

Redes de Computadores

1º Trabalho Laboratorial

11 de novembro de 2018

Beatriz Garrido up201504710

Diogo Santos up201505606

João Costa up201404935

*Índice*

**Sumário2**

**Introdução2**

**Arquitetura3**

**Estrutura do código3**

Camada de Ligação3

Camada de Aplicação 3

**Casos de uso principais**4

Como emissor4

Como recetor 4

**Protocolo de ligação lógica**4

LLOPEN4

LLWRITE

LLREAD

LLCLOSE

**Protocolo de aplicação**

Emissor

Recetor

**Validação**

**Elementos de valorização**

**Conclusões**

**Anexo I - Código fonte**

appLayer.c

dataLink.c

dataLink.h

**Anexo II**

Estatísticas realizadas

*Sumário*

No âmbito da unidade curricular de Redes de Computadores, este relatório tem como objetivo a implementação do primeiro trabalho laboratorial, referente à transferência de dados. Este trabalho consiste no desenvolvimento de uma aplicação capaz de enviar ficheiros de um computador para outro através de uma porta série.

Dito isto, o trabalho foi realizado com sucesso, visto que os objetivos propostos foram concretizados e a aplicação foi desenvolvida na perfeição, sem qualquer perda de dados.

*Introdução*

O objetivo deste trabalho é implementar um protocolo de ligação de dados, de acordo com as especificações descritas no guião fornecido. Bem como testar o protocolo com uma aplicação simples de transferência de ficheiros, igualmente especificada.

Relativamente ao relatório, a sua função é explicar toda a lógica presente no trabalho, seguindo a seguinte estrutura:

* **Arquitetura:** exposição dos blocos funcionais e interfaces;
* **Estrutura do código:** identificação das APIs, principais estruturas de dados, principais funções e sua relação com a arquitetura;
* **Casos de uso principais:** identificação dos principais casos de uso 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 quantificada dos resultados;
* **Elementos de valorização:** identificação dos pontos adicionais implementados e descrição da estratégia de implementação;
* **Conclusão:** síntese da informação apresentada nas secções anteriores e reflexão sobre os objetivos de aprendizagem alcançados;

*Arquitetura*

Este trabalho está dividido em duas principais camadas lógicas, com a finalidade de uma melhor organização do protocolo e aperfeiçoamento da aplicação.

*Estrutura do código*

**Camada de Ligação**

A camada de ligação (ou *Data Link Layer*) proporciona os meios necessários para a transferência de dados entre camadas. Pode também fornecer códigos de deteção e correção de erros, cajo estes ocorram na camada física.

Assim, esta categoria contém as funções necessárias para o estabelecimento de ligação (LLOPEN), escrita e leitura na porta série (LLWRITE e LLREAD respetivamente), fecho da ligação (LLCLOSE) e correção e deteção de erros, como já referido anteriormente.

**Camada de Aplicação**

A camada de aplicação (ou *Application Layer*) é a que se encontra diretamente acima da camada acima mencionada, responsável pela comunicação entre o utilizador e a interface. É a partir desta que se inicia o processo de transferência de dados, sendo a que contém a função *main().*

*Casos de uso principais*

A aplicação desenvolvida pode ser corrida de duas maneiras diferentes:

**Como emissor**

A transmissão de um ficheiro para outro computador necessita de 2 parâmetros. Depois da referente a porta série (normalmente /dev/ttyS0) é necessário introduzir o tipo de comunicação – “*sender*” – e o nome do ficheiro que se deseja enviar.

De notar que, se após um determinado número de *timeouts*, nenhuma ligação for estabelecida por parte do recetor, é invocada a função de fecho de ligação.

**Como recetor**

A receção do ficheiro enviado apenas necessita do parâmetro referente ao tipo de comunicação – “*receiver*” – depois da introdução da porta série.

Ao contrário do emissor, esta comunicação não tem tempo limitado, visto que fica constantemente à espera que o emissor inicie a ligação.

*Protocolo de ligação lógica*

O protocolo de ligação lógica é implementado na camada de ligação (“*data link Layer”),* da qual a camada de aplicação (“*application Layer*”) depende.

Nesta camada são implementadas as seguintes funcionalidades:

* Estabelecer e terminar uma ligação através da porta série, bem como transmitir e receber dados/mensagens na mesma.
* Criar e enviar comandos (tramas I, SET e DISC) e respostas (UA, RR e REJ).
* Receber mensagens através da porta série.
* Fazer stuff e destuff de pacotes da camada de aplicação.
* Verificar se a informação recebida é igual à envida através do cabeçalho em todas as tramas e nas tramas I além do cabeçalho usa-se uma variável de controlo, BCC2.

**LLOPEN**

**int** LLOPEN(**int** fd, **int** com\_type);

A função **LLOPEN** é a função responsável por estabelecer a ligação através da porta série.

Quando esta função é chamada pelo emissor, este envia a trama SET e aguarda resposta do recetor, ou seja, a mensagem UA. Caso o recetor não receba uma resposta ao fim de 3 *timeouts,* cuja duração, até ser atingido, é definida através de um alarme,a ligação é fechada.

Quando esta função é chamada pelo recetor, a função fica à espera de receber a mensagem SET, sendo que, quando a recebe, envia como resposta a mensagem UA e a ligação é estabelecida. No caso do recetor não é implementado qualquer tipo de alarme, ficando este indefinidamente à espera de receber a mensagem SET.

**LLWRITE**

**int**LLWRITE(**int**fd, **char** \*buffer,**int**lenght);

A função **LLWRITE** é a função responsável por tentar escrever na porta série, os dados através da função *write* e fica a aguardar uma resposta. Caso não receba uma resposta ao fim de 3 *timeouts*, a ligação é fechada. Quando a resposta for recebida e, caso esta contenha a variável de controlo RR, a mensagem foi transimitida corretamente.

Caso, a resposta contenha a variável de controlo REJ, a mensagem não foi transimitida corretamente, o que implica a mensagem ser retransmitida e o valor de timeouts permitido restaurado para o valor inicial.

**LLREAD**

**int**LLREAD(**int**fd, **char** \*buffer);

A função **LLREAD** é a função responsável por tentar receber através da porta série os dados enviados.

Caso esta receba uma mensagem inválida, envia a varíavel de controlo REJ através da porta série. Caso a mensagem recebida for válida, a informação é guardada e é realizado o envio da varíavel de controlo RR através da porta série.

Caso receba a mensagem DISC, a ligação deve ser encerrada.

**LLCLOSE**

**int**LLCLOSE(**int**fd,**int**com\_type);

A função **LLCLOSE** é a função responsável por terminar a ligação através da porta série.

Caso seja o emissor a chamar esta função, a mensagem DISC é enviada pela porta série e este fica a aguardar pelo receção do trama DISC. Quando o recebe envia a mensagem UA.

Caso seja o recetor a chamar a função, aguarda pela receção do trama DISC e quando o receber reenvia-o e fica a aguardar pela receção da mensagem UA.

Quando este eventos ocorrem a ligação através da porta série é terminada.

**STUFFING**

**char\* stuffing(char\*** payload**, int\*** length**);**

A função **STUFFING** é a função responsável pelo mecanismo de byte stuffing da transparência de dados.

Caso os dados enviados contenham os carateres 0x7E e/ou 0x7D estes são substituídos por 0x7D 0X5E caso sejam carateres 0x7E e por 0X7D 0X5D caso sejam carateres 0X7D.

Além disso, o cálculo de BCC2 é feito antes da operação de stuffing.

**DESTUFFING**

**char\* destuffing(char\*** msg**, int\*** length**);**

A função **DESTUFFING** é a função responsável pelo mecanismo de byte destuffing da transparência de dados.

Caso os dados recebidos contenham os carateres 0x7D 0X5E e/ou 0x7D 0X5E de forma consecutiva, estes são substituídos por 0x7E caso sejam carateres 0x7D 0X5E e por 0X7D caso sejam carateres 0X7D 0X5D.

Além disso, o cálculo de BCC2 é feito depois da operação de destuffing.

**VERIFY\_BCC2**

**char\* verify\_bcc2(char\*** control\_message**, int\* length);**

A função **VERIFY\_BCC2** é a função responsável pela verificação de se a variável de controlo BCC2 recebida é igual à enviada, que permite saber se os dados foram enviados sem erros.

**CONTROL\_FRAME**

**char\* control\_frame(char\*** filename**, FILE \***file**, int** start**, int\*** frame\_size**);**

A função  **CONTROL\_FRAME** é a função responsável pela criação do pacote de controlo START/END.

**GET\_INFO**

**char\* get\_info(char\*** control**, int\*** file\_size**);**

A função  **GET\_ INFO** é a função responsável por extrair o nome do ficheiro e tamanhos do ficheiro a ser transferido.

**HEADER**

**char\* header(char\*** buffer**, int\*** length**, short** sequence\_number**);**

A função  **HEADER** é a função responsável por adicionar o campo de controlo, o número de sequência, e o tamanho da payload aos tramas que vão ser enviados pelo emissor.

**REMOVE\_HEADER**

**char\* remove\_header(char\*** buffer**, int\*** length**);**

A função  **REMOVE\_HEADER** é a função responsável por remover o campo de controlo, o número de sequência, e o tamanho da payload aos tramas que vão ser recebidos pelo recetor.

*Protocolo de aplicação*

O protocolo de aplicação é implementado na camada de aplicação (“*Application Layer”)* que é a camada de mais alto nível implementada neste projeto e é responsável pelas seguintes funcionalidades:

* Envio e receção de pacotes de controlo;
* Envio e receção de pacotes de dados;
* Envio e receção do ficheiro definido.

**Variáveis do Nível de Aplicação**

* **com\_type:** Indica se a aplicação está a correr no modo de *sender* ou *receiver*.
* **Chunk\_Syze:** Indica o tamanho dos fragmentos de informação que vão ser enviados através da porta série de cada vez.

**Emissor**

O protocolo de aplicação implementado para o modo de emissor, consiste em:

* Abertura do ficheiro em modo de leitura de modo a ser possível usar a informação deste.
* Estabelecimento de comunicação através da função LLOPEN.
* Envio da trama de controlo inicial obtida através da função CONTROL\_FRAME.
* Envio de fragmentos do ficheiro com tamanho definido por Chunk\_syze, com o devido campo de controlo, número de sequência e tamanho obtidos na função HEADER e com transparência assegurada pela função STUFFING.
* Criação de uma barra de progresso que indica a percentagem de informação que está a ser enviada em relação à informação total.
* Envio da trama de controlo final obtida através da função CONTROL\_FRAME.
* Impressão no terminal do tempo de envio e do débito binário.

**Recetor**

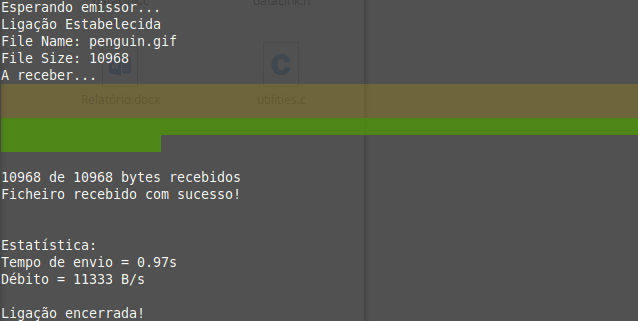
O protocolo de aplicação implementado para o modo de recetor, consiste em:

* Estabelecimento de comunicação através da função LLOPEN.
* Leitura da informação do ficheiro através da função LLREAD.
* Verificação se a informação recebida é a mesma do que a enviada.
* Caso seja válida, envia RR e começa a escrita num ficheiro com o mesmo nome do ficheiro enviado.
* Caso não seja válida, envia RR e não realiza qualquer escrita no ficheiro.
* Fazer *destuffing* e verifica o BCC2.
* Remover o *header* com o campo de controlo, número de sequência e tamanho da informação recebida.
* Criação de uma barra de progresso que indica a percentagem de informação que está a ser recebida em relação à informação total.
* Impressão no terminal do tempo de envio e do débito binário.

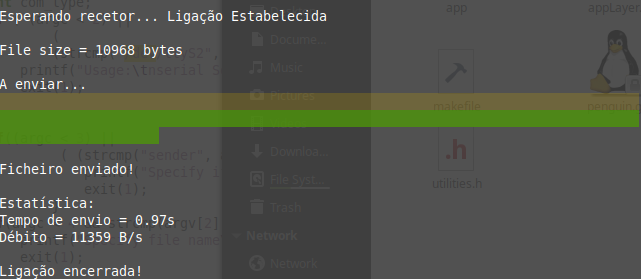
*Validação*

Nesta secção aborda-se os testes realizados com a aplicação relativamente à transferência de ficheiros e comentários acerca disso.

Relativamente à transferência do gif penguin.gif exigida no guião esta é realizada pela nossa aplicação, sendo de seguida apresentados prints da realização dessa parte.

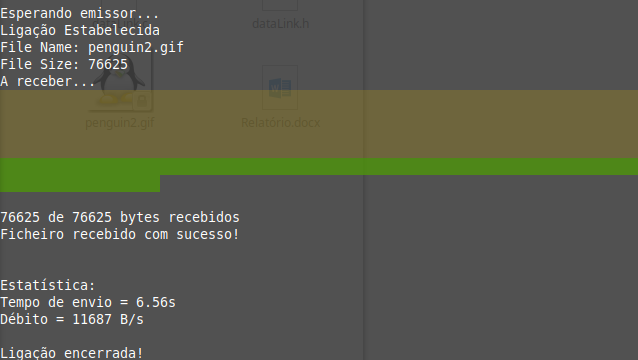


*Fig. 1 - Modo Emissor*

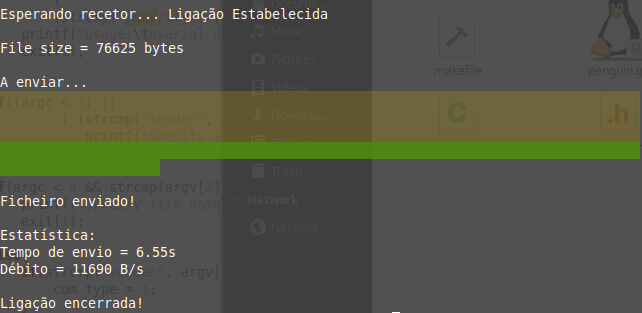
**

*Fig. 2 - Modo Recetor*

Além disso, testamos com um ficheiro gif de maior tamanho sendo o resultado o mesmo, como comprovado em seguida.



*Fig. 3 - Modo Emissor com um ficheiro maior*



*Fig. 4 - Modo Recetor com um ficheiro maior*

Por fim, realizamos testes em que cortamos a ligação da porta série de modo a testar os *timeouts* e se a ligação continuava após o restabelecimento da conexão da porta série e tal foi verificado com sucesso. Também testamos com a ligação à massa de pinos do lado do emissor de modo a introduzir erros e o teste também foi passado com sucesso.

Apesar de não possuirmos imagens estes testes foram comprovados pelo docente na altura da apresentação do projeto.

*Elementos de valorização*

**Seleção de ficheiro a enviar**

O utilizador pode selecionar qualquer ficheiro que pretenda enviar bastando para isso ter o ficheiro na mesma pasta do projeto e indicar o seu nome no terminal.

**Barra de progresso de envio/receção**

Apresentação de uma barra de progresso no envio/receção do ficheiro.

**Processamento das Tramas**

É feito o processamento das tramas, sendo que:

* Tramas são processadas através das funções CONTROL\_FRAME e HEADER;
* Tramas com erros são verificadas através das variáveis de controlo REJ,RR, BCC e BCC2 e não são guardadas;

**Verificação da Integridade dos Dados (!!!!!!ver isto)**

A aplicação verifica se o tamanho do ficheiro recebido é igual ao tamanho do ficheiro enviado.

*Conclusões*

Durante as últimas semanas quer através das aulas laboratoriais, quer através de tempo não letivo, o grupo desenvolveu esta aplicação que permite a transferência de dados entre dois computadores através de uma ligação física à porta série.

O trabalho foi bem entendido pelos elementos do grupo quer através da ajuda do docente da aula, que nos esclareceu diversos pontos acerca do mesmo, quer através do guião que nos serviu de guia.

Acerca da implementação do trabalho, os pontos pedidos no guião foram implementados, desde a divisão em camadas, o tratamento dos dados enviados através de tramas de controlo e o mecanismo de transparência de *byte stuffing.*

Em nota de conclusão, achamos que este trabalho contribuiu para a consolidação dos nossos conhecimentos acerca dos temas lecionados nas aulas teóricas e laboratoriais, permitindo-nos estar mais preparados para desafios futuros relacionados com este tema.

*Anexo I - Código fonte*