

Faculdade de Engenharia da Universidade do Porto

**Data Link Protocol**

Instructor:

Eduardo Nuno Moreira Soares de Almeida

Redes de Computadores

Turma 15 – Grupo 2

- Clarisse Maria Teixeira de Carvalho (up202008444@fc.up.pt)

- Maria Eduarda Pacheco Mendes Araújo ([up202004473@fc.up.pt](mailto:up202004473@fc.up.pt))

Porto, 31 de outubro de 2024

**Summary:**

This project was carried out as part of the Computer Networks course and focuses on implementing a data communication protocol for file transfer over na RS-232 serial port.

foi realizado no âmbito da Unidade Curricular de Redes de Computadores

**Introduction:**

The goal of this project is to implement a data link layer protocol based on the provided specifications. This protocol enables a transmitter and receiver to transfer files stored on the hard disk between two computers connected via na RS-232 serial cable.

We developed and tested a data link protocol, in line with the specifications provided, for file transfers through a serial port.

The report is divided as follows:

* **Architecture:** Functional blocks and interfaces.
* **Code Structure:** APIs, main data structures, key functional and their relation to the architecture.
* **Main Use Cases:** Identification of core project functionalities, including function cal sequences.
* **Logical Link Protocol:** Logical connection functionality and implementation strategies.
* **Application Protocol:** Application layer functionality and implementation strategies.
* **Validation:** Tests performed to assess implementation correctness.
* **Data Link Protocol Efficiency:** Evaluation of the Stop & Wait protocol efficiency in the data link layer.
* **Conclusions:** Summary of information presented in previous sections and refletion on learning outcomes.

**Architecture:**

- Functional Blocks:

The project consistes of two main layers – *LinkLayer* and *ApplicationLayer*:

*LinkLayer*: Implements the data link protocol, found in link\_layer.h and link\_layer.c. This layer is responsible for establishing and terminating connections, creating and sending data frames via the serial port

Application Layer: Implemented in application\_layer.h and application\_layer.c, this layer uses the Link Layer API to transfer and receive file data packets. It provides a user interface for defining parameters like frame size, transfer speed, and maximum retries.

- Interfaces:

Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, número, documento

Descrição gerada automaticamente The program is executed using two terminals, each on a different computer. One computer runs the binary in transmitter mode and the other in receiver mode.

**Estrutura do código:**

*LinkLayer:*

Uma imagem com texto, captura de ecrã, Tipo de letra

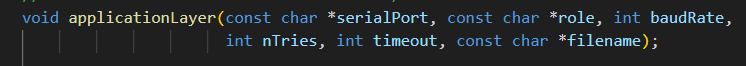
Descrição gerada automaticamente Two auxiliary data structures were used in this layer: LinkLayer, where the parameters associated with data transfer are characterized, and LinkLayerRole, which identifies whether the computer is a transmitter or receiver

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

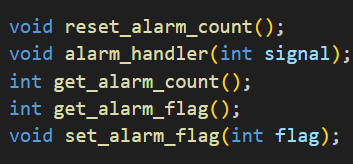
There were the functions implemented

*ApplicationLayer*

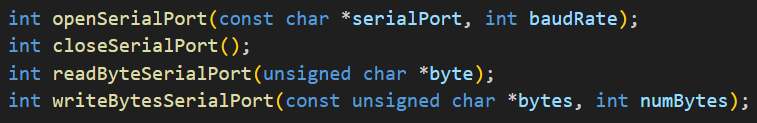


In implementing this layer, there was no need to create auxiliary data structures

*Alarm*



These were the functions implemented

*Serial Port*

There were the functions implemented

*State*

Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

**Main use cases:**

The program can be run in transmitter and receiver modes. The functions to be used and the sequence of calls will differ depending on the choice made.

- Transmiter:

1. **llopen():**  *handshake* between transmitter and receiver, by exchanging connection control packets with the serial port.

- Receiver:

[…]

**Logical Link Protocol:**

The data link layer is the one that interacts directly with the Serial Port and is responsible for communication between the sender and receiver. The Stop & Wait protocol is used to establish and terminate the connection for sending frames.

The connection is established by **llopen()**. After opening and configuring the serial port, the transmitter sends a SET supervisory latch and waits for the receiver to respond with a UA supervisory frame. When the receiver receives SET, it responds with UA (as expected). If the transmitter receives the UA plot, the connection has been well established. After this, the transmitter starts sending information that will be read by the receiver.

The information is sent by **llwrite()**. […]

The information is read by the **llread()** function, which reads the information received by the serial port and checks its validity. It initially destuffs the plot data field and validates BCC1 and BCC2, which check if any errors occurred during transmission.

Finally, we terminate the connection with **llclose()**, which is invoked by the sender when the number of failed attempts is exceeded or when the transfer of data packets is complete.

**Application Protocol:**

**Validation:**

The following tests were carried out throughout the project:

* Partial and/or total interruption of the Serial Port
* Introduction of noise into the Serial Port via a short circuit.
* Rejection of data plots by introducing randomness

The tests have not yet been tested in the presence of the teacher (we have not yet had the opportunity to give the presentation in the laboratory class).

[…]

**Data Link Protocol Efficiency:**

- *Baudrate* variation

- Frame size variation

- Error rate variation

**Conclusions:**

With this project we were able to consolidate what we had learned in theory classes. This project made us understand that *LinkLayer* was responsible for interacting with the serial port and managing the information webs and that *ApplpicationLayer* was responsible for the layer that interacted directly with the file that was going to be transferred.