Project 2: Transmission Control Protocol

# Project Objectives

The task of this project is to implement TCP from the ground up and mainly consists of the reliable data transfer part of TCP.

Please read the complete project description carefully before you start so that you know exactly what is being provided and what functionality you are expected to add.

# Task : Simplified TCP Sender/Receiver

The task is to implement a "Reliable Data Transfer" protocol, following the description of section

* + 1. from the textbook. The idea here is to build a simplified TCP sender and receiver that is capable of handling packet losses and retransmissions.

The following functionalities must be implemented:

* + - * Sending packets to the network based on a fixed sending window size (e.g. WND of 10 packets)
      * Sending acknowledgments back from the receiver and handling what to do when receiving ACKs at the sender
      * A timeout mechanism to deal with packet loss and retransmission

For this task, there is no need to buffer out-of-order packets at the receiver and they can be simply discarded, i.e. no ACKs are sent for out-of-order packets. However, lost packets must be retransmitted by the sender. For the timeout mechanism, you can assume a fixed timeout value that is appropriate for the emulated network scenario using MahiMahi (see below).

We have provided you with a simple (stop-and-wait) starter-code that consists of the following:

* rdt receiver: this holds the implementation of a simple reliable data transfer protocol (rdt) receiver
* rdt sender: this holds the implementation of a simple reliable data transfer protocol (rdt) sender
* Setup document for using mahimahi to emulate different network conditions ([link](https://docs.google.com/document/d/12cKEzeMrM7ia7HxnvmlX3lHmjv7Px8YkfFLevvEcgZ4/edit?usp=sharing))

The simple rdt protocol is implemented on top of the UDP transport protocol. During the lab session, Dena showed you how to use the network emulator MahiMahi to test your sender and receiver functionality in an emulated network environment.

# Grading

|  |  |
| --- | --- |
| **Description** | **Score (/20)** |
| Extending the sender to send 10 packets | 4 |
| Properly sending and handling ACKs | 5 |
| Retransmissions of lost packets | 5 |
| Properly receiving the exact file on the receiver (no errors) | 4 |
| Coding style and usage of meaningful comments | 2 |