Laboration 2

This lab consists mainly of two parts. The majority comes from the book's lab 3, Implementing a Reliable Transport Protocol. The author has simulated a network environment and an application layer. Your task is to write the transport layer. The idea is that you should get a better picture of how the communication works between the different layers without having to implement all the layers yourselves, as this can be very time consuming.

You should create one sender and one receiver of data. (Of course, you may implement data traffic in both directions, but this is not required). The simulated application layer sends data to the sender side of your application. The sender receives the data, repackages it and sends it on to the simulated network that delivers it to the receiving side of your application. The receiver checks that it is the right data and that it has not been corrupted on the way. If something is wrong, the receiver notifies the sender which then makes a retransmission. When the receiver gets the correct data it forwards it to the simulated application layer at the receiving side. What percentage of packages that will be corrupted or lost in the simulated network can be set at the start of the application.

There are two versions of the labs. The version you should do is the first version which is an alternating-bit protocol. Have you not read chapter 3 of the book, do it first! Especially chapter 3.4.1 you will find very helpful for the lab.

Part 1
Before you start with the actual implementation of the transport protocol, you should make a state diagram for an alternating-bit protocol. You should do two state diagrams, one for the sender side and one for the receiver side. The diagram should display the transmission and reception of data.

NOTE! Show the state diagrams to the supervisor before you start to implement.

Part 2
This part contains the actual implementation of the transport protocol. Code for the simulated network environment and for the simulated application layer you can find at the bottom of the page (Jim.c). This code should not be changed and you do not have to read it. Below you also find declarations of the functions that you should write (Stud.c). In the specification you can find detailed descriptions of these functions and what they should do. For the program to work, it is important not to change the name of the functions as they are called by the simulated environment. There is also a header file, which is the interface between your part and the part of the author (Stud.h).
Read through the specification carefully, there are valuable tips in it that makes the work easier.

To pass the lab, the program should be test-run by a supervisor and the documentation should be approved.

Documentation

The documentation should largely follow the template for the lab reports (see "Dokumentationsmall" in it's learning) with a few exceptions described below.

Overview

Describe how the different parts are connected to each other, how the communication between the sender and the receiver are working and what role the various layers have. Be generous with figures, attach also the two state diagrams you did in the first exercise.

Detailed Description

You do not need to describe what each function does, as it is already specified. Describe instead how you have resolved the following points:

· Calculation and control of the checksum
· Management of the timer, how you arrived at the right time-out value

Appendix

The code for the functions that you have written should be attached to the report. Please use explanatory comments even inside functions, and remember to use distinct names for variables, functions etc