data packet Bit Error feedback packet Bit Error packet Loss	checksum, ACK, sequence number, timeout/re-transmission