

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 |
| Writer | 2 |
| Reader | 2 |
| **Casos de uso principais** | 2 |
| **Protocolo de ligação lógica** | 2 |
| **Protocolo de aplicação** | 2 |
| **Validação** | 2 |
| **Eficiência do protocolo de ligação de dados** | 2 |
| **Conclusões** | 2 |
| **Anexo I** | 2 |
| **Anexo II** | 2 |

*Sumário*

Este relatório foi elaborado no âmbito da unidade curricular de Redes e Computadores, e trata-se de uma complementação ao primeiro trabalho laboratorial, 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 camada de ligação de dados e de aplicação isoladamente. 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 um conjunto de argumentos. Sendo emissor, deverá inserir qual a porta de série a ser utilizada (ex: **/dev/ttyS0)**, e o ficheiro a ser enviado (ex: **pinguim.gif**). Sendo recetor, basta inserir a porta de série.

(blocos funcionais e interfaces)

*Estrutura de código*

**Writer**

Descrever o para que serve/o que acontece no writer, brevemente, tipo 3 linhas.

Funções principais da camada de ligação:

* **llopen** – envia trama de supervisão *SET* e recebe trama *UA*
* **llwrite** – realiza *stuffing* das tramas I e envia-as.
* **llclose** – envia trama de supervisão *DISC*, recebe *DISC* e envia *UA*.

Funções principais da camada de aplicação:

* **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.
* **openReadFile** – abre um ficheiro e lê o seu conteúdo.
* **splitMessage** – divide uma mensagem proveniente do ficheiro em *packets*.

Variáveis globais:

* **sumAlarms** – contador de alarmes, inicializada a 0.
* **flagAlarm** – passa a *TRUE* quando é acionado o alarme, inicializada a *FALSE*.
* **trama** – número sequencial da trama (*Ns*) a enviar, inicializada a 0.
* **paragem** – passa a *TRUE* quando *UA* for recebido, inicializada a *FALSE*.
* **numMensagens** – número de sequência das tramas I (*N*), inicializada a 0.
* **numTotalTramas –** contador de tramas enviadas, inicializada a 0.

Macros pertinentes:

* **BAUDRATE –** Capacidade de ligação**.**
* **NUMMAX –** Número máximo de tentativas de reenvio.
* **sizePacketConst –** Número de bytes em cada *packet.*
* **bcc1ErrorPercentage –** Percentagem de erros gerados no *BCC1.*
* **bcc2ErrorPercentage –** Percentagem de erros gerados no *BCC2.*

**Reader**

Descrever o para que serve/o que acontece no reader, brevemente, tipo 3 linhas.

Funções principais da camada de ligação:

* **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*.

Funções principais da camada de aplicação:

* **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.
* **createFile** – cria ficheiro com os dados recebidos nas tramas I.

Variáveis globais:

* **esperado** – trama (*Nr*) esperada, usada para o tratamento de duplicados, inicializada a 0.

Macros pertinentes:

* **BAUDRATE –** Capacidade de ligação**.**

  (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*

**LLOPEN**

Esta função tem a responsabilidade de estabelecer a ligação entre o emissor e o recetor.

No emissor, esta função envia a trama de controlo *SET* e ativa o temporizador que é desativado depois de receber resposta (*UA*). Se não receber resposta dentro de um tempo *time-out*, *SET* é reenviado. Este mecanismo de retransmissão só é repetido um número máximo de vezes, se este número for atingido o programa termina.

No recetor, esta função espera pela chegada de uma trama de controlo SET para responder com um UA.

Para enviar tanto o *UA* como o *SET* é usada a função ***sendControlMessage***. Esta função recebe como argumento o campo de controlo das tramas de Supervisão, o que permite que a função seja usada para enviar qualquer tipo de trama deste tipo.

Como o mecanismo de receção do *UA* necessita de temporizador, usa-se a função ***stateMachineUA*** que tem o que é necessário para tal. Na leitura do *SET* é usada a função ***readControlMessage*** que recebe como argumento o campo de controlo das tramas de Supervisão para verificar se a trama recebida é de facto do tipo que é desejado.

As escritas são feitas trama a trama, no entanto a leitura é feita carater a carater.

**LLWRITE**

Esta é a função no emissor responsável pelo envio das tramas e pelo *stuffing* das mesmas.

Primeiro é feito o *framing* da mensagem, ou seja, acrescentado o cabeçalho do Protocolo de Ligação à mensagem (para calcular o *BCC2* é chamada a função ***calculoBCC2***). Seguidamente é feito o *stuffing* da mensagem e do *BCC2* (feito na função ***stuffingBCC2***). Posto isto, a trama está pronta a ser enviada. Esta escrita é feita trama a trama.

Antes da escrita são introduzidos erros pelas funções ***messUpBCC1*** e ***messUpBCC2***. Estas funções substituem o conteúdo de uma posição aleatória (no caso do *BCC1* da posição 1 a 3 e no caso do *BCC2* nas posições ocupadas pelo campo de dados e *BCC2*) com uma letra aleatória, tendo em conta uma probabilidade erro escolhida.

O envio da trama tem o mesmo mecanismo de *time-out* e retransmissão que o envio do *SET* no ***llopen***. Ou seja, depois de enviar a trama é acionado um alarme até à receção de uma resposta (*RR* ou *REJ*) e se atingido esse alarme a mensagem é reenvida (mecanismo que se pode ocorrer um numero máximo de vezes). Se recebido um *REJ* a mensagem é reenviada. Para fazer a verificação se é *REJ* ou *RR* é usada a função ***readControlMessage*** que retorna o campo de controlo da trama de Supervisão lida.

**LLREAD**

Esta é a função no recetor responsável pela receção das tramas e pelo *destuffing* das mesmas.

A leitura é feita carater a carater. Para verificar o *BCC2* é usada a função ***checkBCC2***, caso esteja correto é enviado *RR*, caso contrário *REJ* usando a função ***sendControlMessage***. O campo de controlo enviado depende do número de sequência da trama (*Nr*). Seguidamente é feito o *destuffing* do campo de dados. Também é feita uma verificação se o número de sequência de tramas é o esperado para ser possível tratar duplicados.

**LLCLOSE**

Esta função tem a responsabilidade de terminar a ligação entre o emissor e o recetor.

No emissor, é enviado a trama de Supervisão *DISC* com ajuda da função ***sendControlMessage*** (que recebe como argumento o campo de controlo da trama a enviar) e esperado outro *DISC* de volta pela função ***readControlMessage*** (que retorna o campo de controlo da trama lida). Para finalizar é enviado um *UA*.

No recetor é esperado um *DISC*, enviado um *DISC* e esperado um UA com as funções ***readControlMessage*** e ***sendControlMessage*** (que recebem ambas o campo de controlo pretendido).

(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 ficheiros de vários tamanhos.
* Geração de curto circuito enquanto se envia um ficheiro.
* Interrupção a ligação por alguns segundos enquanto se envia um ficheiro.
* Envio do um ficheiro com variação na percentagem de erros simulados.
* Envio do um ficheiro com variação do tamanho de pacotes.
* Envio do um ficheiro com variação da capacidades de ligação (baudrate).

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*

Mais texto.

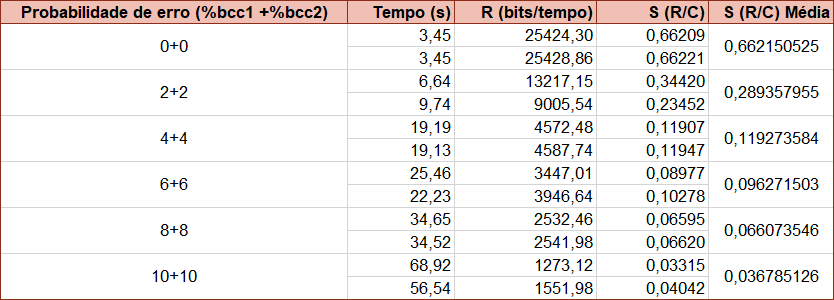
Em suma, o trabalho foi concluído com sucesso, tendo cumprido todos os objetivos, e a sua elaboração contribuiu positivamente para um aprofundamento do conhecimento, tanto teórico como prático, do tema em questão.

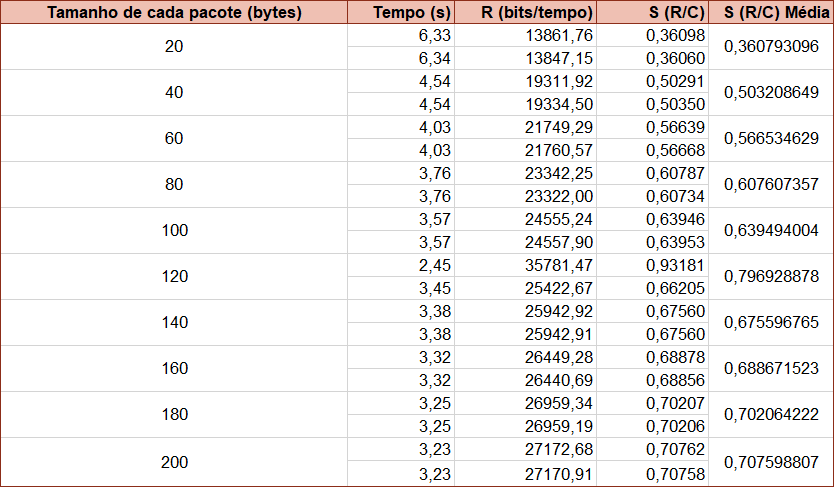
  (síntese da informação apresentada nas secções anteriores; reflexão sobre os objetivos de aprendizagem alcançados)

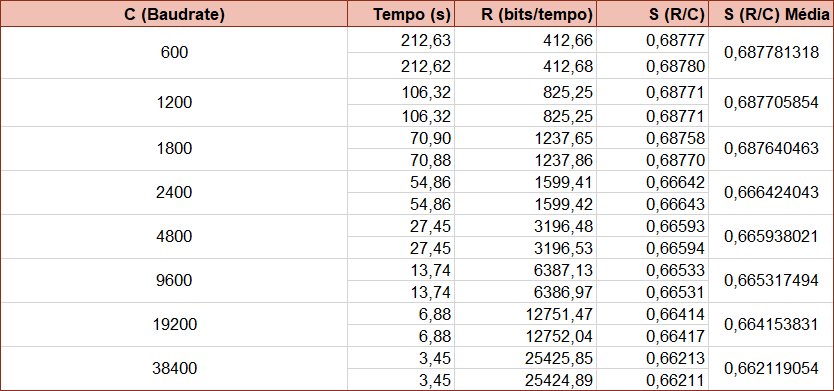
*Anexo I*

Código fonte.

*Anexo II*

**Figura 1**: Tabela de cálculos da eficiência variando o número de erros.

**Figura 2**: Tabela de cálculos da eficiência variando o tamanho da trama I.



**Figura 3**: Tabela de cálculos da eficiência variando a capacidade de ligação.