Faculdade de Engenharia da Universidade do Porto

1º Trabalho Laboratorial

Protocolo de Ligação de Dados

Rede de Computadores

Licenciatura em Engenharia Informática e Computação

Turma 5 - Grupo 8

Ana Rita Baptista de Oliveira - up202004155

Diogo Alexandre da Costa Melo Moreira da Fonte - up202004175

Índice

1. Sumário

Este relatório foi realizado no âmbito da unidade curricular redes de computadores do 3º ano da Licenciatura em Engenharia Informática e de Computação LEIC). O relatório é relativo ao primeiro trabalho laboratorial, que tem como objetivo o desenvolvimento de uma aplicação para transferência de dados. Para isto necessitamos de implementar um protocolo de ligação de dados entre duas máquinas através da porta série RS-232.

Concluímos com sucesso a implementação de transferência de dados quando não ocorre nenhum erro (nas tramas e/ou na porta série) e quando a porta série é desligada, lançando 3 alarmes no máximo, até retomar a transferência dos dados em falta. Com a introdução de ruído na porta, existem problemas na propagação e recebimento das tramas de *acknowledgement* DISC.

Neste relatório vamos detalhar a nossa implementação do protocolo de ligação de dados (quer a camada de ligação, como a camada da aplicação) e abordar os conceitos teóricos que aplicamos na elaboração do trabalho.

2. Introdução

Os dois objetivos principais deste trabalho laboratorial são a implementação da camada de ligação de dados e a implementação da aplicação para aceder à camada anterior.

Quanto ao protocolo de ligação de dados, temos como finalidade prestar um serviço de transferência de dados seguro e fiável entre dois aparelhos ligados por um canal de comunicação. O nosso meio de transmissão será a porta série RS-232.

O objetivo da camada da aplicação é implementar uma forma simples de aceder à camada de ligação de dados acima referida, para podermos usufruir dos serviços disponibilizados.

Este relatório serve para documentar o trabalho laboratorial realizado e explicitar os conceitos fulcrais para a realização do mesmo. Seguimos a seguinte estrutura, proposta pelos professores responsáveis pela unidade curricular:

- **Arquitetura:** identificação de blocos funcionais e interfaces.
- **Estrutura do código:** APIs, principais estruturas de dados, principais funções e a sua relação com a arquitetura.
- **Casos de uso principais:** identificação e sequências de chamadas de funções.
- **Protocolo de ligação lógica:** identificação e descrição da estratégia de implementação dos principais aspetos funcionais.
- **Protocolo de aplicação:** identificação e descrição da estratégia de implementação dos principais aspetos funcionais.
- **Validação:** descrição dos testes efetuados com apresentação quantificada dos resultados.
- **Eficiência do protocolo de ligação de dados:** caracterização estatística da eficiência do protocolo, efetuada recorrendo a medidas sobre o código desenvolvido.
- **Conclusões:** síntese da informação apresentada nas secções anteriores e reflexão sobre os objetivos de aprendizagem que alcançamos.

3. Arquitetura

Os dois blocos funcionais presentes neste trabalho são o transmissor (*transmitter*) e o receptor (*receiver*). Os dois utilizam funções implementadas na aplicação e na camada do protocolo de ligação de dados. São bastante distintos, pois os seus objetivos são contrários, mas utilizam as mesmas funções do protocolo para conseguirem executar as suas ações. Contudo, em cada uma delas, é feita a distinção entre os dois papéis que podem ser tomados.

4. Estrutura do Código

O nosso código está distribuído em duas pastas, ***include*** e ***src***. Na pasta *include* estão todos os nossos *headers files* que necessitamos, e na *src* os ficheiros C que implementam as camadas de aplicação e de ligação de dados. A execução do programa começa no ficheiro ***main.c*** e este chama a função principal da aplicação (*applicationLayer()*) com os devidos parâmetros, para o caso de ser transmissor ou receptor. As principais funções que fazem a distinção clara entre os dois papéis são a ***transmitter_app()*** e a ***receiver_app()*** presentes no ficheiro ***application_layer.c***. Ambas fazem chamadas a funções da camada da aplicação e da camada do protocolo de ligação de dados.

Estruturas de Dados Comuns:

Na camada de aplicação:

```
typedef struct {
    int size;                   // file size
    char name[255];       // file name
    FILE* file;
} FileInfo;


typedef struct {
    int port;              // serial port
    int  status;         // tx | rx
} ApplicationLayer;
```

Na camada de ligação de dados:

```
typedef struct {
    char serialPort[50];
    int baudRate;
    int nRetransmissions;
    int timeout;
    int iFrameType;
    int packetNumber;
} LinkLayer;
```

Variáveis Globais Comuns:

- ApplicationLayer **al**;
- FileInfo **file**;
- LinkLayer **ll**;

TRANSMITTER - transmitter_app()

A transmitter_app() é a função onde são executadas todas as instruções relativas ao transmissor.

Funções principais da aplicação utilizadas:

- ***createControlPacket()*** - cria um pacote de controlo para poder enviar ao receptor através da camada de ligação de dados;
- ***createDataPacket()*** - cria um pacote de dados para poder enviar ao receptor através da camada de ligação de dados.

Nota: Todas as restantes operações, como abrir o ficheiro, determinar o tamanho do ficheiro, ler o ficheiro em partes, enviar cada uma das partes e fechar o ficheiro, são executadas na função principal *transmitter_app()*.

Funções principais do protocolo de ligação de dados utilizadas:

- ***llopen()*** - estabelece a ligação com o receptor através da porta série, envia uma trama SET e recebe uma UA;
- ***llwrite()*** - envia a trama I pretendida, após executar *byte stuffing*;
- ***llclose()*** - termina a ligação da porta série, mas antes envia uma trama DISC, recebe outra igual a esta como resposta e envia uma trama UA para o receptor.

RECEIVER - receiver_app()

A receiver_app() é a função onde são executadas todas as instruções relativas ao receptor.

Funções principais da aplicação utilizadas:

- ***checkControlPacket()*** - verifica o pacote de controlo recebido do emissor através da camada de ligação de dados e copia os dados importantes para as estruturas de dados correspondentes;
- ***checkDataPacket()*** - verifica e lê o pacote de dados recebido através da camada de ligação de dados e copia os dados para o apontador em atributo.

Nota: Todas as restantes operações, como criar o ficheiro, escrever para o ficheiro cada uma das partes recebidas e fechar o ficheiro são executadas na função principal

receiver_app().

Funções principais do protocolo de ligação de dados utilizadas:

- ***llopen()*** - estabelece a ligação com o transmissor através da porta série, recebendo uma trama SET e enviando uma UA;
- ***llread()*** - lê a trama I que o emissor envia, executa *byte destuffing* e verifica se é a trama esperada;
- ***llclose()*** - termina a ligação da porta série, recebendo primeiramente uma trama DISC, envia outra igual a esta como resposta e por fim recebe uma trama UA do emissor.

5. Casos de uso principais

Interface - Execução do programa

Existem duas formas gerais para correr o programa. A primeira é utilizando o comando ***make*** seguido do papel que quer tomar, ***run_tx*** (para o transmissor) e ***run_rx*** (para o receptor). Com esta primeira forma, é utilizado o ficheiro definido por predefinição (penguin.gif). A segunda maneira consiste na execução do seguinte comando: ***./bin/main*** <**porta série** (***/dev/ttyS10*** ou ***/dev/ttyS11*** por predefinição)**)**> <papel a tomar (***tx*** ou ***rx***)> <**nome do ficheiro**>. As predefinições podem ser alteradas no ficheiro ***Makefile*** do projeto.

Sequência de Eventos para a Transmissão de Dados

- É escolhido o ficheiro a ser enviado pelo transmissor;
- A ligação entre os dois dispositivo é configurada e estabelecida através da função ***llopen()***;
- O transmissor lê o ficheiro pretendido em partes, de tamanho estipulado, e envia cada trama com os dados, uma a uma, utilizando a função ***llwrite()***;
- O receptor recebe as tramas, uma a uma, a partir da função ***llread()*** e guarda os dados no ficheiro que criou (possui o mesmo nome do enviado pelo emissor);
- A ligação é terminada com o uso da função ***llclose()***.

6. Protocolo de Ligação Lógica

Os principais aspetos funcionais são as funções principais ***llopen()***, ***llwrite()***, ***llread()***

e ***llclose()***. Vamos detalhar cada uma delas ao longo desta secção.

llopen()

```
/**
 * Establishes a connection using "serialPort" parameter of the LinkLayer
 * Returns the file descriptor on success or -1 otherwise
 * Arguments:
 *  - ll - link layer struct that contains information such as the number of retransmissions, serial port and timeout
 *  - role - application layer role ("tx" - 0 or "rx" - 1)
*/
int llopen(LinkLayer *ll, int role);
```

A função começa por abrir a ligação da porta série, cria o *file descriptor* e configura as definições da ligação. Após isto divide-se para o caso de ser transmissor ou receptor. No caso do transmissor, configura o alarme e cria uma trama SET, a partir da função ***createSUFrame()***, que constrói a trama de supervisão/não-numerada pretendida, dependendo do argumento passado (se é um SET, UA, RR, REJ ou DISC). De seguida, envia a trama SET e espera receber um UA. No caso do receptor, recebe a trama SET com a função ***receiveSUFrame()***, cria a trama UA e envia-a. A função que recebe as tramas de supervisão/não-numeradas, recebe byte a byte e utiliza uma máquina de estados ***machine()*** presente no ficheiro ***state_machine.c*** para alterar os estados e verificar se o estado final é válido.

llwrite()

```
/**
 * Sends the data in the buf with size bufSize and receives an acknowlegment
 * Returns the number of bytes written, or -1 on error
 * Arguments:
 *  - buf - buffer with the data that should be sent
 *  - bufSize - size of buf
 *  - ll - link layer struct that contains the information of the number of retransmissions, type of I frame (I0 or I1) and timeout
 *  - port - serial port
 *  - role - application role ("tx" - 0 or "rx" - 1)
*/
int llwrite(const unsigned char *buf, int bufSize, LinkLayer *ll, int port, int role);
```

Esta função apenas é utilizada pelo transmissor para poder enviar os pacotes de controlo e de dados, encapsulados em tramas I. Começa por criar uma trama I com a função ***createInfoFrame()***. Nesta, os dados são passados como argumentos e o byte stuffing é tratado a partir de uma chamada a ***byteStuffing()***. Após isto, a trama é enviada e espera-se receber a resposta (RR/REJ) através da função ***receiveSUFrame()***. Após uma resposta RR, a execução da função termina.

llread()

```
/**
 * Receives the data packet and send an acknowlegment
 * Returns the number of bytes read, or -1 on error
 * Arguments:
 *  - packet - buffer that will contain the packet being received
 *  - ll - link layer struct that contains the information of the number of retransmissions, type of I frame (I0 or I1), timeout and packet number
 *  - port - serial port
 *  - role - application role ("tx" - 0 or "rx" - 1)
*/
int llread(unsigned char *packet, LinkLayer *ll, int port, int role);
```

Ao contrário de ***llwrite()***, esta função é utilizada apenas para o receptor poder receber os pacotes de controlo e de dados, também em tramas I. Começa por fazer um ciclo, dependendo do número de tentativas estipulado, em que chama a função ***receiveInfoFrame()***, verifica a trama que recebeu, cria uma resposta adequada (verificando o BCC2) com a função ***createSUFrame()*** falada anteriormente e envia o *acknowledgment*. A função *receiveInfoFrame()* trata de ler a trama, byte a byte, e utiliza a máquina de estados ***machine_info()*** presente no ficheiro ***state_machine.c***, para alterar os estados e verificar se o final é válido e trata de executar a função ***byteDestuffing()***.

llclose()

```
/**
 * Closes the previously established connection
 * Returns 1 on success or -1 otehrwise
 * Arguments:
 *  - ll - link layer struct that contains the information of the number of retransmissions, type of I frame (I0 or I1) and timeout
 *  - port - serial port
 *  - role - application role ("tx" - 0 or "rx" - 1)
*/
int llclose(LinkLayer ll, int port, int role);
```

Esta função recebe o papel desempenhado porque chama duas funções, ***discT()*** e ***discR()***, dependendo se é transmissor ou receptor, respectivamente. Depois da execução de uma destas funções, sem qualquer erro, fecha a ligação. Na discT() é enviado uma trama DISC, recebida outra igual e enviada uma trama UA. Já na discR() é recebida uma trama DISC, enviada outra igual e recebida uma trama UA.

7. Protocolo de Aplicação

TRANSMITTER

A execução do transmissor começa com ***transmitter_app()*** que tem como principais ações as seguintes:

- Abrir o ficheiro pretendido e determinar o seu tamanho;
- Estabelecer a ligação com uma chamada a ***llopen()***;
- Chamar ***createControlPacket()*** e enviar o 1º pacote de controlo com ***llwrite()***;
- Executar um ciclo while até ler o ficheiro todo. À medida que lê o ficheiro (em partes de tamanho máximo estipulado), chamar ***createDataPacket()*** e de seguida ***llwrite()*** para enviar os pacotes de dados;
- Enviar o 2º pacote de controlo da mesma forma que o primeiro;
- Chamar ***llclose()*** para terminar a ligação e posto isto fechar o ficheiro.

Na função ***createControlPacket()*** são colocados, no pacote de controlo, os dados principais do ficheiro, isto é, o tamanho do ficheiro e o nome do ficheiro. A cada um destes parâmetros, antecede o tamanho de cada um deles.

Em ***createDataPacket()*** são colocados, no pacote de dados, os campos de cabeçalho necessários (campo de controlo, número de sequência e o número de octetos ocupados pelos dados) e os dados em concreto (presentes no apontador em argumento).

RECEIVER

A execução do receptor começa com ***receiver_app()*** que tem como principais ações as seguintes:

- Estabelecer a ligação com uma chamada a ***llopen()***;
- Fazer uma chamada a ***llread()*** para ler o 1º pacote de controlo e utilizar ***checkControlPacket()*** para verificar e copiar os dados nele contidos;
- Abrir o ficheiro pretendido com os dados do pacote de controlo;
- Executar um ciclo while até receber o ficheiro todo. À medida que recebe os dados, com chamadas a ***llread()***, utiliza a função ***checkDataPacket()*** (para verificar o pacote e copiar os dados) e escreve para o ficheiro;
- Receber o 2º pacote de controlo da mesma forma que o primeiro;
- Chamar ***llclose()*** para terminar a ligação e posto isto, fechar o ficheiro.

Na função ***checkControlPacket()***, os parâmetros referidos na criação dos pacotes de controlo são lidos e copiados, por ordem, para o apontador passado como argumento.

Em ***checkDataPacket()*** são verificados os primeiros campos e copiados os dados para o apontador passado como argumento.

8. Validação

Testes:

1. Transmissão de vários ficheiros, de dimensões diferentes;
2. Interrupção da ligação entre as portas série;
3. Inclusão de ruído na ligação entre as portas série
4. Variação do valor da *baud rate* (velocidade de transmissão);
5. Variação do tamanho estipulado para as tramas de informação.

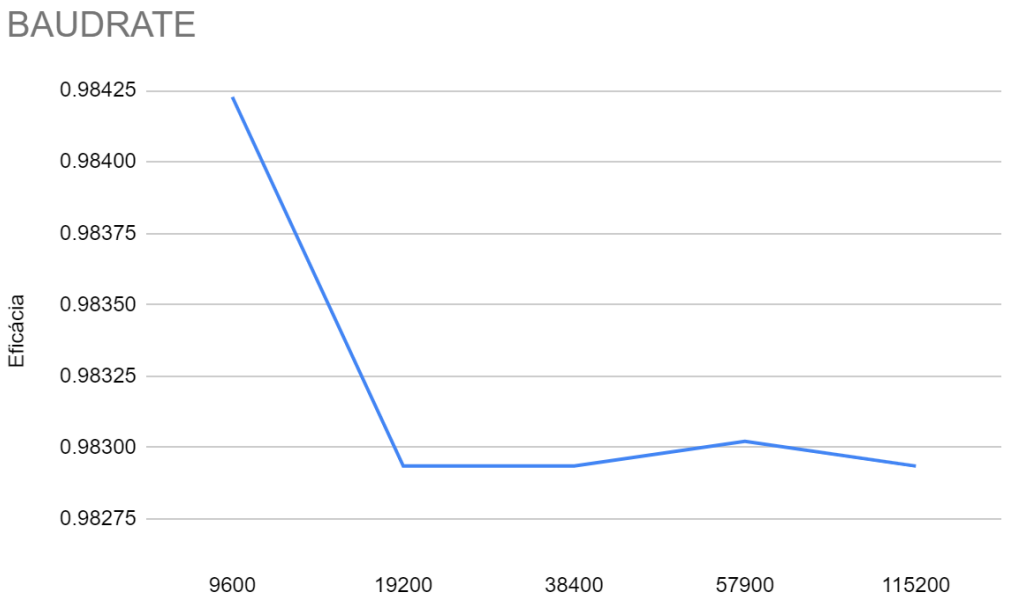
Resultados:

Os testes 1, 2 e 4 foram bem sucedidos, comprovando o bom funcionamento do protocolo de ligação de dados. Contudo, os testes 3 e 5 mostraram um comportamento inesperado por parte do protocolo.

9. Eficiência do Protocolo e Ligação

Para avaliar a eficiência do protocolo, efetuamos teste de variação da BAUDRATE, usando os valores 9600, 19200, 38400, 57900 e 115200.
O teste foi realizado com o ficheiro penguin.gif que tem aproximadamente 11KB.

Este foi o resultado que obtivemos:



Apesar de a partir de certo valor a diferença ser reduzida, é possível concluir que com o aumento da BAUDRATE, a eficiência é menor.

Seria também possível testar o protocolo através de:

- variação da FER (Frame Error Ratio), através da introdução de nas tramas, implicando o reenvio destas;
- variação do T_{prop} , usando sleep() ou usleep() para simular o aumento deste tempo;
- variação do tamanho da trama I, variando o MAX_PAYLOAD_SIZE;

10. Conclusões

Este projeto consiste no desenvolvimento de uma aplicação para transferência de dados. Para isto implementamos um protocolo de ligação de dados entre duas máquinas através da porta série RS-232. Com este trabalho laboratorial conseguimos perceber como funciona o protocolo de ligação de dados e os seguintes pontos são de salientar:

- processo de estabelecimento e terminação da ligação entre os dois dispositivos;
- criação e verificação de pacotes de controlo e de dados;
- estrutura das tramas de supervisão/não numeradas e das tramas de informação;
- processo de envio e receção de informação;
- independência e transparência entre a camada da aplicação e da camada da ligação de dados.