COMPUTER NETWORKS

Transport Layer

Animesh Giri
Department of Computer Science & Engineering

COMPUTER NETWORKS

Principles of reliable date transfer

Animesh Giri
Department of Computer Science & Engineering

In this segment

- rdt2.0: channel with bit errors
- rdt2.0: FSM specification
- rdt2.0: operation with no errors
- rdt2.0: error scenario
- rdt2.0 has a fatal flaw!
- rdt2.1: sender, handles garbled ACK/NAKs
- rdt2.1: discussion
- rdt2.2: a NAK-free protocol
- rdt2.2: sender, receiver fragments
- Summary

rdt2.0: channel with bit errors

- underlying channel may flip bits in packet
  - checksum to detect bit errors
- *the* question: how to recover from errors:

How do humans recover from “errors” during conversation?

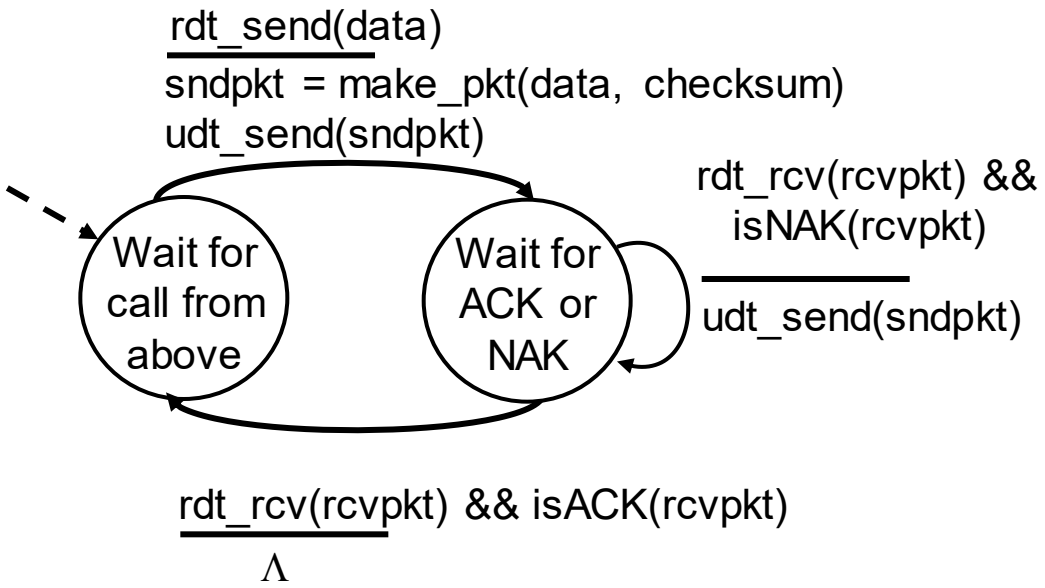
rdt2.0: channel with bit errors

- underlying channel may flip bits in packet
  - checksum to detect bit errors
- *the* question: how to recover from errors:
  - *acknowledgements (ACKs):* receiver explicitly tells sender that pkt received OK
  - *negative acknowledgements (NAKs):* receiver explicitly tells sender that pkt had errors
  - sender retransmits pkt on receipt of NAK
- new mechanisms in **`rdt2.0`** (beyond **`rdt1.0`**):
  - error detection
  - feedback: control msgs (ACK,NAK) from receiver to sender

stop and wait

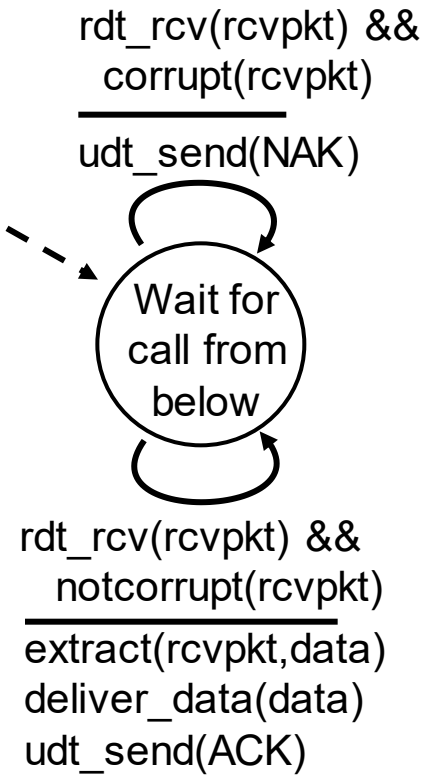
sender sends one packet, then waits for receiver response

rdt2.0: FSM specification

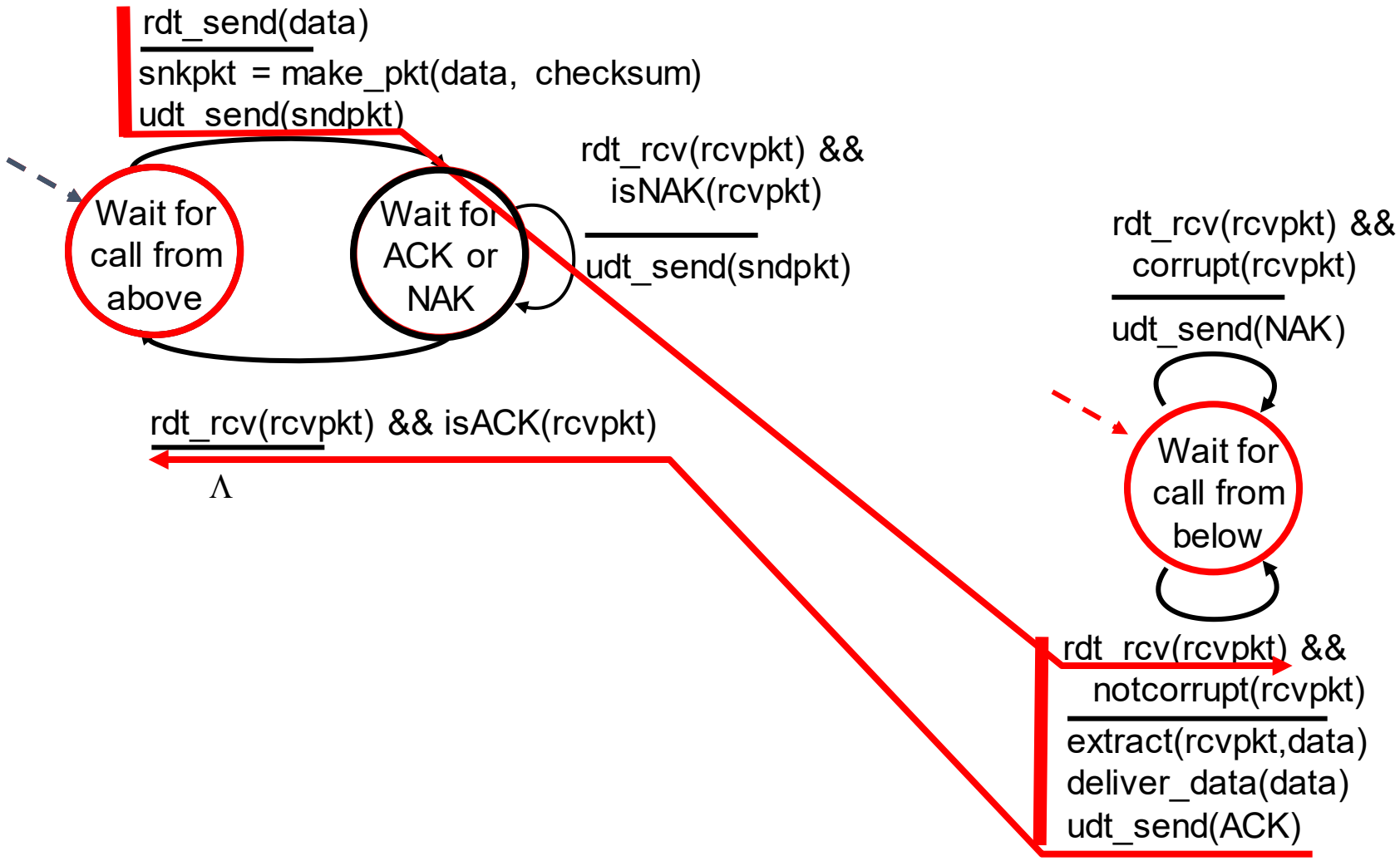


sender

receiver

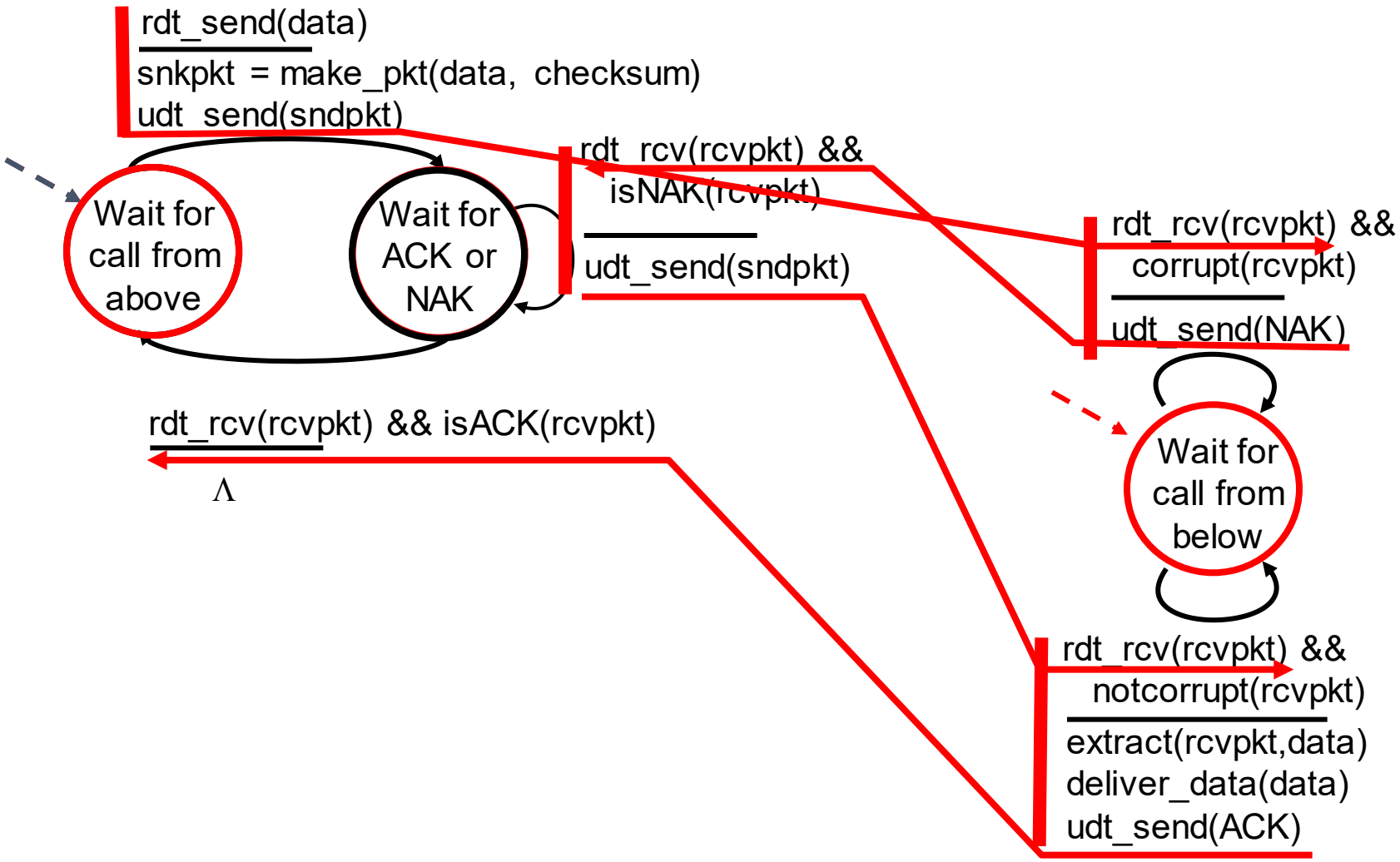


rdt2.0: operation with no errors



COMPUTER NETWORKS

rdt2.0: error scenario.



rdt2.0 has a fatal flaw!

what happens if ACK/NAK corrupted?

- sender doesn't know what happened at receiver!
- can't just retransmit: possible duplicate

handling duplicates:

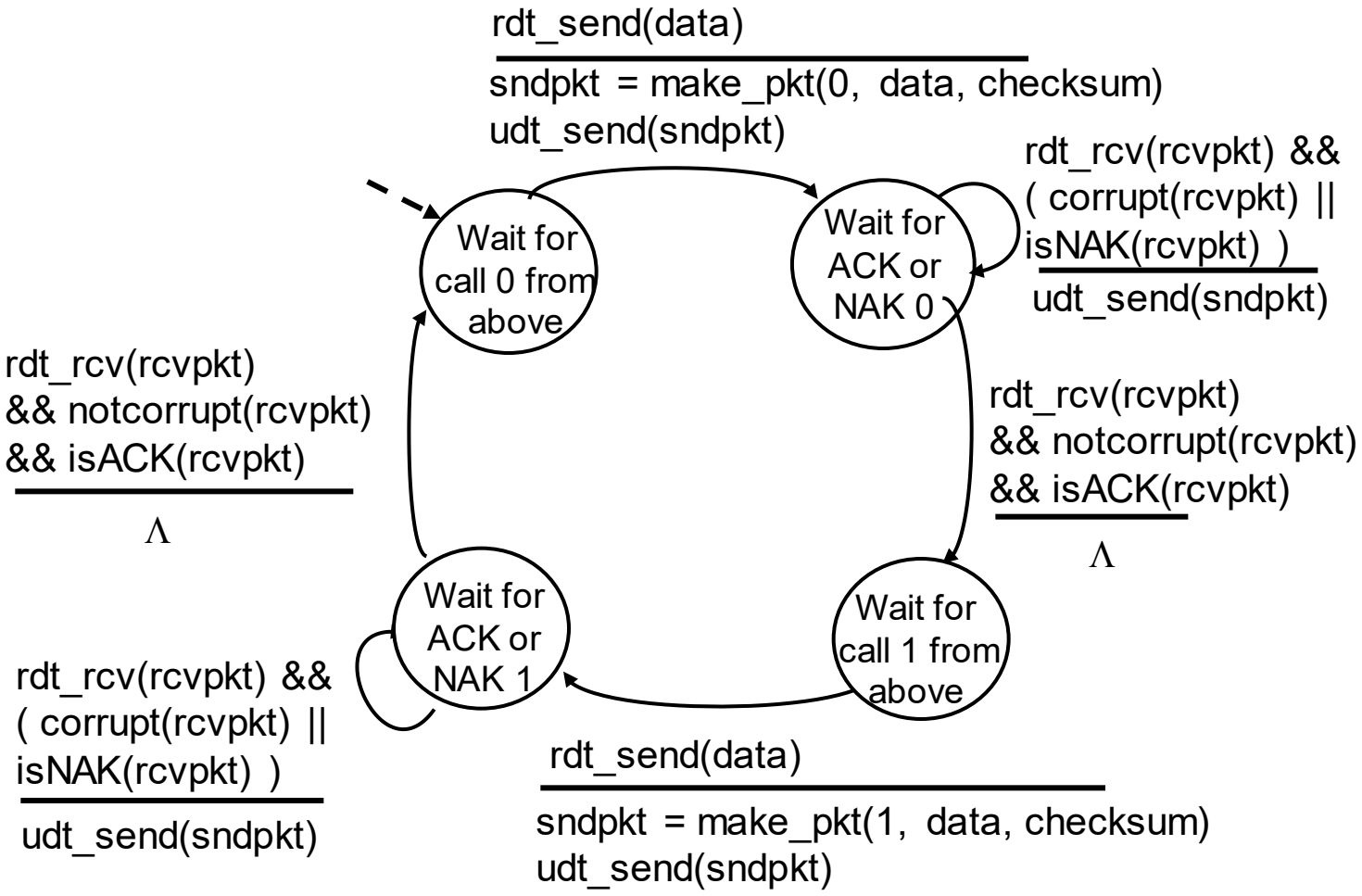
- sender retransmits current pkt if ACK/NAK corrupted
- sender adds sequence number to each pkt
- receiver discards (doesn't deliver up) duplicate pkt

stop and wait

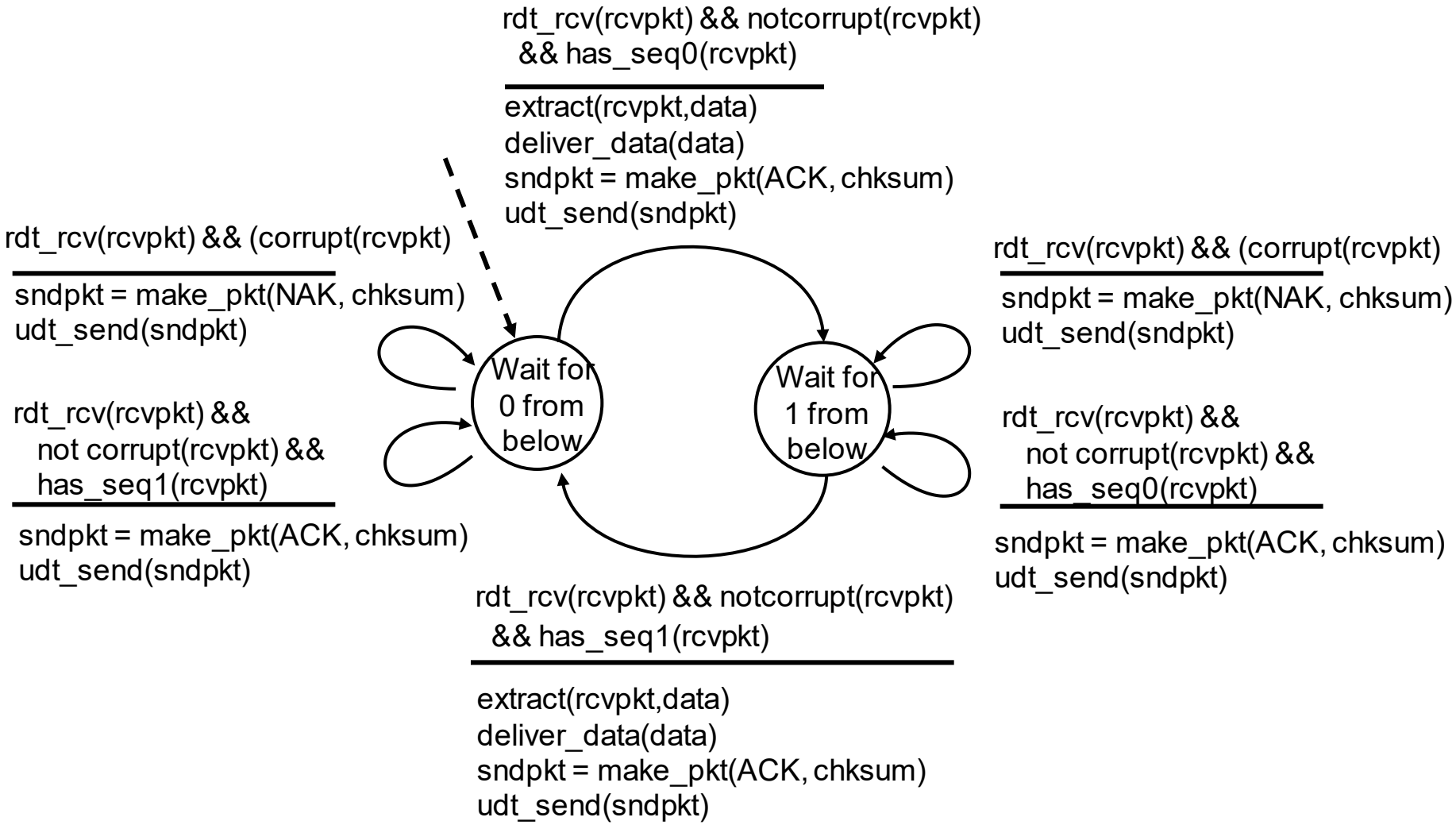
sender sends one packet, then waits for receiver response

COMPUTER NETWORKS

rdt2.1: sender, handles garbled ACK/NAKs



rdt2.1: receiver, handles garbled ACK/NAKs



rdt2.1: discussion

sender:

- seq # added to pkt
- two seq. #'s (0,1) will suffice. Why?
- must check if received ACK/NAK corrupted
- twice as many states
  - state must "remember" whether "expected" pkt should have seq # of 0 or 1

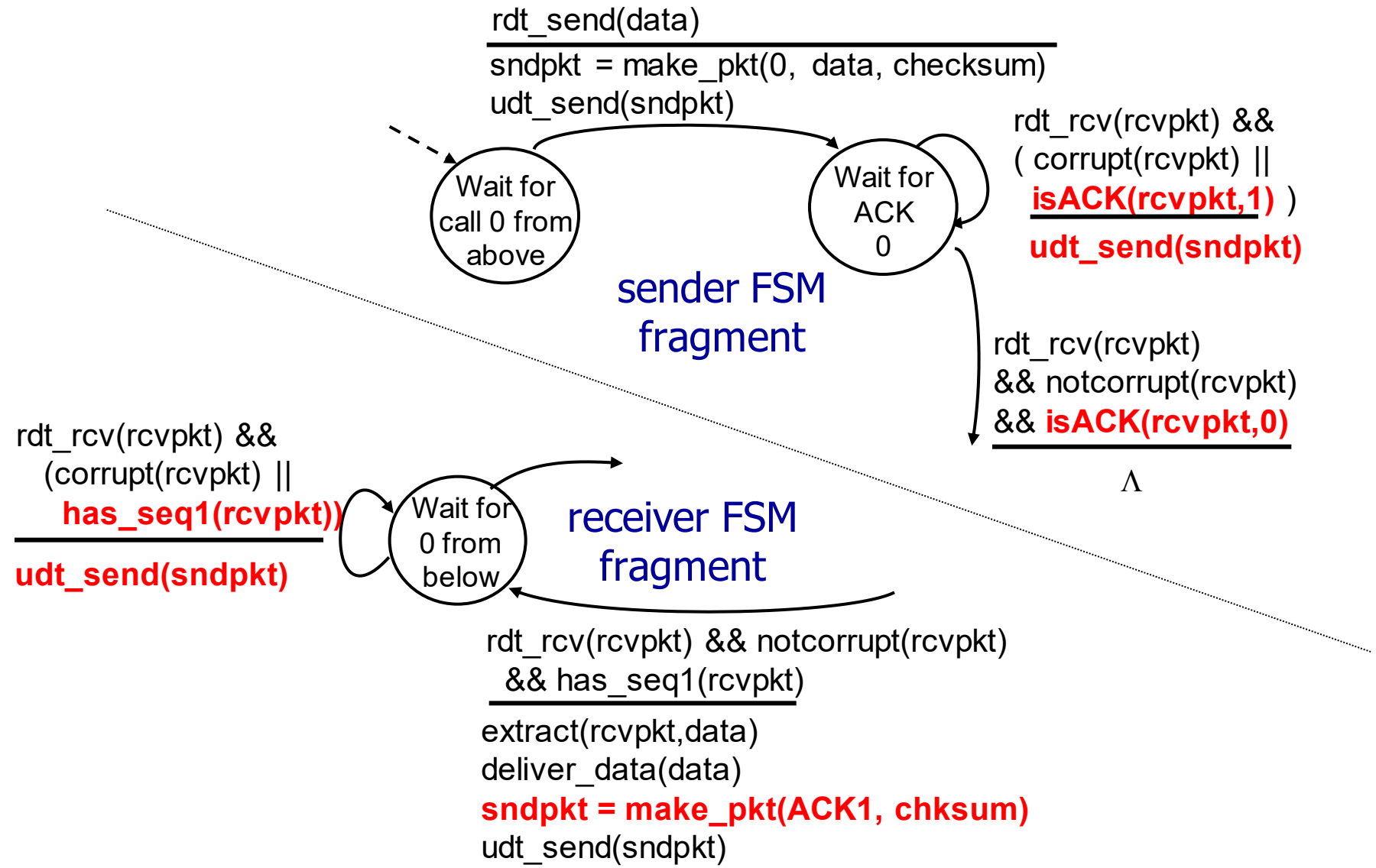
receiver:

- must check if received packet is duplicate
  - state indicates whether 0 or 1 is expected pkt seq #
- note: receiver can *not* know if its last ACK/NAK received OK at sender

rdt2.2: a NAK-free protocol

- same functionality as rdt2.1, using ACKs only
- instead of NAK, receiver sends ACK for last pkt 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



COMPUTER NETWORKS

Summary

THANK YOU

Animesh Giri
Department of Computer Science & Engineering

animeshgiri@pes.edu

+91 80 6618 6603