Chapter 3: roadmap

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
- Principles of congestion control
- TCP congestion control
- Evolution of transport-layer functionality

Principles of reliable data transfer

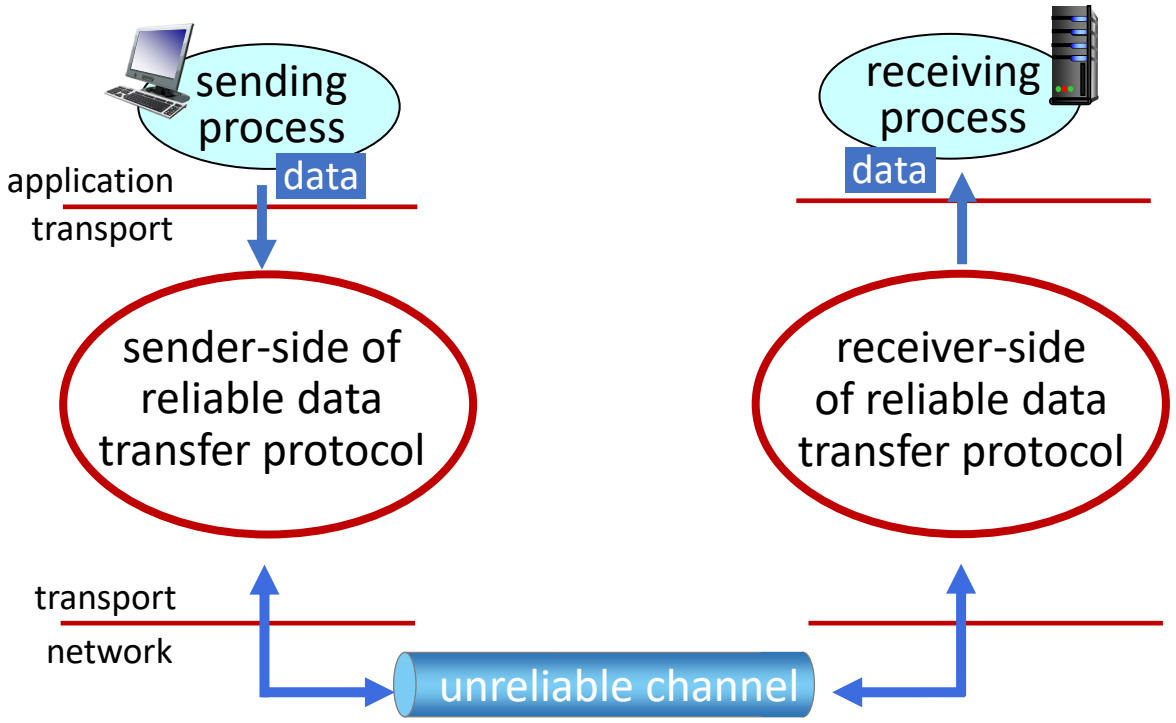
- Channel model
- Realistic assumptions
- Protocol mechanisms

Principles of reliable data transfer

reliable service *abstraction*

Principles of reliable data transfer

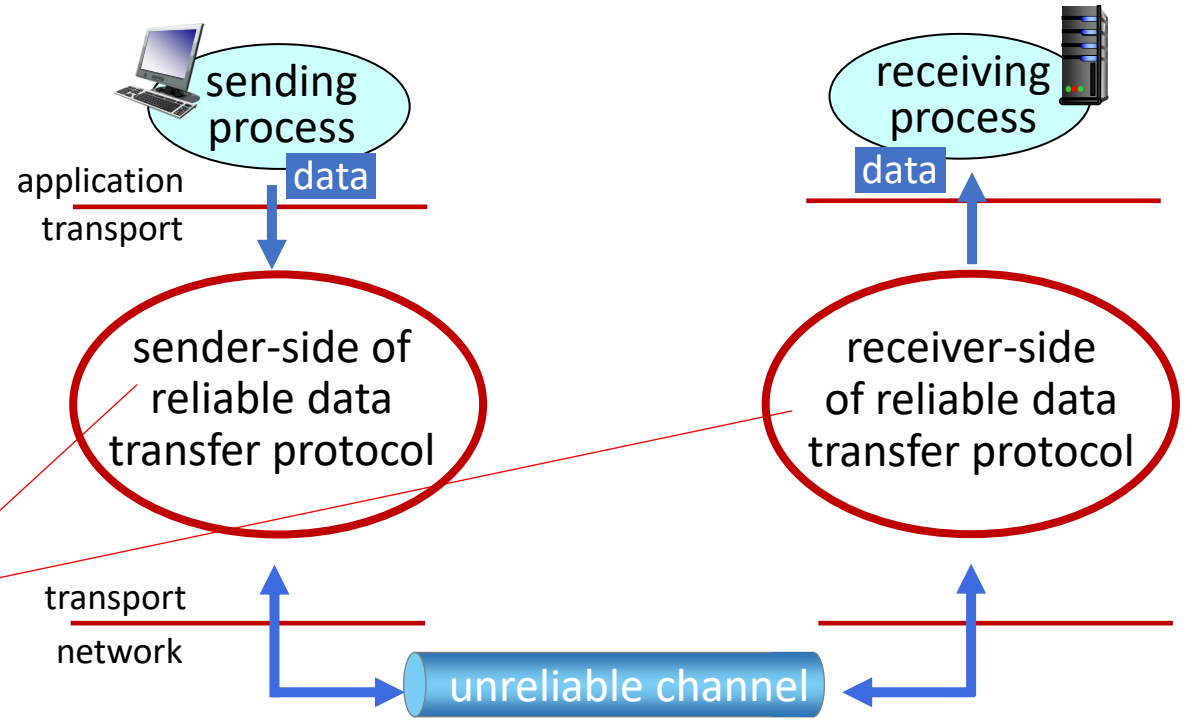
reliable service *abstraction*



reliable service *implementation*

Principles of reliable data transfer

Complexity of reliable data transfer protocol will depend (strongly) on characteristics of unreliable channel (lose, corrupt, reorder data?)

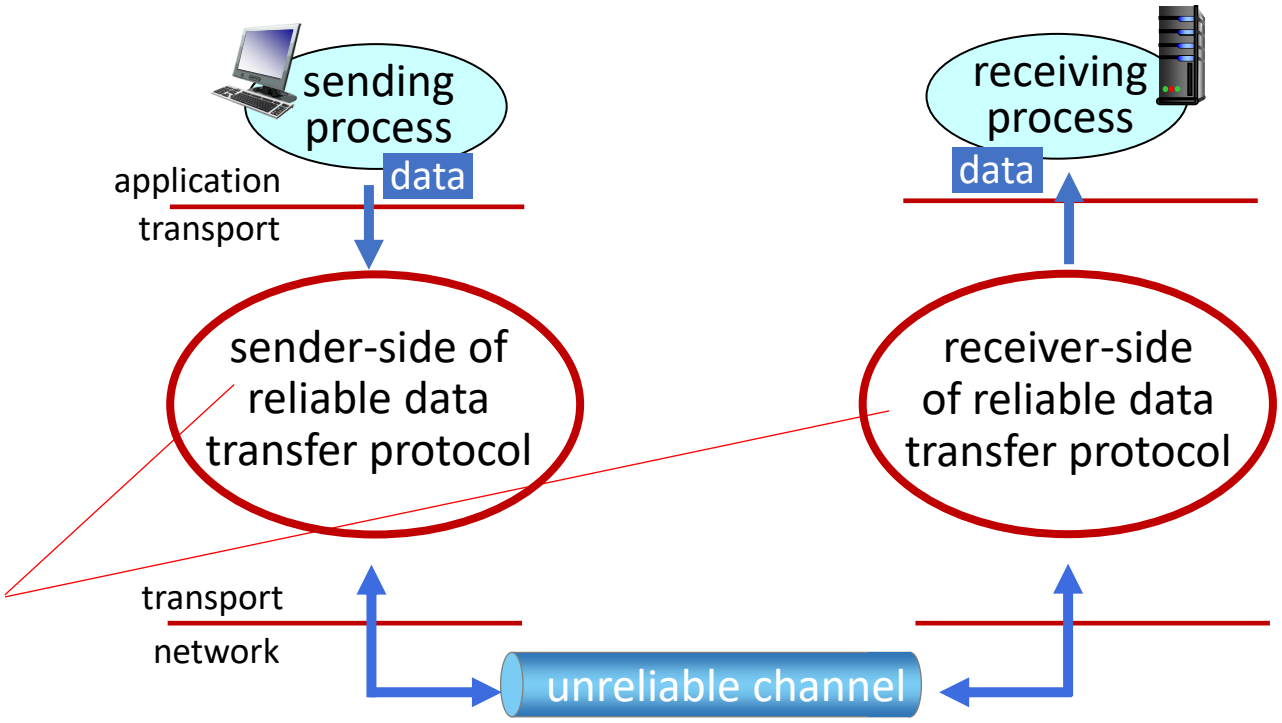


reliable service *implementation*

Principles of reliable data transfer

Sender, receiver do *not* know the "state" of each other, e.g., was a message received?

- unless communicated via a message

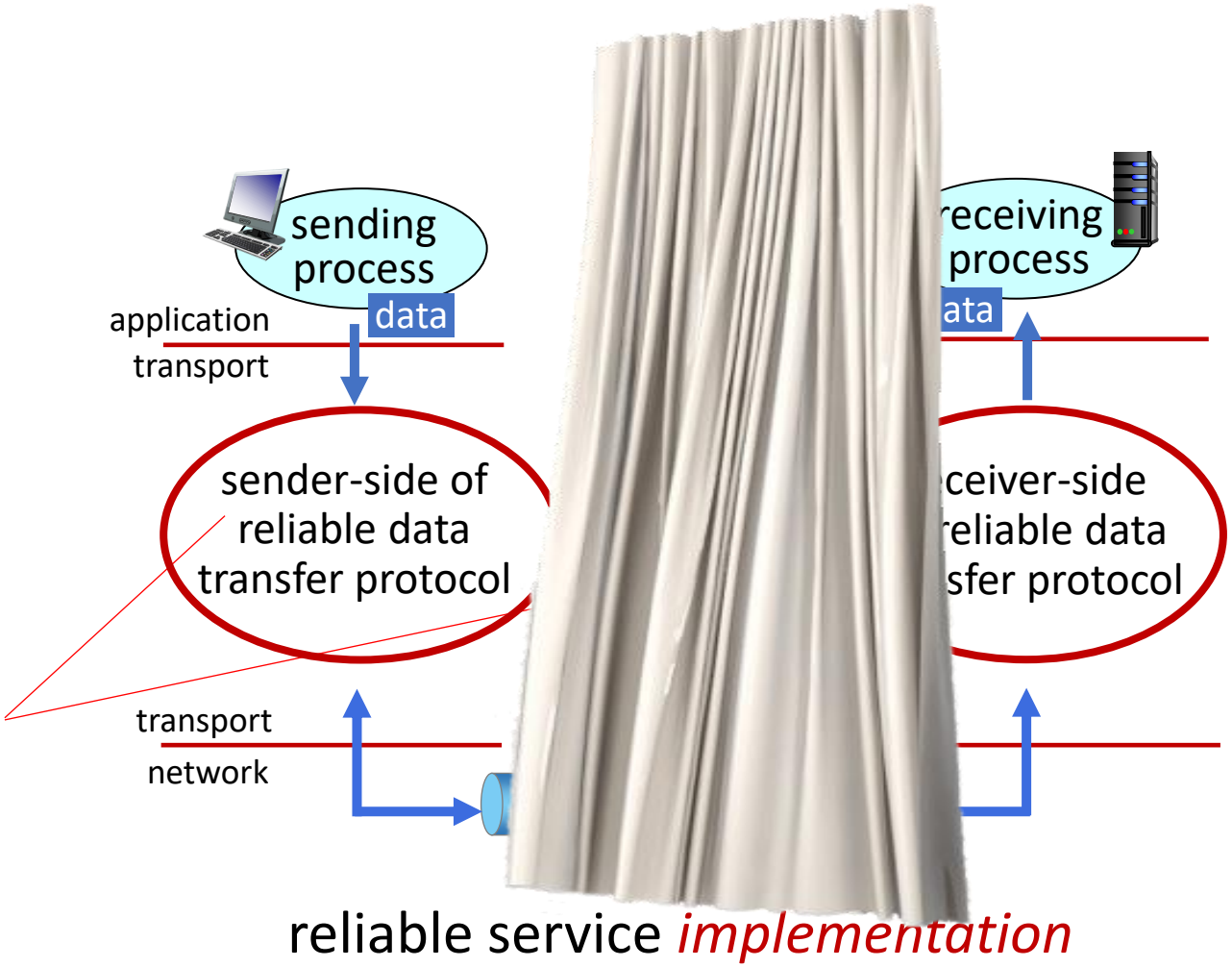


reliable service *implementation*

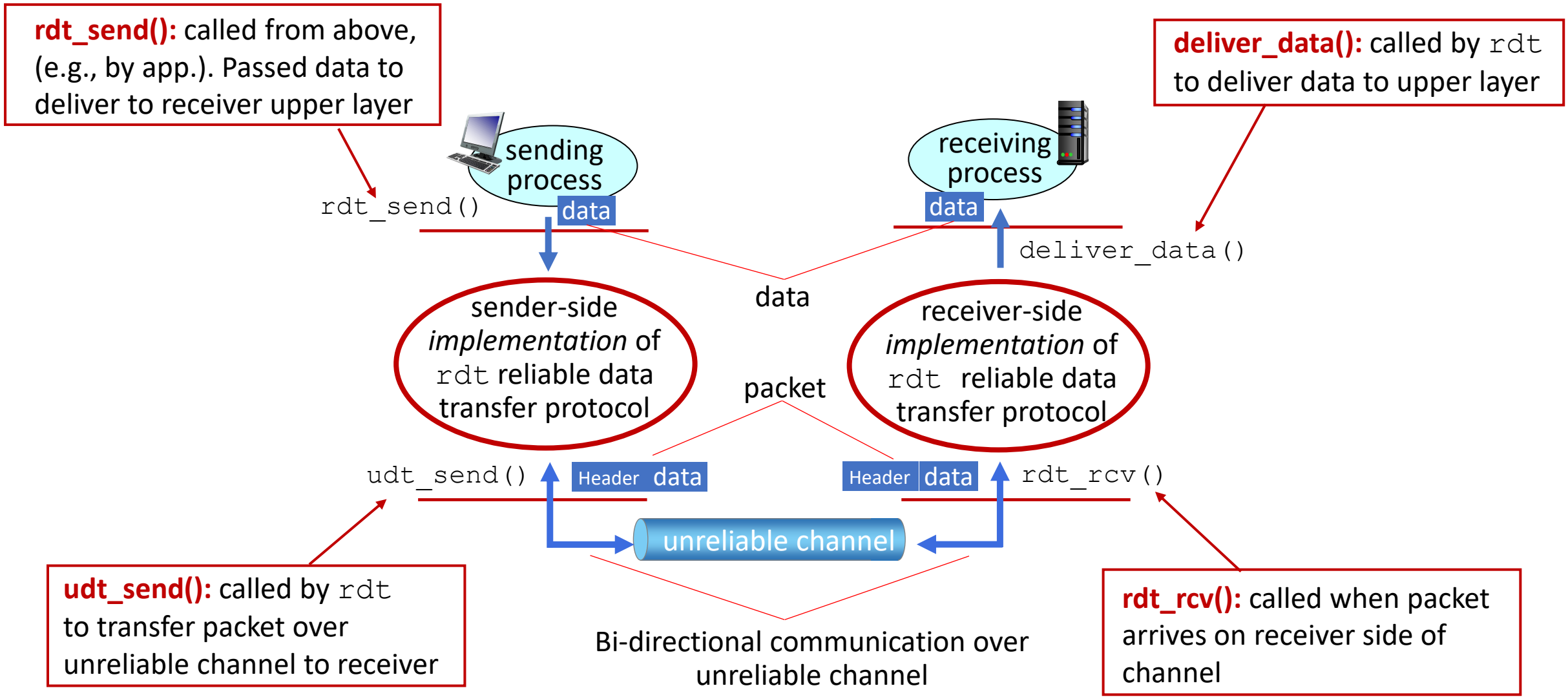
Principles of reliable data transfer

Sender, receiver do *not* know the "state" of each other, e.g., was a message received?

- unless communicated via a message



Reliable data transfer protocol (rdt): interfaces

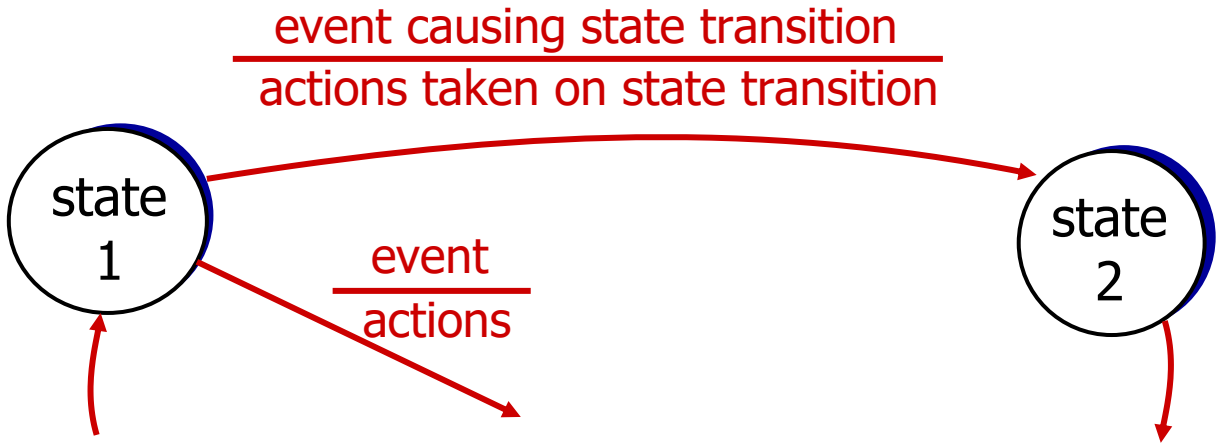


Reliable data transfer: getting started

We will:

- incrementally develop sender, receiver sides of reliable data transfer protocol (`rdt`)
- consider only unidirectional data transfer
  - but control info will flow in both directions!
- use finite state machines (FSM) to specify sender, receiver

state: when in this "state" next state uniquely determined by next event



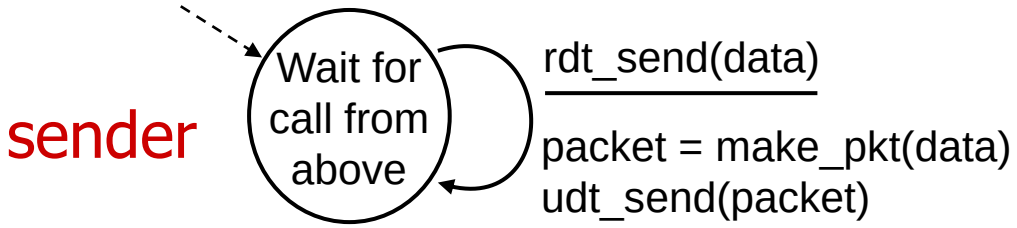
Reliable data transfer: getting started

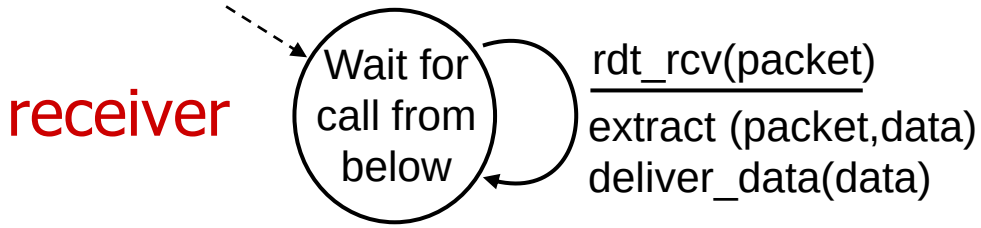
- What do we really mean when we say RDT senders in a given state or a receiver is in a given state?
- Think about a link being in a transmitting state or in an idle State.
- Think about notion of there being transitions between states.
- Transitions happening because of an event that takes place.
- think about actions that are taken by the system.

rdt1.0: reliable transfer over a reliable channel

- underlying channel perfectly reliable
  - no bit errors
  - no loss of packets
- *separate* FSMs for sender, receiver:
  - sender sends data into underlying channel
  - receiver reads data from underlying channel

rdt1.0: reliable transfer over a reliable channel





rdt2.0: channel with bit errors

- underlying channel may flip bits in packet
  - checksum (e.g., Internet 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

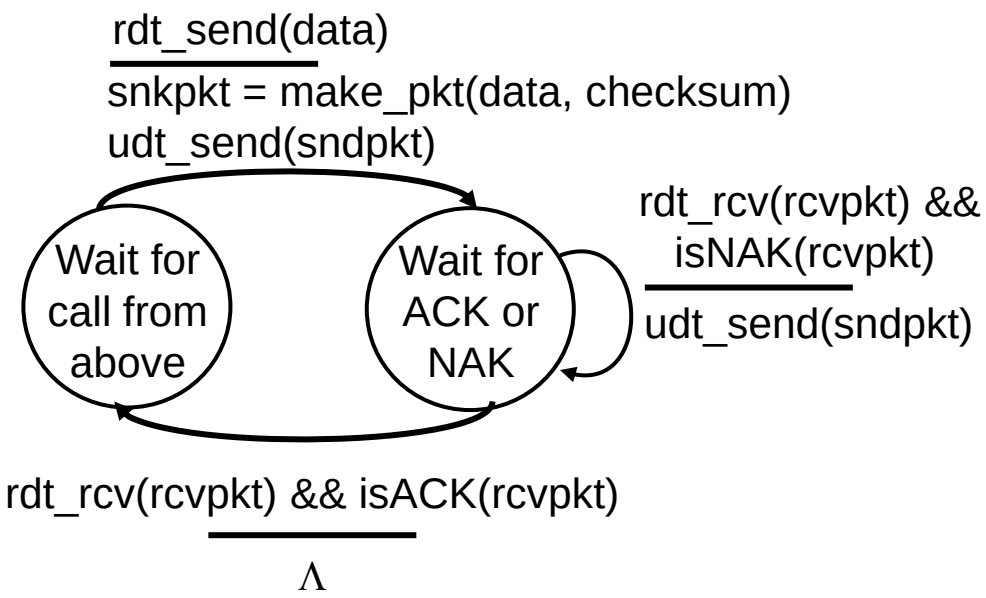
- underlying channel may flip bits in packet
  - checksum to detect bit errors
- *the* question: how to recover from errors?
  - *acknowledgements (ACKs):* receiver explicitly tells sender that pkt received OK
  - *negative acknowledgements (NAKs):* receiver explicitly tells sender that pkt had errors
  - sender *retransmits* pkt on receipt of NAK

stop and wait

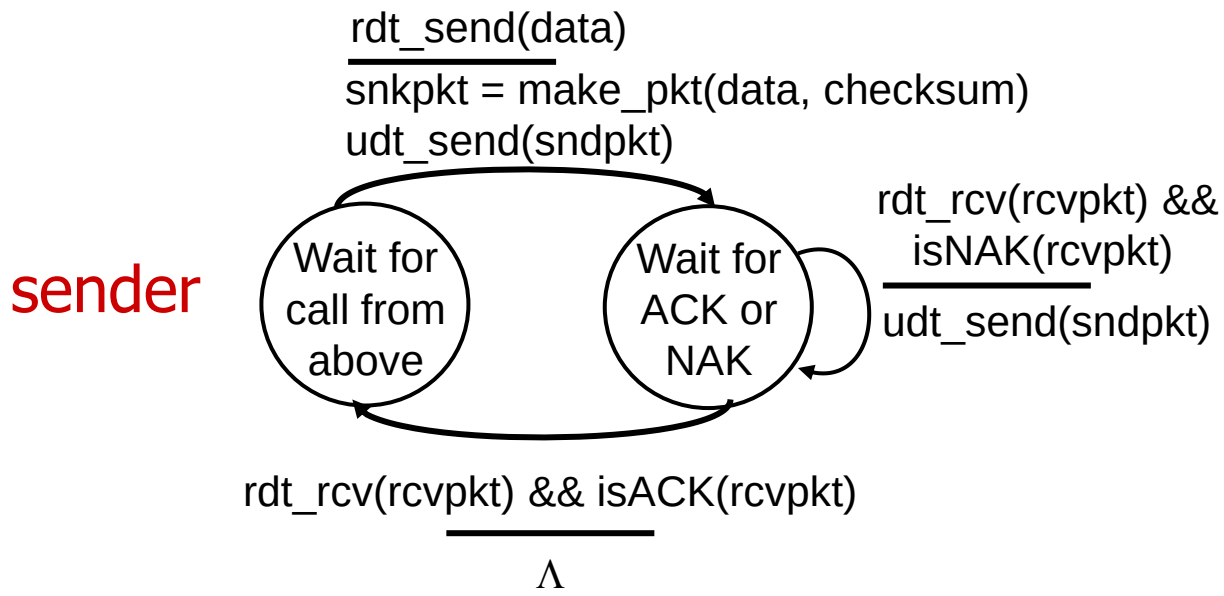
sender sends one packet, then waits for receiver response

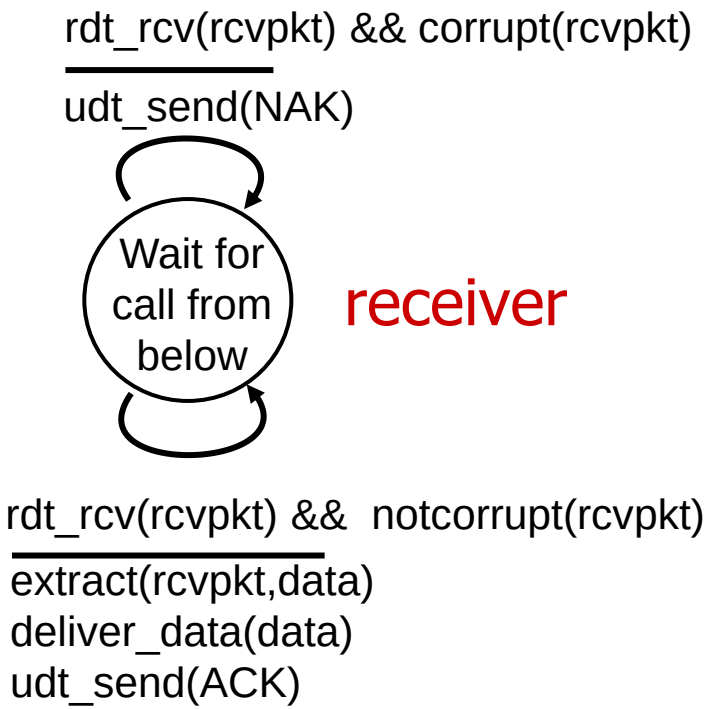
rdt2.0: FSM specification

sender



rdt2.0: FSM specification

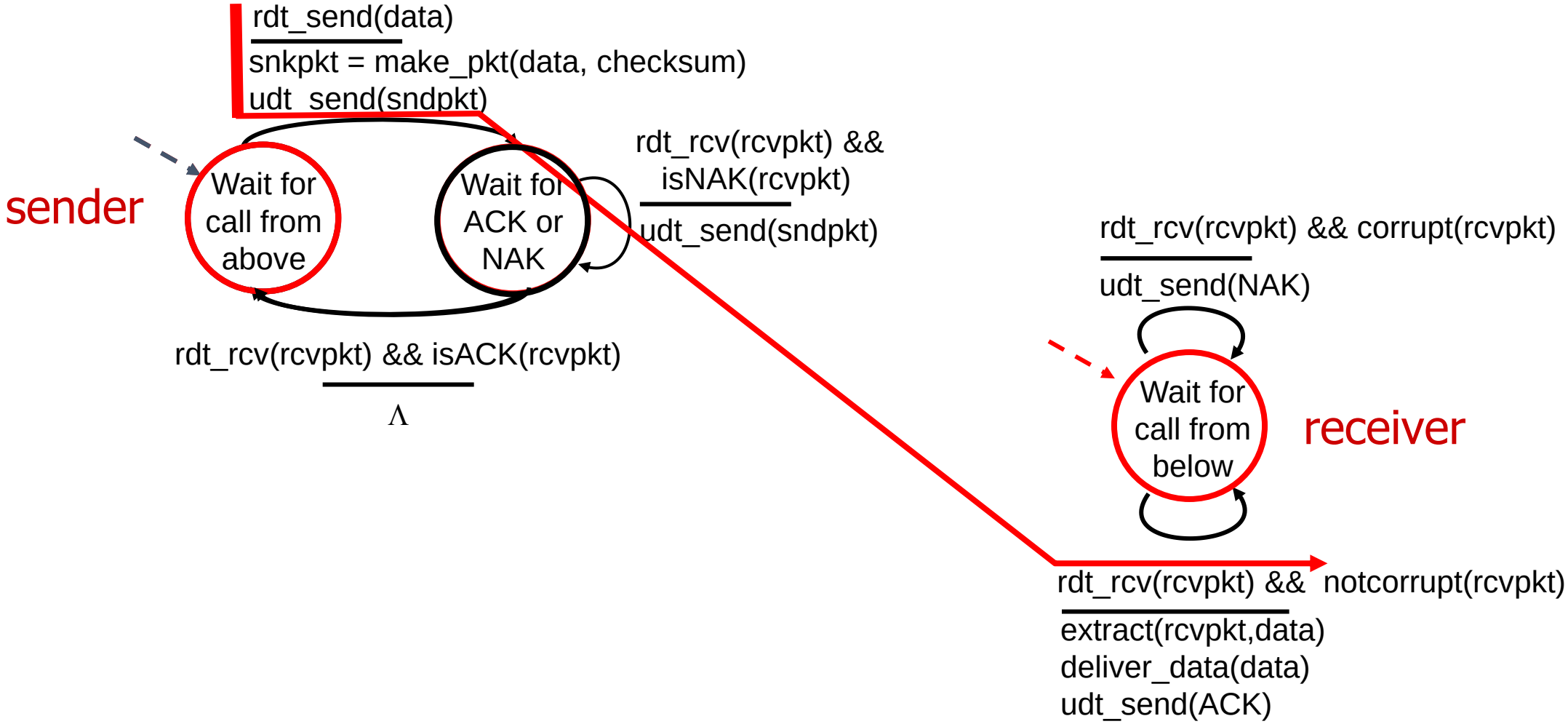




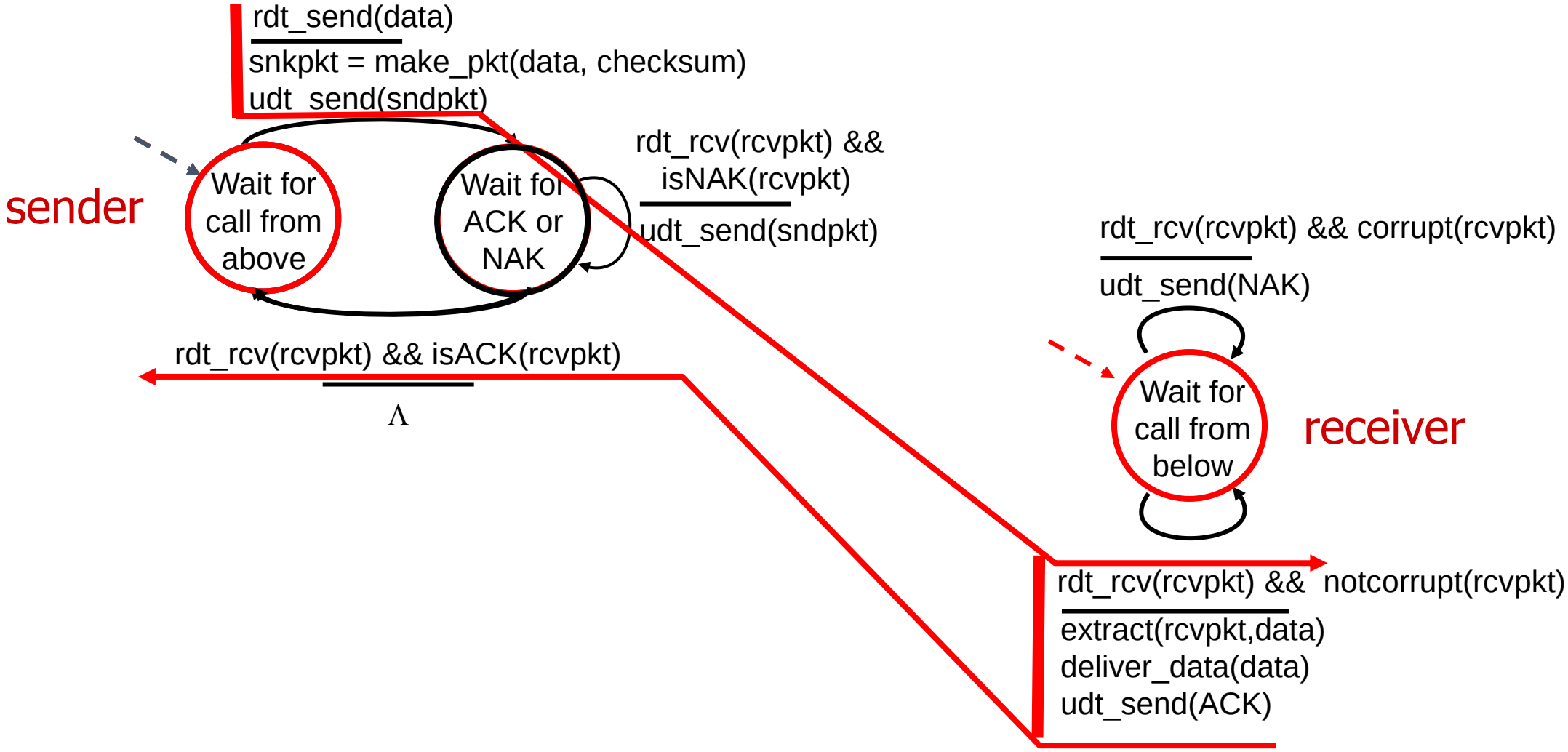
Note: "state" of receiver (did the receiver get my message correctly?) isn't known to sender unless somehow communicated from receiver to sender

- that's why we need a protocol!

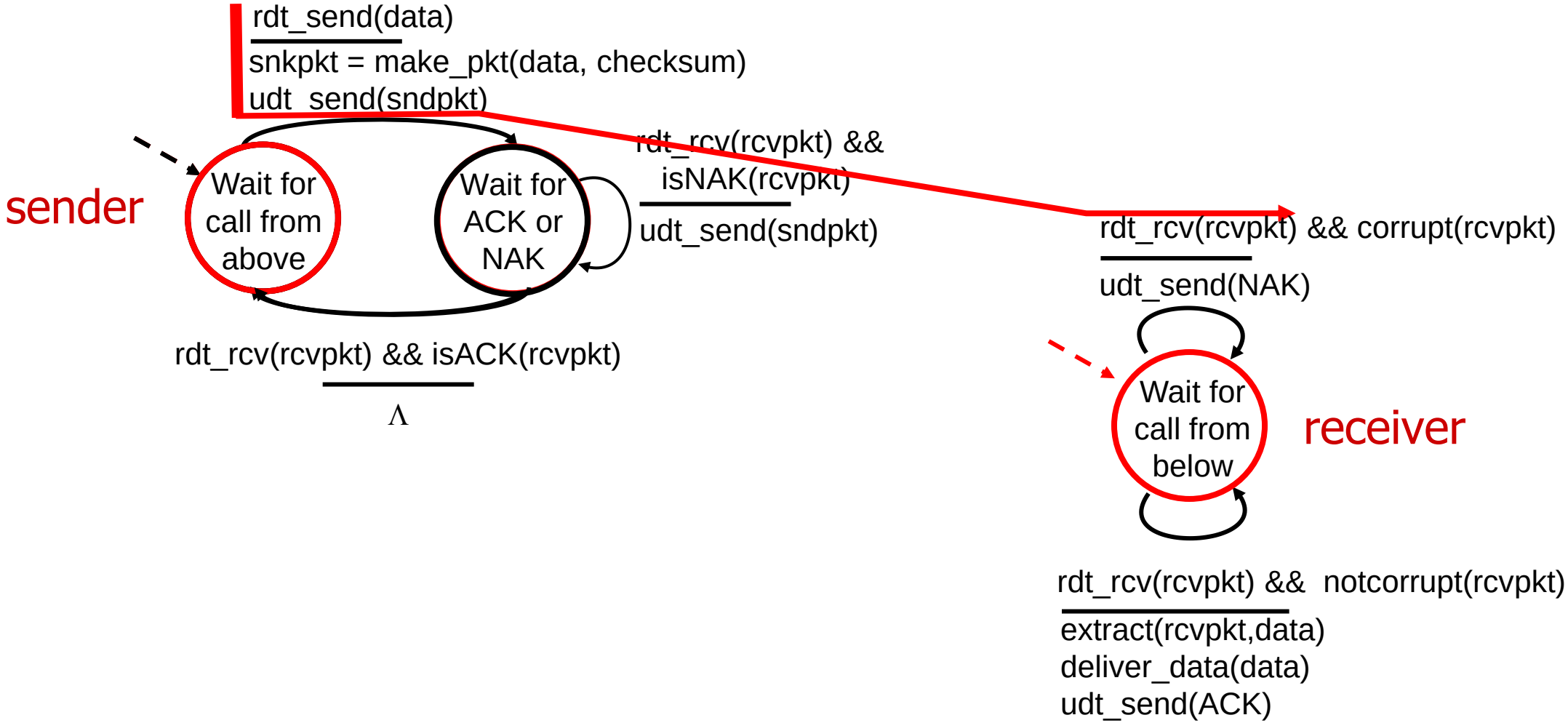
rdt2.0: operation with no errors



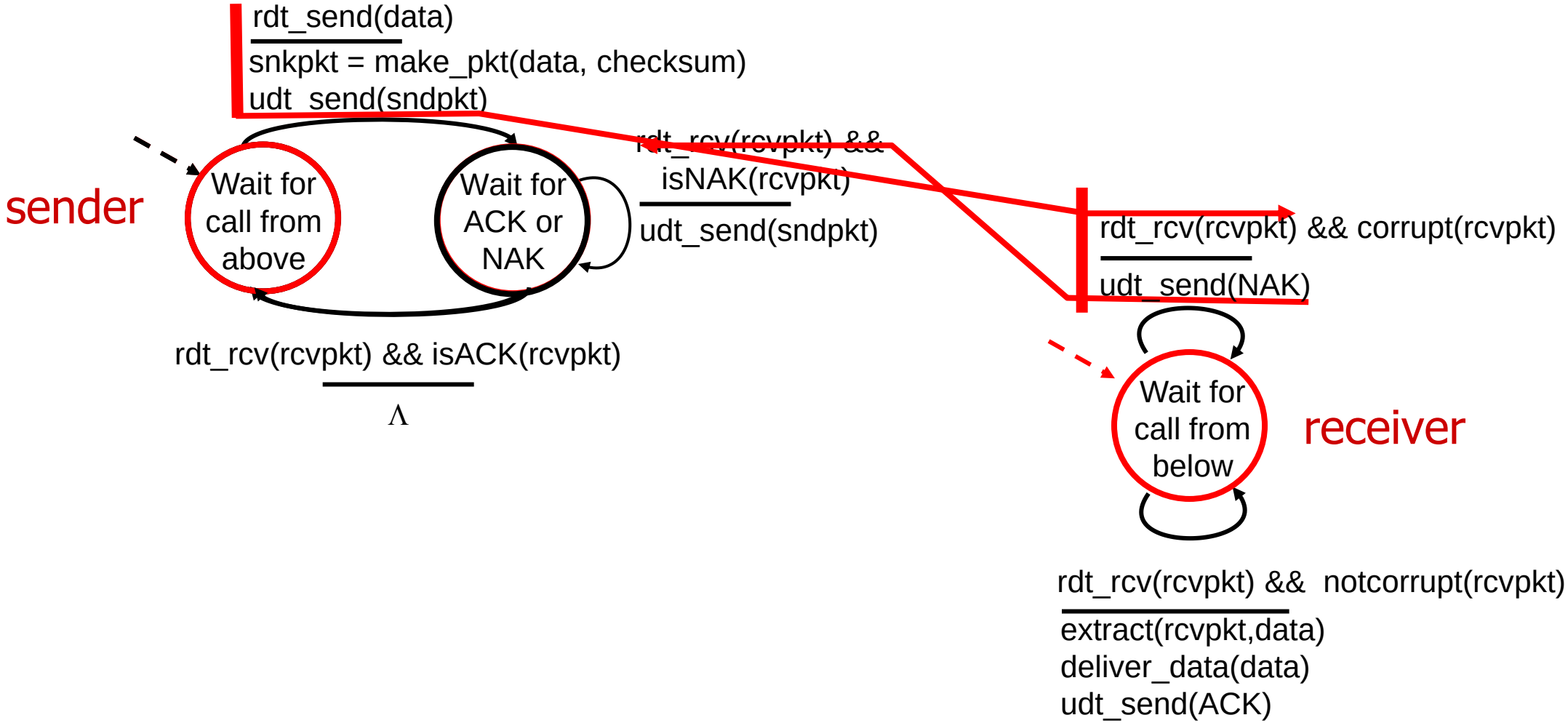
rdt2.0: operation with no errors



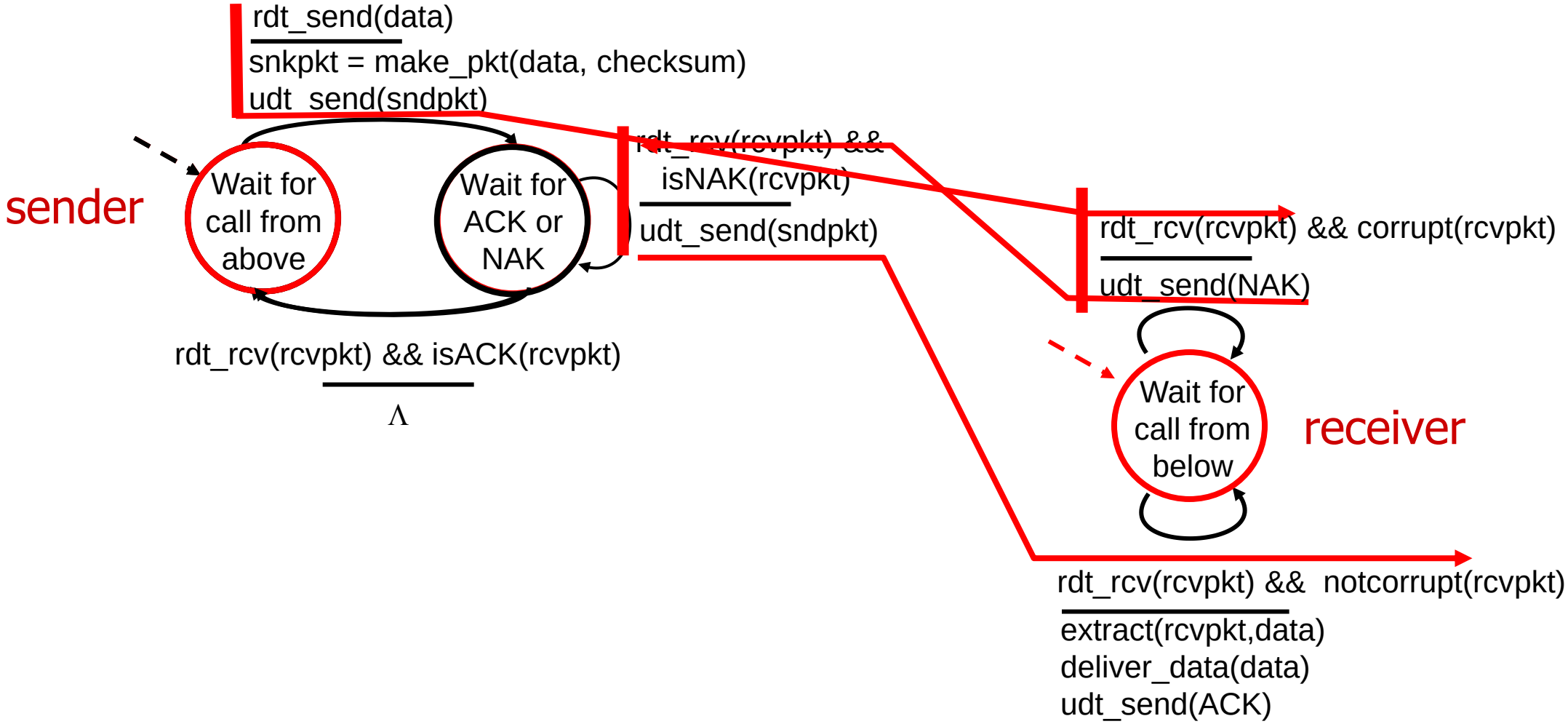
rdt2.0: corrupted packet scenario



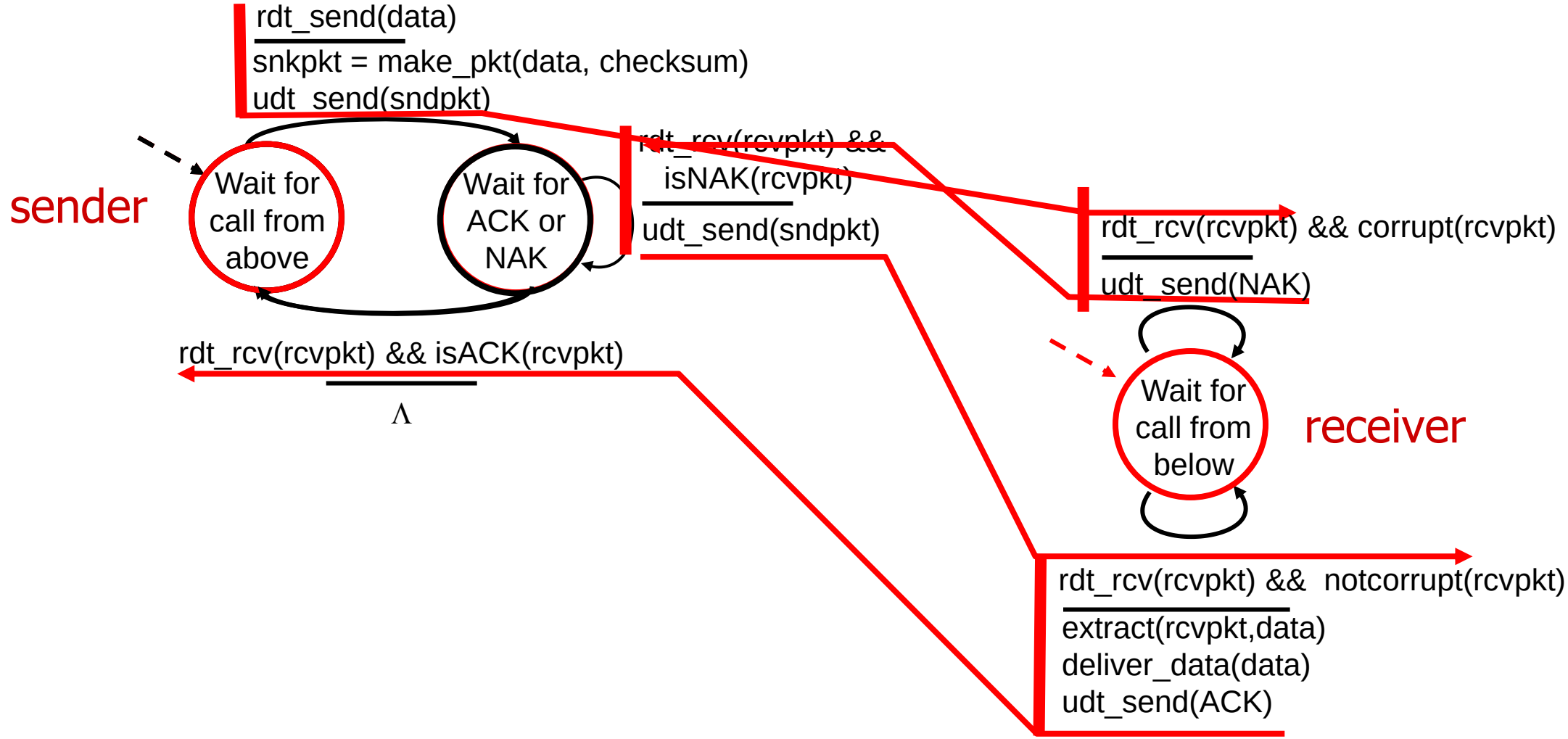
rdt2.0: corrupted packet scenario



rdt2.0: corrupted packet scenario



rdt2.0: corrupted packet 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

rdt2.0 has a fatal flaw!

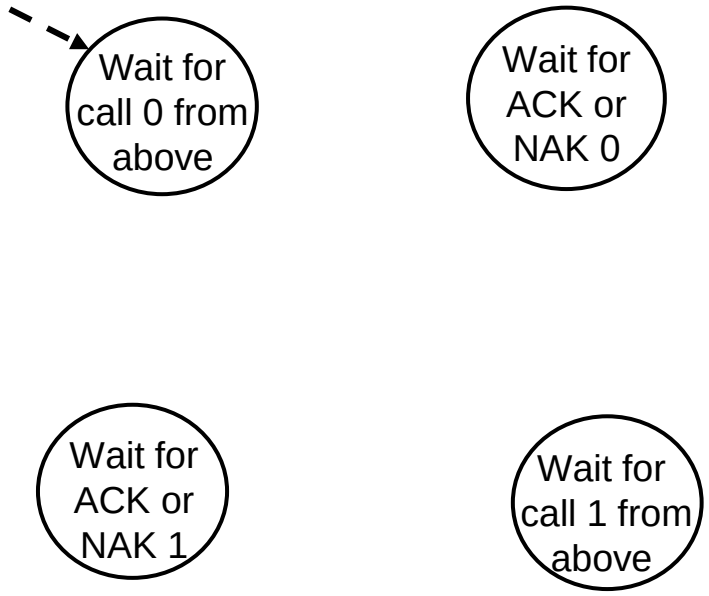
handling duplicates:

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

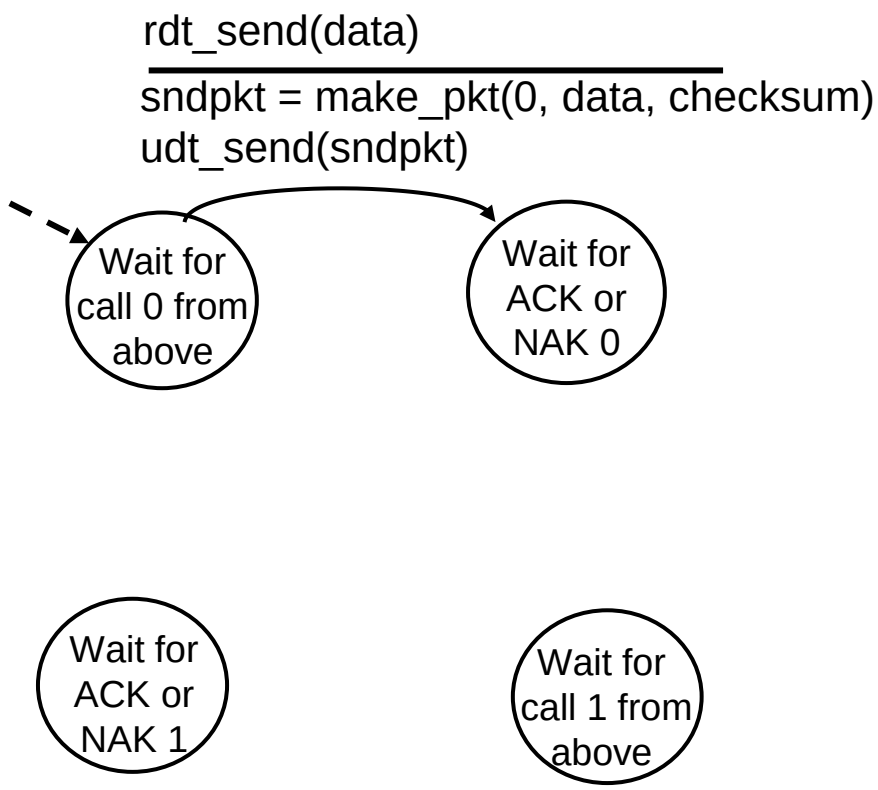
stop and wait

sender sends one packet, then waits for receiver response

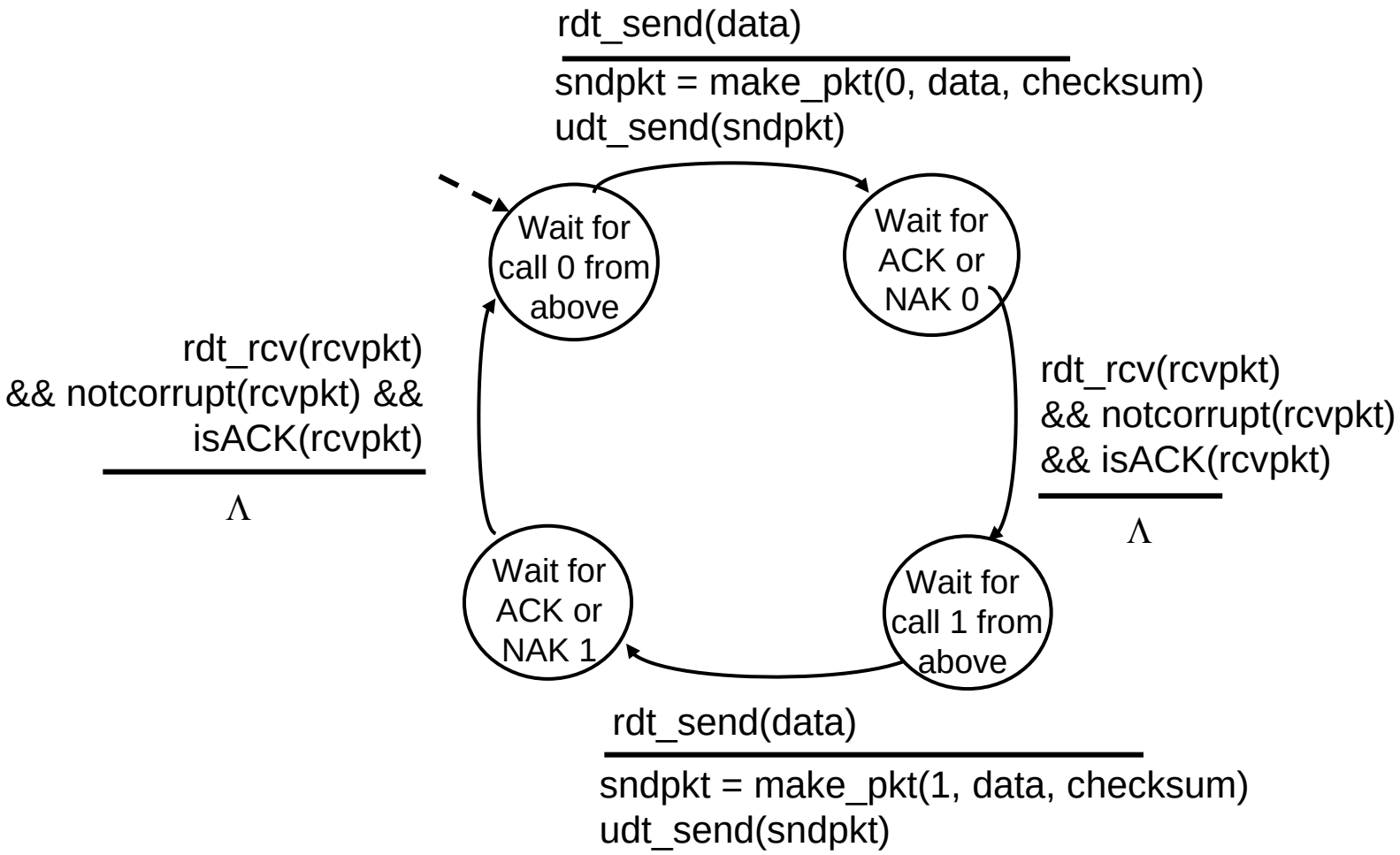
rdt2.1: sender, handling garbled ACK/NAKs



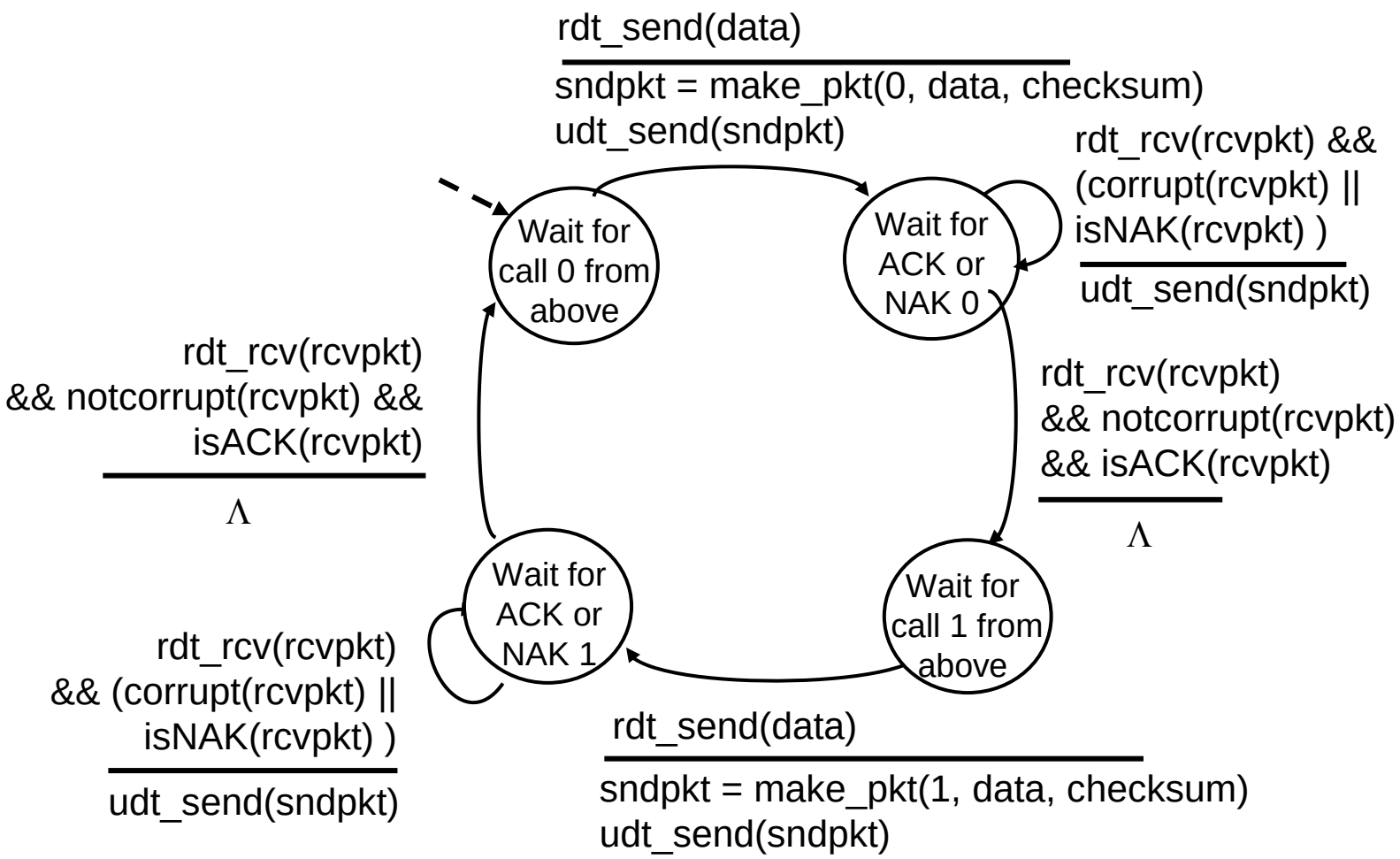
rdt2.1: sender, handling garbled ACK/NAKs



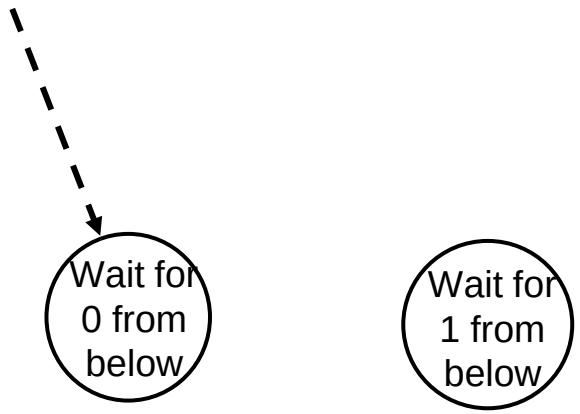
rdt2.1: sender, handling garbled ACK/NAKs



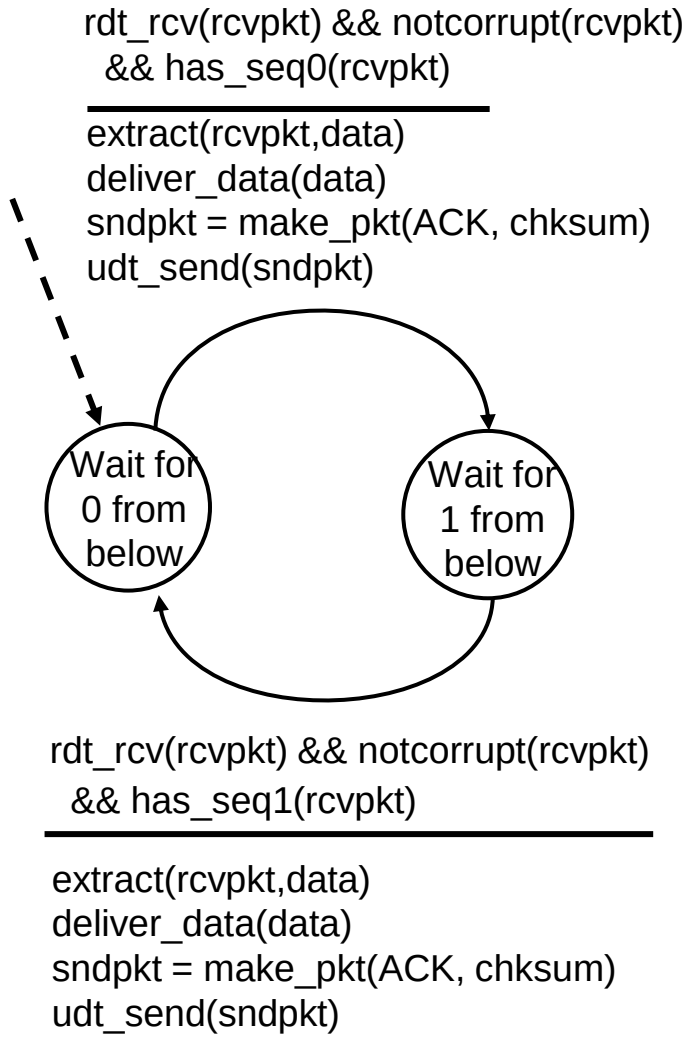
rdt2.1: sender, handling garbled ACK/NAKs



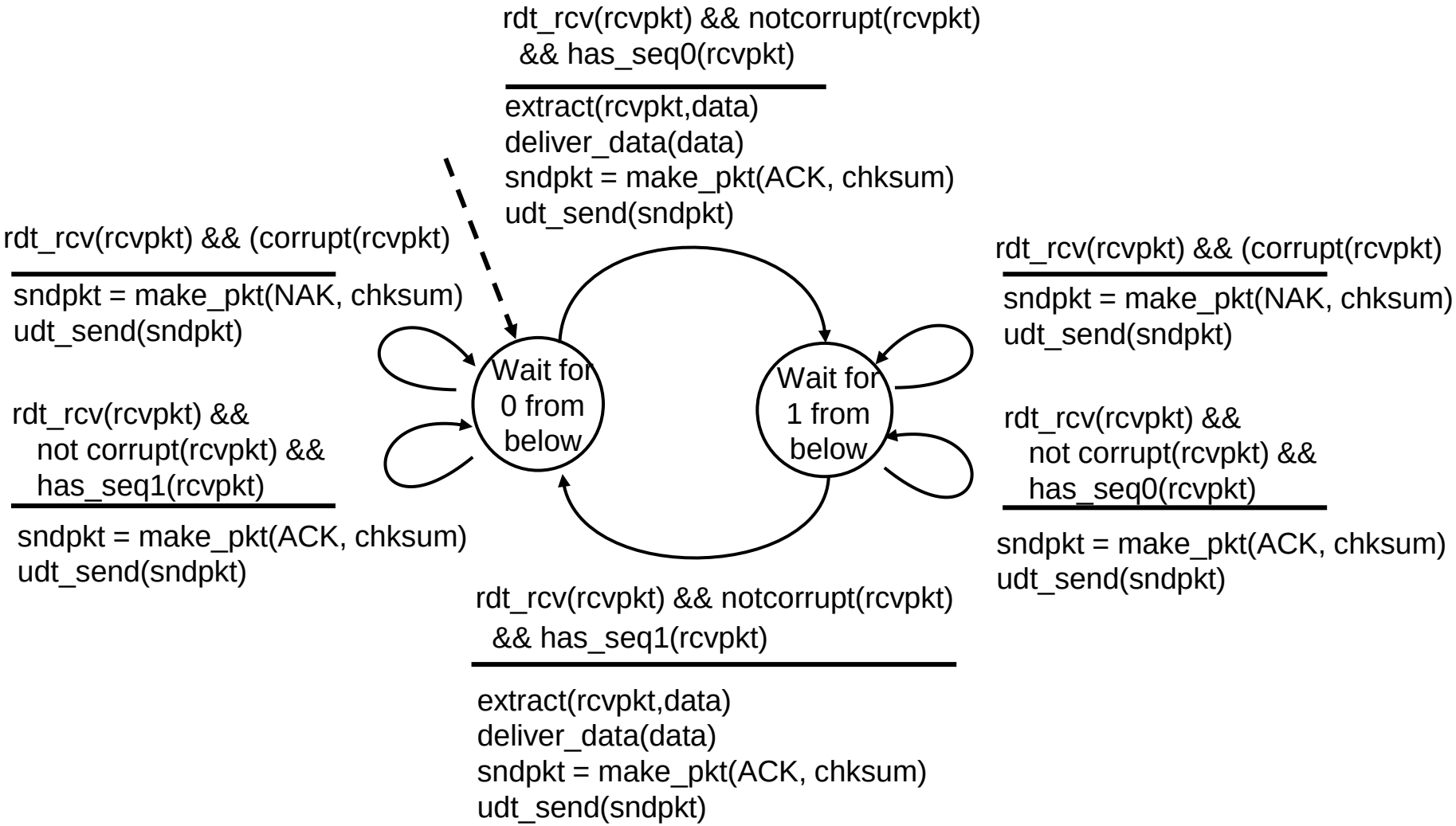
rdt2.1: receiver, handling garbled ACK/NAKs



rdt2.1: receiver, handling garbled ACK/NAKs



rdt2.1: receiver, handling 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*

As we will see, TCP uses this approach to be NAK-free

rdt protocol mechanisms

- Error detection (e.g., checksum)
- ACKs, NAKs
- Retransmission
- Sequence numbers (duplicate detection)

Important Dates

06-21-2024(Friday) – HW 2

06-24-2024(Monday) – Quiz 1

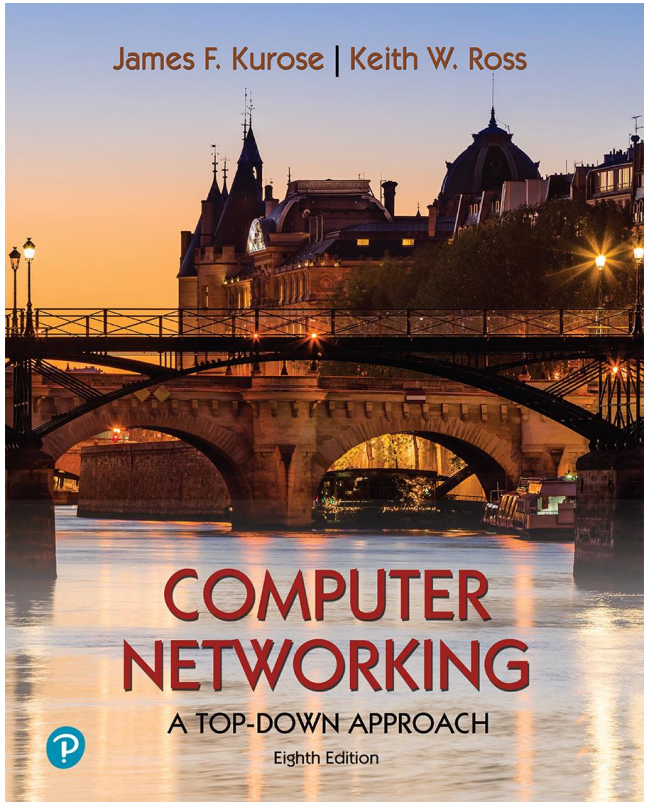
Quiz 1 Guidelines

- 06-24-2024(Monday)– 11:00 AM-12:15 PM
- Please arrive by 11 AM and leave at least two seats vacant between you and other students while taking the quiz.
- The quiz is closed-book, but one side of an 8 ½'' by 11'' sheet may be used, and the quiz is worth 100 points.(Only formulas are allowed)
- The quiz will consist of three main questions, each with different sub-questions.
- The quiz will cover all the concepts

  From LectureD_1_Chapter_1

  To LectureD_8_Chapter_3
- No other electronic devices are allowed except a calculator for the quiz.

Copyright Information



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

The Slides are adapted from,