1º Trabalho Laboratorial: Ligação de Dados

Relatório

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Turma 1 Grupo 2:

André Cruz - 201503776
Bruno Piedade - 201505668
Edgar Carneiro - 201503748

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

5 de Novembro de 2017

Sumário

Pimeiro parágrafo sobre o contexto do trabalho:
O trabalho, realizado no âmbito da cadeira de Redes de Computadores, tinha como objetivo a implementação de um protocolo de ligação de dados, de acordo como uma especificação dada pelos docentes da cadeira. A ligação de dados era feita através da Porta Série, que conectava assim dois computadores. Era também pedido aos alunos que testassem esse protocolo com uma aplicação simples de transferência de dados.

Segundo parágrafo sobre as principais conclusões do relatório.
TODO

1 Introdução

Objetivos do trabalho e do relatorio? (COPIAR do guiao?) O trabalho tinha como objetivo a implementação de um protocolo de ligação de dados. A ligação de dados era estabelecida através da Porta Série, conectando assim dois computadores. A ligação de dados tinha três camadas ... TODO

descrição da lógica do relatório com indicações sobre o tipo de informação que poderá ser encontrada em cada uma secções seguintes

2 Arquitetura

Blocos funcionais e interfaces

Blocos Funcionais

No Trabalho é possível distinguir a existência de duas camadas bem definidas: a camada do protocolo de ligação de dados - *LinkLayer* - e a camada da aplicação - *AplicationLayer*. Os ficheiros *LinkLayer.h* e *LinkLayer.c* representam a camada de ligação de dados. Os ficheiros *AplicationLayer.h, AplicationLayer.c, Packets.h* e
texit Packets.c representam a camada da aplicação.

A camada de ligação de dados é a camada responsável pelo estabelecimento de ligação e, portanto, tem todas as funções que asseguram a consistência do protocolo, como o tratamento de erros, envio de mensagens de comunicação, entre outros. É também nesta camada que a interação com a porta série é feita, nomeadamente, a sua abertura, a escrita e leitura desta e o seu fecho.

A camada da aplicação é responsável pela envio e receção de ficheiros, segmentando o ficheiro a enviar em tramas de tamanho definivel pelo utilizador. Esta camada faz uso da inteface da camada de ligação de dados, chamando as suas funções para o envio e receção de segmentos do ficheiro a receber / enviar. A camada da aplicação é sub-dividida em duas sub-camadas, dai o uso dos ficheiros *AplicationLayer.h* e *AplicationLayer.c* para representar a camada mais abstrata, responsável pelo envio do fiicheiro e a receção do ficheiro, e que faz uso da camada menos abstrata, representada nos fiheiros *Packets.h* e *Packets.c*, que é responsável pela segmentação do ficheiro em pacotes e envio de pacotes de controlo e informação.

Interface

Na interface da linha de comandos é permitido ao utilizador correr o programa usando o mesmo binário, independentemente de ser o recetor ou o emissor. É necessário o utilizador especificar se será o emissor / recetor, qual o Serial Port a ser usado e, no caso do recetor, qual o ficheiro a transmitir. No entanto, existem parâmetros opcionais que permitem definir outras definições relcionadas com a transmissão de informação, tais como: *baudrate*, tamanho dos segmentos de informação, número de tentativas no reenvio de tramas e tempo esperado até ao reenvio de uma trama. Assim, a aplicação pode correr com valores inseridos pelo utilizador, ou com os seus valores por defeito.

O módulo da interface do utilizador interage depois com a camada de aplicação, inicializando esta e indicando o ficheiro a transmitir, no caso do emissor, ou se receberá um ficheiro, no caso do recetor.

3 Estrutura do Código

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

Application Layer

Tal como já foi referido, na secção **Arquitetura**, a implementação da camada da aplicação é feita através dos ficheiros *AplicationLayer.h, AplicationLayer.c, Packets.h* e *Packets.c*.

Os ficheiros *AplicationLayer.h* e *AplicationLayer.c* , representantes da sub-camada mais abstrata da camada da aplicação, fazem uso de uma estrutura de dados que guarda o descritor do ficheiro da porta série, o nome do ficheiro a ser transmitido, o tamanho máximo de mensagem a ser transmitido e ainda o tipo de conexão a ser usado - emissor ou recetor.
Meter figurinha da struct (TODO COMENTAR)

As funções da API desta sub-camada são:
Meter figurinha da struct (TODO COMENTAR)

As prinicpais funções desta sub-camada são:
Meter figurinha da struct (TODO COMENTAR)

Os ficheiros *Packets.h* e *Packets.c* , representantes da sub-camada menos abstrata da camada da aplicação, fazem uso de três estruturas de dados: a estrutura *Packet* que guarda um apontador para a informação, e o tamanho dessa informação; a estrutura *DataPacket]* que guarda o número sequencial do pacote a ser enviado, o seu tamanho e o apontador para essa informação; a estrutura *ControlPacket* que guarda o tipo de pacote de Controlo - inicio ou fim -, o nome do ficheiro, o tamanho do ficheiro e o número de argumentos do pacote de controlo.
Meter figurinha da struct (TODO COMENTAR)

As funções da API desta sub-camada são:
Meter figurinha da struct (TODO COMENTAR)

As principais funções desta sub-camada são:
Meter figurinha da struct (TODO COMENTAR)

Link Layer

Tal como já foi referido, na secção **Arquitetura**, a implementação da camada de ligação de dados é feita através dos ficheiros *LinkLayer.h* e *LinkLayer.c*.

A camada da ligação de dados é representada através de uma estrutura de dados onde é guardado a porta série utilizada, o *baudrate* utilizado, o número de sequência da trama esperada, tempo esperado até ao reenvio de uma trama, e o número de tentativas de reenvio de uma trama.
Meter figurinha da struct (TODO COMENTAR)

As funções da API desta camada são:
Meter figurinha da struct (TODO COMENTAR)

As principais funções desta camada são:
Meter figurinha da struct (TODO COMENTAR)

4 Casos de uso principais

(identificação; sequências de chamada de funções)
Existem dois casos de uso principais bem distintos: correr o programa como emissor ou correr o prorama como recetor. Em cada um destes casos é possível correr o programa usando o mesmo binário, apenas dependendo os argumentos usados na chamada do programa, sendo estes:
TODO meter imagem do print usage

No caso em que o programa é executado como **recetor** a sequência de chamada de funções, considerando as de maior relevência, é:

- **receiveFile**, que tem como objetivo receber o ficheiro indicado e que faz uso de funções como a **receiveControlPacket**, **receiveDataPacket**, **llopen** e **llclose**.
- **receiveControlPacket**, que tem como objetivo enviar um pacote de controlo, do tipo *START* no início da transmissão e do tipo *END* no fim da transmissão, e que faz uso de funções como **fillControlPacketArg** e **llread**.
- **receiveDataPacket**, que tem como objeitvo enviar um pacote de informação,e que faz uso de funções como **llread**.
- **fillControlPacketArg**, que tem como objetivo preencher os argumentos de um control packet, com informação recebida.
- **llread**, que tem como objetivo ler da Porta Série informação, aplicando-lhe *Byte Destuffing* e *Deframing*. Faz uso das funções **byteDestuffing, deframingInformation, sendControlFrame** e **read**.

No caso em que o programa é executado como **emissor** a sequência de chamada de funções, considerando as de mairo relevância, é:

- **sendFile**, que tem como objetivo enviar o ficheiro indicado e que faz uso de funções como a **sendControlPacket**, **sendDataPacket**, **llopen** e **llclose**.
- **sendControlPacket**, que tem como objetivo enviar um pacote de controlo, do tipo *START* no início da transmissão e do tipo *END* no fim da transmissão, e que faz uso de funções como **makeControlPacket** e **llwrite**.
- **sendDataPacket**, que tem como objeitvo enviar um pacote de informação, e que faz uso de funções como **makeDataPacket** e **llwrite**.
- **makeControlPacket**, que tem como objetivo criar uma pacote de controlo.
- **makeDataPacket**, que tem como objetivo criar um pacot de informação.
- **llwrite**, que tem como objetivo escrever para a Porta Série a informação recebida como argumento, após aplicar uma *frame* e *Byte Stuffing* à informação. Faz uso das funções **framingInformation, byteStuffing, readControlFrame** e **write**.

As funções de mais baixo nível, associadas à camada da ligação de dados, são usadas quer pelo emissor quer pelo recetor, sendo estas:

- **llopen**, que tem como objetivo abrir a ligação da Porta Série. Faz uso das funções **openSerialPort, sendControlFrame** e **readControlFrame**.
- **llclose**, que tem como objetivo terminar a ligação da Porta Série. Faz uso das funções **llcloseTransmitter, llcloseReceiver** - conforme seja Emissor ou Recetor - e **close**.
- **sendControlFrame**, que tem como objetivo enviar uma trama de controlo. Faz uso das funções **createControlFrame** e **write**.
- **readControlFrame**, que tem como objetivo receber uma trama de controlo. Faz uso da função **readFromSerialPort**.

5 Protocolo de ligação lógica

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

6 Protocolo de aplicação

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

7 Validação

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

8 Eficiência do protocolo de ligação de dados

(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).

9 Eficiência do protocolo de ligação de dados

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

10 Anexo I