

REDES DE COMPUTADORES

PROTOCOLO

DE

LIGAÇÃO DE DADOS



Henrique Silva – up202007242

Tiago Branquinho – up202005567

**SUMÁRIO**

Este relatório foi elaborado no âmbito do primeiro trabalho prático da unidade curricular “Redes de Computadores”, pertencente à Licenciatura em Engenharia Informática e Computação. Neste trabalho, foi-nos proposta a transferência de dados usando uma aplicação que recorresse a um protocolo de ligação de dados fiável, mesmo em condições de interrupção e de adição de “ruído” à transmissão. Este trabalho foi implementado em C, no Linux, usando portas série RS-232.

Conclui-se que a proposta foi cumprida com sucesso pois todos os objetivos estabelecidos foram alcançados.

**INTRODUÇÃO**

O objetivo deste trabalho divide-se em duas partes, uma relativa ao protocolo de ligação de dados e outra à aplicação em si. O objetivo do protocolo de ligação de dados é fornecer um serviço de comunicação de dados fiável entre dois sistemas ligados por um canal de transmissão, neste caso uma porta de série. O objetivo da aplicação é desenvolver um protocolo de aplicação muito simples para transferência de um ficheiro, usando um serviço fiável oferecido pelo protocolo de ligação de dados.

O objetivo do relatório é documentar o código, explicando a nossa implementação em função do guião que nos foi apresentado. O relatório tem a seguinte estrutura:

* Arquitetura – Esclarecimento sobre os blocos funcionais e interfaces.
* Estrutura do código – Explicitação das *APIs*, das principais estruturas de dados e funções, e da sua relação com a arquitetura.
* Casos de uso principais – Identificação destes e das suas sequências de chamada de funções
* Protocolo de ligação lógica – Identificação dos seus principais aspetos funcionais e descrição da estratégia de implementação dos mesmos, apresentando extratos de código.
* Protocolo de aplicação – Identificação dos seus principais aspetos funcionais e descrição da estratégia de implementação dos mesmos, apresentando extratos de código.
* Validação – Descrição dos testes efetuados, (com apresentação quantificada dos resultados)
* Eficiência do protocolo de ligação de dados – Caracterização da mesma, efetuada recorrendo a estatísticas medidas sobre o código desenvolvido. Comparação desta com a de um *protocolo Stop&Wait*.
* Conclusões – Sintese da informação apresentada e reflexão dos objetivos de aprendizagem alcançados.

**ARQUITETURA**

O trabalho segue uma arquitetura em camadas, já que é possível identificar duas camadas independentes: a da aplicação, e a da ligação de dados.

A camada da ligação de dados é a responsável pelo estabelecimento da ligação, garantindo a integridade do protocolo. Assim, é a responsável pela comunicação direta com a porta de série, gerindo a abertura, fecho, leitura e escrita na mesma. É nesta camada que ocorre a verificação de erros, garantindo-se assim a fiabilidade da transferência. A camada da ligação de dados oferece serviços à camada da aplicação, e, deste modo, é a de mais baixo nível deste projeto.

A camada da aplicação é a responsável pelo envio e receção de ficheiros, estando dividida em dois blocos, o emissor e o recetor. Esta camada acede aos serviços fornecidos pela camada de ligação de dados, invocando funções da mesma. Deste modo, é considerada a de mais alto nível deste projeto.

**ESTRUTURA DO CÓDIGO**

A camada da ligação de dados é representada por uma estrutura de ligação de dados. Esta estrutura contém o dispositivo associado à porta de série, a velocidade de transmissão, o número de tentativas em caso de falha, o valor do temporizador e a entidade que está a usufruir dos seus serviços, recetor ou emissor, que é representada por uma enumeração.

typedef enum

{

LlTx,

LlRx,

} LinkLayerRole;

typedef struct

{

char serialPort[50];

LinkLayerRole role;

int baudRate;

int nRetransmissions;

int timeout;

} LinkLayer;

As principais funções da camada de ligação de dados são llopen(), que abre uma conexão entre o recetor e o emissor, llread(), que permite a leitura de um pacote, llwrite(), que permite o envio de um pacote, llclose(), que fecha a conexão, permitindo a impressão de estatísticas referentes à mesma e, por fim, int set\_up\_port() e int close\_port(), que permitem a abertura e o fecho da porta de série, respetivamente. As duas últimas encontram-se no ficheiro *port\_operations.c*, enquanto que as restantes se encontram no ficheiro *link\_layer.c*

int llopen(LinkLayer connectionParameters);

int llwrite(const unsigned char \*buf, int bufSize);

int llread(unsigned char \*packet);

int llclose(int showStatistics);

int set\_up\_port(char \*serialPortName);

int close\_port(int fd);

A camada da aplicação é representada por estrutura de aplicação. Esta estrutura contém o descritor correspondente à porta de série e o estado, recetor ou emissor.

struct applicationLayer {

int fileDescriptor;

int status;

};

As principais funções da camada da aplicação são receiveFile(), sendFile(), buildDataPacket(), parseDataPacket(), buildControlPacket(), parseControlPacket(), e applicationLayer(), a função principal. Esta última, para além de preencher a estrutura LinkLayer, vai invocar, de acordo com o *status*, sendFile(), que permite enviar um ficheiro através da porta de série, ou receiveFile(), que permite a receção de um ficheiro através da porta de série. As funções de *parse* são invocadas pelo recetor, enquanto que as de *build* são invocadas pelo emissor.

int buildDataPacket(unsigned char \*packet, int sequenceNumber, unsigned char \*data, int dataLength);

int parseDataPacket(unsigned char \*packet, unsigned char \*data, int \*sequenceNumber);

int buildControlPacket(unsigned char \*packet, unsigned char control, int fileSize, const char \*fileName);

int parseControlPacket(unsigned char \*packet, int \*fileSize, unsigned char \*fileName);

int sendFile(const char \*filename, char \*serialPort);

int receiveFile(char \*filename, char \*serialPort);

void applicationLayer(const char \*serialPort, const char \*role, int baudRate,

int nTries, int timeout, const char \*filename);

**CASOS DE USO PRINCIPAIS**

O utilizador pode escolher as configurações, o nome do ficheiro a ser enviado e o nome com que se deverá criar no recetor através da consola. Em alternativa pode recorrer ao *Makefile* existente, iniciando o sistema na configuração padrão.

No caso do emissor, o sistema irá invocar a função sendFile(), que, primeiramente, tenta abrir o ficheiro a ser lido. Em seguida, é invocada a função llopen(), de forma a estabelecer uma conexão entre o emissor e o recetor, através do envio de uma trama SET e receção de uma trama UA. Em seguida, é invocada a função buildControlPacket(), que cria um *control packet*, que contém a indicação de inicio de transmissão. Esse *packet* é posteriormente enviado para o recetor, através a função llwrite(). Após este início, é implementado um ciclo que vai lendo porções do ficheiro, criando *data packets* que as contêm, e enviando esses *packets*, recorrendo à função llwrite(). Seguidamente, envia um *control packet* criado pela função buildControlPacket() que sinaliza o final da transmissão. Por fim é invocada a função llclose(), de forma a fechar a conexão com o recetor.

No caso do recetor, o sistema irá invocar a função receiveFile(), que invoca a função llopen(), desta vez recebendo a trama SET e enviando a trama UA. Em seguida, é invocada a função llread() de forma a receber o *control packet* enviado pelo emissor, que posteriormente é processado, através da função parseControlPacket(). Depois disso, o ficheiro a ser criado é aberto, e é implementado um ciclo que vai recebendo e processando *data packets* através da função llread() e parseDataPacket(), respetivamente e, por fim, adicionando a informação presente nesses *packets* ao novo ficheiro criado. Seguidamente, é invocada a função llread() de forma a receber o *control packet* enviado pelo emissor, que posteriormente é processado através da função parseControlPacket(). Por fim, é invocada a função llclose(), de forma a fechar a conexão com o emissor.

**PROTOCOLO DE LIGAÇÃO LÓGICA**

a