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**

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 different processes
- Port numbers are used to implement multiplexing

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
  - Congestion control
- UDP
  - Best-effort data transfer
  - Connectionless
  - Possible out-of-order delivery
  - No congestion control

TCP vs. UDP (II)

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

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

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) one's complement sum of first two words:

```
  1010 1010 1010 1010
+ 1011 1011 1011 1011
  --------------------
  0110 0110 0110 0110
```

- Why 0 at last digit? The carry is added in the last step!

UDP checksums (cont‘d)

- 2) add third word:

```
  0110 0110 0110 0110
+ 1100 1100 1100 1100
  --------------------
  0011 0011 0011 0011
```

UDP checksums (cont‘d)

- 3) add to 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)

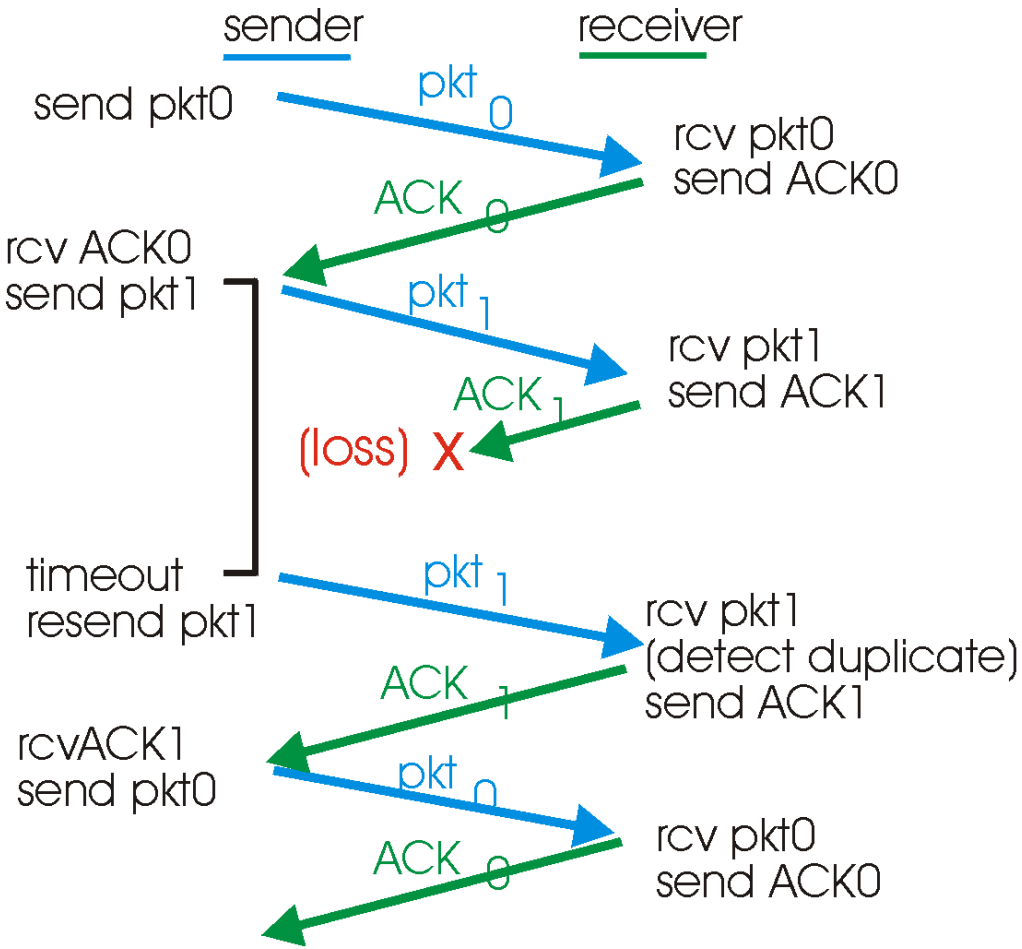
- Send ACK if packet was received without errors, send NACK otherwise
- Flaw: If ACK/NACK gets corrupted, sender doesn't know if packet was received without errors

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.

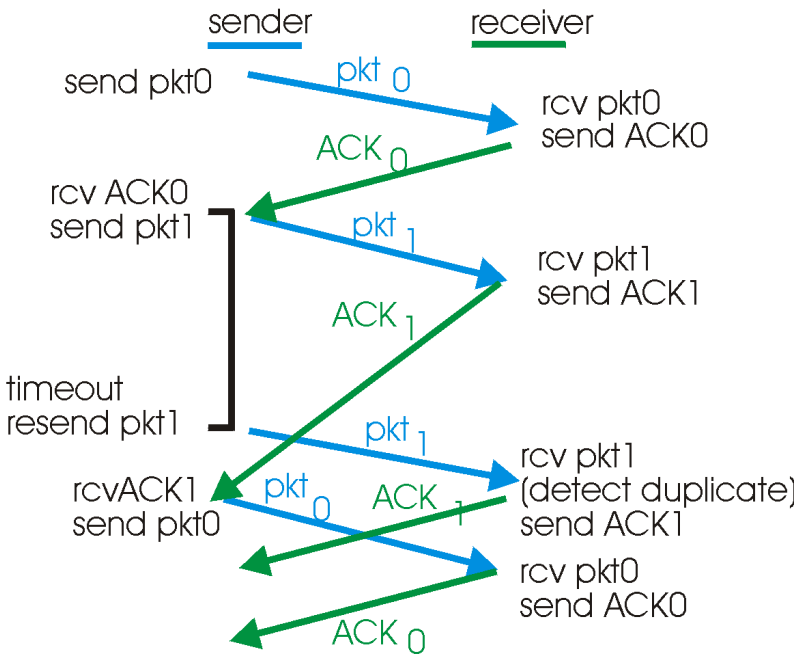
- Sender needs to maintain a timer for unacknowledged packets so it can re-transmit if an ACK is not received within a certain timeframe.

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 let to a retransmit)
- Therefore, in the example, the second ACK_1 is ignored



Thank you

Any questions?