

COMP 431

Internet Services & Protocols

The Transport Layer

Principles of Reliable Data Delivery

Jasleen Kaur

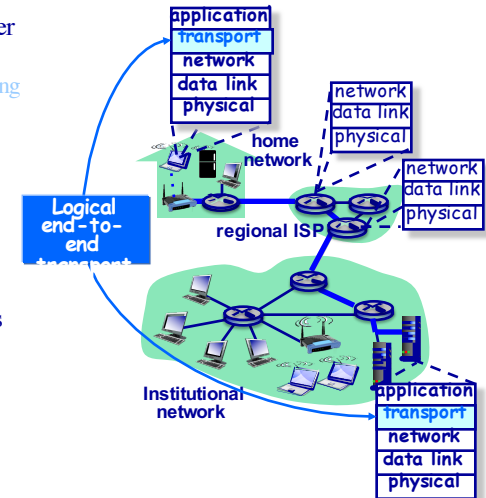
Feb 20, 2020

1

Transport Layer Protocols & Services

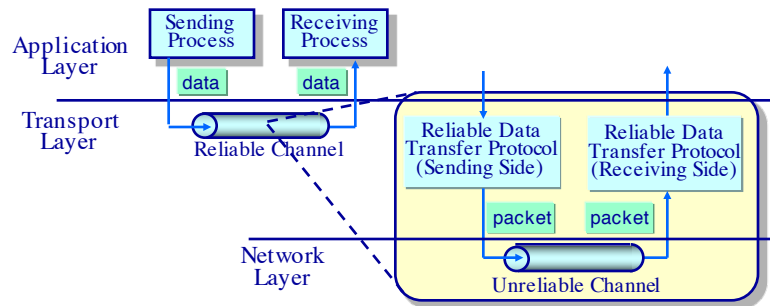
Outline

- ◆ 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

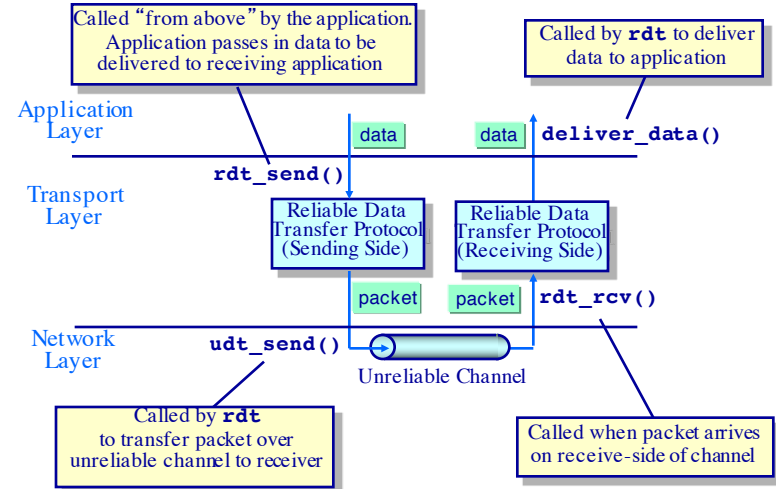


- ◆ Goal: Provide a reliable channel abstraction
 - » The characteristics of the underlying channel will determine the complexity of providing reliable communications
- ◆ Issues: *State* required at sender and receiver and number of *control messages* exchanged

3

Reliable Data Transfer

Programming interfaces



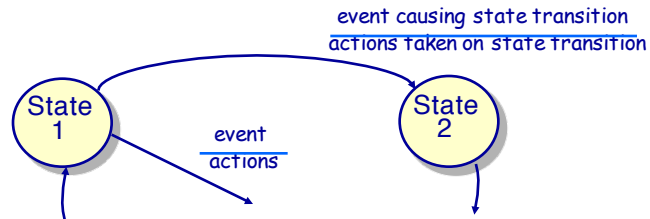
4

RDTP 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



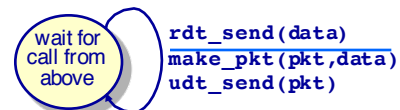
5

Reliable Data Transfer Protocol 1.0

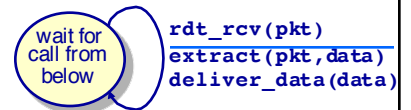
Reliable transfer over a reliable channel

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

◆ Sender state machine



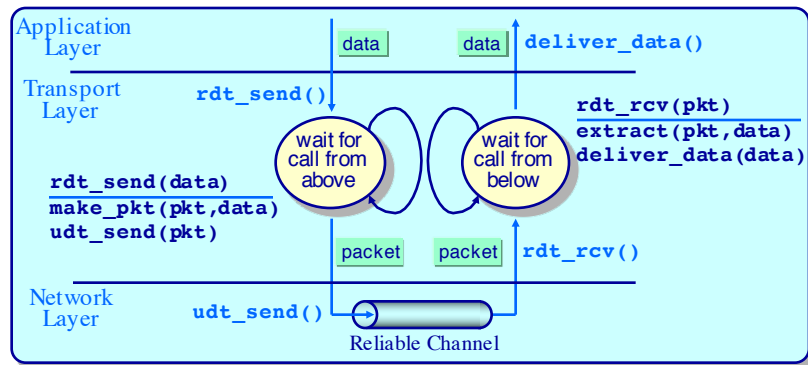
◆ Receiver state machine



6

Reliable Data Transfer Protocol 1.0

Programming interfaces



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

7

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

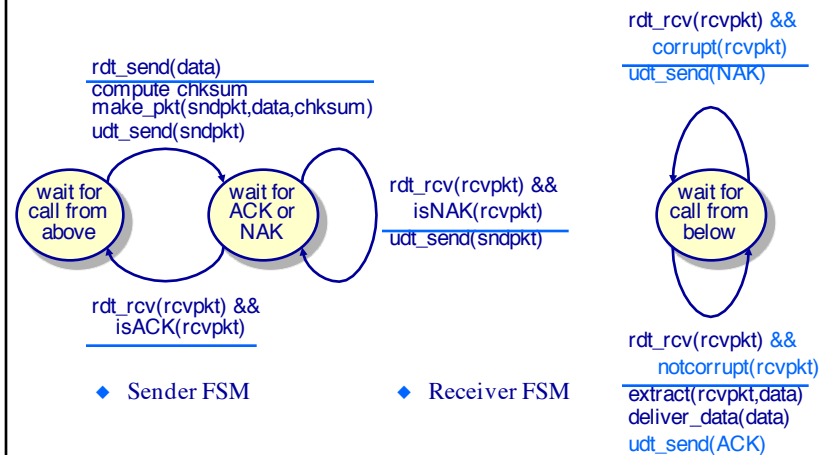
8

Detect errors with checksum

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

Reliable Data Transfer Protocol 2.0

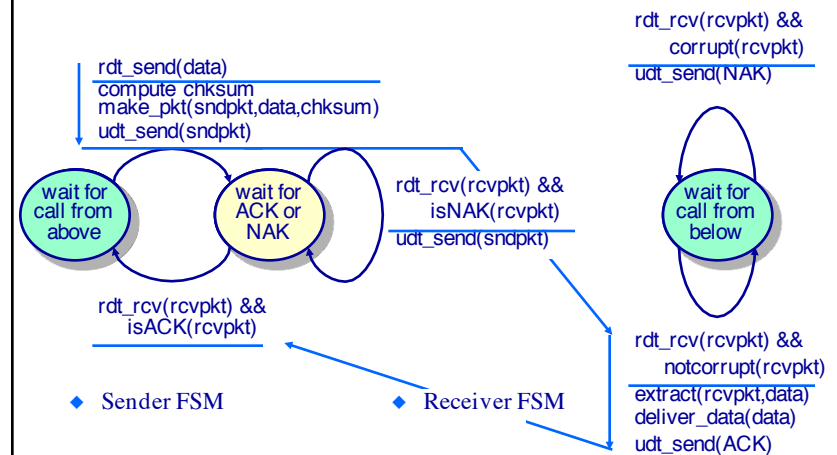
Reliable transfer over a channel with bit errors only



9

Reliable Data Transfer Protocol 2.0

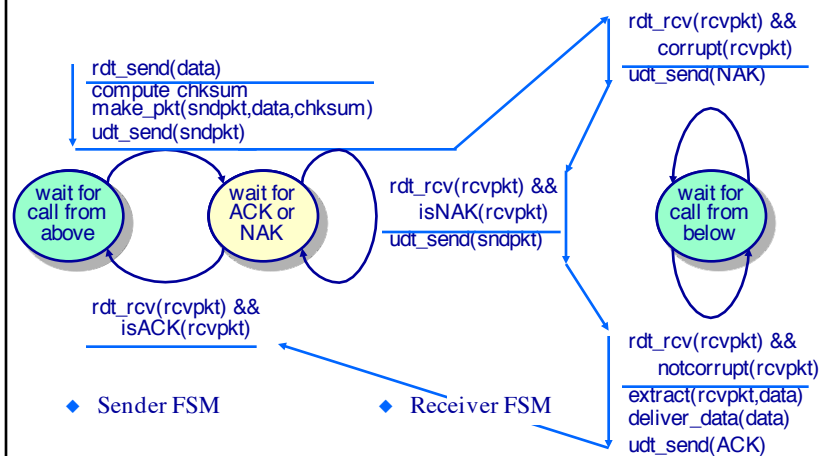
Example 1: No errors occur



10

Reliable Data Transfer Protocol 2.0

Example 2: A corrupted packet arrives at the receiver

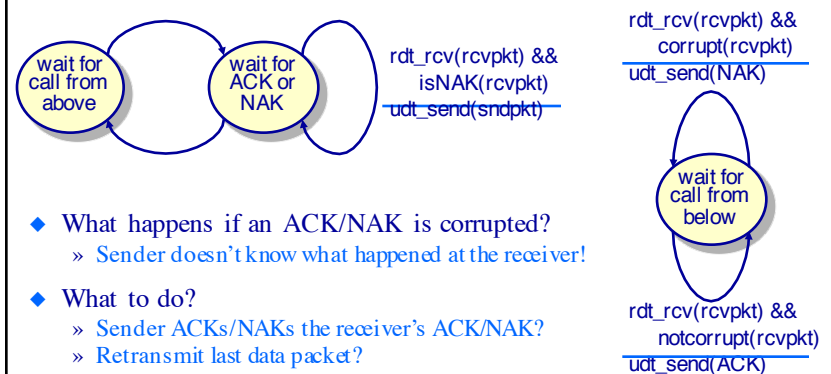


Sender and receiver both calculate checksum
look up CRC(cyclic redundancy checksum) online

What if the ack is corrupted?

Reliable Data Transfer Protocol 2.0

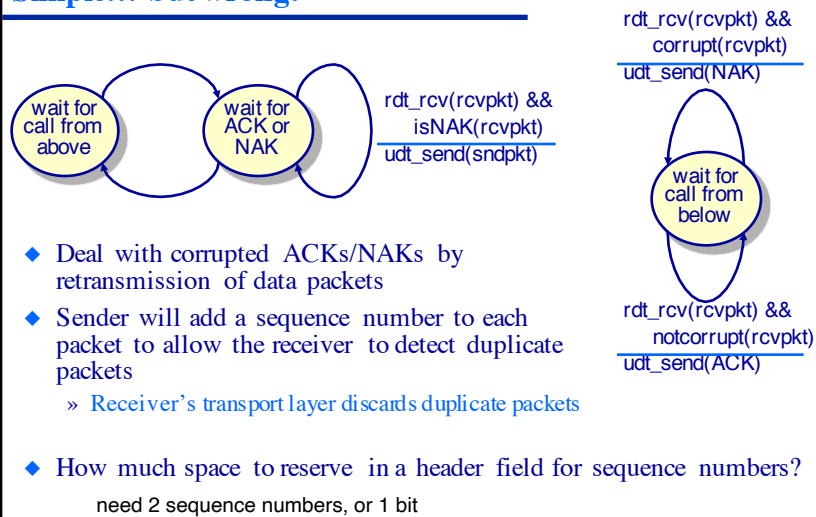
Simple... but wrong!



- ◆ 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?

Reliable Data Transfer Protocol 2.0

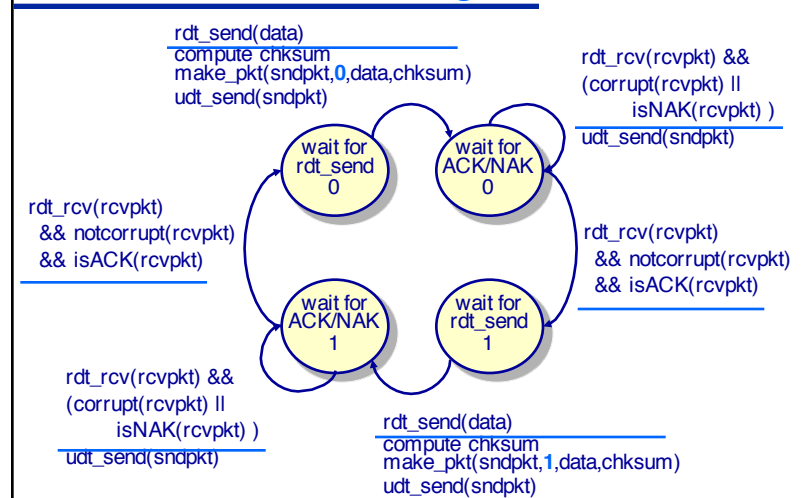
Simple... but wrong!



13

Reliable Data Transfer Protocol 2.1

Sender state machine to handle garbled ACKs/NAKs



14

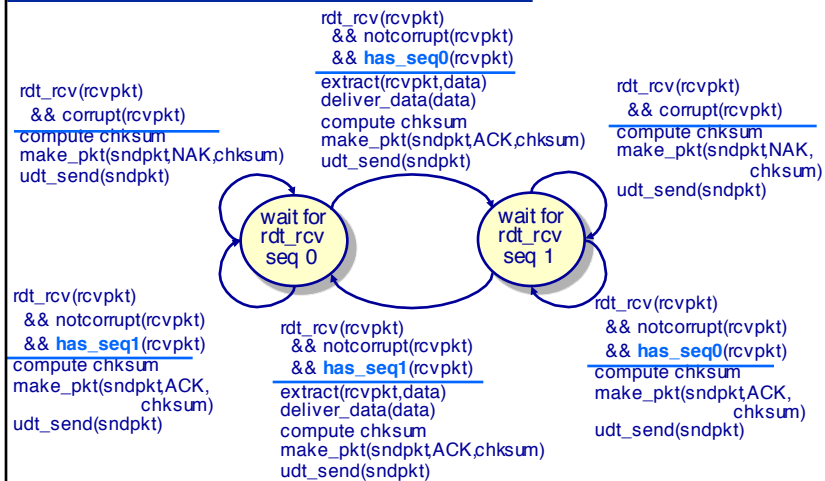
Midterm

1st question: compare and contrast 2 definition

example: compare circuit switching and packet switching

Reliable Data Transfer Protocol 2.1

Receiver state machine to handle garbled ACKs/NAKs



15

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

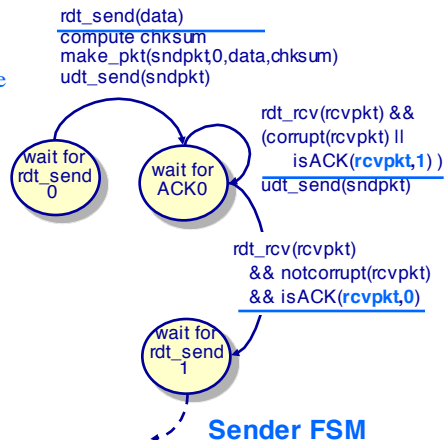
◆ Receiver issues

16

Reliable Data Transfer Protocol 2.2

A NAK-free protocol

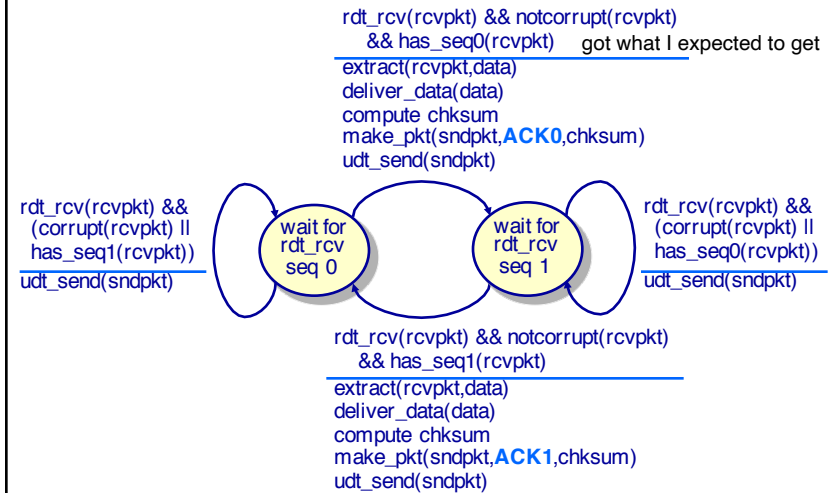
- ◆ Instead of NAKing, receiver sends ACK for last packet received OK
 - » Receiver must include the sequence number of packet being ACKed in ACK
- ◆ Receipt of duplicate ACKs at sender is equivalent to a NAK
 - » Sender retransmits current packet



17

Reliable Data Transfer Protocol 2.2

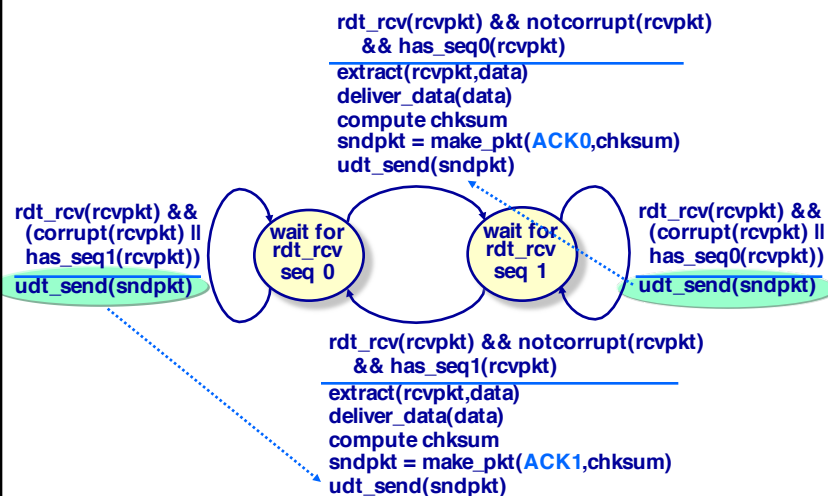
Receiver state machine to eliminate NAKs



18

Reliable Data Transfer Protocol 2.2

A buggy receiver state machine?



19

Reliable Data Transfer Protocol 3.0

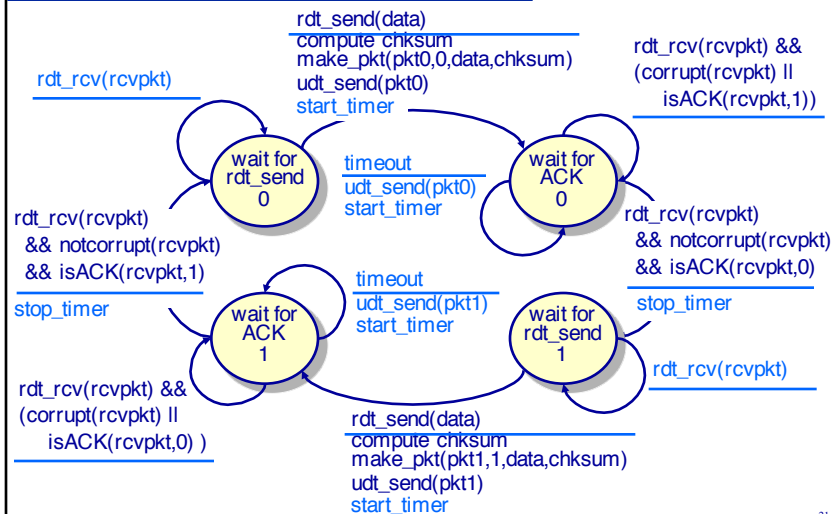
Dealing with channels with errors *and* loss

- ◆ 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 isn't 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)

20

Reliable Data Transfer Protocol 3.0

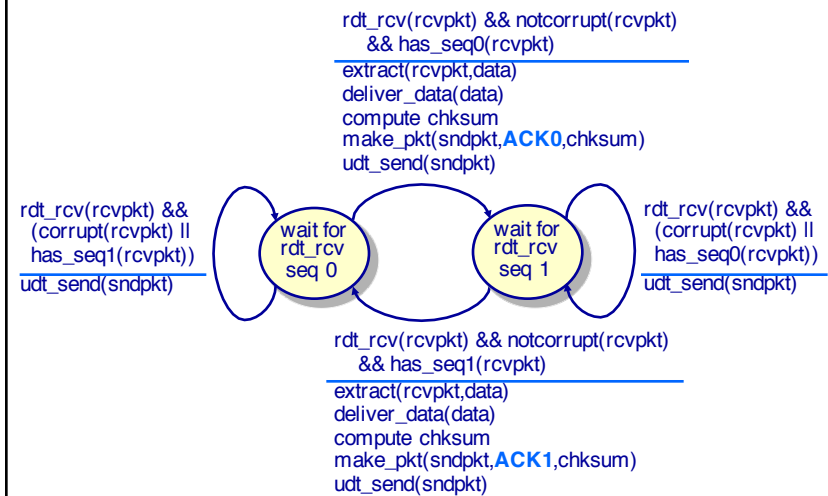
Sender state machine to handle lost/garbled packets



21

Receiver State Machine for RDT 2.2

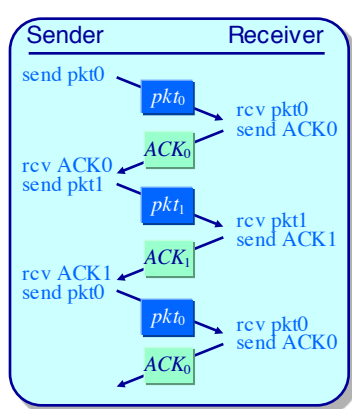
What changes are needed to handle lost/garbled packets?



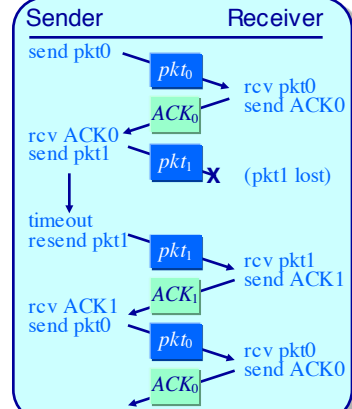
22

Reliable Data Transfer Protocol 3.0

Execution examples



- ◆ Protocol operation with no loss

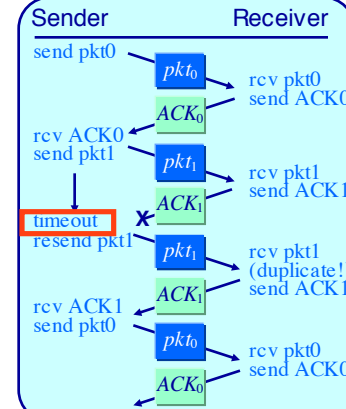


- ◆ Protocol operation with a lost packet

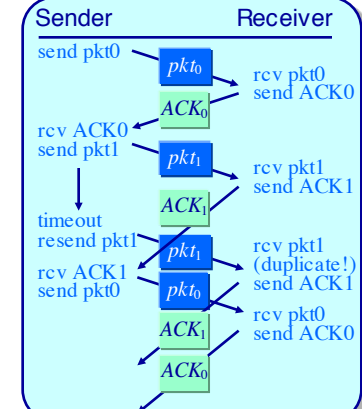
23

Reliable Data Transfer Protocol 3.0

Execution examples



- ◆ Protocol operation with a lost ACK

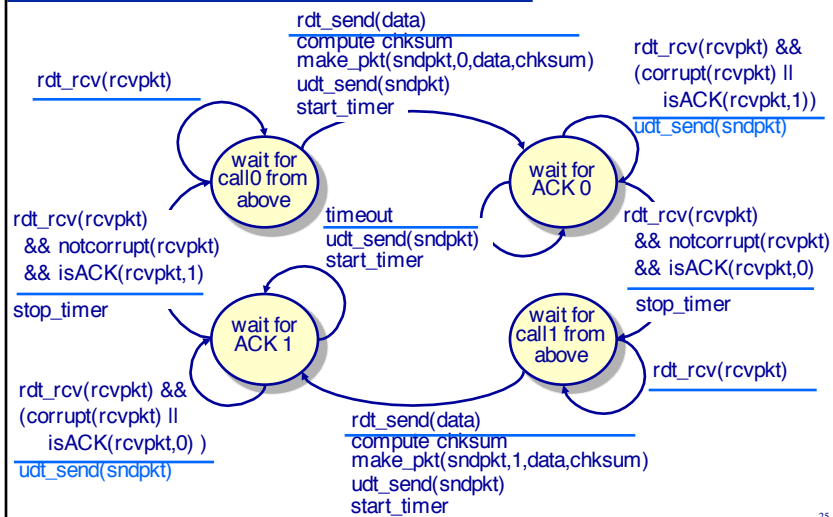


- ◆ Protocol operation with a poor timeout value

24

Reliable Data Transfer Protocol 3.1

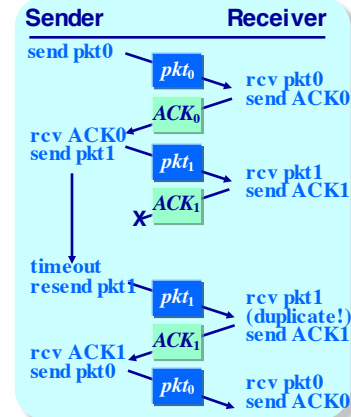
Sender state machine to handle lost/garbled packets



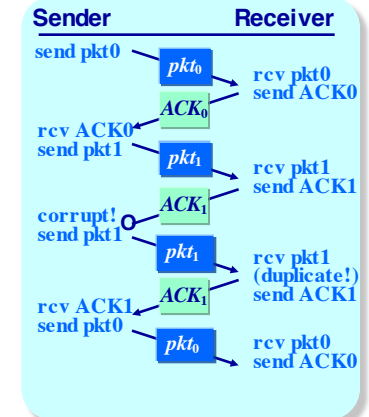
25

Reliable Data Transfer Protocol 3.1

Execution examples



- ◆ Protocol operation with a lost ACK



- ◆ Protocol operation with a corrupted ACK

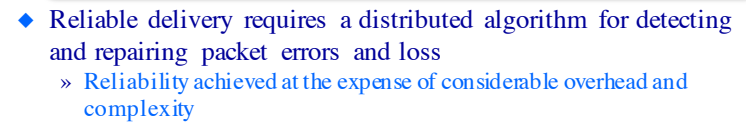
26

Execution examples



27

Summary



30

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