

Redes de Computadores

*1º Trabalho Laboratorial*

*Mestrado Integrado em Engenharia Informática e Computação*

*(7 de novembro de 2017)*

Bárbara Silva  **up201505628**@fe.up.pt

Catarina Ferreira **up201506671**@fe.up.pt

Julieta Frade **up201506530**@fe.up.pt

Índice

|  |  |
| --- | --- |
| **Sumário** | 0 |
| **Introdução** | 2 |
| **Arquitetura** | 2 |
| **Estrutura do código** | 2 |
| **Casos de uso principais** | 2 |
| **Protocolo de ligação lógica** | 2 |
| Identificação dos principais aspetos funcionais | 2 |
| Descrição da estratégia | 2 |
| **Protocolo de aplicação** | 2 |
| **Validação** | 2 |
| **Eficiência do protocolo de ligação de dados** | 2 |
| **Conclusões** | 2 |
| **Anexo I** | 2 |

*Sumário*

Este relatório trata-se de uma complementação ao primeiro trabalho laboratorial da unidade curricular de Redes de Computadores, cuja essência é a transferência de dados. O trabalho consiste no desenvolvimento de uma aplicação capaz de transferir ficheiros de um computador para o outro através de uma porta de série.

Isto posto, o trabalho foi concluído com sucesso, visto que todos os objetivos estabelecidos foram cumpridos e foi finalizada uma aplicação perfeitamente funcional e capaz de transferir ficheiros sem perda de dados.

(dois parágrafos: um sobre o contexto do trabalho; outro sobre as principais conclusões do relatório)

*Introdução*

O objetivo deste trabalho é implementar um protocolo de ligação de dados, de acordo com o guião fornecido, e testar o protocolo com uma aplicação simples de transferência de ficheiros, recorrendo a uma porta de série. Quanto ao relatório, o seu objetivo é expor e explicar toda a componente teórica presente neste primeiro trabalho, tendo a seguinte estrutura:

* **Arquitetura**

Exibição dos blocos funcionais e interfaces presentes.

* **Estrutura do código**

Demonstração das APIs, principais estruturas de dados, principais funções e sua relação com a arquitetura.

* **Casos de uso principais**

Identificação dos mesmos e demonstração das sequências de chamada de funções.

* **Protocolo de ligação lógica**

Identificação dos principais aspetos funcionais e descrição da estratégia de implementação dos mesmos com apresentação de extratos de código.

* **Protocolo de aplicação**

Identificação dos principais aspetos funcionais e descrição da estratégia de implementação dos mesmos com apresentação de extratos de código.

* **Validação**

Descrição dos testes efetuados com apresentação quantificada dos resultados.

* **Eficiência do protocolo de ligação de dados**

Caraterização estatística da eficiência do protocolo, feita com recurso a medidas sobre o código desenvolvido.

* **Conclusão**

Síntese da informação apresentada nas secções anteriores e reflexão sobre os objetivos de aprendizagem alcançados.

 (indicação dos objetivos do trabalho e do relatório; descrição da lógica do relatório com indicações sobre o tipo de informação que poderá ser encontrada em cada uma secção seguinte)

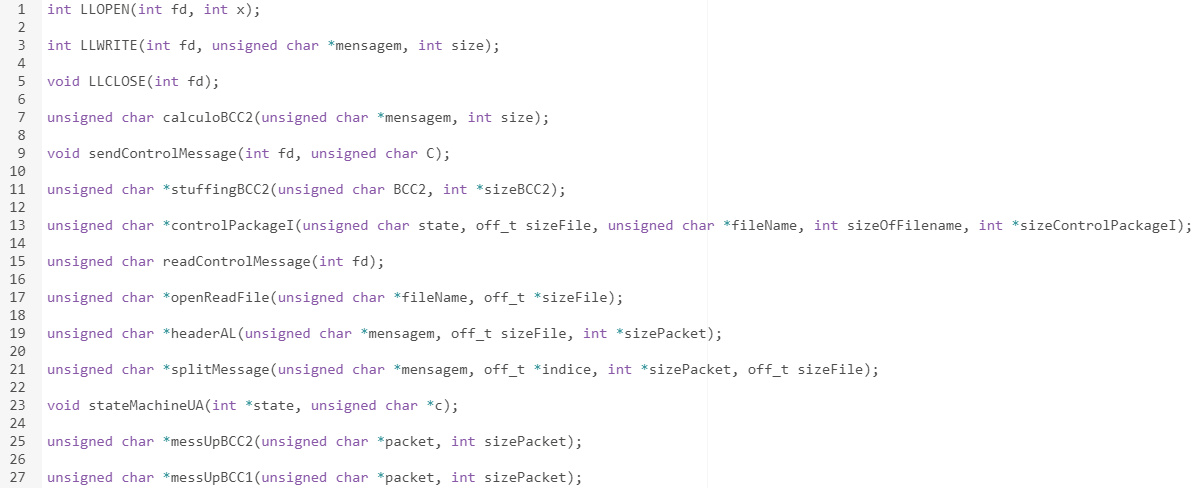
*Arquitetura*

O trabalho está dividido em dois blocos funcionais, focando-se no emissor e no recetor. Cada um destes blocos incorpora a sua própria camada de ligação de dados e de aplicação. Note-se que, foi decidido dividir desta forma com o intuito de isolar os casos.

Quanto à interface, o utilizador, de forma a dar início à aplicação, deverá inserir uma linha no terminal com os respetivos argumentos.

(blocos funcionais e interfaces)

*Estrutura de código*



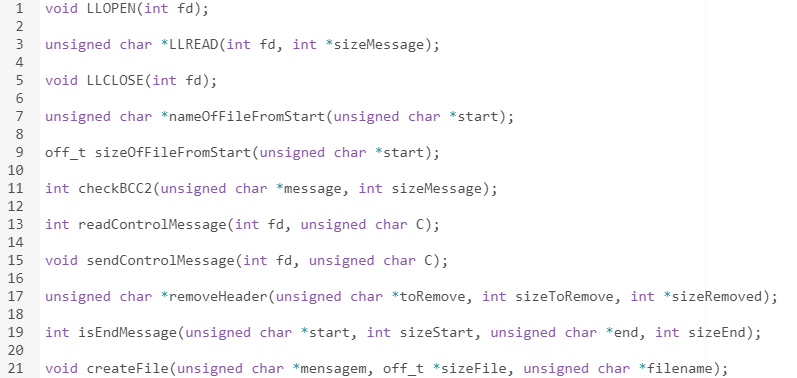
**Figura 1**: Funções presentes no ficheiro *writer.h*

No que diz respeito à camada de ligação de dados temos as seguintes funções:

* **LLOPEN** – envia trama de controlo SET e recebe trama UA
* **LLWRITE** – realiza *stuffing* das tramas I e envia-as.
* **LLCLOSE** – envia trama de controlo DISC, recebe DISC e envia UA.
* **calculoBCC2** – calcula o valor do BCC2 de uma mensagem.
* **stuffingBCC2** – realiza o *stuffing* do BCC2.
* **readControlMessage** – lê tramas de controlo.
* **stateMachineUA** – verifica se UA foi recebido.
* **messUpBCC1** – geração aleatória de erros no BCC1.
* **messUpBCC2** – geração aleatória de erros no BCC2.

Analisando agora a camada de aplicação, temos as seguintes funções:

* **main** – base da camada de aplicação pois é esta que controla todo o processo que ocorre nesta camada e que faz as chamadas às funções da camada de ligação.
* **sendControlMessage** – envia uma trama de controlo, sendo o C recebido como argumento da função a diferença de cada trama enviada.
* **controlPackageI** – cria os pacotes de controlo START e END.
* **openReadFile** – abre um ficheiro e lê o seu conteúdo.
* **headerAL** – acrescenta o cabeçalho do nível de aplicação às tramas.
* **splitMessage** – divide uma mensagem proveniente do ficheiro em packets.



**Figura 2**: Funções presentes no ficheiro reader*.h*

No que diz respeito à camada de ligação de dados temos as seguintes funções:

* **LLOPEN** – lê trama de controlo SET e envia a trama UA.
* **LLREAD** – lê tramas I e faz *destuffing*.
* **LLCLOSE** – lê trama de controlo DISC, envia DISC de volta e recebe UA.
* **checkBBC2** – verifica se o BCC2 recebido na mensagem está correto.

Analisando agora a camada de aplicação, temos as seguintes funções:

* **main** – base da camada de aplicação pois é esta que controla todo o processo que ocorre nesta camada e que faz as chamadas às funções da camada de ligação.
* **nameOfFileFromStart** – obtém nome do ficheiro a partir da trama START
* **sizeOfFileFromStart** – obtém tamanho do ficheiro a partir da trama START.
* **readControlMessage** – ciclo de leitura que quebra após ler uma trama de controlo C que seja igual ao C recebido como argumento da função.
* **sendControlMessage** – envia uma trama de controlo, sendo o C recebido como argumento da função a diferença de cada trama enviada.
* **removeHeader** – remove o cabeçalho do nível de aplicação das tramas I.
* **isEndMessage** – verifica se a trama recebida e a trama END.
* **createFile** – cria ficheiro com os dados recebidos nas tramas I.

Dizer que não foram usadas estruturas de dados? Hmm qual a relação?

  (APIs, principais estruturas de dados, principais funções e sua relação com a arquitetura)

*Casos de uso principais*

Os principais casos de uso desta aplicação são: a interface, que permite o transmissor escolher o ficheiro a enviar, e a transferência desse mesmo ficheiro, via porta de série, entre dois computadores, o transmissor e o recetor. A sequência de chamada de funções é a seguinte:

* Configurar a ligação entre os dois computadores.
* Transmissor escolhe o ficheiro a enviar.
* Estabelecimento da ligação.
* Transmissor envia dados.
* Recetor recebe os dados.
* São fechados todos os ficheiros abertos.
* Terminar ligação.

  (identificação; sequências de chamada de funções)

*Protocolo de ligação lógica*

Texto.

  (identificação dos principais aspetos funcionais; descrição da estratégia de implementação destes aspetos com apresentação de extratos de código)

*Protocolo de aplicação*

Texto.

  (identificação dos principais aspetos funcionais; descrição da estratégia de implementação destes aspetos com apresentação de extratos de código)

*Validação*

De forma a estudar a aplicação desenvolvida, foram efetuados os seguintes testes:

* Envio de um ficheiro.
* Etc

Todos os testes foram concluídos com sucesso.

  (descrição dos testes efetuados com apresentação quantificada dos resultados, se possível)

*Eficiência do protocolo de ligação de dados*

Texto.

 (caraterização estatística da eficiência do protocolo, feita com recurso a medidas sobre o código desenvolvido. A caracterização teórica de um protocolo Stop&Wait, que deverá ser usada como termo de comparação, encontra-se descrita nos slides de Ligação Lógica das aulas teóricas).

*Conclusões*

Texto.

  (síntese da informação apresentada nas secções anteriores; reflexão sobre os objetivos de aprendizagem alcançados)

*Anexo I*

Código fonte.