#### COEN 146: Computer Networks – Fall 2018

**Lab assignment 6: Data transfer protocol – unreliable channel**

**Objectives**

##### To build client/ server UDP application for an unreliable channel

##### **Part 1: Client – Server with UDP/IP using Stop-and-Wait protocol [26 points]**

Stop and Wait and Sliding Window protocols are generally used to provide a reliable transport service. In this lab, you will be guided to build Stop and Wait protocol on top of UDP, and it is supposed to provide a reliable transport service. The file transfer application you have developed in Lab 3 will be used as the UDP/IP service.

Messages are sent one at a time, and each message needs to be acknowledged when received, before a new message can be sent. Your reliable data transfer will basically implement the protocol rdt2.2 presented in the text book. Let us assume the communication is unidirectional, i.e., data flows from the client to the server. Here is the sequence of operation:

1. The server starts first and waits for messages.
2. The client starts the communication. Messages have sequence number 0 or 1. Before sending each message, a checksum is calculated and added to the header. After sending each message, the client waits for a corresponding ACK. When it arrives, if it is not the corresponding ACK (or if the checksum does not match), the message is sent again. If it is the corresponding ACK, the client changes state and returns to the application, which can now send one more message. This means that your client blocks on writes until an ACK is received.
3. The server, after receiving a message, checks its checksum. If the message is correct and has the right sequence number, the server sends an ACK message (according to the sequence number) to the client, changes state accordingly, and deliver data to the application.

The protocol should deal properly with duplicate data messages and duplicate ACK messages. Follow the FSM in the book! The message contains the header and the application data. No reordering is necessary, since the protocol is sending the exact message given by the application, one by one.

PROTOCOL

HEADER

* seq\_ack int (32 bits) // SEQ for data and ACK for Acknowledgement
* len int (32 bits) // Length of the data in byes (zero for ACKS)
* cksum int (32 bits) // Checksum calculated (by byte)

PACKET

* header
* data char (10 bytes)

SENDER

* Member seq\_ack is used as SEQ, and the data is in member data.
* Each packet may have 10 or less bytes of data, and the sender only sends the necessary bytes.
* After transmitting the file, a packet with no data (len = 0) is sent to notify the receiver that the file is complete.

RECEIVER

* Member seq\_ack is used as ACK, and data is empty (len = 0).

Demonstrate client/server communication to the TA and upload your source code to Camino.

**Part 2: Quiz [4 points]**

Complete the quiz on Part 2 of Google Lab6\_Turn-in Form through Camino.

**Requirements to complete the lab**

1. Show the TA correct execution of the program you wrote for Part 1 and upload source code to Camino.
2. Complete the quiz on Google Lab6\_Turn-in Form through Camino.