Project Parts 3 and 4: A Simple Data Link Layer Protocol

Overview

This is Part 3 and 4 of a project consisting of four parts. Refer to your Suggested Course Schedule to confirm your due dates.

Part 3 of the project is worth 13% of your final course grade. Part 4 of the project is worth 5% of your final course grade. Refer to your Suggested Course Schedule to confirm your due dates.

Read all project directions and questions before you begin preparing your answers, and consult your Open Learning Faculty Member if you have any questions about the projects.

In Project Part 3, you will design and implement a simple protocol, called SWAP.

In Project Part 4, you will write a file transfer application, called FTA, that uses SWAP.

Objectives

To apply previously learned course concepts in order to design and develop a simple data link layer protocol.

Project Part 3: Instructions

Read the section called *Project Part 3: Background Information* as you will need this information to complete Part 3 of the project.

After you have read and understand the Background Information section, you will design and implement a simple block transfer protocol called Stop-and-Wait ARQ Protocol (SWAP). SWAP will use the Stop-and-Wait ARQ protocol to support reliability. This protocol and its implementation (described below) are explained in *Project Part 3: Background Information*.

When you implement SWAP, you only need to design and implement SWAP in **swap_read()** and **swap_write()**, using the Simple Data Link Protocol (SDP) and Application Program Interface (API). All the other functions in SWAP have been already coded in the source files swap_client.c and swap_server.c. SDP is also already completely implemented in sdp.c.

(A Project Resource folder can be found on the Homepage of your course management system. It contains your source code files. Locate the following files: swap_client.c and swap_server.c and sdp.c.)

In `swap_client.c`, there are three functions with some global variables:

- int swap_open(unsigned int addr, unsigned short port);
- **int swap_write(int sd, char *buf, int length); // you need to design and implement**
- void swap_close(int sd);

In `swap_server.c`, there are three functions with some global variables:

- int swap_wait(unsigned short port);
- **int swap_read(int sd, char *buf); // you need to design and implement**
- void swap_close(int sd);

You may use the global variable S for the control variable on the SWAP client in swap_client.c, and *R* for the control variable on the SWAP server in `swap_server.c`.

For error checking, you need to use **1 byte checksum algorithm**. Do not forget that you need to use the error checking algorithm for DATA frames as well as ACK frames.

For testing your swap_client.c and swap_server.c after you implement swap_write() and swap_read(), an application to transfer messages, test_swap_client.c and test_swap_server.c. (Go to the Project Resource Folder on your home page to locate the test_swap_client.c and test_swap_server.c files.) Here is an example how to compile the programs. In order to compile the client program, use the following:

```
     $ gcc test_swap_client.c swap_client.c sdp.c –o
test_client
```

In order to compile the server program, use:

```
     $ gcc test_swap_server.c swap_server.c sdp.c –o
test_server
```

After the compilation, assuming 9988 for the port number, you can run the server program first on a terminal:

```
$ ./test_server 9988
```

And you can run the client program on another terminal:

```
$ ./test_client 127.0.0.1 9988
```

If there is no error in your `swap_client.c` and `swap_server.c`, then `test_server` will print the message sent from `test_client`.

Requirements for Project Part 3

Submit `swap_client.c` and `swap_server.c` with screen shots showing how your programs are compiled and run. Please refer to the general guidelines for preparing and submitting your project found under the Assignments Overview tab of your course. Also note that the resource files necessary for these parts of your project can be found under the Project tab.

Marking Criteria	**Weighting**
No syntax error: All requirements are fully implemented without syntax errors. Submitted screen shots will be reviewed with source code.	/5
Correct implementation: All requirements are correctly implemented and produce correct results Submitted screen shots will be reviewed with source code.	/5
Total	/10

Project Part 4: Instructions

In Part 4, you will write a file transfer application (FTA) that uses SWAP. The FTA application will have a server program and a client program, `fta_server.c` and `fta_client.c`., that use SWAP. Refer to `test_swap_client.c` and `test_swap_server.c` to help you understand how to use SWAP in applications. (Go to the Project Resource Folder on your home page to locate the test_swap_client.c and test_swap_server.c files.) The client program sends a file to the server program, which receives the file from the client and saves it as another filename.

Note: The source file to be sent can be long, which means you will need to use a loop for the transfer.

Project Part 4 Requirements

- The source file name and the destination file name must be given to the client program as arguments.
- The client program must transfer the destination file name to the server program before the data transfer.
- **You may need to design a packet format for your file transfer application.**
- The client and server programs should use the above API of SWAP, not the TCP/IP socket, and both of them should also print your name and student number.
- When you compile `fta_client.c` or `fta_server.c`, compile the source files with `swap_client.c` or `swap_server.c`.
- Here are examples:

```
$ gcc fta_client.c swap_client.c sdp.c –o fta_client
$ gcc fta_server.c swap_server.c sdp.c –o fta_server
```

- After you copy a file using FTA, compare the two files to see if the destination file was copied correctly, using the *diff* command.

Submit `fta_client.c` and `fta_server.c` with screen shots showing how your programs are compiled and run. Please refer to the general guidelines for preparing and submitting your project found under the Assignments Overview tab of your course. Also note that the resource files necessary for these parts of your project can be found under the Project tab.

Marking Criteria	**Weighting**
No syntax error: All requirements are fully implemented without syntax errors. Submitted screen shots will be reviewed with source code.	/5
Correct implementation: All requirements are correctly implemented and produce correct results Submitted screen shots will be reviewed with source code.	/5
Total	/10