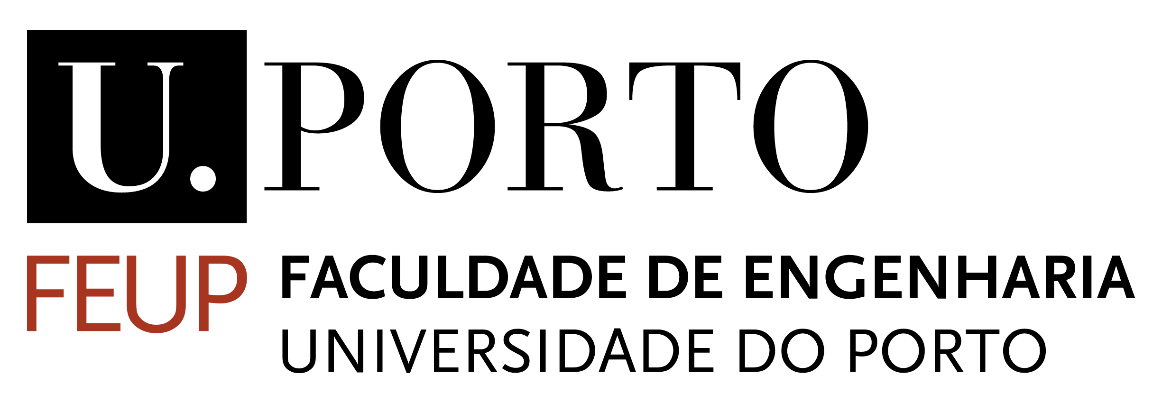
**Protocolo de Ligação de Dados**

**Relatório do 1º Trabalho Laboratorial**



Mestrado integrado em Engenharia Informática e Computação

Redes de Computadores

Bruno Marques - 201405781

João Loureiro - 201405652

José Cruz - 201403526

José Costa - 201402717

Faculdade de Engenharia da Universidade do Porto

Rua Roberto Frias, sn, 4200-465 Porto, Portugal

3 de novembro de 2016

Índice

[Sumário 3](#_Toc466500250)

[1. Introdução 3](#_Toc466500251)

[2. Arquitetura 3](#_Toc466500252)

[3. Estrutura do Código 4](#_Toc466500253)

[4. Casos de Uso Principais 5](#_Toc466500254)

[4.1. Sequência de chamada de funções 5](#_Toc466500255)

[5. Protocolo de Ligação Lógica 6](#_Toc466500256)

[6. Protocolo de Aplicação 6](#_Toc466500257)

[7. Validação 6](#_Toc466500258)

[8. Elementos de Valorização 6](#_Toc466500259)

[9. Conclusões 6](#_Toc466500260)

# Sumário

O presente relatório serve de apoio ao 1º projeto realizado no âmbito da Unidade Curricular “Redes de Computadores” do 3º ano do Mestrado integrado em Engenharia Informática e Computação. Projeto este intitulado “Protocolo de Ligação de Dados” e cujo objetivo era transferir ficheiros entre dois computadores usando a porta de série tendo em conta os conteúdos abordados nas aulas teóricas e práticas, dos quais se destacam *Application Layer, Data Link Layer* e *Physical Layer.*

# Introdução

O trabalho prático em questão visa a implementação de um protocolo de ligação de dados, de acordo com a especificação do guião, e o teste deste mesmo protocolo com uma aplicação simples de transferência de ficheiros, igualmente especificada. Este relatório serve para especificar o projeto tanto do ponto de vista prático como teórico e serão caracterizadas todas as funcionalidades implementadas no projeto. Todo o projeto foi desenvolvido em ambiente Linux utilizando a linguagem C e portas de série RS-232, com comunicação assíncrona.

O relatório está dividido em diferentes secções, sendo estas:

* **Arquitetura:** Especificação dos Blocos Funcionais e da Interface.
* **Estrutura do Código:** Descrição das principais APIs e Estruturas de Dados utilizadas, bem como as principais Funções e a sua relação com a Arquitetura.
* **Casos de Uso Principais:** Fazer a sua Identificação e abordar as sequencias de chamada de funções.
* **Protocolo de Ligação Lógica:** Identificação dos principais aspetos funcionais da *LinkLayer*, descrevendo a estratégia da sua implementação.
* **Protocolo de Aplicação:** Identificação dos principais aspetos funcionais da *ApplicationLayer*, descrevendo a estratégia da sua implementação.
* **Validação:** Descrição dos testes efetuados ao programa com apresentação quantificada dos resultados.
* **Elementos de Valorização:** Identificação dos elementos de valorização implementados e descrição da estratégia da sua implementação.

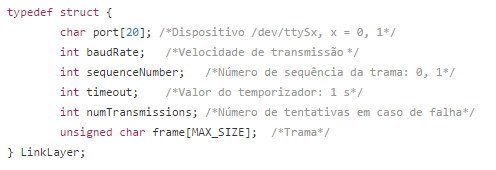
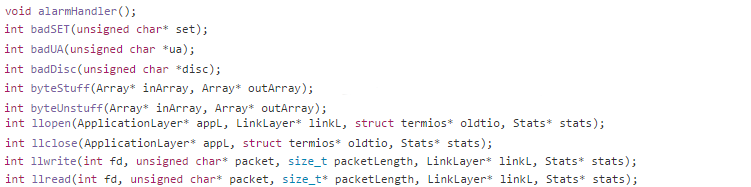
# Arquitetura

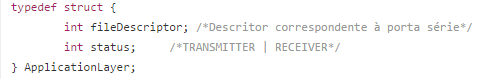
O projeto está dividido em duas camadas funcionais: a camada do protocolo de ligação de dados e a camada de aplicação (implementados em diferentes ficheiros source e header, respetivamente \*.c e\*.h). Os ficheiros link.c e link.h representam a camada do protocolo de ligação de dados e os ficheiros application.c e application.h representam a camada de aplicação. A camada do protocolo de ligação de dados tem as funções de sincronismo e abertura, fecho e configuração da porta de série, *stuffing* e *unstuffing* de bytes. A camada de aplicação cria as tramas e é responsável pelo envio e receção do ficheiro.

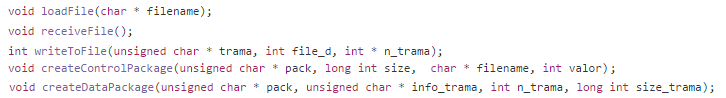
A interface está implementada em interface.c e interface.h onde o utilizador começa por selecionar se é o recetor ou o emissor e permite a escolha de valores de alguns parâmetros referentes à transferência do ficheiro referindo também os limites destes mesmos valores, sendo estes: o *Baudrate* , que porta utilizar(se /dev/ttyS0 ou /dev/ttyS1), o tamanho máximo do campo de informação das tramas, o intervalo *Timeout*, o número máximo de retransmissões e o nome do ficheiro a ser enviado.

# Estrutura do Código

Ambas as camadas são representadas por uma estrutura de dados, estruturas estas que estão implementadas em utils.h.

A estrutura da camada do protocolo de ligação de dados, bem como as suas principais funções estão abaixo.

 A estrutura da camada de aplicação e as suas principais funções.



Para além dos ficheiros onde as camadas estão implementadas (link.c, link.h,application.c e application.h), existem ainda outros ficheiros: utils.c e utils.h (onde estão definidas as estruturas de dados das duas camadas e algumas funções auxiliares, por exemplo, de manipulação de arrays) e interface.c e interface.h(onde está definida a interface).

# Casos de Uso Principais



## Sequência de chamada de funções

Em traços gerais, o nosso programa segue a seguinte sequência de funções durante a sua execução:

1. Obter a partir do utilizador as definições a usar na conexão (menu\_cycle()).
2. Criar e abrir a conexão partilhada, tanto no emissor como no recetor (ll\_open()).
3. No emissor:
   1. Ler a informação do ficheiro escolhido para transmissão e empacotá-la (loadFile()).
   2. Criar e enviar tramas que contenham os pacotes de informação, decidindo o próximo passo pela resposta do emissor (ll\_write()).
4. No recetor:
   1. Receber tramas de informação e enviar uma resposta ao emissor, retirando das tramas os pacotes de informação (ll\_read()).
   2. Gerar um novo ficheiro a partir dos bytes dos pacotes recebidos (receiveFile()).
5. Terminar a ligação (ll\_close()).

# Protocolo de Ligação Lógica

O protocolo de ligação lógica é definido e implementado no ficheiro *link.c*, que inclui as funções que o protocolo de aplicação utiliza para enviar informações de um controlador para o outro.

## Principais aspetos funcionais

* Definir a configuração da porta série.
* Criar a ligação entre portas série.
* Usar o byte *stuffing* e *destuffing* para criar tramas.
* Enviar e receber tramas de informação ou de controlo.
* Terminar a ligação, repondo as configurações prévias da porta-série.

## Estratégia de implementação

### send\_cycle

A transmissão de informação pela porta série segue um padrão de envio de comando e receção de resposta. Assim, foi elaborada uma função que permite escrever um determinado comando para um descritor de ficheiro (que no presente caso é a porta-série) e que reenvia esse comando um determinado número de vezes, utilizando o alarme, até receber resposta por esse mesmo descritor. As funções ll\_open, ll\_close, ll\_read e ll\_read recorrem todas ao send\_cycle quando se pretende trocar informação entre controladores.

int send\_cycle**(**int fd**,** unsigned char **\*** sendMsg**,** int size**,** unsigned char **\*** received**,** Stats**\*** stats**){**

flag **=** 1**;**

numOfTries **=** 0**;**

int res**;**

**(**void**)**signal**(**SIGALRM**,** alarmHandler**);** /\* sets alarmHandler function as SIGALRM handler\*/

int writtenChars **=** 0**;**

//Cycle that sends the SET bytes, while waiting for UA

**while(**numOfTries **<** max\_num\_tries**){**

**if(**flag**)**

**{**

alarm**(**time\_out**);** /\* waits x seconds, then activates a SIGALRM \*/

flag **=** 0**;** /\* doesn't resend a signal until an alarm is handled \*/

tcflush**(**fd**,** TCIOFLUSH**);**

writtenChars **=** write**(**fd**,** sendMsg**,** size**);** // writes the flags

**}**

res **=** read**(**fd**,** received**,** size**);**

**if(**res **>=** 1**)**

**{**

**return** writtenChars**;**

**}**

**if(**numOfTries **==** max\_num\_tries**)**

**{**

stats**->**numTimeouts**++;**

**}**

**}**

**return** **-**1**;**

**}**

### ll\_open e ll\_close

A função **ll\_open** configura a porta-série a partir dos atributos da struct *applicationLayer*, que foi definida para o programa. Do lado do emissor é enviado o comando SET, através da função send\_set, até ser recebida a resposta UA vinda do emissor.

**if(**appL**->**status **==** TRANSMITTER**)**

send\_set**(**fd**,** stats**);**

**else**

receive\_set**(**fd**);**

A função **ll\_close** envia o comando DISC a partir do emissor, quando os pacotes de informação acabam de ser transmitidos. O recetor, ao receber o comando de terminação, responde também com uma mensagem DISC, que, ao ser interpretada pelo emissor, é retornada com UA.

**if(**appL**->**status **==** TRANSMITTER**)**

close\_set**(**fd**,** stats**);**

**else**

close\_ua**(**fd**);**

### ll\_write e ll\_read

Como as funções ll\_write e ll\_read necessitam de alocação dinâmica, foi elaborada uma struct *Array* que possui funções capazes de simular um vetor dinâmico em C, de forma a não ser necessário lidar sempre com alocações de memória. O byte *stuff* e *destuff* é feito recorrendo a funções desta struct, como por exemplo:

void insertArray**(**Array **\***a**,** unsigned char element**)** **{**

**if** **(**a**->**used **==** a**->**size**)** **{**

a**->**size **\*=** 2**;** // double size if full

a**->**array **=** **(**unsigned char**\*)**realloc**(**a**->**array**,** a**->**size **\*** **sizeof(**unsigned char**));**

**}**

a**->**array**[**a**->**used**++]** **=** element**;**

**}**

A função **ll\_write** é executada pelo transmissor e começa por inicializar a trama que se vai enviar com os bytes de controlo do cabeçalho, incluindo o *BCC1*. De seguida, pega no pacote que vai enviar e calcula o *BCC* dos bytes do pacote, anexando o BCC à informação de envio. O *Array* resultante passa depois por byte *stuffing* e recebe o cabeçalho e a FLAG final. O conjunto de bytes obtido depois deste processo é enviado através da send\_cycle para o recetor.

Array packetArray**;**

initArray**(&**packetArray**,** 1**);**

copyArray**(**packet**,** **&**packetArray**,** packetLength**);**

Array stuffedArray**;**

initArray**(&**stuffedArray**,** 1**);**

initializeInformationFrame**(&**stuffedArray**,** linkL**);**

addBodyBCC**(&**packetArray**);**

byteStuff**(&**packetArray**,** **&**stuffedArray**);**

endInformationFrame**(&**stuffedArray**);**

A função **ll\_open** tem o mesmo funcionamento, mas com o sentido contrário ao da ll\_write, fazendo uso de uma função chamada generateResponse() para verificar a validade da trama que recebeu e enviar a devida resposta de volta ao transmissor.

# Protocolo de Aplicação

# Validação

Durante a realização do trabalho procuramos testa-lo de diversas maneiras. Para além de transferirmos a imagem fornecida pelo docente(pinguim.gif), testámos também utilizando outras imagens incluindo uma de maior tamanho e definição. Além das transferências normais, testámos, como viria a ser analisado mais tarde, interromper a ligação durante a transferência e introduzir resíduos raspando um fio no circuito que ligava os dois cabos de série.

# Elementos de Valorização

**Seleção de parâmetros pelo utilizador**

Ao executar o programa, o apresentada uma interface ao utilizador, onde este é capaz de escolher se é o emissor ou o recetor, o *BaudRate*, qual a porta de série que vai usar, o tamanho máximo do campo de informação das tramas, o intervalo de *TimeOut* em segundos, o número máximo de retransmissões, e o nome do ficheiro a enviar.

**Geração aleatória de erros em tramas de informação**

**Implementação de REJ**

Caso a mensagem recebida tenha um erro no campo BCC2 na função llread, a trama REJ é enviada para que o emissor reenvie a mensagem que não chegou ao recetor corretamente. Esta implementação está na função int reject(unsigned char\* rej) do ficheiro link.c.

**Verificação da integridade dos dados pela aplicação**

Após o envio é verificado o tamanho do ficheiro recebido. Esta verificação está implementada em long int verifyControlData(unsigned char \* data ,char \* file\_name, int value) no ficheiro application.c.

**Registo de ocorrências**

Na estrutura de dados *Stats* declarada em utils.h são guardadas, ao longo da execução do programa, diversos valores, nomeadamente o número de tramas I enviadas e recebidas, o número de ocorrências de *TimeOuts*, e o número de REJ e RR enviados e recebidos.

# Conclusões

Os objetivos do projeto foram cumpridos na totalidade, tendo sido implementado um protocolo de ligação de dados e um protocolo de aplicação capazes de enviar ficheiros de um computador para outro através da porta de série.

Ao longo da realização do trabalho prático foram sentidas algumas dificuldades nomeadamente com o controlo de erros e com a sincronização na execução da aplicação nos dois computadores, mas estas foram ultrapassadas.

Em suma, através da realização deste projeto conseguimos consolidar de melhor maneira alguns dos conceitos abordados nas aulas teóricas e aprofundamos o nosso conhecimento do funcionamento das comunicações em rede, através do uso da porta de série.