Chapter 3: roadmap

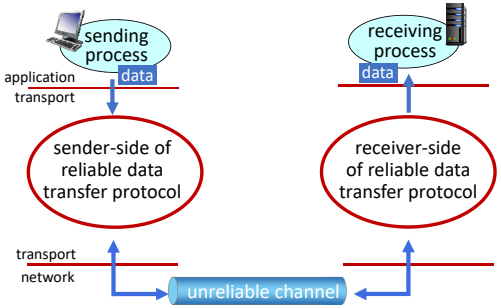
- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
- Principles of congestion control
- TCP congestion control
- Evolution of transport-layer functionality

Principles of reliable data transfer

reliable service *abstraction*

Principles of reliable data transfer

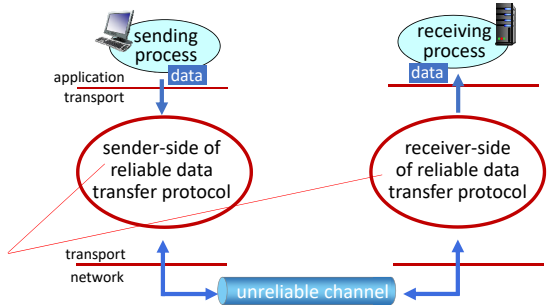
reliable service *abstraction*



reliable service *implementation*

Principles of reliable data transfer

Design (and complexity) of reliable data transfer protocol will depend strongly on characteristics of unreliable channel (corrupt, loss, reorder data?)

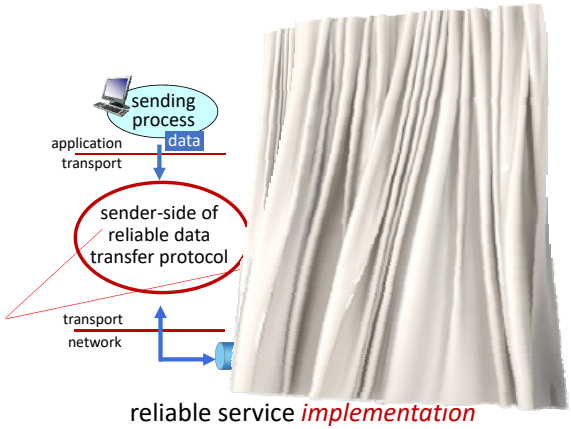


reliable service *implementation*

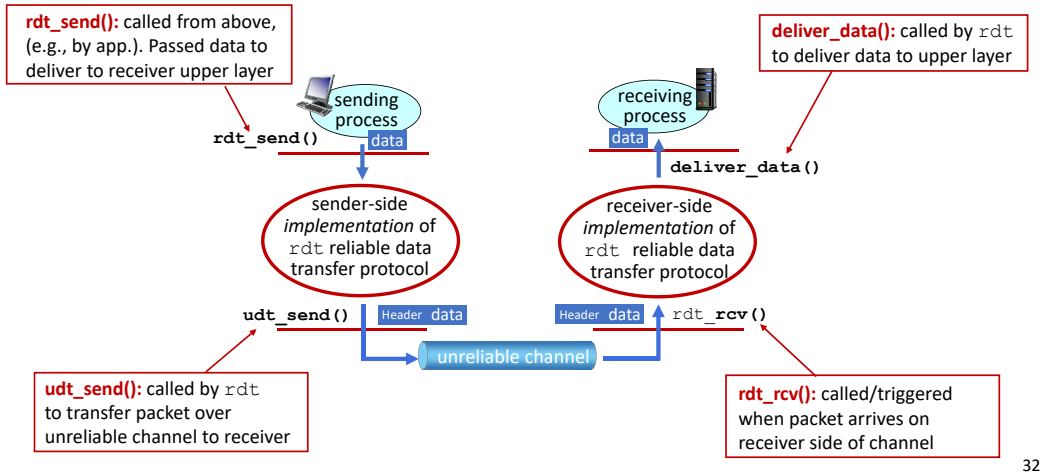
Principles of reliable data transfer

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

- unless communicated via a message

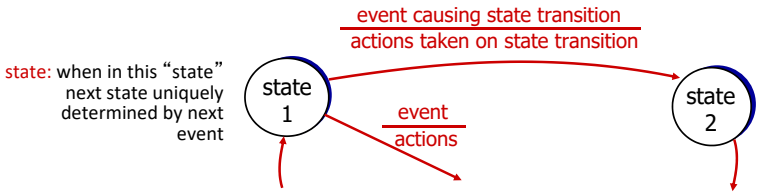


Reliable data transfer protocol (rdt): interfaces



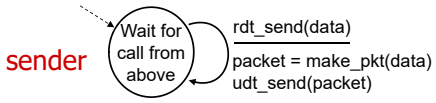
Reliable data transfer: getting started

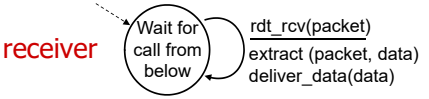
- We will incrementally develop sender and receiver sides of reliable data transfer protocol (rdt)
  - from simple to complex
- consider only unidirectional data transfer
  - but control information will flow in both directions!
- use finite state machines (FSM) to specify sender and receiver



rdt1.0: reliable transfer over a reliable channel

- underlying (L3) channel is perfectly reliable
  - no bit errors
  - no loss of packets
  - all packets will arrive in order
- *separate* FSMs for sender, receiver:
  - sender sends data into underlying channel
  - receiver reads data from underlying channel





rdt2.0: channel with forward bit errors (but w/o loss)

- 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?

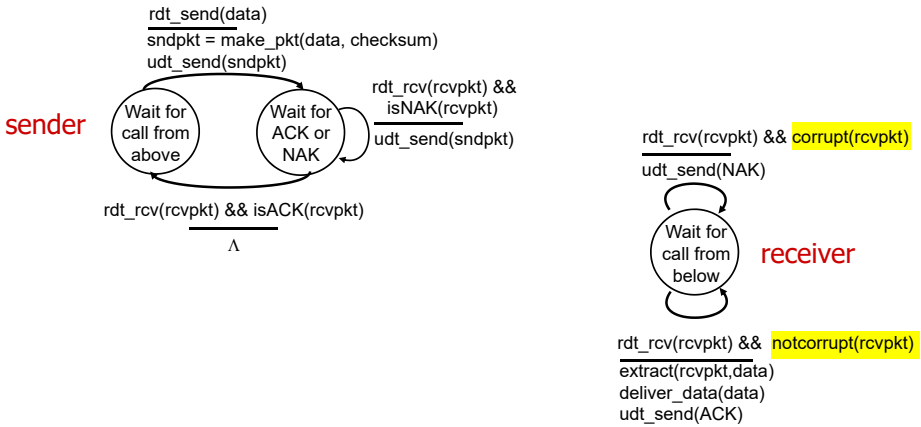
rdt2.0: channel with forward bit errors (but w/o loss)

- underlying channel may flip bits in packet
  - checksum to detect bit errors
- how to know whether bit errors occur and then get recovered?
  - *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

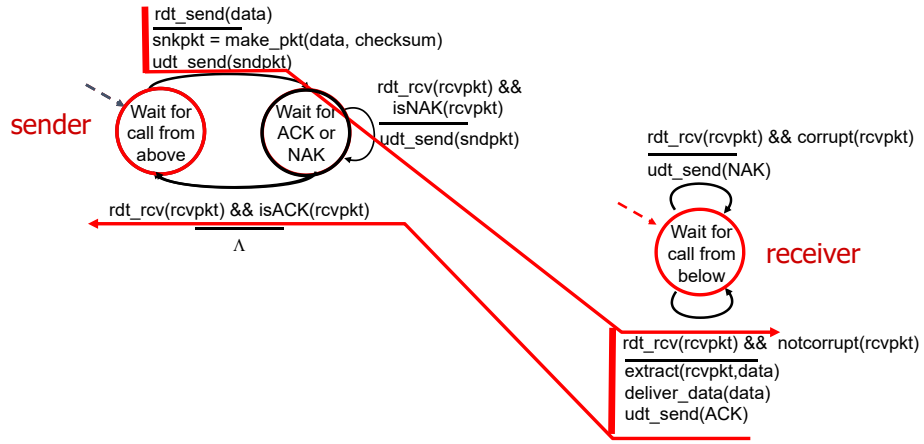
stop and wait

sender sends one packet, then waits for receiver response

rdt2.0: FSM specifications



rdt2.0: operation with no errors



rdt2.0: corrupted packet scenario