# Computer Networks WS20/21

Exercise 3

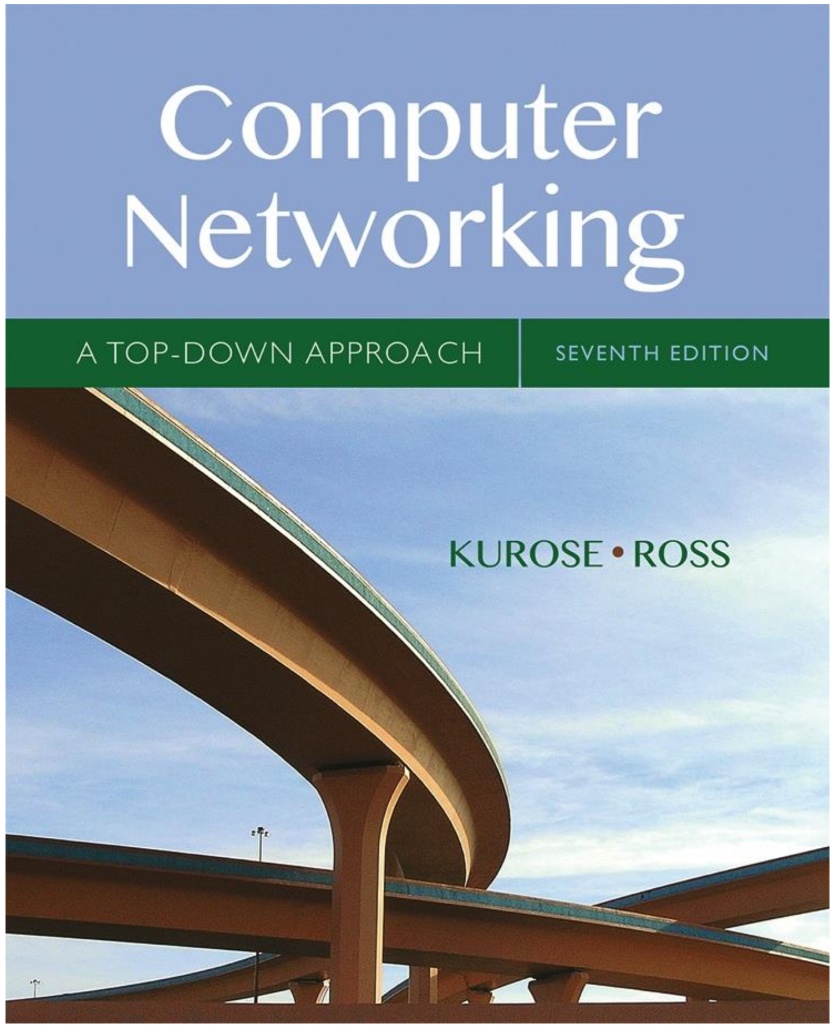
## Recommendation

Try to borrow (or buy) this book:

Computer Networking: A Top Down Approach

7th edition. Jim Kurose, Keith Ross, Pearson, 2019.

It is very good to understand!



## Transport Layer

- Q: What entities does the transport layer connect in contrast to the network layer?

- The transport layer connects **processes** while the network layer connects **hosts/networks**

# Multiplexing

- Q: Why is multiplexing and demultiplexing used for at the transport layer and what has the concept of ports to do with this?

- Multiplexing and demultiplexing is needed to reliably match incoming and outgoing packets to the right processes
- Multiplexing at sending host:
  - Gathering data from multiple sockets (belonging to different processes)
  - Enveloping data with header information for socket identification (at receiving host)
  - Segmentation of data
  - Passing data to the network-layer
- Demultiplexing at receiving host:
  - Identification of socket each incoming segment belongs to and
  - Delivering received segment to correct socket
- Port numbers are used to identify the sockets

# TCP vs. UDP

Q: Please compare TCP and UDP in terms of the services they offer to the application layer.

- TCP
  - Reliable data transfer
  - Connection-oriented transport
  - In-order delivery
  - Retransmission
  - Congestion control
    - imposes transmission-rate constraints
  - Functions can not be turned off

- UDP
  - Best-effort data transfer
  - Connectionless
  - Possible out-of-order delivery
  - Segments may get lost
  - Faster as TCP
    - No congestion control
    - No connection establishment

## TCP vs. UDP (II)

- Q: For which kinds of applications would you prefer UDP over TCP.

- When transmission rate constraints enforced by TCP are not wanted/needed
- When reliability of TCP is not wanted/needed
- Examples: Multimedia (e.g.VoIP, streaming media) or “simple” services like SNMP, DNS

## UDP checksums

- Assume a UDP transport has received a datagram which consists of the following 16-bit words and the given checksum. Please verify the checksum.

- 1010 1010 1010 1010 (1st 16 bit word)
- 1011 1011 1011 1011 (2nd 16 bit word)
- 1100 1100 1100 1100 (3rd 16 bit word)
- 1110 1100 1100 1100 (recvd. checksum)

## UDP checksums (cont‘d)

- 1) Sum of first two words:

```
  1010 1010 1010 1010
+ 1011 1011 1011 1011
  -------------------
1 0110 0110 0110 0101

  0110 0110 0110 0101
+ 0000 0000 0000 0001
  -------------------
  0110 0110 0110 0110
```

The overflowing 1 must be added to the last digit!

# UDP checksums (cont'd)

- 2) Add third word:

```
    0110 0110 0110 0110
  + 1100 1100 1100 1100
    -------------------
  1 0011 0011 0011 0010

    0011 0011 0011 0010
  + 0000 0000 0000 0001
    -------------------
    0011 0011 0011 0011
```

The overflowing 1 must be added to the last digit!

## UDP checksums (cont‘d)

- 3) Add received checksum:

```
    0011 0011 0011 0011
  + 1110 1100 1100 1100
    -------------------
  1 0001 1111 1111 1111
```

- 4) Result ≠ 1111 1111 1111 1111 therefore verification failed

## Reliable data transfer

- Q: Assume you want to reliably transfer data over a channel with bit errors but no loss. An error detection mechanisms is already implemented. Which simple mechanism can you use to recover from errors? What flaw does this simple mechanism have?

## Reliable data transfer (cont’d)

- Method: Packet arrives
  - Without errors: Receiving host sends acknowledgement message (ACK) to the sending host
  - With errors: Receiving host sends negative acknowledgement message (NAK) to the sending hot

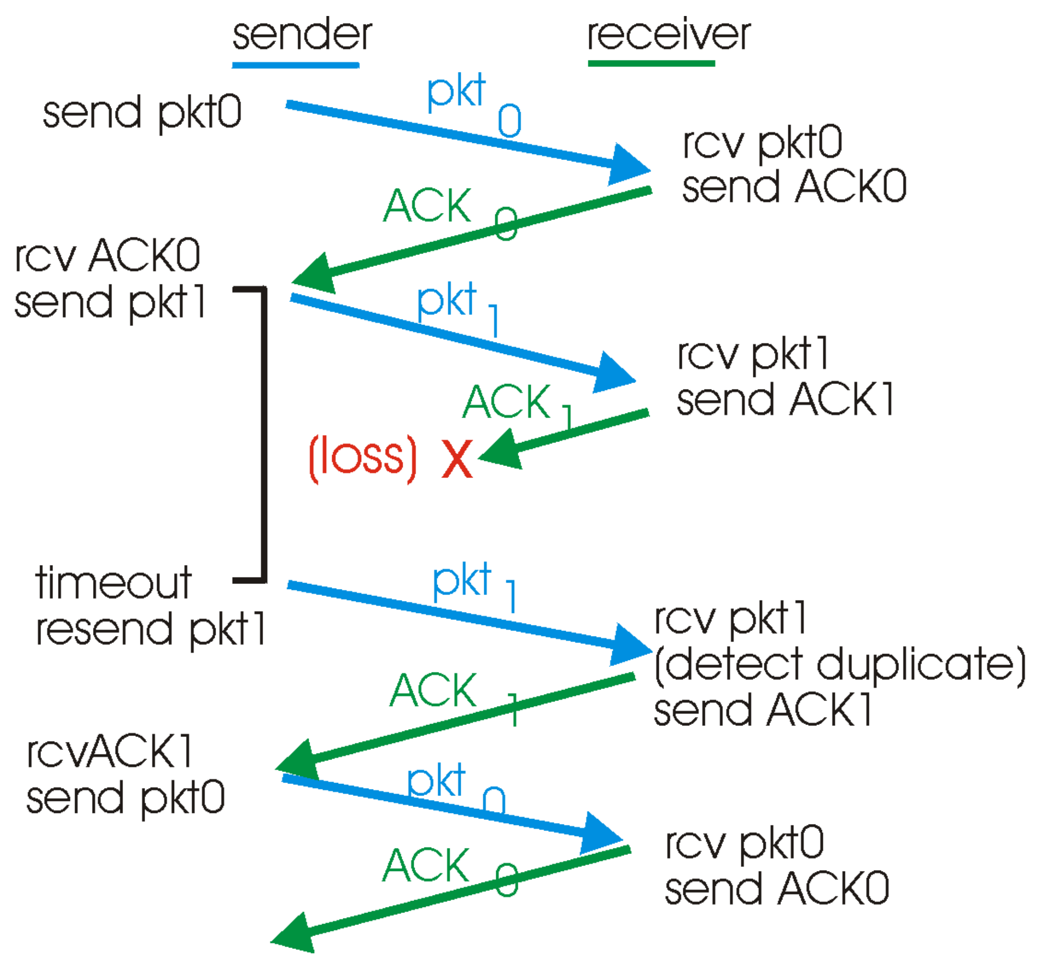
- Flaw: If ACK/NAK message gets corrupted, sender doesn’t know if packet was received without errors

## Reliable data transfer (II)

- Q: Assume you want reliably transfer data over a channel with bit errors and loss. What additional mechanism do you need to introduce. Give an example of how this mechanism can recover from the loss of a packet.

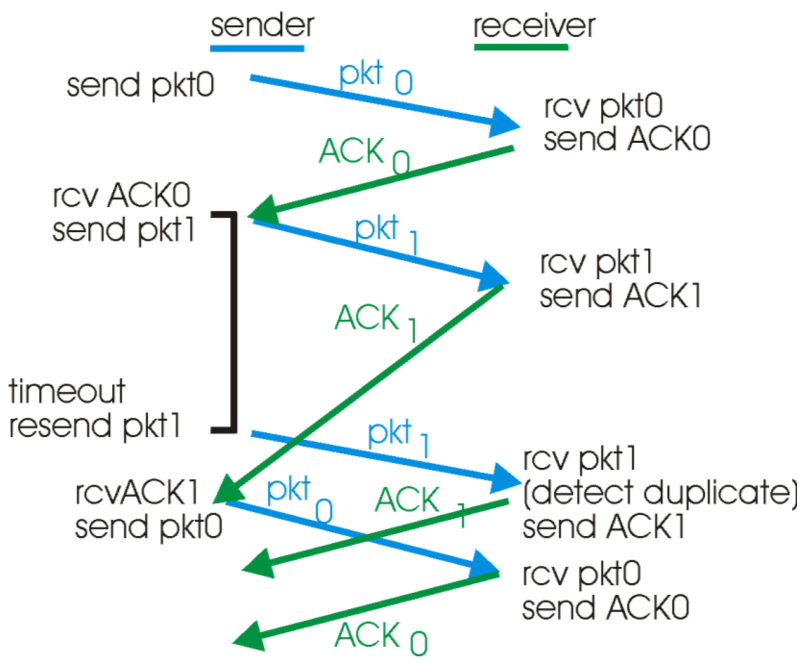
- Sender needs to maintain a timer for unacknowledged packets. If an ACK for a packet is not received within a certain time frame, the sender can transmit the packet again.

# Reliable data transfer (II) (cont’d)



# Premature timeout in rdt3.0 (Lecture addendum)

- In rdt3.0 an ACK for the wrong sequence number is ignored (as opposed to rdt2.2 where it leads to a retransmit)
- Therefore, in the example, the second $ACK_1$ is ignored



## Any Questions?

Mail us:

Yachao Shao: yachao.shao@cs.uni-goettingen.de

Fabian Wölk: fabian.woelk@cs.uni-goettingen.de