First Laboratory Project

Data Link Protocol

Master in Computer Engineering and Informatics

Computer Networks

Grupo Nodes_3:
Carolina Centeio Jorge - up201403090
Tiago Almeida - up201305665

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

3 de Novembro de 2016

1 Summary

This laboratory project was done in the scope of the subject Computer Networks of the Master in Computer Engineering and Informatics course in the school year of 2016/2017. All required knowledge and skills were provided both in the theoretical and practical classes of this subject, giving special attention to the Application Layer and Data Link Layer slides. Also, the laboratory guide was an essential tool for the development, as it contained all the requirements and all the necessary guides for this project.

This project's development was extremely important, since it allowed us to understand better how the theoretical concepts are actually applied to real world problems. We concluded that to successfully transfer files between two machines, we must guarantee the integrity of all transfered data and therefore, the usage of a data link protocol is crucial.

2 Introduction

In the first few practical classes, we learnt how to transfer supervision frames and how to set up an alarm based on a timeout system, so the transmitter knew when to resend data. The goal of this project was to implement a data link protocol between two computers through a serial port and test it by transferring a gif file, although, due to the fact that the file is read as a binary file, other formats would also be supported. The project was developed in a computer with an operating system based on LINUX, using only the C programming language and RS-232 serial ports with asynchronous communication.

This report will explain in better detail the solution we chose to implement for this problem. In the sections 2 and 3, respectively, we describe the project's architecture and structure of the code. Also, in section 4, we demonstrate the main use cases. In the section 5 and 6, we analyze in detail the data link protocol and application protocol, respectively. In section 7, we describe the tests that were made to validate the program and present some results and finally, in section 8, we declare the appreciation elements that were implemented.

3 Project's Architecture

3.1 Layers

The main program can be divided in two modules, the **receiver** and the **transmitter**. Despite this division, the main program is just one and the user is the one who decides which computer is the transmitter and which one is the receiver by passing it as an argument.

There are two main layers that allow the program to run correctly called **application layer** and **link layer**. The application layer is responsible for the transfer and the reception of files, since it's in this layer that the functions to send and receive data packages are executed. The link layer is responsible for aspects related with the serial port, since it's in this layer that the functions to send and receive supervision and information frames are executed. Although, none of this would be possible with the API functions to write and send frames.

3.2 Interface

The user is able to choose the port, the data package size, the number of tries in case of error, the timeout, the file and if the computer is going to behave as the transmitter or the receiver by passing those as arguments when calling the main function. While the program is running, it is displayed in the console every frame that is being sent and received as well as the respective responses. If the transfer is successful, the program will print the size of the received file or, in case of error, it will print an error message with what went wrong.

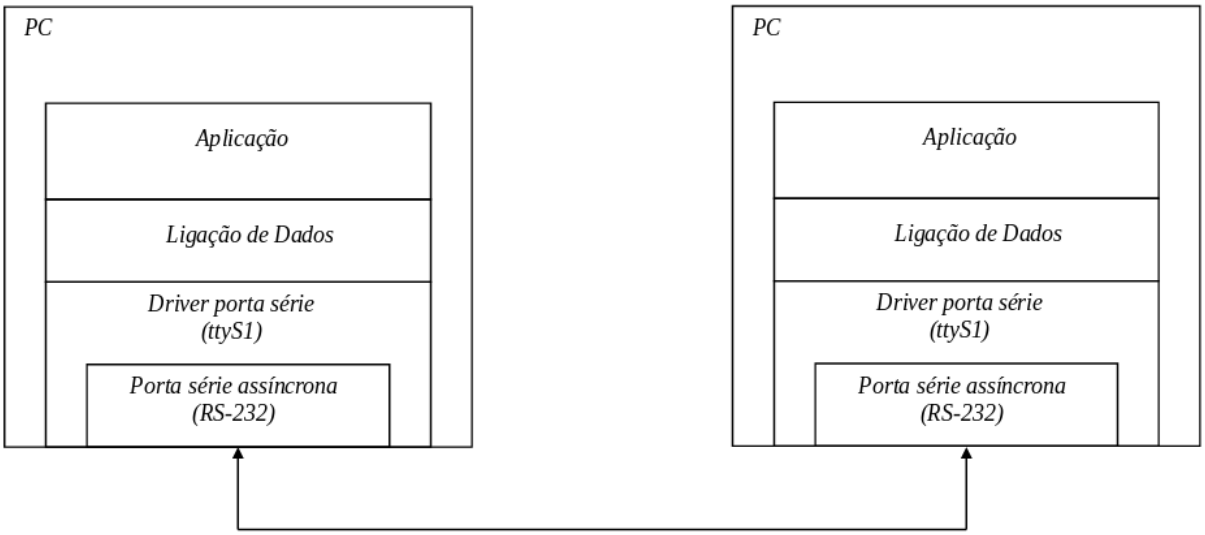

Fig. 1: Program Architecture

4 Code Structure

The application layer is implemented in the files ApplicationLayer.c and ApplicationLayer.h. As suggested in the laboratory guide, we used a struct to store this layer data.

```
struct applicationLayer {
    int fd;
    int status; //Connection mode, 0 - Receiver, 1 - Transmitter
    unsigned int messageSize;
    char* fileName;
};
```

The functions of the application layer are:

```
int writeControlPackage(int control, char* fileName, char * fileSize);
int initializeApplicationLayer(char* port, unsigned int messageSize, int retries, int timeout,
    char* fileName, int status);
int send();
int receive();
```

The link layer is implemented in the files LinkLayer.c and LinkLayer.h. Again, as suggest in the laboratory guide, we used a struct to store the data of this layer.

```
struct linkLayer {
    char port[20];
    unsigned int sequenceNumber;
    unsigned int timeout;
    unsigned int triesMAX;
};
```

The functions of the link layer are:

```
void handleAlarm();
int sendMessage(int fd, char* message);
int receiveMessage(int fd, char* message);
```

```
int initializeLinkLayer(int fd, char * port, int triesMAX, int timeout);
int llopen(int fd, int connectionMode);
int llwrite(int fd, char* buffer, unsigned int length);
int llread(int fd, char* buffer);
int llclose(int fd, int mode);
unsigned int dataStuffing(char* buffer, unsigned int frameSize);
unsigned int dataDestuffing(char* buffer, unsigned int frameSize);
char findBCC2(char* data, unsigned int size);
```

At the Lab1.c file we create the function main and initialize some variables.

5 Use Cases

The application has many use cases. The function llopen is used to establish the connection and the function llclose is used to close the connection. When the program is executed as transmitter, we use the function send to open the file and create the initial buffer (send) which is send by the function llwrite. When the program is executed as receiver, buffers are read through the function llread. The destination file is open so data can be written there with the function receive.

6 Link Layer Protocol

This protocol main objective is to organize the file information. However, opposite to the supervision frames, frames used in this protocol send information about the file. In the following code, we can see that all frames have 6 common fields: FLAG (frame[0] and frame[5 + length]), sequence bit (frame[2]), it can be 0 or 1, verification bit (frame[3]), address field (frame[1]), BCC control field (frame[4 + length]).

We also have the following function to find our BCC control field.

Our implementation includes a timeout, that is used when the connection is lost. Timeout is a value in seconds, at the end of that time, if the connection is not established the program finish. We also have a variable that says us how many tries can the program do to resend packages. If timeout didn't finished and we have already finished all tries of resending, the program finishes without timeout reach the end.

7 Application Protocol

The application layer contains all the higher-lever protocols. Particularly, this layer opens the port, transfers a file (sending it if it is the transmitter or receiving it if it is the receiver) and closes the port. More specifically, the application layer is responsible for the control packages (start and end) and for the data packages. In the function initializeApplicationLayer (function called in main), the port is opened, the Link Layer is initialized, it then calls send or receive function and closes the port.

In send function, after the connection is established through Link Layer, the Control Package is written and sent as START package. If no errors occur, the data packages (parts of the file that is being transferred) starts being sent. At the end, the Control Package is written and sent as END package and the connection is finished through Link Layer.

In receive function, after the connection is established through Link Layer, the Control Package is received as START package. If no errors occur, the data packages (parts of the file that is being transferred) starts being received and the file is built. At the end, the Control Package is received as END package and the connection is finished through Link Layer.

8 Validation

While the implementation was being done, a lot of tests were done to the program, such as, reception capacity, "pinguim.gif" image sending process. We tested with normal conditions, and after we also

tested in case of losing and returning connection. In both situations we had the expected response, overcoming duplicated files and interruptions in sending packages. Receiving "pinguim.gif" image on the receiver computer was done with success.

9 Valuing Elements

This program allows the user to choose the maximum size of the data packages that are going to be sent, how many times the program tries to resend packages in case of error, and even the time it waits to retry it. If any error (through BCC) is detected in a frame and it is a frame that has not been sent before, the receiver send a REJ (reject message) so it can be sent again. The program also checks the size of the file received and compares it to the size of the file sent. If any error that disables the file to be transferred entire and correctly occurs, the program is finished and error is reported to the user.

10 Conclusion