

**Redes de Computadores**

**Mestrado Integrado de Engenharia Informática e Computação, 3ºano**

10 de novembro de 2020

Protocolo de Ligação de Dados

Daniel Garcia Silva, up201806524@fe.up.pt

Mariana Truta, up201806543@fe.up.pt

Turma 3

Índice

[1. Sumário 2](#_Toc55769016)

[2. Introdução 2](#_Toc55769017)

[3. Arquitetura 3](#_Toc55769018)

[4. Estrutura de código 3](#_Toc55769019)

[appSender 3](#_Toc55769020)

[5. Casos de uso principais 4](#_Toc55769021)

[6. Protocolo de ligação lógica 4](#_Toc55769022)

[7. Protocolo de aplicação 4](#_Toc55769023)

[8. Validação 4](#_Toc55769024)

[9. Eficiência do protocolo de ligação de dados 4](#_Toc55769025)

[10. Conclusão 4](#_Toc55769026)

# Sumário

No âmbito da unidade curricular de *Redes de Computadores*, foi elaborado um projeto que consistia no desenvolvimento de um *software* que permitisse a **transferência** de ficheiros de um computador para o outro, estando estes ligados por um **cabo série**.

Ao longo deste relatório, será explicado como foram **cumpridos** todos os objetivos do projeto, tendo sido concluída uma aplicação funcional e sem perdas de dados.

# Introdução

Este primeiro projeto tinha dois grandes objetivos: implementar um **protocolo de ligação de dados**, especificado no guião fornecido pelos docentes, e testá-lo com uma **aplicação** simples de transferência de ficheiros. Relativamente ao ambiente de desenvolvimento, o trabalho foi realizado em LINUX, utilizando a linguagem de programação C e portas série RS-232 cuja comunicação é assíncrona.

Neste relatório, pretende-se tornar claro como foi possível a elaboração de um **serviço de comunicação fiável** entre dois computadores por via de uma porta de série assíncrona, apresentando detalhes de toda a **teoria** utilizada. Este relatório está estruturado da seguinte forma:

* **Arquitetura:** descriçãodosblocos funcionais e interfaces implementadas;
* **Estrutura de código:** apresentação das *APIs*, principais estruturas de dados e funções e sua relação com a arquitetura;
* **Casos de uso principais:** identificação dos casos de uso e sequências de chamadas de funções;
* **Protocolo de ligação de dados:** identificação dos principais aspetos funcionais bem como a descrição da estratégia de implementação dos mesmos, sendo complementada com a apresentação de extratos de código;
* **Protocolo de aplicação:** identificação dos principais aspetos funcionais bem como a descrição da estratégia de implementação dos mesmos, sendo complementada com a apresentação de extratos de código;
* **Validação:** descrição dos testes efetuados com apresentação quantificada dos resultados;
* **Eficiência de protocolo de ligação de dados:** caraterização estatística da eficiência do protocolo, recorrendo a medidas sobre o código desenvolvido;
* **Conclusão:** síntese da informação apresentada nas secções anteriores; reflexão sobre os objetivos de aprendizagem alcançados.

# Arquitetura

O projeto está dividido em duas camadas muito bem definidas: a camada de **protocolo de ligação de dados** e a camada da **aplicação**.

O objetivo do **protocolo de ligação de dados** é fornecer um serviço de comunicação de dados **fiável** entre dois sistemas ligados por um **cabo série**. Esta camada contem, assim, todas as funções necessárias para a **abertura**, **fecho**, **escrita** e **leitura** da porta de série. Para além disso, tem também a seu encargo o controlo de **erros** e **fluxo** e o *stuffing*/*destuffing* de pacotes.

A camada da **aplicação** está situada acima da camada de ligação de dados, sendo responsável pelo **envio** e **receção** de ficheiros. Tem de permitir enviar e identificar os **pacotes de controlo** bem como **dividir** o ficheiro em vários pacotes, no caso do emissor, ou **concatenar** e **interpretar** toda a informação recebida, no caso do recetor.

É importante referir que esta arquitetura baseia-se no **principio de independência entre camadas**, isto é, ao nível da camada de ligação de dados, não existe qualquer distinção ou processamento que incida sobre o cabeçalho dos pacotes a transportar em tramas de informação e a camada da aplicação não conhece os detalhes da outra camada mas apenas a forma como acede ao serviço.

# Estrutura de código

A **camada da** **aplicação** está dividida em dois ficheiros: *appSender*, que é a parte utilizada pelo emissor, e *appReceiver*, que é utilizada pelo recetor. As funções e estruturas de dados utilizadas são as seguintes:

## appSender

* ***int main(int argc, char \*\*argv)***

É responsável pela escrita de um ficheiro, sendo o *path* fornecido pelo utilizador.

## appReceiver

* ***int main(int argc, char \*\*argv)***

É responsável pela leitura de um ficheiro, sendo este guardado em *“./imagesToReceive/”*.

* ***void freeFile()***

Liberta a memória que foi alocada à *struct* *File.*

* ***int parseInfo(unsigned char \*info, int size)***

Encaminha o pacote de dados para função respetiva, tendo em conta o seu primeiro byte (*START\_BYTE*, *DATA\_BYTE* e *END\_BYTE*).

* ***int parseControlPacket(unsigned char \*info, int size)***

Interpreta a informação guardada na *info*, armazenando o nome e o tamanho do ficheiro que irá ser recebido na *struct File*.

* ***int checkControlPacket (unsigned char \*info, int size)***

Verifica se o pacote de controlo de finalização contém a mesma informação que o pacote de controlo que sinalizou o inicio da transmissão, diferenciando apenas no campo de controlo.

* ***int parseDataPacket(unsigned char \*info, int size)***

Guarda os dados do ficheiro no campo *data* da *struct File*.

* ***int initFile()***

Inicializa a *struct File*, alocando a memória necessária.

* ***struct File***

Esta *struct* armazena as várias informações necessárias para escrever um ficheiro, como por exemplo o seu tamanho e o seu nome.

## dataLink

Por sua vez, a **camada de ligação de dados** está dividida em quatro principais funções que recorrem a ficheiros e funções auxiliares:

* ***int llopen(int port, int status)***

Estabelece a ligação entre o emissor e o recetor, usando um mecanismo de *STOP and WAIT* e garantindo que se reúnem as condições necessárias para começar a transmissão de dados.

* ***int llwrite(int fd, char \*buffer, int length)***

É responsável pela incorporação, numa trama *I*, de um conjunto de *bytes*, *buffer,* sendo realizado o *byte stuffing* do pacote, e pelo envio desta mesma trama para o recetor. Para além disso, tem de ser capaz de agir de acordo com uma mensagem *ACK* que o recetor irá enviar como resposta.

* ***int llread(int fd, char \*buffer)***

É responsável pela leitura de uma mensagem enviada pelo recetor e após o *byte destuffing* do pacote, envia uma resposta de rejeição, no caso do pacote não ser válido, ou de aceitação, se o pacote for válido.

# Casos de uso principais

# Protocolo de ligação lógica

# Protocolo de aplicação

# Validação

# Eficiência do protocolo de ligação de dados

# Conclusão