Transport Layer

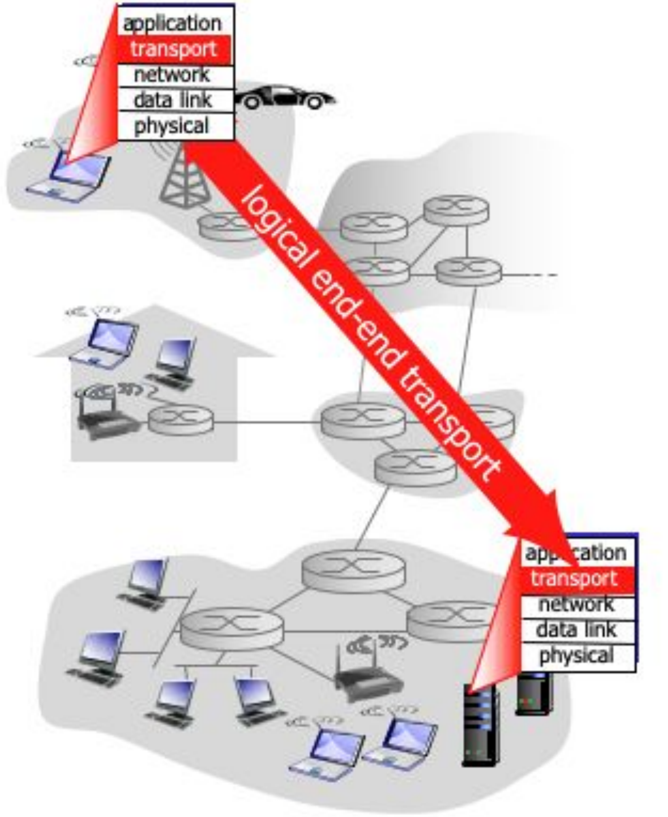
CS5700 Fall 2019

Agenda

- Transport layer services
- UDP
- Reliable data transfer
- TCP
- Congestion control

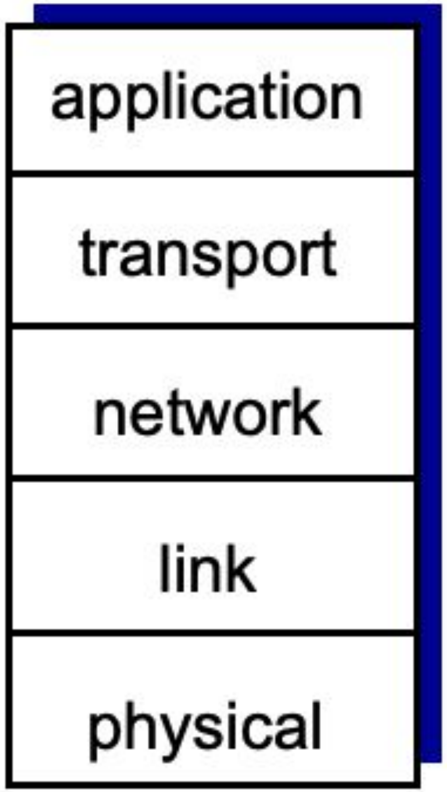
Transport services and protocols

- Provide logical communication between application processes
- Run in end systems (not the core)
- More than one transport protocol available to applications
  - TCP and UDP



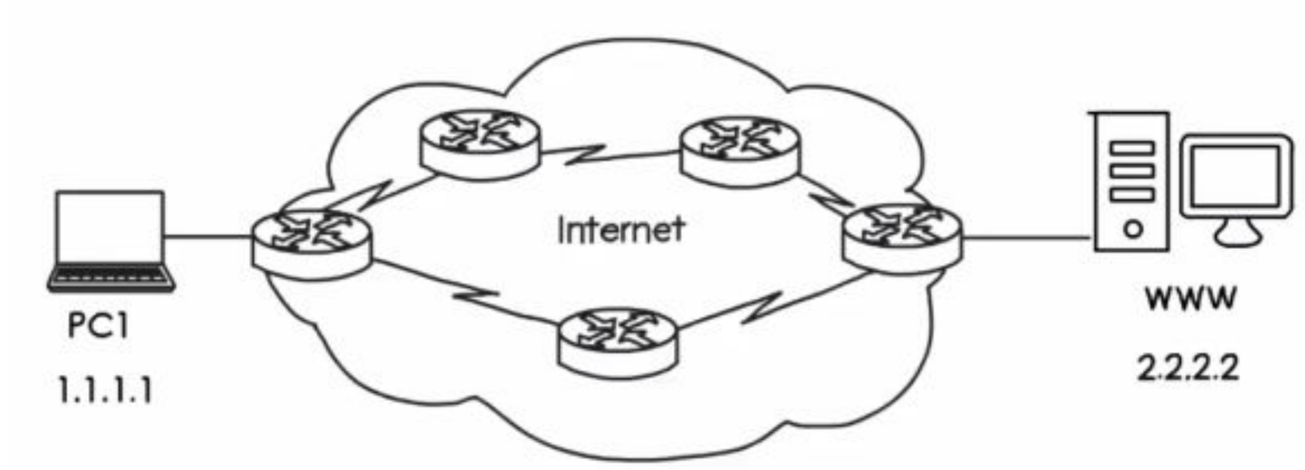
What does network layer do?

- What’s the difference between transport layer and network layer?
- What services are provided by network layer?



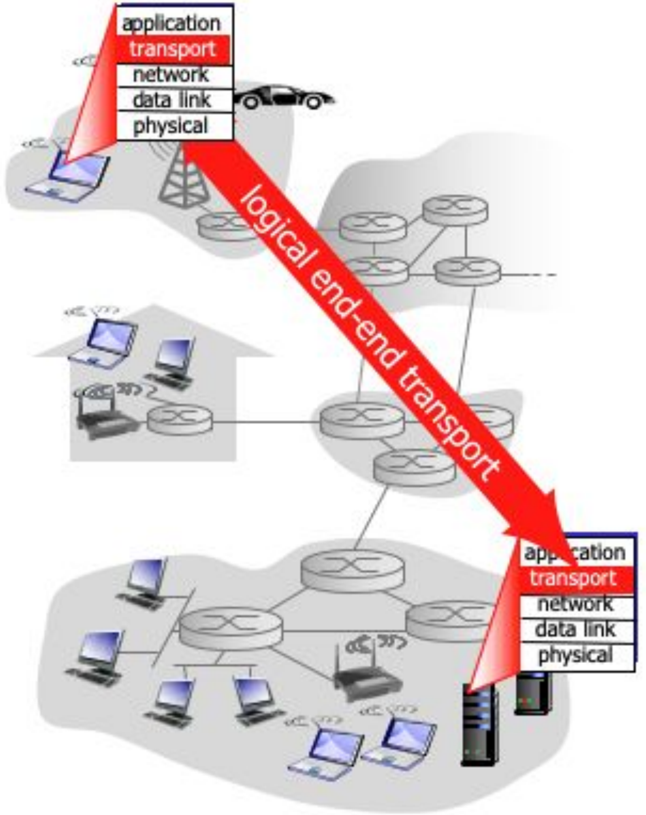
Network layer service model

- Logical communication between hosts
- Every packet is treated individually and separately
- ***Best effort***. No guarantee of delivery.



Transport layer protocols

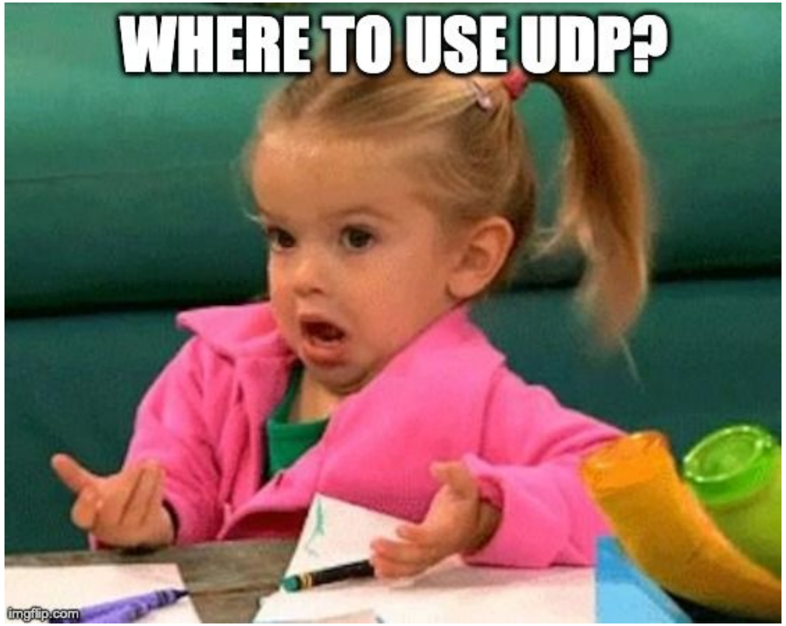
- TCP
  - Reliable in-order delivery
  - Connection oriented
  - Flow control
  - Congestion control
- UDP
- Services not available
  - Delay or bandwidth guarantee



UDP

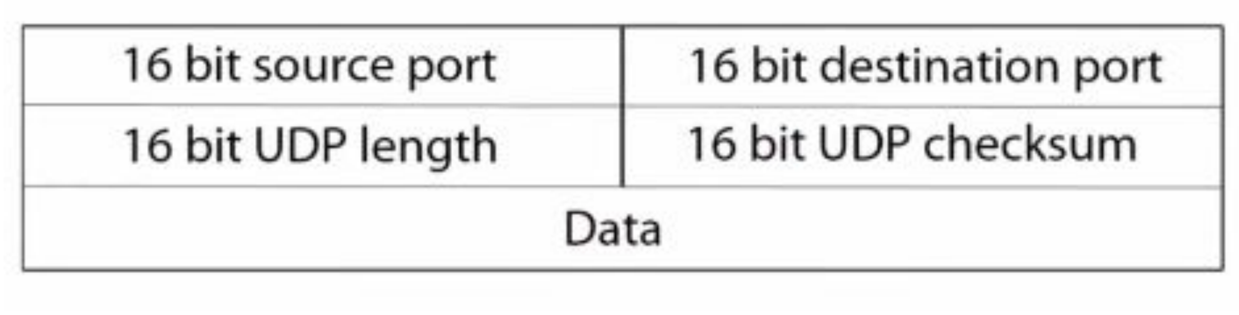
UDP

- User Datagram Protocol
  - Connection less
  - No guarantee of delivery
- Where do you see UDP used? Do you know why?

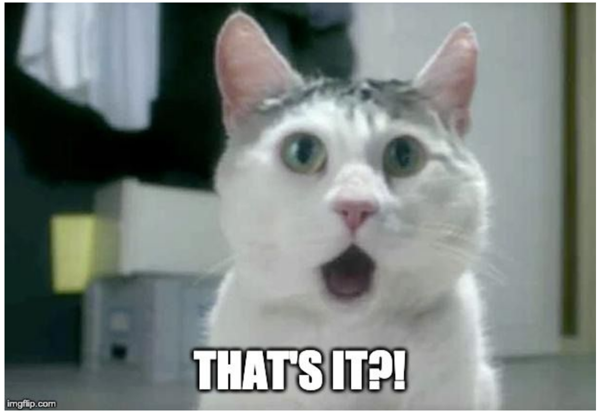


UDP header

- Do you know what's each field for?



16 bit source port	16 bit destination port
16 bit UDP length	16 bit UDP checksum
Data	



UDP checksum

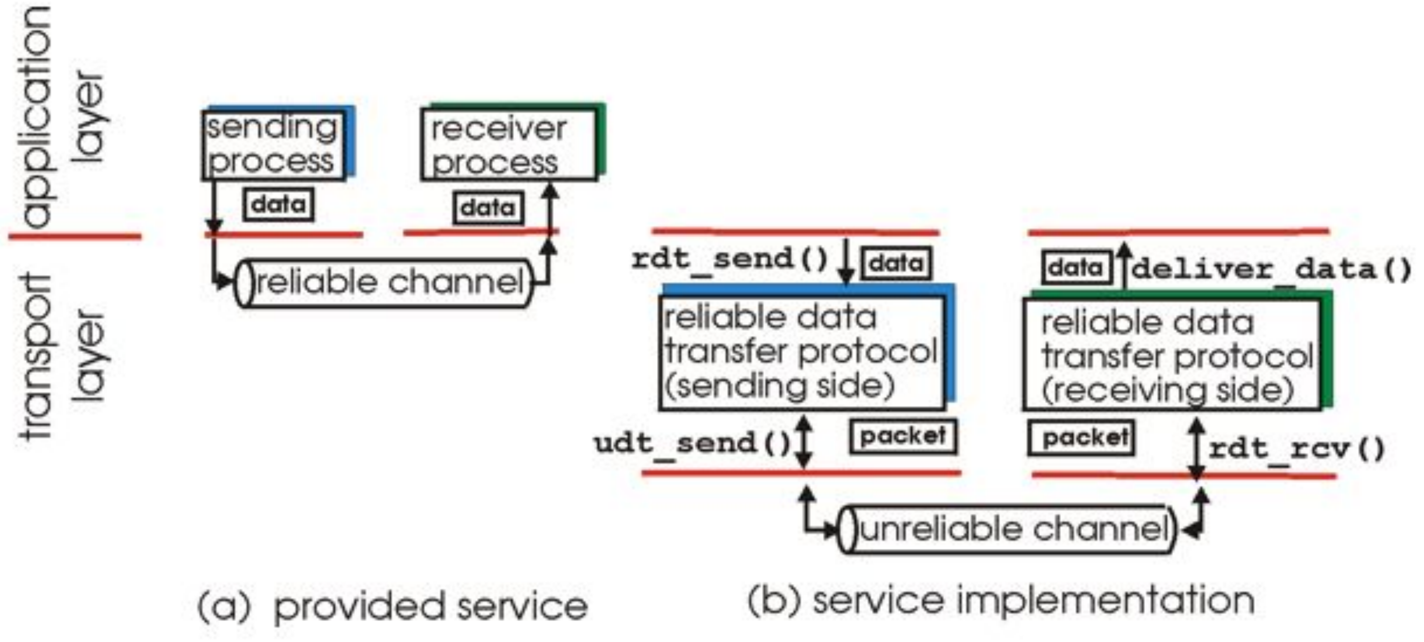
- Detect “errors” (e.g. flipped bits) in transmitted segment
- Sender
  - Treat data (include header) as seq of 16-bit integers
  - Add them up (1’s complement), call it checksum
  - Put checksum into UDP header
- Receiver
  - Same algorithm, compute checksum and compare

UDP socket

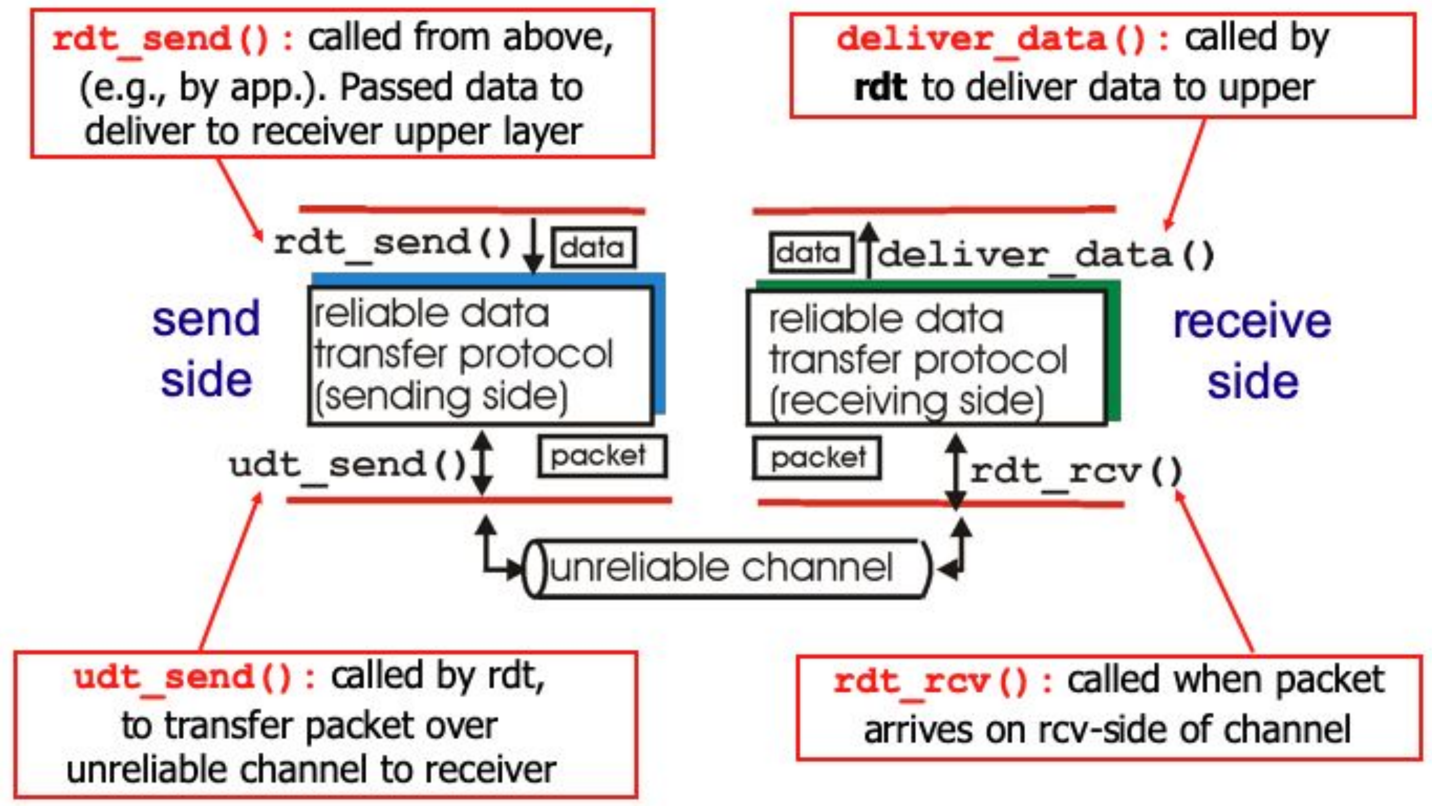
Reliable data transfer

Reliable data transfer

- Important in application, transport, and link layers

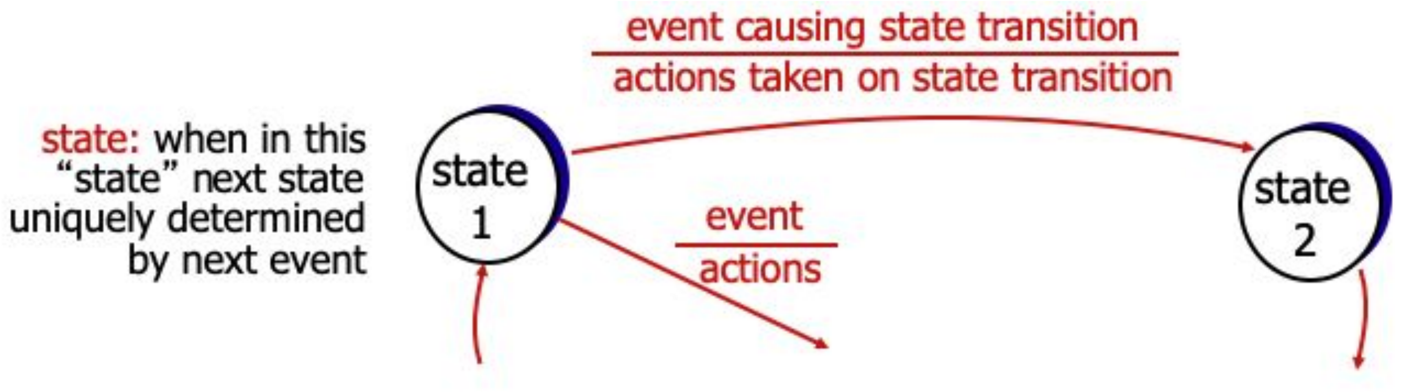


Reliable data transfer


rdt_send(): called from above, (e.g., by app.). Passed data to deliver to receiver upper layer
deliver_data(): called by rdt to deliver data to upper
rdt_send()
data
data
deliver_data()
send side
reliable data transfer protocol (sending side)
reliable data transfer protocol (receiving side)
receive side
udt_send()
packet
packet
rdt_rcv()
unreliable channel
udt_send(): called by rdt, to transfer packet over unreliable channel to receiver
rdt_rcv(): called when packet arrives on rcv-side of channel

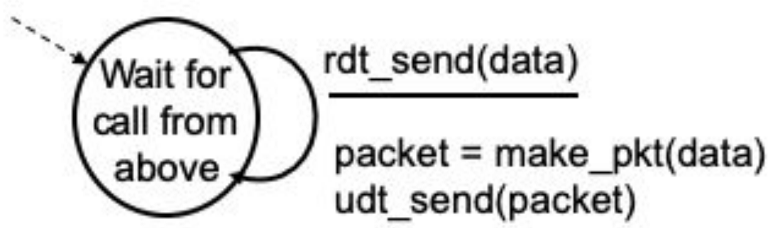
Reliable data transfer

- Incrementally design sender/receiver of rdt
- Consider only unidirectional data transfer
  - But control info will flow on both directions
- Use FSM (finite state machines) to design algorithm

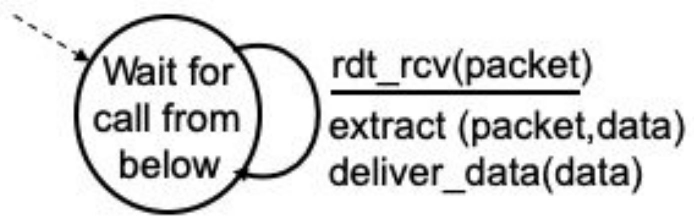

event causing state transition
actions taken on state transition
state: when in this "state" next state uniquely determined by next event
state 1
event
actions
state 2

rdt1.0: over a reliable channel

- Underlying channel is perfectly reliable
  - No bit errors
  - No loss of packets



sender



receiver

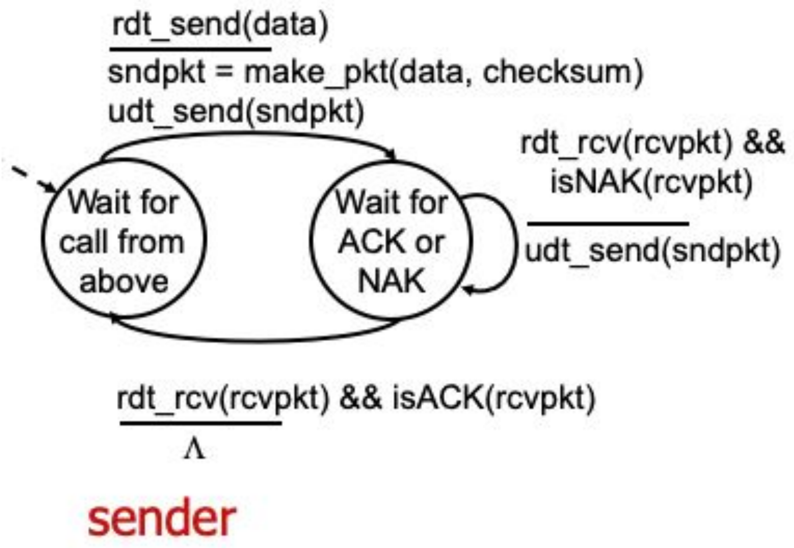
rdt2.0: channel with bit errors

- Underlying channel may flip bits in packet
  - Checksum to detect bit errors
- How to recover from errors?

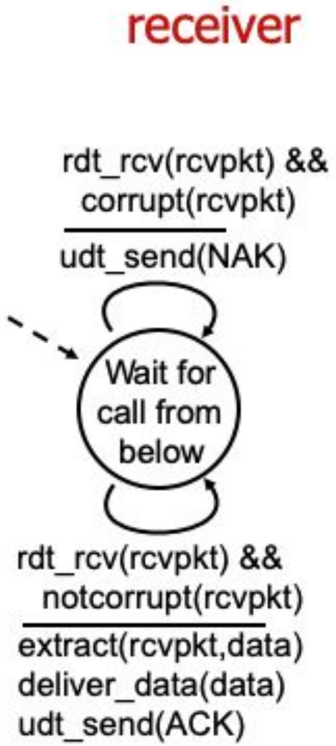
rdt2.0: channel with bit errors

- ACK (acknowledgement)
  - Receiver explicitly tells sender that pkt received OK
- NAK (negative acknowledgement)
  - Receiver explicitly tells sender that pkt had errors
- Sender needs to retransmit pkt on receipt of NAK
- Summary
  - Error detection
  - Feedback with control message ACK and NAK

rdt2.0: FSM



rdt_send(data)
sndpkt = make_pkt(data, checksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) &&
isNAK(rcvpkt)
udt_send(sndpkt)
Wait for call from above
Wait for ACK or NAK
rdt_rcv(rcvpkt) && isACK(rcvpkt)
Λ
sender



receiver
rdt_rcv(rcvpkt) &&
corrupt(rcvpkt)
udt_send(NAK)
Wait for call from below
rdt_rcv(rcvpkt) &&
notcorrupt(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
udt_send(ACK)

rdt2.0: anything looks wrong?

rdt2.0

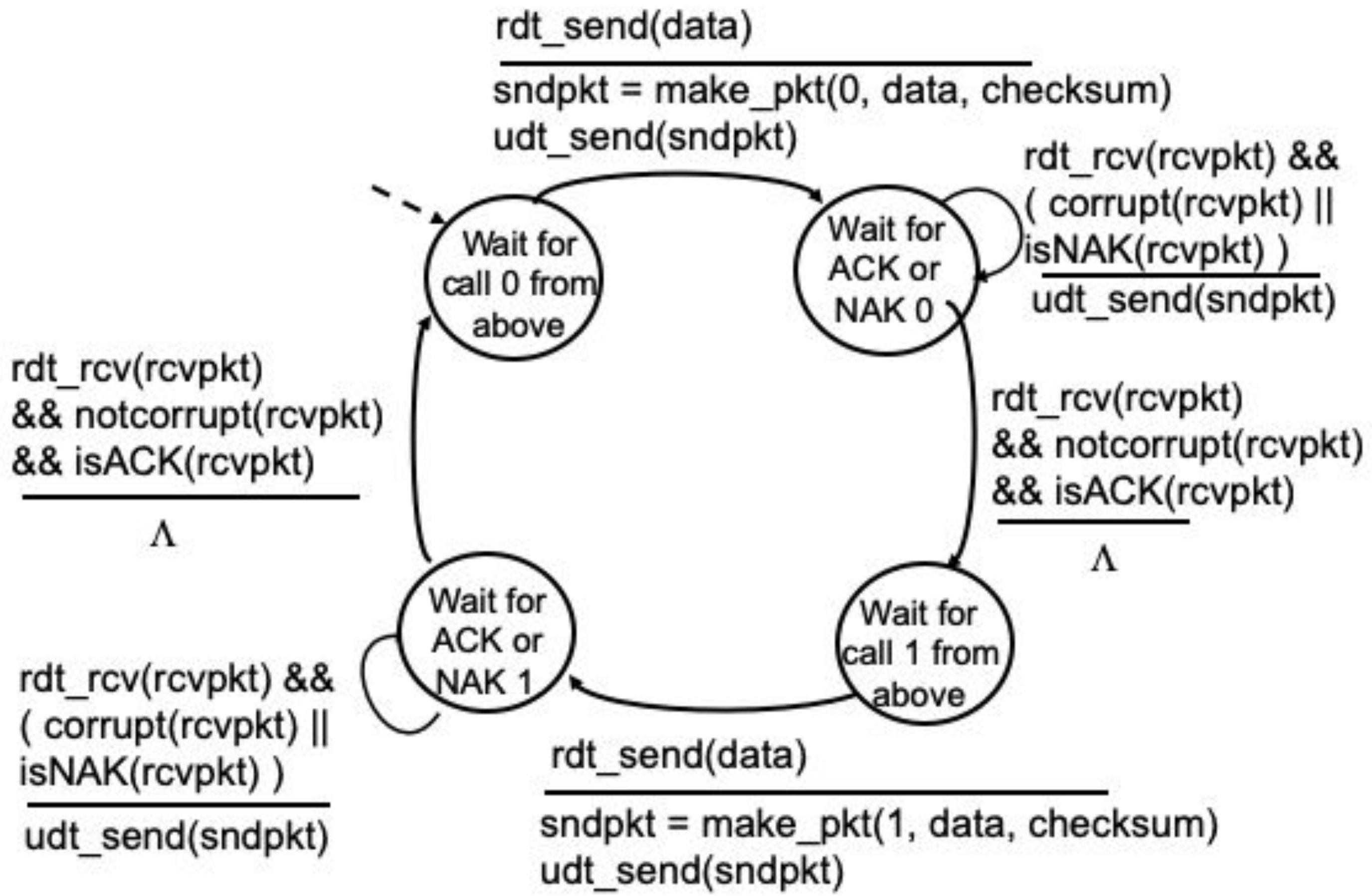
- What happens if ACK or NAK is corrupted?
    - Sender doesn’t know what happened at receiver
- Can sender just retransmit?

rdt2.0

- Receiver needs to handle duplicates when sender retransmit
- Need to use sequence number!
- Stop and wait algorithms
    - Sequence number either 0 or 1

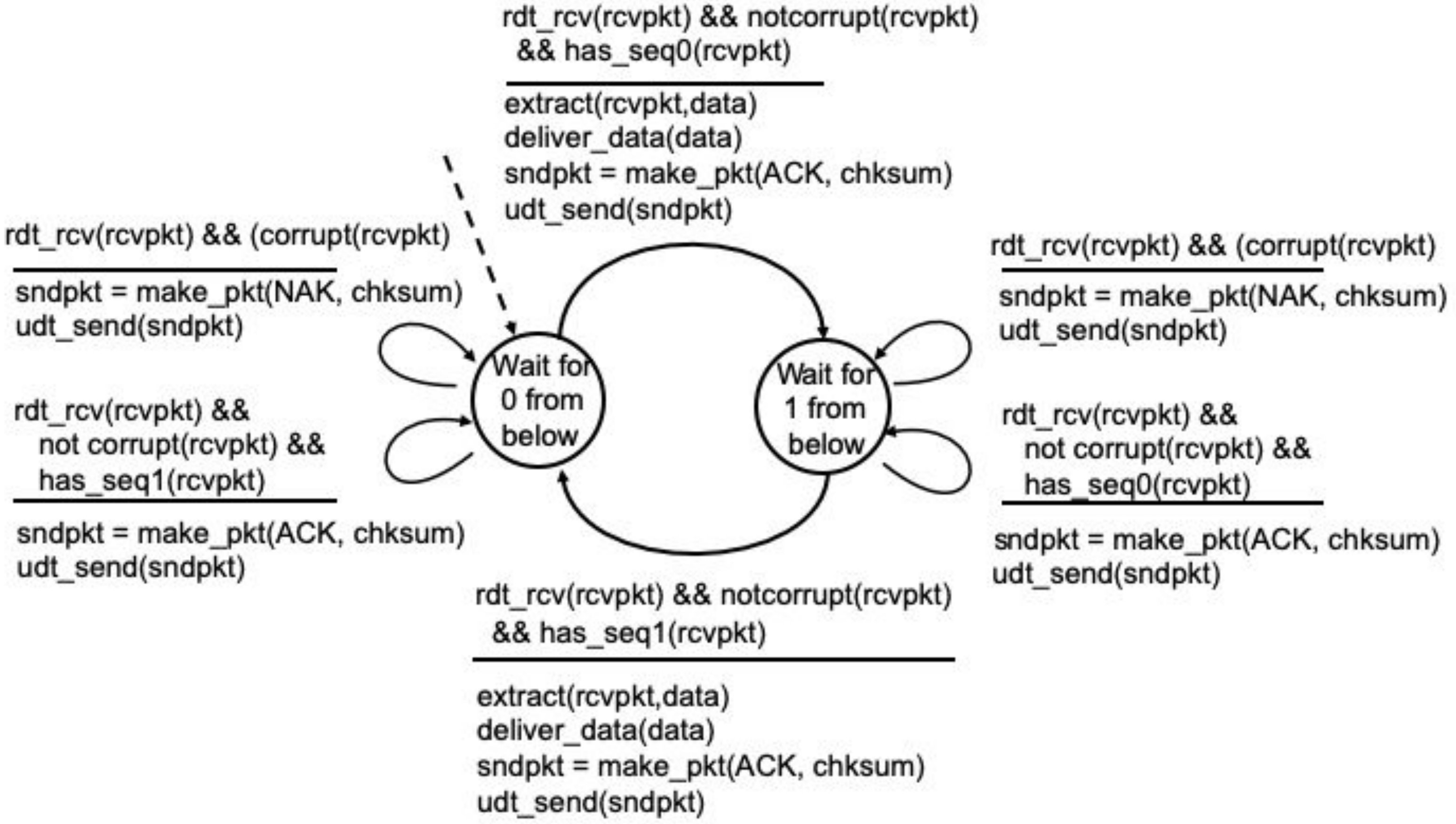
rdt2.1

Sender



rdt2.1

Receiver


rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq0(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)
sndpkt = make_pkt(NAK, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && not corrupt(rcvpkt) && has_seq1(rcvpkt)
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)
Wait for 0 from below
Wait for 1 from below
rdt_rcv(rcvpkt) && (corrupt(rcvpkt)
sndpkt = make_pkt(NAK, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && not corrupt(rcvpkt) && has_seq0(rcvpkt)
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq1(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt = make_pkt(ACK, chksum)
udt_send(sndpkt)

rdt2.1: summery

- Sender
  - Add sequence number to packets (either 0 or 1)
  - Retransmit if receives NAK
  - Retransmit if ACK/NAK is corrupted
- Receiver
  - Check if received packet is duplicate (use seq #)
  - Send ACK or NAK for each packet