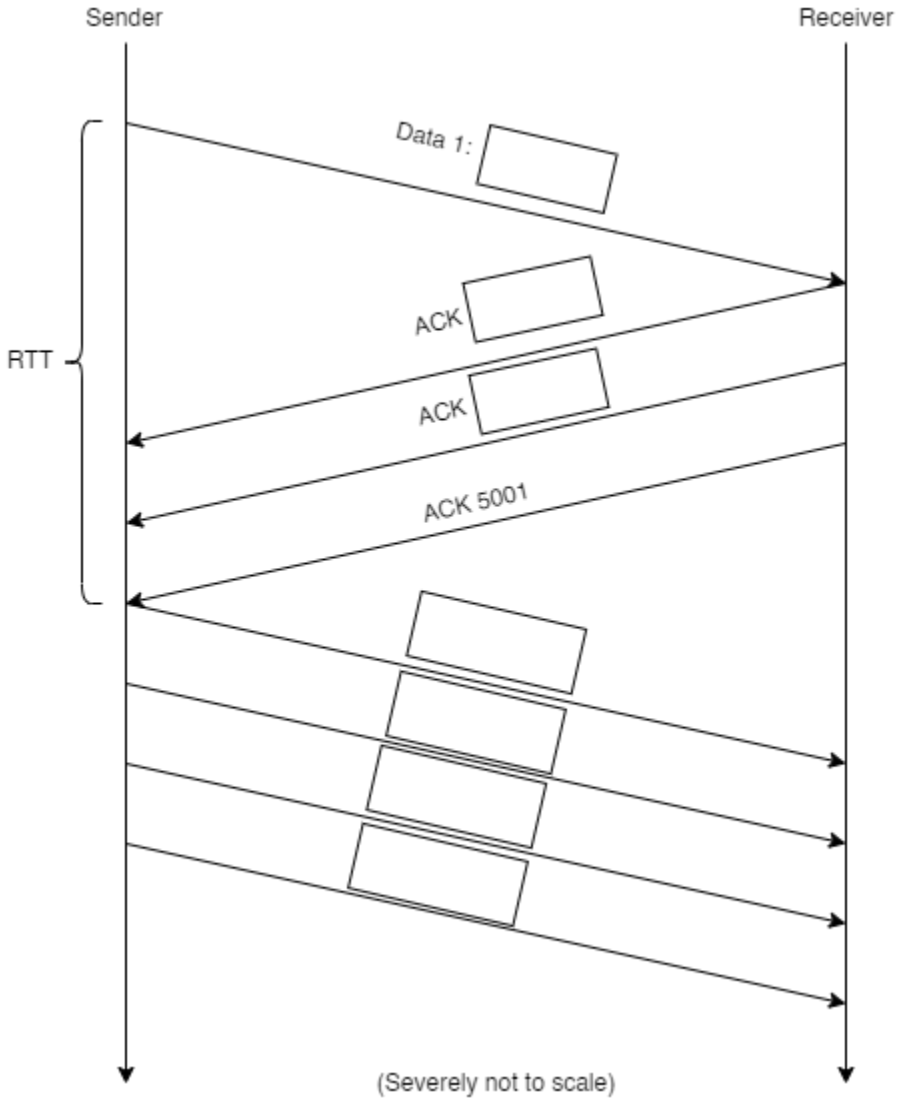
Proposed Final Exam Questions (Group 2)

1) TCP Congestion Control with a Misbehaving Receiver

In this course, we have discussed multiple network protocols in the various layers of the TCP/IP Model. For all of the specified network protocols, we have always assumed that all parties mutually agree to follow said protocols so as to have a stable network that will benefit everyone. We will now explore what will happen if the receiver of a TCP connection deviates from the protocol, which could potentially happen, intentionally or unintentionally, in real life.

a) In Group 2's project presentation, there was a discussion on the 3 different attack mechanisms that might be possibly conducted by a TCP receiver: ACK Division, DupACK Spoofing, and Optimistic ACKing. Which method is not immune to end-to-end losses and does not actually provide reliable data transfer? [1m]

b) Complete the following incomplete spacetime diagram for an ACK Division attack and calculate the total time taken to finish the attack over a link of RTT 100ms and transmission bandwidth 1 Gbit/s, starting from the time the sender starts transmitting data, assuming that the total data size to be downloaded from the server is 25000 bytes. Assume that the size of an ACK packet is negligible and that the time interval between each ACK packet is negligible. Also assume that processing delay and queueing delay are negligible. [4m]



c) Section 3.2 of RFC 5681 proposed a possible defense mechanism against DupACK Spoofing, which is by limiting the artificially inflated cwnd to the number of outstanding packets. Complete the following code snippet in Python which is implementing this protection. Note that you do not have to define any new variables (i.e., you just need to use the variables explicitly defined below). [3m]

```
MSS = ...

class DefendedTCPRenoServer:
    def __init__(self, ...):
        # Define relevant class variables here
        self.state = "slow_start"
        self.cwnd = 1 * MSS
        self.seq = 0
        self.next_seq = 1
        self.dupack = 0
        # Some other irrelevant class variables...
        # ...

    # Other irrelevant methods
    # ...

    def receive_duplicate_ack(self, ...):
        self.dupack += 1
        # Some other code here...
        # ...

        if self.dupack > 3 and self.state == "fast_recovery":
            # TODO! (Put your answer here!)




    # Other irrelevant methods
    # ...
```

d) Suppose that for the Optimistic ACKing attack, on top of the number of optimistic ACK packets to be sent, we also parameterize the time interval between each optimistic ACK packet in milliseconds. For a given **fixed** number of optimistic ACK packets to be sent, plot the expected general trend in average throughput as we vary the time interval. Assume that the number of optimistic ACK packets is already **fixed**. Note that only the general trend is required. [2m]