1º Trabalho Laboratorial

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Nuno Ramos - up201405498
Maria João Mira Paulo - up201403820
Pedro Duarte Costa - up201403291

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

2 de Novembro de 2016

Conteúdo

1 Sumário

Este trabalho tem por objetivo aplicar na prática os conceitos teóricos lecionados na cadeira de Redes de Computadores, mais concretamente os protocolos de transferência de dados. A realização do trabalho laboratorial "Protocolo de Ligação de Dados"consiste no envio de dados de um computador para outro através do uso porta de série.

O trabalho foi concluído com sucesso. A transferência de dados, foi concretizada sem perdas ou erros.

2 Introdução

O objetivo deste trabalho é implementar um mecanismo de transferência de dados entre dois computadores recorrendo a uma porta de série. O uso da porta de serie deve-se ao facto de ser um dos mecanismos mais básico de ligação permitindo-nos compreender a um mais baixo nível como é realizada a transferência de dados. Este objetivo mais geral necessitou da implementação de protocolos de encapsulamento e o uso mecanismos de deteção e controlo de erros, para certificar que a informação que era recebida era correta independentemente da ocorrência falhas de conexão. O relatório serve como documentação do trabalho realização, seguindo portanto a seguinte estrutura:

- **Introdução**, indicação dos objetivos do trabalho e do relatório;
- **Arquitetura**, secção que explica o funcionamento da aplicação e da interface do utilizador;
- **Estrutura do código**, aborda-se as principais estruturas de dados e funções e a sua relação com a arquitetura;
- **Casos de uso principais**;
- **Interface com o Utilizador**, descreve-se o módulo de interface com o utilizador em modo de texto;
- **Protocolo de ligação lógica**, identificação dos principais aspetos funcionais, descrição da estratégia de implementação destes aspetos com apresentação de extratos de código;
- **Protocolo de aplicação**, explicando como no ponto anterior a lógica de implementação seguida nesta fase do trabalho;
- **Validação**, descrição dos testes realizados e resultados obtidos;
- **Elementos de Valorização**, descrição e explicação de alguns elementos extras;
- **Conclusão**, considerações finais acerca do trabalho que realizámos.

3 Arquitetura

O nosso projeto está construido em camadas baseando-se no principio de independência entre camadas. Ao nível da camada de ligação de dados encontram-se implementado o protocolo de ligação de dados, com funções que estabelecem e terminam a ligação e garantem uma transmissão sincronizada e com controlo de erros, recorrendo a mecanismos como, por exemplo, *stuffing*. Ao nível da camada de aplicação, ocorre a leitura e escrita no ficheiro e a emissão e recessão de tramas. Estas duas camadas estão diretamente relacionadas, sendo que a camada de aplicação depende diretamente da camada de ligação de dados.

A nível de interface com o utilizador, no inicio do programa o utilizador pode preencher os parâmetros como deseja.

4 Estrutura do código

As principais funções da camada de ligação são as seguintes:

```
1 int llopenTransmiter(char *SerialPort);
2
3 int llopenReceiver(char *SerialPort);
4
5 int llwrite(int fd, unsigned char *buffer, int length);
6
7 int llread(int fd, unsigned char *buffer);
8
9 int llclose(int fd, enum Functionality func);
```

Listing 1: Code example

As funções llopenTransmiter() e llopenReceiver() estabelecem a ligação entre os dois computadores, garantido que esta tudo pronto para começar a transferência de dados. O emissor chama a llwrite() que envia os dados que lhes são passados como argumento e fica à espera de receber uma confirmação de sucesso, caso tal não aconteça, reenvia os dados e volta a esperar uma resposta afirmativa, isto até conseguir uma resposta de sucesso. No recetor, a função llread() fica à espera de receber uma trama, aquando da receção da mesma, conforme receba com sucesso ou não envia a resposta adequada.

Na camada de aplicação usamos a seguinte estrutura de dados para guardar informação do ficheiro:

```
typedef struct {
  unsigned int size;
  char filename[MAX_SIZE];
} FileInfo;
```

Listing 2: Code example

As principais funções usadas nesta camada são as seguintes:

```
int sendData();

int receiveData();

int sendControlPackage(int state, FileInfo file, unsigned char
    *controlPacket);

int processingDataPacket(unsigned char *packet, int length,
    FileInfo *file, int fp);

int sendDataPackage(unsigned char *dataPacket, FILE *fp, int
    sequenceNumber, int *length);
```

Listing 3: Code example

A função sendData() é responsável por enviar o pacote de controlo *Start* que marca o inicio da transferência de dados, os pacotes de dados e o pacote de controlo *End* que marca o final da transferência e proporciona o fechar da ligação. A função sendDataPackage() envia um pacote de dados. A função sendControlPackage() trata de enviar um pacote de controlo. Por fim, a função processingDataPacket() é responsável por processar os pacotes recebidos pelo receptor, e enviar uma resposta que depende do sucesso ou do insucesso da leitura.

5 Casos de uso principais

f O trabalho laboratorial realizado inclui dois diferentes casos de uso: a transmissão de um ficheiro entre dois computadores e a interface que permite ao transmissor a escolha do ficheiro a enviar, ou seja a inserção de dados do utilizador.

- Configurar Ligação e escolha do ficheiro a enviar.
- Estabelecer Ligação.

- Envio de dados, pelo transmissor.
- Receção de dados, pelo receptor e escrita no ficheiro de output.
- Fecho dos ficheiros abertos para a realização da transferência.
- Terminar Ligação.

6 Interface com o Utilizador

No que diz respeito à interface com o utilizador existem funções no inicio da aplicação que permitem que o utilizador decida o tamanho máximo do campo de Informação das tramas,o número máximo de retransmissões e o intervalo de time-out. Assim que o utilizador decide os critérios a usar na transferência do ficheiro, o programa está pronto para correr.

```
1 void askNumberOfTries();
2 void askTimeOfTimeout();
3 int askMaxFrameSize();
```

Listing 4: Code example

7 Protocolo de ligação lógica

O objectivo do protocolo de ligação de dados é fornecer um serviço de comunicação de dados fiável entre dois sistemas ligados por um meio (canal) de transmissão – neste caso, um cabo série. O protocolo de ligação de dados está implementado na Data Link Layer.

7.1 Aspectos Fundamentais

A camada de ligação de dados é responsável por:

- Tratar das novas configurações da porta de série e ao mesmo tempo,guardar as configurações anteriores para as repor facilmente.
- Estabelecer e terminar uma ligação através da porta de série.
- Envio de comandos e, também, pelo envio e receção de mensagens através da porta de série.
- Delimitação de tramas e processo de Stuffing e Destuffing dos packets recebidos da camada Application Layer - dados orgamizados em tramas (framing). A delimitação de tramas é feita por meio de uma sequência especial de oito bits (flag) e a transparência é assegurada pela técnica de byte stuffing.
- Controlo de erros e de fluxo.

7.2 Funções Implementadas na Data Link Layer

A função **llopen()** é responsável por estabelecer a ligação entre os dois computadores através de um cabo de série. O transmissor trata de enviar o SET ao recetor, ficando depois à espera para receber uma resposta deste, a trama UA. Desta forma, a ligação é estabelecida com sucesso e o programa continua em execução. Caso o transmissor não receba nenhuma reposta passado 3 segundos (tempo estabelecido pelo utilizador),este volta a enviar um SET ao recetor, com o apoio da função alarme. Em caso de novo insucesso, o processo repete-se, no máximo 3 vezes. O número máximo de tentativas feitas pelo transmissor é estabelecido pelo utilizador. Se o número de tentativas for excedido, o programa termina, retornando um erro.

A função **llwrite()** é responsável por enviar os dados ao recetor. A função recebe como parâmetro um buffer ,isto é, um pacote quer de dados quer de controlo e constrói a trama de informação correspondente.

F	A	C	BCC1	D_1	Dados	D_N	BCC2	F

Esta trama é depois enviada ao recetor e o transmissor aguarda uma resposta. Caso o transmissor não receba nenhuma resposta, a trama é reenviada, processo que pode ser repetido o número de tentativas estipulado pelo utilizador. Caso esse número seja ultrapassado a função termina com erro. A resposta recebida pelo recetor pode ser de dois tipo: **RR**, indicando que a transferência de informação correu com sucesso e que pode continuar a transferência, enviando a próxima trama com o próximo número de sequência caso não seja uma trama repetida, ou um **REJ** que indica que houve um erro ao ler a trama e que por isso esta deve ser reenviada.

A função **llread()** é responsável por ler a trama de informação enviada pelo transmissor. Após a receber é feito o *destuffing* e é processada a informação. A trama é verificada com apoio a uma maquina de estados que volta ao estado inicial sempre que recebe um byte inesperado. Caso a trama recebida seja a trama de controlo ***Start*** é aberto o ficheiro para escrita de informação. Caso tenha recebido um a trama de informação de dados pode, enviar um **REJ** em caso de erro de leitura, enviar um **RR** em caso de sucesso ou caso seja um trama de controlo ***End*** terminar a ligação.

A função **llclose()** é responsável por terminar a ligação entre os dois computadores. O transmissor envia um **DISC** ao transmissor que mal o

receba, envia um **DISC** de volta ao receptor. A ligação termina quando o transmissor recebe um **UA** enviado pelo transmissor.

8 Protocolo de aplicação

Assim que é iniciada a Application Layer, é chamada a função **appLayer()** que está encarregue de, através da função **openSerialPort()** inicializar a estrutura respetiva recebendo o descritor da porta de série. Além disso é inicializada a camada Data Link pela função **llopen()**. Por fim, dependendo da funcionalidade atribuída a cada um dos computadores, seguem-se caminhos diferentes.

```
1 enum Functionality { TRANSMITER, RECEIVER };
```

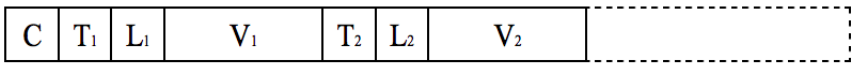
Listing 5: Code example

No caso do **Transmiter** é chamada a função **sendData()**, no caso do **Receiver()** é chamada a função **receiveData()**.

8.1 Funções Implementadas na App Layer

A função **sendData()** é responsável por enviar os pacotes de controlo e abrir, ler e enviar o ficheiro em varias tramas. Trata assim de enviar através da função **sendControlPackage()** o pacote de controlo ***Start***. A partir da função **sendDataPackage()** escreve a informação do ficheiro, pacote por pacote, e por fim chama de novo a função **sendControlPackage()** mas desta vez com o pacote de controlo ***End***.

A função **sendControlPackage()** é responsável por preencher convenientemente o pacote de controlo, acrescentando campo de controlo e informação relativa ao nome e tamanho do ficheiro.



A função **sendDataPackage()** é responsável por formar convenientemente a trama de dados. A função recebe como parâmetros um array de dados e um número de sequência, pretendendo-se atualizar o campo de controlo, número de sequência, número de octetos do campo de dados e o campo de dados do pacote.

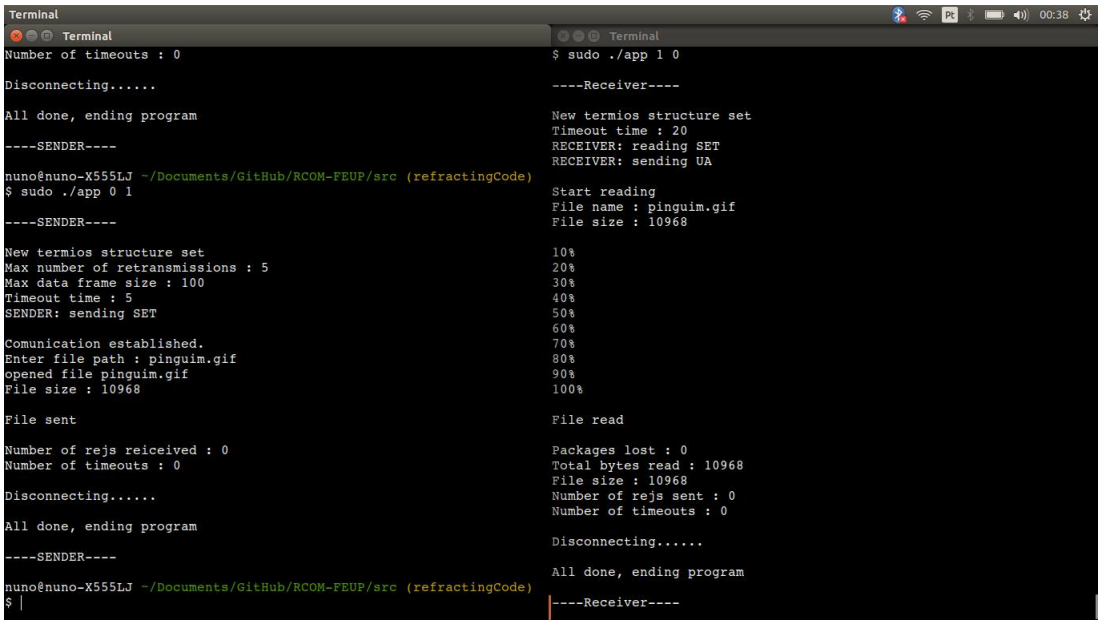
C	N	L_2	L_1	P_1		P_K

A função **receiveData()** é responsável por ler as tramas enviadas pelo transmissor. Através da função **processingDataPacket()** processa uma trama de informação. O retorno desta função pode indicar que se trata de um pacote de controlo ***Start***, que a leitura da trama ocorreu com um erro e por isso deve ser enviado um **REJ** ao transmissor, que a leitura ocorreu com sucesso e neste caso é enviado um **RR** ao transmissor ou um pacote de controlo ***End*** acabando a receção de dados, o ficheiro foi lido.

A função **processingDataPacket()** é responsável por receber um pacote de informação. Caso o campo de controlo seja do tipo ***Start*** ou ***End*** é processado um pacote de controlo, caso contrário é processado um pacote de dados. O processamento de um pacote de dados trata de escrever no ficheiro de destino a informação contida neste.

9 Validação

Com o fim de testar a aplicação, efetuou-se a transferência de vários ficheiros com diferentes extensões e tamanhos. As transferências foram concluídas com sucesso mesmo quando a ligação da porta de série era interrompida ou até quando se interferia com o envio de dados, criando ruído.



10 Elementos de valorização

- **Selecção de parâmetros pelo utilizador** - Quando o programa é inicializado, é apresentada uma interface que permite ao utilizador escoher o tamanho máximo do campo de informação das tramas, o número máximo de retransmissões e o intervalo de time-out.
- **Implementação de REJ** - Sempre que existe algum erro ao nível do BCC2, o comando REJ é enviado para que o transmissor reenvie a informação que nao chegou ao receptor corretamente.
- **Verificação da integridade dos dados pela Aplicação** - A aplicação verifica se o tamanho do ficheiro recebido é o mesmo que indicado nos pacotes de controlo. Além disso verifica quais os pacotes de dados que foram perdidos ou duplicados, através do campo de numeração do pacote.
- **Registo de ocorrências** - A aplicação ao longo do tempo, vai também registando o estado da transferência, em percentagem. Além disso a aplicação calcula o número de ocorrências de timeout assim como o número de REJ enviados / recebidos.

11 Conclusões

Por fim, concluímos que os objetivos propostos foram cumpridos. O projeto está dividido em duas camadas independentes entre si.

Na camada de ligação de dados não é feito qualquer processamento que incida sobre o cabeçalho dos pacotes a transportar em tramas de Informação – esta informação é considerada inacessível ao protocolo de ligação de dados. Além disso não existe qualquer distinção entre pacotes de controlo e de dados, nem é relevante a numeração dos pacotes de dados.

A camada de aplicação não conhece os detalhes do protocolo de ligação de dados, mas apenas a forma como acede ao serviço. O protocolo de aplicação desconhece a estrutura das tramas e o respectivo mecanismo de delineação, a existência de stuffing e destuffing, o mecanismo de protecção das tramas e eventuais retransmissões de tramas de Informação.

Desta forma, concluimos que o objectivo foi alcançado com sucesso, pois conseguimos estabelecer uma comunicação de dados fiável entre dois sistemas ligados por um cabo de série.