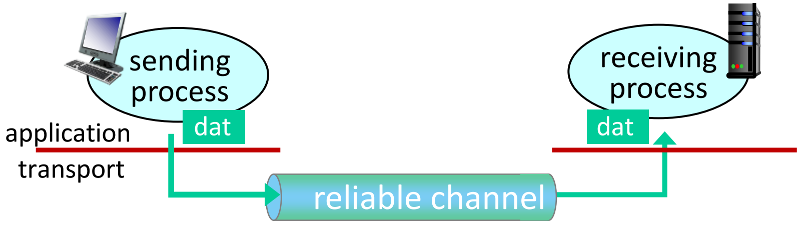
Chapter 3: roadmap

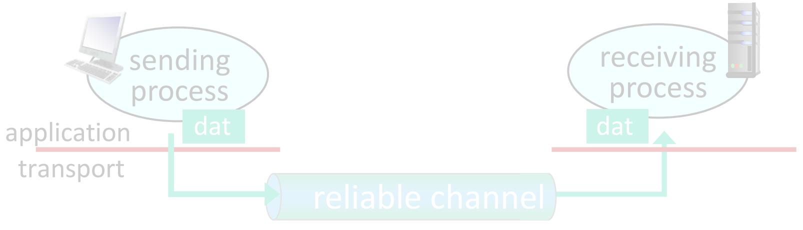
- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
- Principles of congestion control
- TCP congestion control
- Evolution of transport-layer functionality

Principles of reliable data transfer



reliable service *abstraction*

Principles of reliable data transfer



sending process

receiving process

application

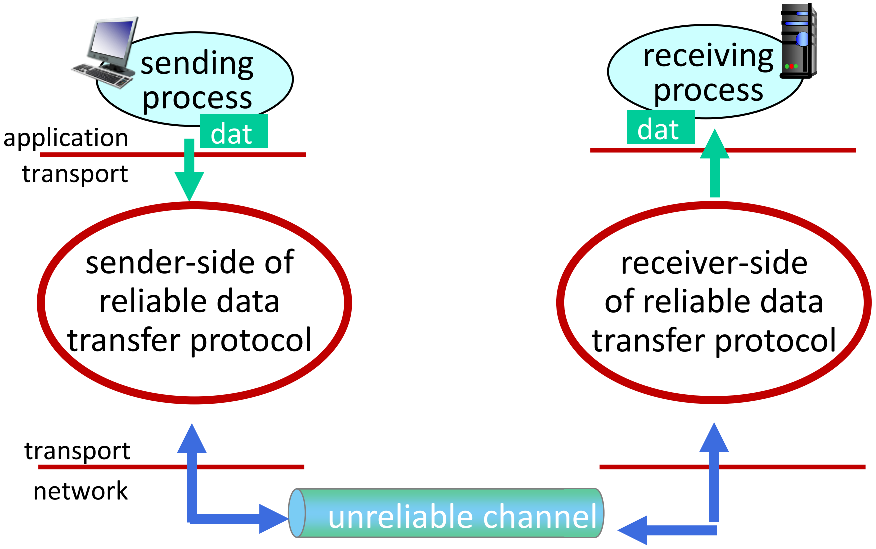
transport

dat

dat

reliable channel

reliable service *abstraction*



sending process

receiving process

application

transport

dat

dat

sender-side of reliable data transfer protocol

receiver-side of reliable data transfer protocol

transport

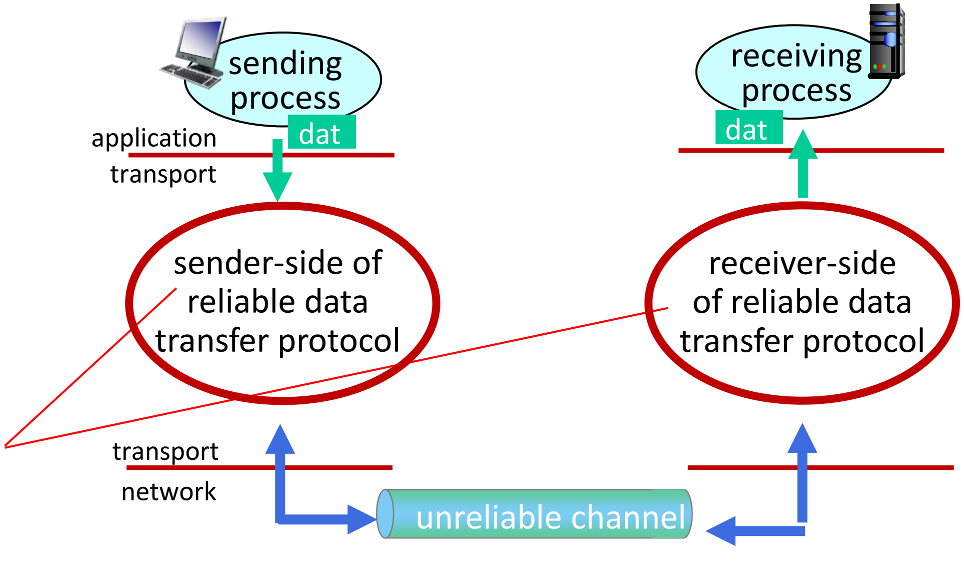
network

unreliable channel

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?)

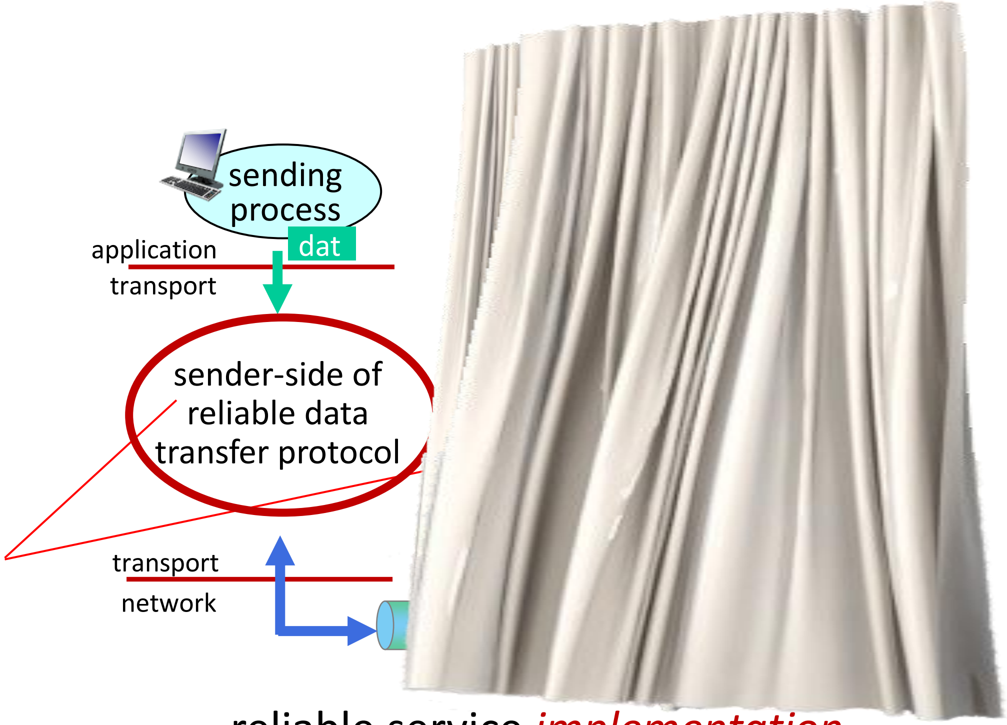

sending process
dat
application
transport
sender-side of reliable data transfer protocol
transport
network
unreliable channel
receiving process
dat
receiver-side of reliable data transfer protocol

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

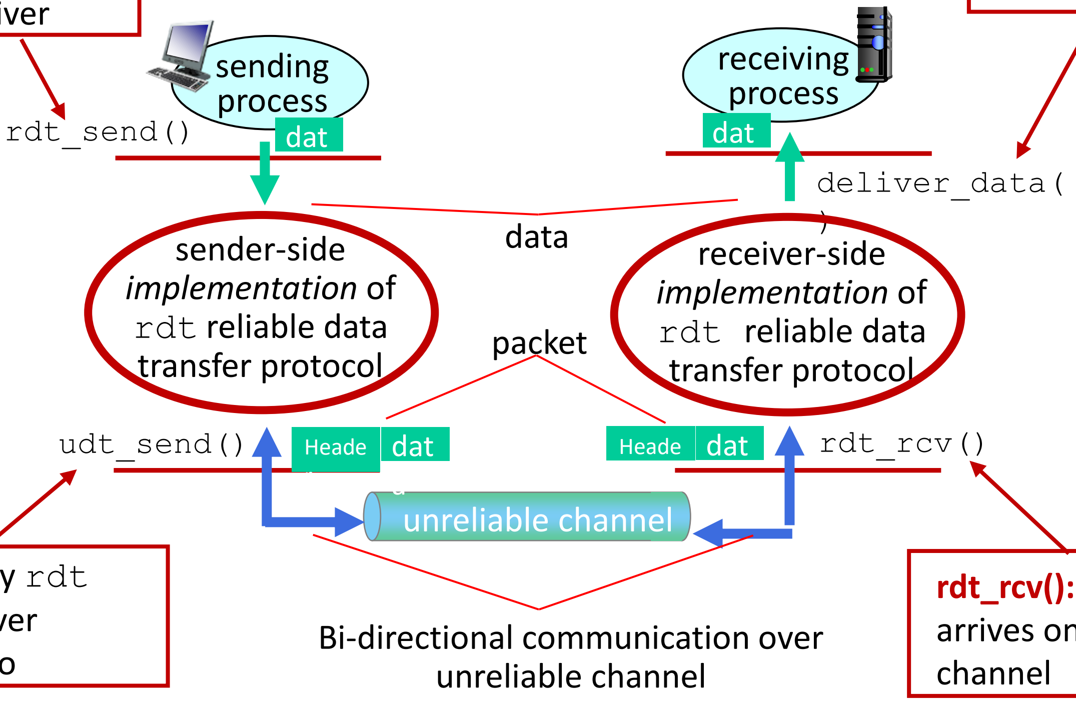


reliable service *implementation*

Reliable data transfer protocol (rdt): interfaces

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 layer


sending process
receiving process
dat
dat
rdt_send()
deliver_data()
data
sender-side *implementation* of rdt reliable data transfer protocol
receiver-side *implementation* of rdt reliable data transfer protocol
packet
udt_send()
Heade
dat
Heade
dat
rdt_rcv()
unreliable channel
Bi-directional communication over unreliable channel

udt_send(): called by `rdt` to transfer packet over unreliable channel to receiver

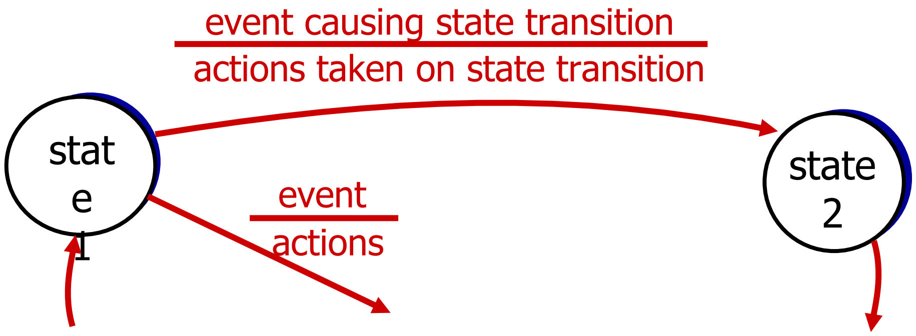
rdt_rcv(): called when packet arrives on receiver side of channel

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

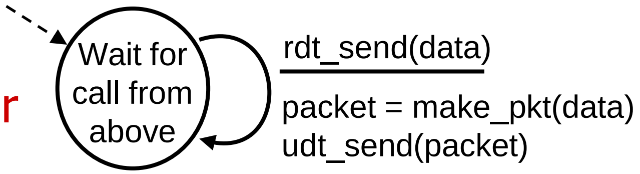
state: when in this "state" next state uniquely determined by next event


event causing state transition
actions taken on state transition
stat
e
1
event
actions
state
2

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

sender

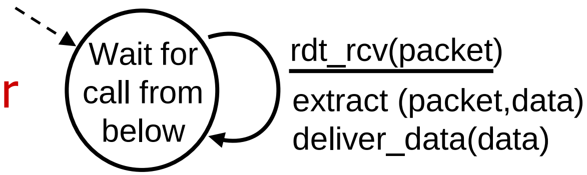


Wait for call from above

rdt_send(data)

packet = make_pkt(data)
udt_send(packet)

receiver



Wait for call from below

rdt_rcv(packet)

extract (packet,data)
deliver_data(data)

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

Announcements

- Term project will be posted on May 22
- Midterm exam 2: next week
- Final exam: June 12

Homework #1

(Submission of your answers is entirely ***optional****. Feel free to e-mail it to me (NOT TA) if you'd like, but there's no separate grade for it and no penalty at all for not submitting.)*

1. What are the key components of an FSM?
2. During our lecture today, we discussed various challenges associated with reliability. Explain the specific error types we need to manage in designing protocols for reliable communications
3. What does the statement below mean?

   "Sender, receiver do *not* know the "state" of each other"