1º Trabalho Laboratorial - RCOM

Diogo Guimarães do Rosário - 201806582

Henrique Melo Ribeiro - 201806529

# Sumário

Este trabalho foi desenvolvido no âmbito da unidade curricular Redes de computadores. Este projeto consistia no desenvolvimento de uma aplicação

capaz de transferir dados de um computador para outro através de uma porta série assíncrona. A aplicação é resistente a erros na transmissão dos pacotes de dados

e desconexão da porta série.

A aplicação foi desenvolvida com sucesso, sendo possível transferir ficheiros entre dois computadores sem qualquer perda de informação.

1.Introdução

Este relatório tem o propósito de expor o modo como a nossa aplicação está organizada bem como o funcionamento desta.

O objetivo deste trabalho é implementar um protocoolo de ligação de dados especificado no guião do trabalho, de modo a permitir transferência fiável de dados entre dois dispositivos

conectados pela porta série.

Assim o relatório estará organizado da seguinte forma:

2. Arquitetura e Estrutura do código - Demonstração dos blocos funcionais e interfaces e exposição das principais estruturas de dados, funções e sua relação com a arquitetura

3. Casos de uso principais - Identificação das sequências de chamada de funções

4. Protocolo de ligação lógica - Identificação dos principais aspetos funcionais, descrição da estratégia de implementação destes aspetos com apresentação de extratos de código

5. Protocolo de aplicação - Identificação dos principais aspetos funcionais, descrição da estratégia de implementação destes aspetos com apresentação de extratos de código

6. Validação - Descrição dos testes efetuados com apresentação dos resultados

7. Eficiência do protocolo de ligação de dados - Caraterização estatística da eficiência do protocolo

8. Conclusões - Síntese da informação apresentada nas secções anteriores

2.Arquitetura e Estrutura de código

O nosso projeto foi desenvolvido com duas camadas principais (protocolo e aplicação).

2.1.Camada do protocolo

A camado do protocolo está definida no ficheiro common.h e é a camada de nível mais baixo do nosso programa. É responsável pela comunicação entre os dois computadores através da porta série.

Para além disso, faz uso dos ficheiros writenoncanonical.c, writenoncanonical.h, noncanonical.c e noncanonical.h.

Esta separação foi feita devido a certas funções não serem necessárias dos dois lados do protocolo (escrita e leitura), como por exemplo a função de leitura de dados e a sua máquina de estados.

Foi utilizada a seguinte estrutura de dados para facilitar este processo:

struct linkLayer

{

  char port[20];                 /*\*Dispositivo /dev/ttySx, x = 0, 1\**/

  int baudRate;                  /*\*Velocidade de transmissão\**/

  unsigned int sequenceNumber;   /*\*Número de sequência da trama: 0, 1\**/

  unsigned int timeout;          /*\*Valor do temporizador: 1 s\**/

  unsigned int numTransmissions; /*\*Número de tentativas em caso de falha\**/

  int currentTry;                /*\*Número da tentativa atual em caso de falha\**/

  char frame[MAX\_SIZE\*2+7];          /*\*Trama\**/

  int frameSize;

};

2.2. Camada de aplicação

A camada da aplicação está definida nos ficheiros application.c e application.h e é a camada que está imediatamente acima do protocolo.

Esta camada é responsável pela leitura do ficheiro no lado da escrita e criar os pacotes de dados que serão transmitidos à camada do protocolo.

Foi utilizada a seguinte estrutura de dados para facilitar este processo:

struct applicationLayer

{

    int fileDescriptor; /\*Descritor correspondente à porta série\*/

    int status;         /\*TRANSMITTER | RECEIVER\*/

};

3.Casos de uso principais

A aplicação necessita de diferentes parâmetros dependendo se é recetor ou transmissor.

Do lado do transmissor é necessário o número da porta série a ser usada, a string "transmitter" e o nome do ficheiro.

Exemplo : ./application 0 transmitter pinguim.gif

Do lado do recetor é necessário o número da porta série a ser usada e a string "receiver".

Exemplo : ./application 0 receiver

O recetor quando iniciado fica à espera de um transmissor para iniciar a conexão. Após estabelecer esta conexão o ficheiro começa a ser transmitido pelo transmissor em pacotes de dados.

4.Protocolo de ligação lógica

O protocolo de ligação lógica tem como objetivos:

\* Configurar a porta série

\* Estabelecer a conexão entre as duas portas série

\* Transferência dos pacotes de dados após operações de *stuffing* e *destuffing*

\* Deteção de erros nas transmissões

4.1. Configuração da porta série

openWriter() e openReader()

Estas funções recebem como argumento a porta séria com a qual será estabelecida uma ligação e são invocadas na função na função **llopen()** sendo que esta função consegue distinguir qual a função do programa (transmitter ou receiver) que a chama.

Pala além do referido anteriormente também são responsáveis por preencher a struct da camada de protocolo com os valores corretos.

Caso seja impossível alcançar a porta série desejada o programa lança uma mensagem e exit code -1.

4.2 Estabelecer a conexão entre as duas portas série

setupWriterConnection() e setupReaderConnection()

Estas funções são também invocadas pelo **llopen()** que lhes fornece o file descriptor das portas abertas pelas funções openWriter() ou openReader().

O estabelecimento da conexão começa com o programa emissor a enviar uma mensagem de controlo *set* que quando recebida e processada pelo programa recetor responde com uma mensagem *UA.*

Nestas funções também se dá *setup* ao ao sinal de alarm (SIG\_ALARM). O alarm serve como controlo nas comunicações, fazendo o emissor reenviar a última mensagem escrita caso este não receba resposta por parte do recetor. Existe uma ocasião em que os papei estão invertidos, sendo esta quando os programas se estão a desconectar da porta série.

4.3 Transferência dos pacotes de dados após operações de *stuffing* e *destuffing*

Esta transferência é garantida pelas função **sendInfo()** e **readInfo()**.

A função **sendInfo()** da camada de protocolo recebe o file descriptor da porta série, a mensagem e o seu tamanho, e retorna o número de bytes escritos. Nesta função são feitas operações de *stufing* da mensagem de forma a garantir a comunicação correta entre programas

Depois de escrever para a porta série a função **readRR()** é invocada, fazendo o emissor esperar por uma mensagem enviada pelo recetor. Se esta mensagem recebida for uma mensagem *receiver ready* o programa passa para o tratamento do próximo conjunto de dados que tem que enviar, caso seja de *reject* o programa reenvia a última mensagem e volta a ficar à espera de uma mensagem.

A função **readInfo()** responsabiliza-se por fazer a leitura da informação da porta série e de enviar a resposta correta ao programa emissor. Na função **readInfo()** são feitas operações de *destuffing* de forma a restaurar a mensagem recebida para o estado anterior ao *stuffing*  que esta sofreu.

4.4 Deteção de erros nas transmissões

A deteção de erros é feita ao mesmo tempo que a leitura da mensagem é feita, comparando a mensagem recebida com o **BCC** final (o **BCC** é igual à operação lógica ou exlusivo entre todos os bytes da informação), e verificando o **numero de série** da mensagem (varia entre 0 e 1 quando a mensagem recebida conter a informação correta).

Caso o **BCC** estiver errado é enviado um resposta *reject,* se o **número de** **série** estiver errado a informação recebida é considerada repetida e é enviada uma resposta *receiver ready* para o emissor enviar o próximo pacote de informação.