Challenge 2 – Reliable Data Transfer

In our implementation, we decided to use a sliding window approach. Therefore we implemented selective repeat.

In a first step, the file which should be sent is divided into packets of equal and preset size (except the last one which can potentially be smaller). These packets then are sent to the receiver.

Each packet gets a header, which contains a calculated sequence number, which is used to identify the packet when it is received or lost. With the sliding window approach, the amount of packets „on the way“ are limited. Therefore we can limit the maximum sequence number to twice the greatest amount of packets „on the way“ possible, while maintaining identifiability. This can save a lot of packet size when the file which should be sent is very big.

Upon sending, each packet gets a personal timeout, all set to a former specified value. These timeouts are used to resend the corresponding packets. Therefore, the sender checks if the packet, which timer has run out, has already been acknowledged by the receiver.

For the acknowledgement, the receiver answers to each received data packet by sending back an acknowledgement packet with the sequence number of the obtained data packet. The sender then can keep track of received and unreceived packets. The timeouts are useful if a packet gets lost on the way. Since no acknowledgement will ever be received for that particular packet, it will automatically be resent upon its timeout. Even if a data packet is received, but the acknowledgement takes too long to get back to the sender or gets lost itself, the corresponding data packet will be resent. This may produce overhead, but does not affect the correctness of the received file, since the duplicate can be identified by its sequence number.

We separated the sender and receiver into different classes to remain readability. The corresponding “main” method for sending and receiving is then called by our ReliableDataTransferProtocol.

Jonas Becker (428292) & Daniel Beckmann (416096)