COMP 431
Internet Services & Protocols

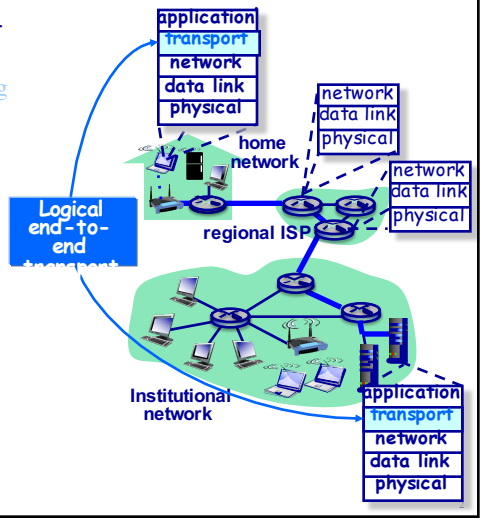
The Transport Layer
Principles of Reliable Data Delivery

Jasleen Kaur

Feb 20, 2020

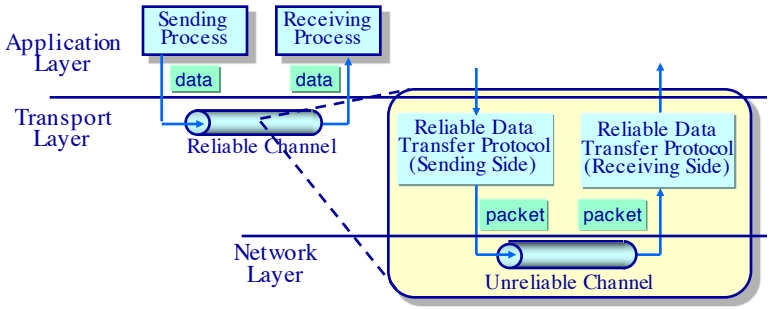
Transport Layer Protocols & Services
Outline

- Fundamental transport layer services
  - » Multiplexing/Demultiplexing
  - » Error detection
  - » Reliable data delivery
  - » Pipelining
  - » Flow control
  - » Congestion control
- Service implementation in Internet transport protocols
  - » UDP
  - » TCP



Fundamental Transport Layer Services

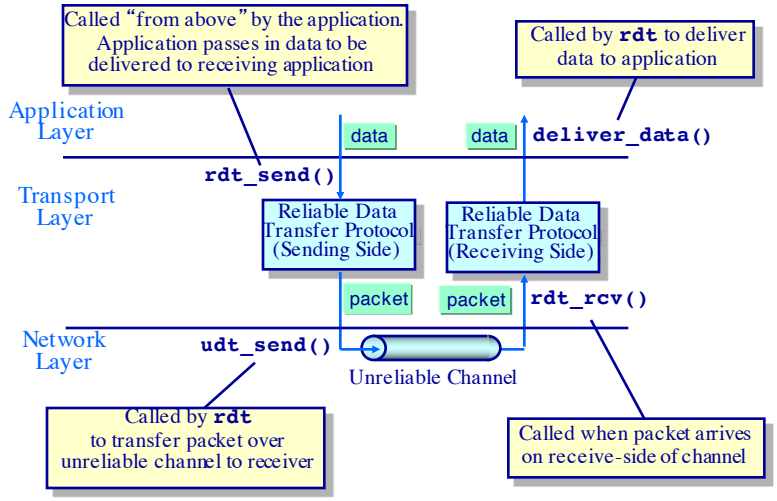
Principles of reliable data transfer



- Goal: Provide a reliable channel abstraction
  - » The characteristics of the underlying channel will determine the complexity of providing reliable communications
- Issues: *State* required at sender and receiver and number of *control messages* exchanged

Reliable Data Transfer

Programming interfaces

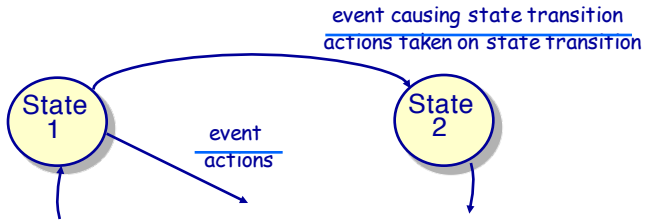


RDT reliable data transfer
UDT unreliable data transfer

Reliable Data Transfer

Protocol specification method

- Use finite state machines to specify sender and receiver algorithms
  - » When in a given state, the next state (and actions) are uniquely determined by the next event

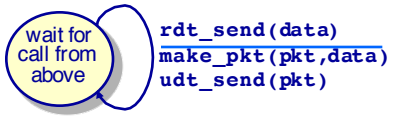


Reliable Data Transfer Protocol 1.0

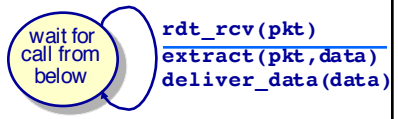
Reliable transfer over a reliable channel

- The underlying channel is assumed to be perfectly reliable
  - » No bit errors
  - » No loss of packets

- Sender state machine

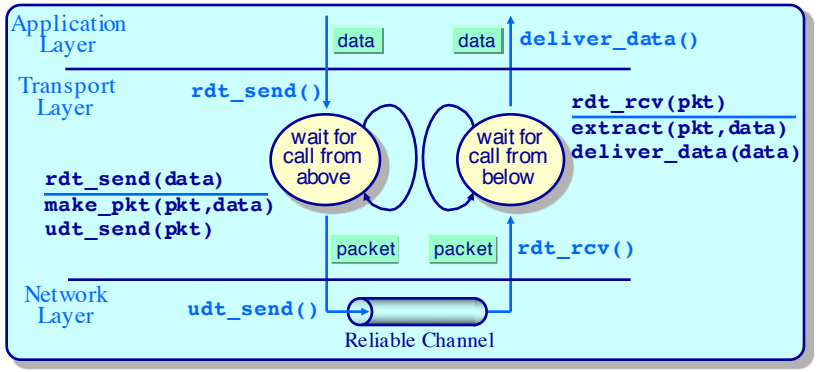


- Receiver state machine



Reliable Data Transfer Protocol 1.0

Programming interfaces



- This is the complete protocol under the assumption of a reliable network channel

Reliable Data Transfer Protocol 2.0

Reliable transfer over a channel with bit errors

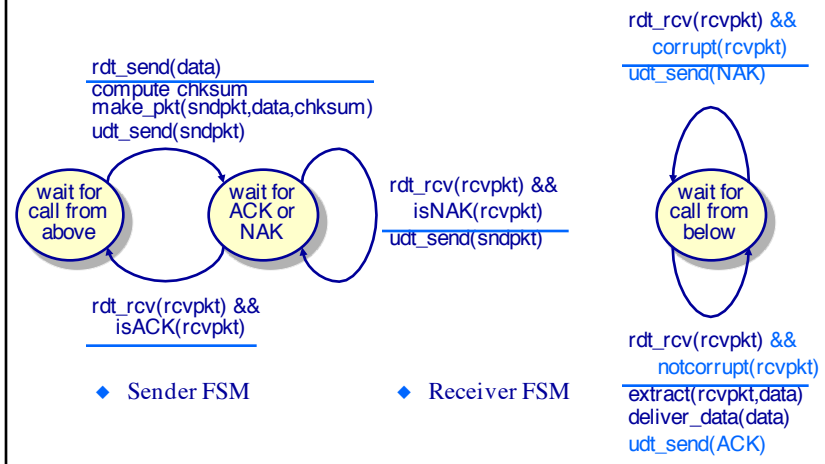
- Now assume the underlying channel may "flip" random bits in a packet
- How to detect errors?
- How to recover from errors:
  - » *acknowledgements* (*ACKs*) — the receiver explicitly tells the sender that a packet was received OK
  - » *negative acknowledgements* (*NAKs*) — the receiver explicitly tells the sender that a packet had errors
  - » Sender retransmits packet on receipt of NAK
- New mechanisms to deal with bit errors:
  - » Error detection
  - » Control messages (ACK, NAK) from a receiver to the sender
  - » Retransmission

Detect errors with checksum

Sender would have to store a packet until it gets an ack

Reliable Data Transfer Protocol 2.0

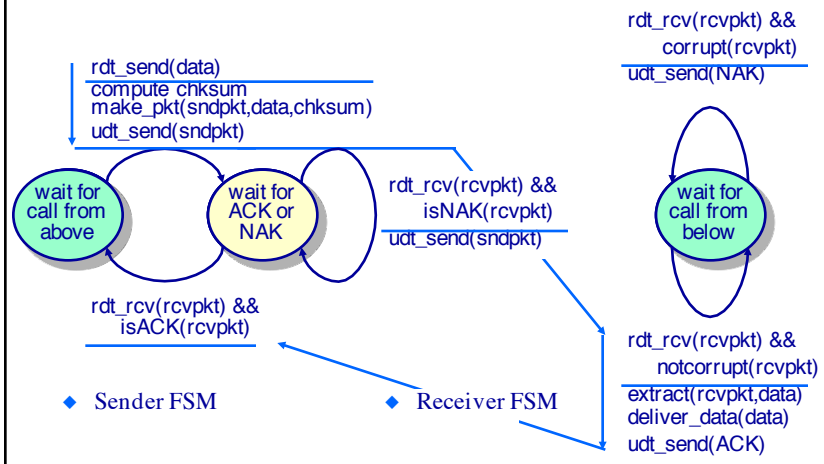
Reliable transfer over a channel with bit errors only



- Sender FSM
- Receiver FSM

Reliable Data Transfer Protocol 2.0

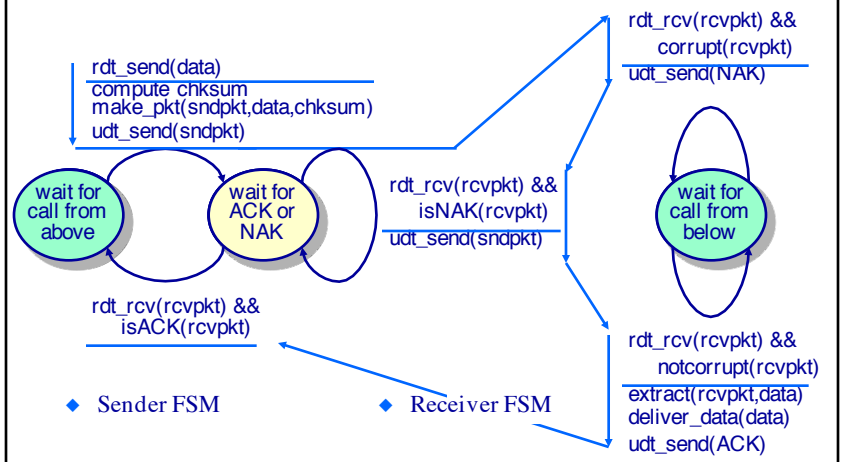
Example 1: No errors occur



- Sender FSM
- Receiver FSM

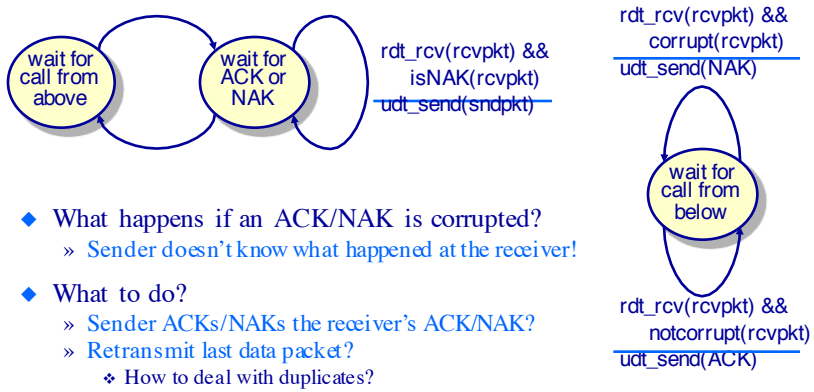
Reliable Data Transfer Protocol 2.0

Example 2: A corrupted packet arrives at the receiver



Reliable Data Transfer Protocol 2.0

Simple… but wrong!



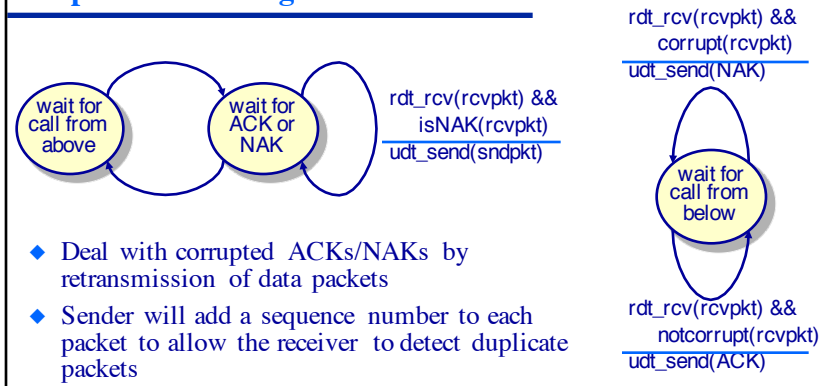
- What happens if an ACK/NAK is corrupted?
  - » Sender doesn't know what happened at the receiver!
- What to do?
  - » Sender ACKs/NAKs the receiver's ACK/NAK?
  - » Retransmit last data packet?
    - ❖ How to deal with duplicates?

Sender and receiver both calculate checksum
look up CRC(cyclic redundancy checksum) online

What if the ack is corrupted?

Reliable Data Transfer Protocol 2.0

Simple… but wrong!

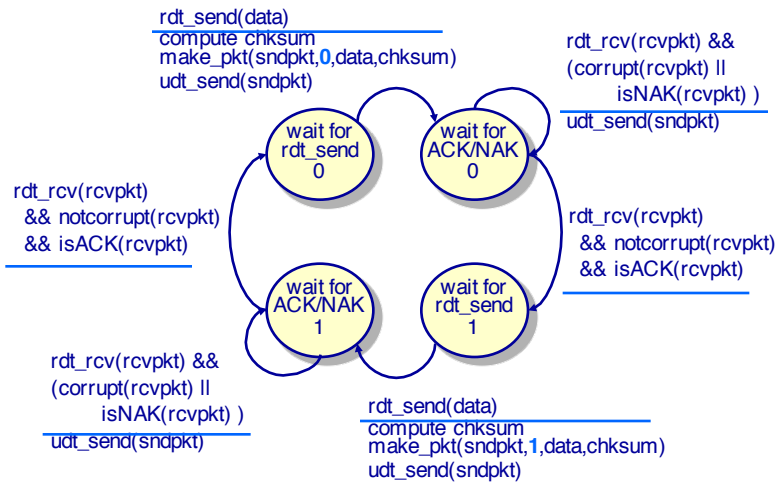


- Deal with corrupted ACKs/NAKs by retransmission of data packets
- Sender will add a sequence number to each packet to allow the receiver to detect duplicate packets
  - » Receiver's transport layer discards duplicate packets
- How much space to reserve in a header field for sequence numbers?

  need 2 sequence numbers, or 1 bit

Reliable Data Transfer Protocol 2.1

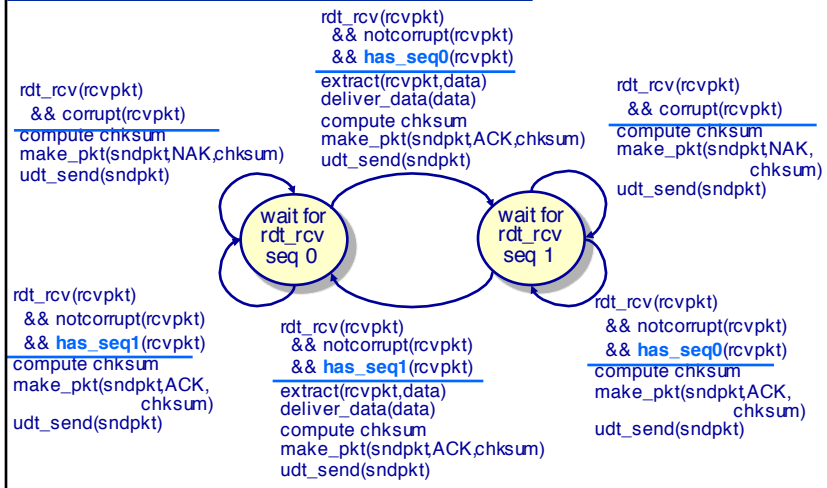
Sender state machine to handle garbled ACKs/NAKs



Midterm
1st question: compare and contrast 2 definition
example: contrast circuit switching and packet switching

Reliable Data Transfer Protocol 2.1

Receiver state machine to handle garbled ACKs/NAKs



Reliable Data Transfer Protocol 2.1

Discussion (Handling garbled ACKs/NAKs)

Sequence number added to header
- » Two sequence numbers suffice

Must check if received ACK/NAK is corrupted

Number of states doubles
- » State encodes whether current packet has sequence number 0 or 1

- Sender issues

Must check if received packet is duplicate
- » State encodes whether expected packet sequence number is 0 or 1

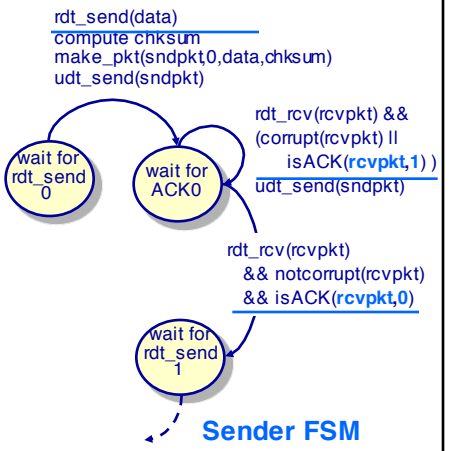
Note: receiver can *not* know if its last ACK/NAK received OK at sender

- Receiver issues

Reliable Data Transfer Protocol 2.2

A NAK-free protocol

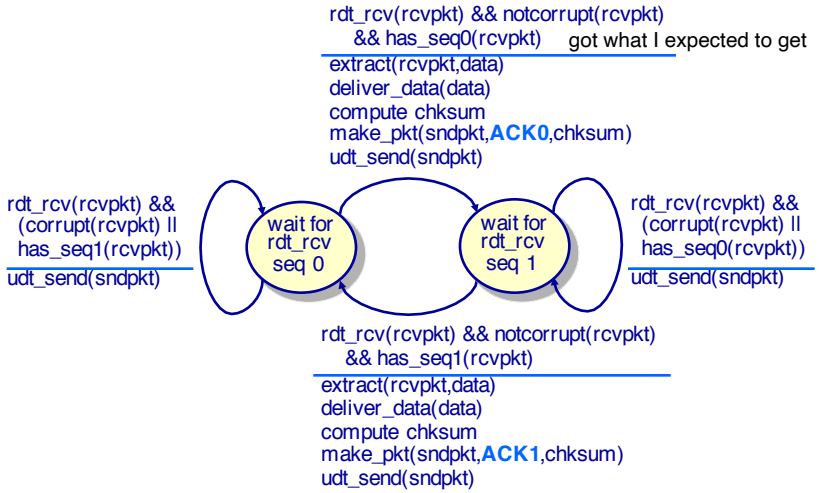
- Instead of NAKing, receiver sends ACK for last packet received OK
  - » Receiver must include the sequence number of packet being ACKed in ACK
- Receipt of duplicate ACKs at sender is equivalent to a NAK
  - » Sender retransmits current packet



Sender FSM

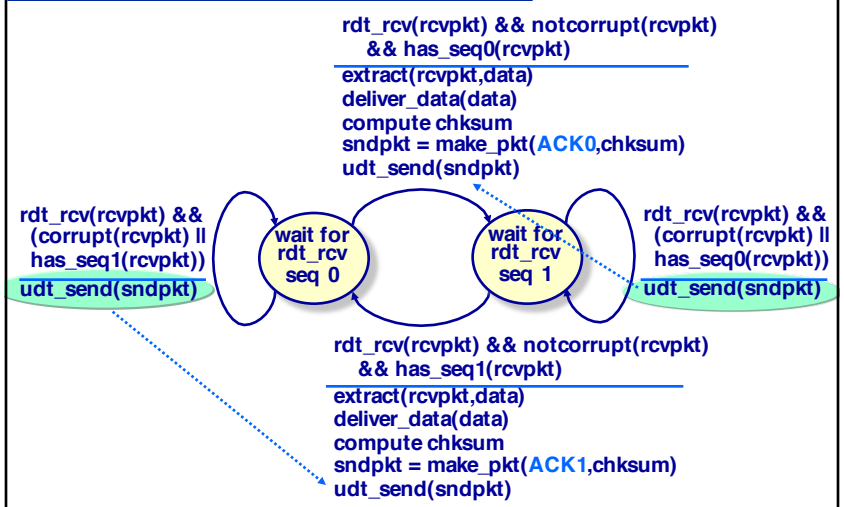
Reliable Data Transfer Protocol 2.2

Receiver state machine to eliminate NAKs



Reliable Data Transfer Protocol 2.2

A buggy receiver state machine?



Reliable Data Transfer Protocol 3.0

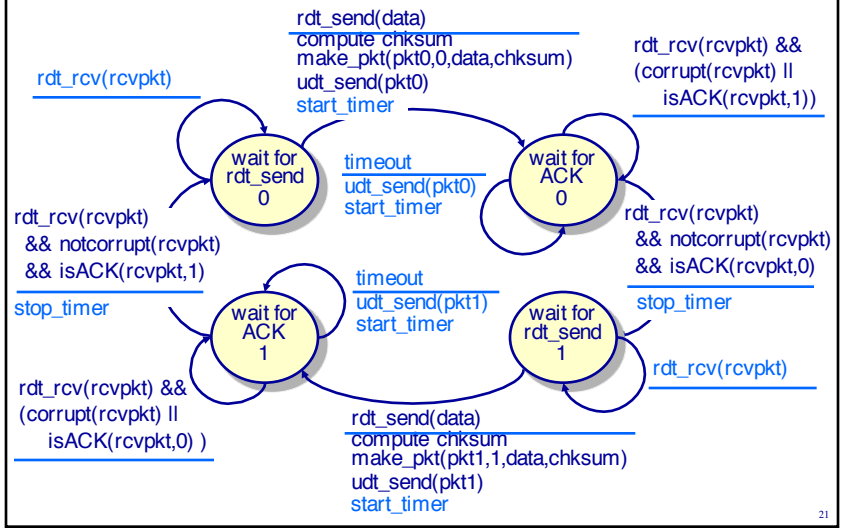
Dealing with channels with errors *and* loss

- Now assume the underlying channel can also **lose** packets
- New problem: How to detect loss?
  - » Are checksums, ACKs, sequence numbers, retransmissions enough?
    add a timer for how long to wait before concluding that the ack isnt arriving
- Approach: sender waits "reasonable" amount of time and retransmits if no ACK received in this time
  - » Requires the use of a countdown timer
- What if packet (or ACK) just delayed beyond its timer?
  - » Retransmission will be duplicate…
  - » But use of sequence numbers already handles this!

How long should you wait?
a little longer than the typical RTT
3 way handshake gives a sample of the RTT (round trip time)

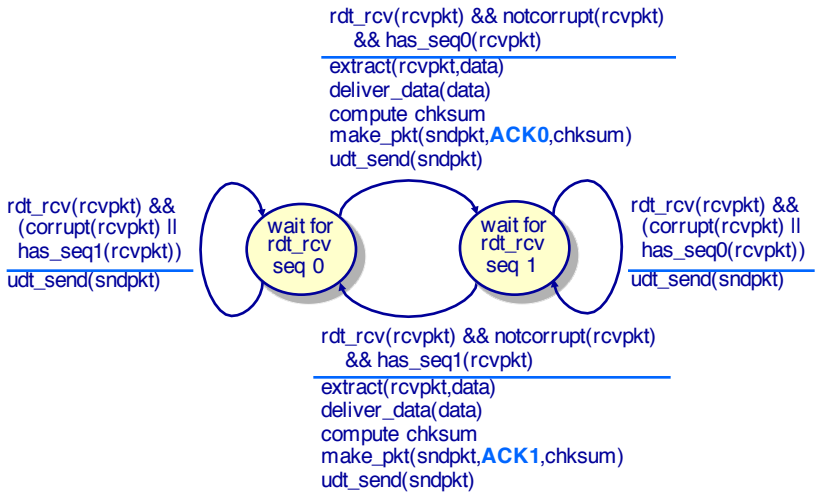
Reliable Data Transfer Protocol 3.0

Sender state machine to handle lost/garbled packets



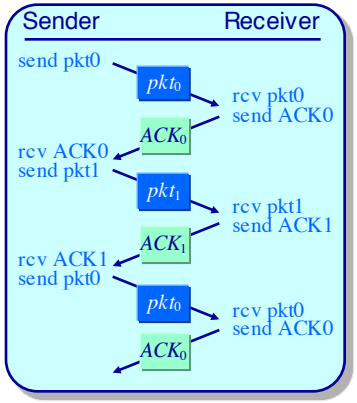
Receiver State Machine for RDT 2.2

What changes are needed to handle lost/garbled packets?

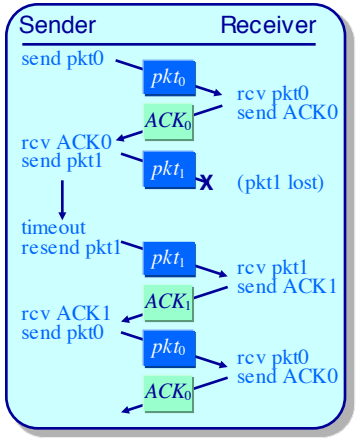


Reliable Data Transfer Protocol 3.0

Execution examples



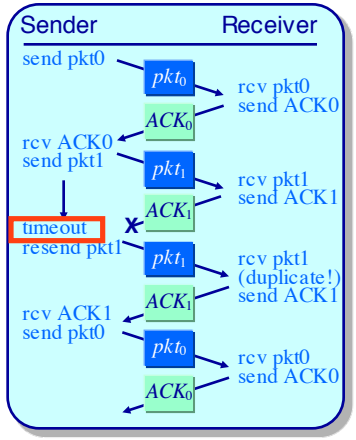
- Protocol operation with no loss



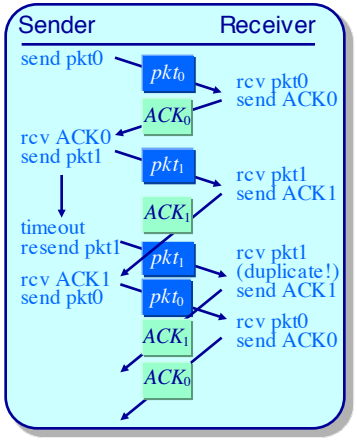
- Protocol operation with a lost packet

Reliable Data Transfer Protocol 3.0

Execution examples



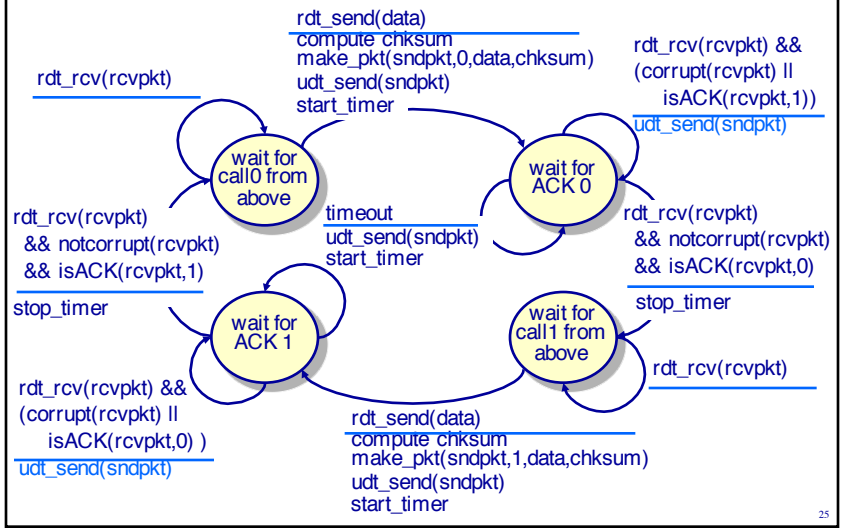
- Protocol operation with a lost ACK



- Protocol operation with a poor timeout value

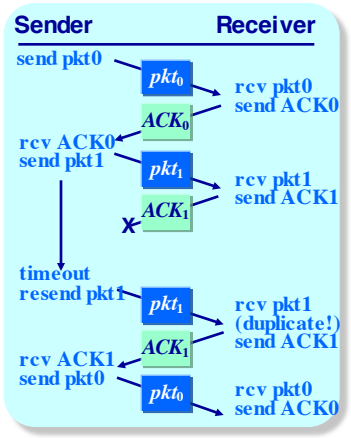
Reliable Data Transfer Protocol 3.1

Sender state machine to handle lost/garbled packets

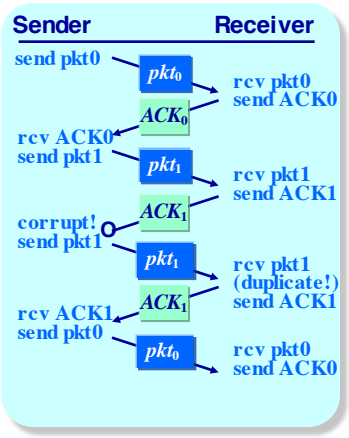


Reliable Data Transfer Protocol 3.1

Execution examples



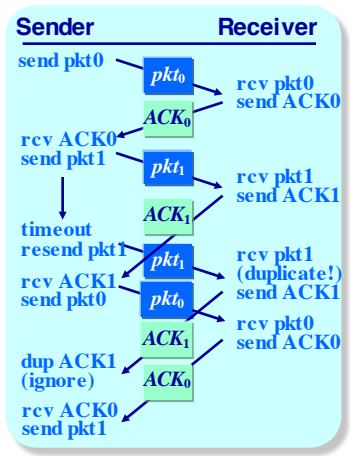
- Protocol operation with a lost ACK



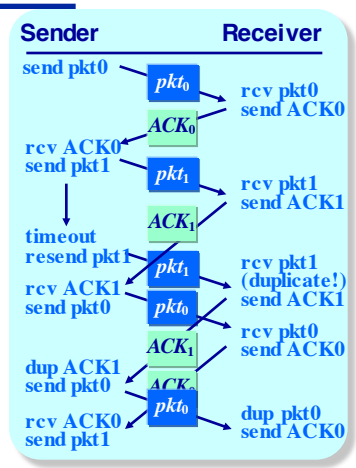
- Protocol operation with a corrupted ACK

Reliable Data Transfer Protocol 3.1

Execution examples



- RDT 3.0 operation with a duplicate ACK

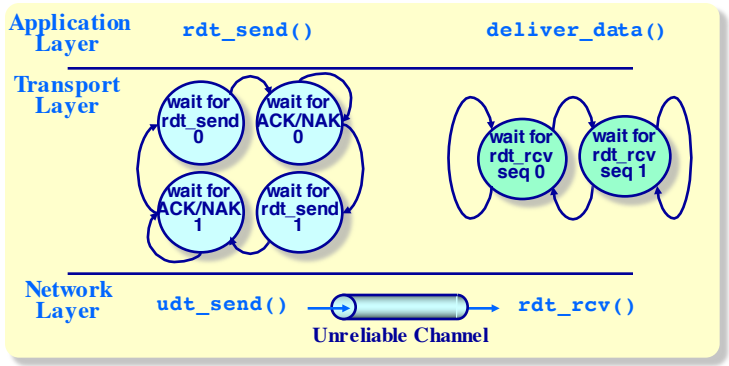


- RDT 3.1 operation with a duplicate ACK

27

Reliable Data Delivery

Summary



- Reliable delivery requires a distributed algorithm for detecting and repairing packet errors and loss
  - » Reliability achieved at the expense of considerable overhead and complexity

30

One solution is to update the timeout

adding congestion to the network but reliable data transfer is still achieved

RTT isn't measured for any packet that has to be sent again