FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Protocolo de Ligação de Dados
1ª Trabalho Laboratorial

Ana Rita Torres, up201406093@fe.up.pt
Catarina Correia, up201405765@fe.up.pt
Ricardo Neves, up201405868@fe.up.pt

Faculdade de Engenharia da Universidade do Porto
R. Dr. Roberto Frias, 4200-464 Porto

11 de novembro de 2015

Índice

1.Sumário

No âmbito da unidade curricular de Redes de Computadores (RCOM), foi-nos proposta a realização de um projeto denominado “Protocolo de Ligação de Dados”. Este tem por base a comunicação de dados entre computadores, por meio de uma porta de série. Tendo em conta este objetivo, foram implementados diversos métodos de envio e receção que permitiram a escrita e a leitura de informação com verificação de erros.

O objetivo principal era fazer passar uma imagem de um pinguim, de um computador emissor para um computador recetor, exibindo a imagem no último. Este objetivo, foi concretizado, assim como o envio e receção dos protocolos a seguir.

2.Introdução

O principal propósito deste relatório é explicitar de forma clara a solução implementada para a resolução da proposta colocada. A linguagem de programação utilizada foi C e o sistema operativo a que recorremos o “Linux”. As portas de série, centro do trabalho, comunicavam de forma assíncrona.
O protocolo implementado combina características de protocolos de ligação de dados existentes, este garante também a transmissão de dados independentes de códigos, a chamada transparência.

A transmissão efetuada é organizada em três tipos de tramas tratadas na camada de ligação de dados:

- Informação(I)
- Supervisão(S)
- Não Numeradas(N)

Estas apresentam, de forma geral, um formato semelhante, com exceção das tramas de informação que possuem um campo destinado ao transporte de dados, mas todas são protegidas por um código detetor de erros.

Quando ocorre a transmissão do ficheiro, este é fragmentado em pacotes de dados que são transportados no campo de dados das tramas de Informação. O tratamento dos pacotes, que podem ser de Controlo ou de Dados é realizado pela camada de aplicação.
Posto isto, o relatório terá as secções que se seguem:

- Arquitetura: blocos funcionais e interfaces
- Estrutura de Código: estruturas de dados e funções principais, assim como a sua relação com a arquitetura
- Casos de Uso Principais: identificação destes e sequências de chamada de funções
- Protocolo de Ligação Lógica: identificação dos principais aspetos funcionais e descrição da estratégia de implementação destes com apresentação de extratos de código
- Protocolo de Aplicação: identificação dos principais aspetos funcionais e descrição da estratégia de implementação destes com apresentação de extratos de código
- Validação: descrição dos testes efetuados com apresentação de resultados

3.Arquitetura

A arquitetura do projeto está organizada e distribuída por duas camadas: a de Aplicação e a de Ligação de Dados. Os ficheiros que as representam são, respetivamente, *applicationLayer.c* e *applicationLayer.h*, *dataLinkLayer.c* e *dataLinkLayer.h*.

A camada de **Aplicação** visa construir os pacotes, tanto de controlo como de dados, que serão inseridos nas tramas de informação. Os pacotes de controlo contêm informação sobre o tipo de pacote (START ou END), tamanho do ficheiro, o nome do ficheiro e o tamanho que ocupa o nome do ficheiro. Os pacotes de dados contêm o conteúdo do ficheiro que, neste projeto em particular, seria o conteúdo da imagem do

pinguim, dividido em partes. Esta camada também é responsável pela leitura do conteúdo antes de criar os pacotes de dados a serem enviados.

A camada de **Ligação de Dados** concentra as funções de envio e receção de tramas. No caso das tramas de informação, o campo de dados é preenchido pelo pacote de dados gerado na camada de Aplicação, garantindo sempre a transparência através de métodos de *stuffing* e *destuffing*. Esta é também responsável pela configuração da porta série.

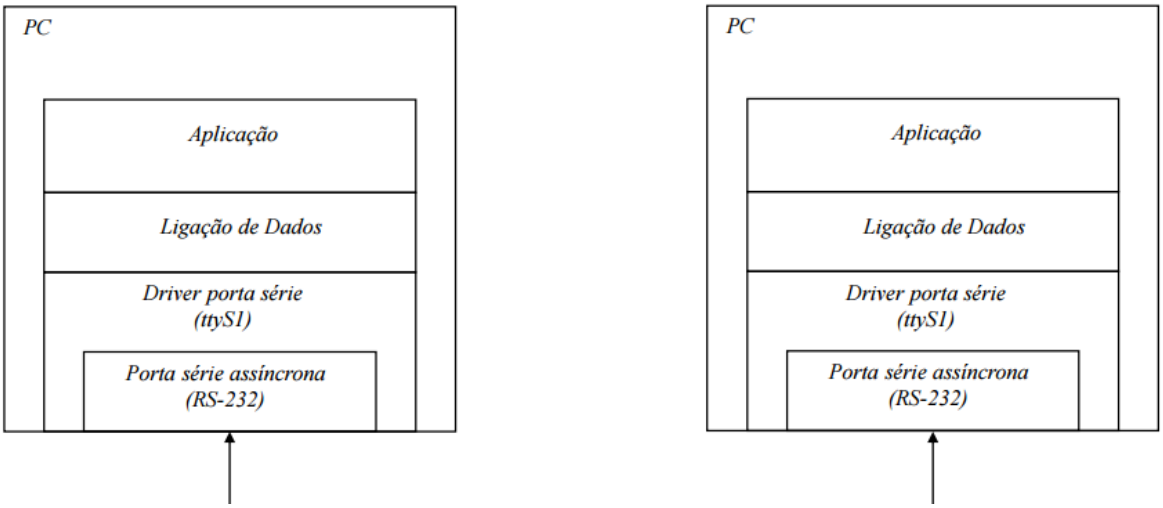

Imagem 1- Arquitetura

4.Estrutura de Código

A *struct* abaixo apresentada está no ficheiro *applicationLayer.h*. Esta é composta por dois elementos: o *fd*, que representa o descritor do ficheiro a ser usado, seja para leitura ou escrita, e o *status*, que define se o código a executar será de um emissor, no caso do valor ser zero, ou de um recetor, no caso de ser um. Foram definidas as macros na imagem três para tal efeito.

```
typedef struct{
    int fd;
    int status;
}applicationLayer;
```

Imagem 2- Struct da Camada de Aplicação

```
#define TRANSMITTER 0
#define RECEIVER 1
```

Imagem 3- Macros de status

As funções desenvolvidas para o bom funcionamento da camada de Aplicação são as que se seguem:

```
void createControlPacket(unsigned char *controlPacket, char * filename, int fileSize, unsigned char c);
int createDataPacket(unsigned char *dataPacket, unsigned char *data, int dataSize);
int readFile(char * fileName, unsigned char *content, int fileSize);
int getFileSize(char *fileName);
j
```

Imagem 4- Funções essenciais da Camada de Aplicação

A *linkLayer*, como o nome indica, está definida no ficheiro *dataLinkLayer.h*. Passemos à descrição dos seus elementos:

- *portname* - representa a porta de série selecionada
- *baudRate* - a velocidade de transmissão
- *sequenceNumber* - o número de sequência da trama, podendo ser zero ou um
- *timeout* – valor do temporizador (três segundos)
- *numTransmissions* – número de tentativas em caso de falha (3 tentativas)
- *frame* – trama

```
typedef struct{
    char * portName;
    int baudRate;
    unsigned int sequenceNumber;
    unsigned int timeout;
    unsigned int numTransmissions;
    unsigned char frame[MAX_SIZE];
}linkLayer;
```

Imagem 5 - Struct da Camada de Ligação de Dados

As funções que consideramos mais importantes nesta camada são as seguintes:

```
int ll_open(int filedes, int status, linkLayer * linkL);

int ll_close(int status);

int ll_write(unsigned char * packet, int length, linkLayer * linkL);

int ll_read(linkLayer * linkL, unsigned char *final_packet);
```

Imagem 6- Funções principais da Camada de Ligação de Dados

5. Casos de Utilização Principais

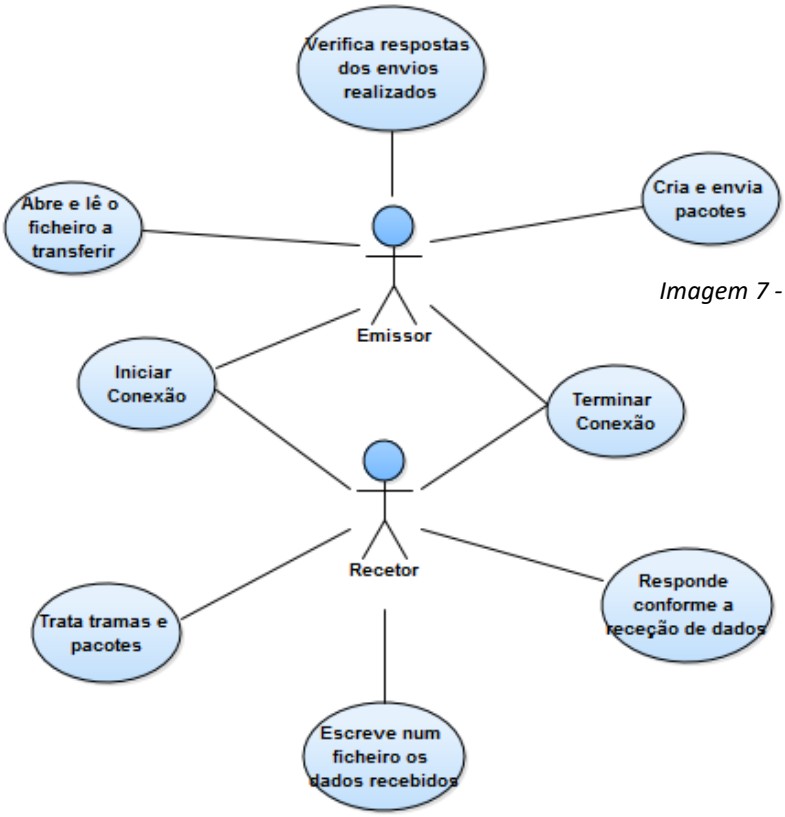

Imagem 7 - Casos de Utilização

6.Protocolo de Ligação Lógica

O protocolo tratado nesta secção do relatório está implementado no ficheiro *dataLinkLayer.c*.
Os principais aspetos funcionais desta camada passam pela configuração da porta série e transmissão de dados através da mesma. O envio e receção de tramas é também um ponto fulcral, assim como os processos que garantem a transparência, *stuffing* e *destuffing*.
As funções fundamentais para o funcionamento desta camada já foram referidas na secção de estrutura de código e, portanto, serão aqui descritas.

A função **ll_open** permite iniciar a conexão através da porta série e inicializa o alarme. Esta executa uma função auxiliar **sendSET**, no caso de se tratar do emissor, ou a **receiveSET**, no caso do recetor.
Quando o emissor está ativo, a função envia uma trama com o comando SET, esperando pela resposta do recetor, o envio de uma trama com o comando UA, que significa que o processo pode prosseguir. Se o envio da trama UA não for efetuado o programa reenvia a trama SET um número de vezes estipuladas, que ao ser atingido termina o programa em estado de erro. O lado do recetor aguarda a receção do SET e quando esta é efetuada envia o comando UA.

```
int sendSET();
```

Imagem 8 - Declaração das funções auxiliares de ll_open

A função **ll_close**, tal como a **ll_open,** tem duas funções auxiliares: a **sendDISC** e **receiveDISC**. A **sendDisc,** utilizada pelo emissor, envia o comando DISC e espera a confirmação de receção pelo lado do recetor. Caso este não receba a confirmação, reenvia o comando DISC; caso contrário, verifica a confirmação e envia um comando UA. Em contrapartida, a **receiveDISC**, utilizada pelo recetor, espera até receber o comando DISC. Após esta receção, envia outro comando DISC, que difere do anterior no campo de endereço. Quando esta recebe o comando UA, a porta série é fechada.

```
int sendDISC();
int receiveDISC();
```

Imagem 9 - Declaração das funções auxiliares de ll_close

A função **ll_write** recebe um pacote de dados, que vai sofrer *stuffing* e mais tarde, será inserido numa trama de informação, sendo utlizadas as funções auxiliares **stuffing** e **buildIFrame** para esse efeito. De seguida, através da função **writeInfo** a trama é enviada pelo descritor.

```
int writeInfo(const unsigned char *buffer, int length);

void stuffing(char *to_stuff,char *stuffing, int length);

int buildIFrame(unsigned char * stuffed_packet, unsigned char *frame, unsigned char a, unsigned char c, int length);
```

Imagem 10 - Declaração das funções auxiliares de ll_write

A função **ll_read** recebe uma trama cuja composição é verificada pela função **receiveIFrame**, no caso desta ser aceite é realizado o seu *destuffing*. Posteriormente, verifica se os BCCs recebidos são os corretos e, se sim, envia uma trama resposta RR, caso contrário, envia uma trama resposta REJ.

```
int readInfo(unsigned char *buffer, int length);

int destuffing(char *to_destuff,char*destuffing, int length);

int receiveIFrame(unsigned char * frame);
```

Imagem 11 - Declarações das funções auxiliares de ll_read

7.Protocolo de Aplicação

O protocolo indicado está definido no ficheiro *applicationLayer.c*. Aqui vai ser lido o ficheiro para extrair o seu conteúdo e todas as informações úteis sobre o mesmo. Com este protocolo pretende-se organizar a informação em pacotes de modo a que possam vir a ser incorporados em tramas de informação. Posto isto, este protocolo efetua a criação de pacotes de controlo, que contêm informação do ficheiro e pacotes de dados que contêm o conteúdo do ficheiro.

A função **createControlPacket** recebe um pacote a preencher, o nome do ficheiro, o tamanho do ficheiro e um identificador do tipo de pacote. Esta função vai inserir no pacote recebido as informações do ficheiro pela ordem correta e com o tamanho requerido (nome, comprimento do nome e tamanho do conteúdo).

```
void createControlPacket(unsigned char *controlPacket, char * fileName, int fileSize, unsigned char c){

    int fileNameSize = (int)strlen(fileName), packetSize, i;

    controlPacket[0]=c;//start ou end

    controlPacket[1]=0;//T1
    controlPacket[2]=sizeof(fileSize);//L1

    for (i=3; i<7; i++){
       controlPacket[i]=0xFF & (fileSize >> (8 * (i-3)));
    }

    controlPacket[7]=1;//T2
    controlPacket[8]=fileNameSize;//L2
    memcpy(&controlPacket[9], fileName, fileNameSize);//V2

}
```

Imagem 12 - Função que cria um pacote de controlo

A função **createDataPacket** recebe um pacote, o conteúdo e o tamanho do conteúdo. Tal como na função anterior, o pacote é um argumento de saída e esta função vai preenchê-lo com o conteúdo do ficheiro. Na segunda posição deste pacote é guardado o número de sequência.

```
int createDataPacket(unsigned char *dataPacket, unsigned char *data, int dataSize){

    int aux=numSequence;

    dataPacket[0]= DATA;
    dataPacket[1]=  numSequence;
    dataPacket[2]= (dataSize >> 8) & 0xFF;
    dataPacket[3]= dataSize & 0xFF;
    memcpy(&dataPacket[4], data, dataSize);

    if(aux==0){
        numSequence=1;
    }
    else if(aux==1){
        numSequence=0;
    }

    return 0;
}
```

Imagem 13 - Função que cria pacotes de dados

A função **readFile** recebe o nome do ficheiro, um *char* e o tamanho do ficheiro. O objetivo desta função é passar todo o conteúdo do ficheiro para o *char*, tornando-o também num argumento de saída.

A função **getFileSize** recebe como argumento o nome do ficheiro, percorre o mesmo até ao fim e retorna o tamanho do ficheiro.

A função **getNumPackets** recebe um inteiro, que representa o tamanho do ficheiro. Tal como o nome indica, esta função vai apenas efetuar a divisão do tamanho do ficheiro pelo tamanho máximo do pacote para saber quantos pacotes serão necessários para o envio do conteúdo do ficheiro.

8.Validação

Para efeitos de teste, foi enviada uma imagem “pinguim.gif” de um emissor para um recetor. Este processo de envio e receção da imagem juntamente com o protocolo completa grande parte do objetivo do trabalho e através de escrita para a consola conseguimos provar que, tanto o conteúdo como os SET e UA foram enviados e recebidos corretamente.

9.Conclusão

Aquando da elaboração deste trabalho, o grupo foi compreendendo que as camadas devem ser o mais independentes possível para corresponder com o *princípio da independência de camadas*. O grupo percebeu também a importância dos protocolos de ligações de dados, assim como a quantidade de erros que podem existir e que devem ser tratados.

A realização deste projeto serviu para consolidar a matéria lecionada nas aulas teóricas, permitindo aprofundar conhecimentos acerca da comunicação em redes de computadores, nomeadamente, o funcionamento da porta de série e os tipos de ligação não canónica e assíncrona.

10.Anexo I

O código fonte encontra-se na pasta RCOM_projeto1.zip.