Redes de Computadores

1º Trabalho Laboratorial

Mestrado Integrado em Engenharia Informática e Computação

(INSERIR DATA)

Diogo Samuel Fernandes [up2018XXXXX@fe.up.pt](mailto:up2018XXXXX@fe.up.pt)

Paulo Jorge Salgado Marinho Ribeiro up201806505@fe.up.pt

**Indice**

**Sumário**

**Introdução**

**Arquitetura**

**Estrutura do código**

**Sumario**

Este relatório foi elaborado no âmbito da unidade curricular “Redes de Computadores” (RCOM), no sentido de complementar o primeiro trabalho laboratorial, que consiste em implementar um protocolo de transferência de ficheiros de um computador para o outro através de uma porta série.

**Introdução**

Tendo como principal objetivo implementar um protocolo de transferência de dados recorrendo a uma porta série, este trabalho deve resultar num programa capaz de resistir a fenómenos como a interrupção da porta série ou a receção de informação corrompida, provocada pela indução de “ruído” na porta série. Este relatório procura explicar toda a teoria envolvida neste primeiro trabalho, de forma bem estrutura, nos seguintes tópicos:

- *Arquitetura*

Descrição dos blocos funcionais e interfaces

- *Estrutura do código*

Explicação das APIs, enumeração das principais estruturas de dados utilizadas, das funções de maior importância e relação com a arquitetura

- *Casos de Uso Principais*

Identificação dos casos de uso mais importantes, e demonstração sequencial das chamadas às funções.

- *Protocolo de ligação lógica*

Identificação dos principais aspetos funcionais da camada de Ligação de Dados, descrição da estratégia de implementação destes aspetos, com o apoio de extratos de código.

- *Protocolo de aplicação*

Identificação dos principais aspetos funcionais da camada da Aplicação, descrição da estratégia de implementação destes aspetos, com o apoio de extratos de código.

- *Validação*

Descrição dos testes efetuados com apresentação quantificada dos resultados.

- *Eficiência do protocolo de ligação de dados*

Caracterização estatística da eficiência do protocolo, feita com recurso a medidas sobre o código desenvolvido.

- *Conclusões*

Síntese da informação apresentada nas secções anteriores, e reflexão sobre os objetivos de aprendizagem alcançados.

**Arquitetura**

O programa desenvolvido desdobra-se em duas camadas bem definidas: a camada de Ligação de Dados (*Link Layer*) e a camada da Aplicação (*Application Layer)*.

A primeira, como é responsável pelo estabelecimento da ligação, torna o protocolo sólido, garantindo a sua consistência. Por este motivo, é considerada a camada de mais baixo nível do programa, sendo que trata da abertura da porta série, da transmissão de informação (escrita e leitura), e do seu posterior fecho. É também da sua responsabilidade testar se a informação foi escrita/recebida corretamente, através de *byte stuffing*, e de testes de erros como os BCC, que serão aqui detalhados mais tarde.

Por outro lado, a camada da Aplicação é apenas responsável pelo envio e receção da informação dos ficheiros, pelo que é de um nível superior à camada de ligação de dados. Assim, esta camada chama as funções da camada da ligação de dados, para envio/receção da informação de dados, mantendo-se, no entanto, completamente independente desta, uma vez que desconhece os seus métodos de atuar.

**Estrutura do código**

Esta independência entre camadas está também explícita na sua implementação, uma vez que o código relativo à camada de ligação de dados se encontra desenvolvido nos ficheiros “llfunctions.h” e “llfunctions.c”, enquanto que o código relativo à camada da Aplicação encontra-se nos ficheiros “writenoncanonical.c” (parte do Emissor) e “noncanonical.c” (parte do Recetor). Os ficheiros “messages.h” e “messages.c” contêm funções auxiliares usadas pelas principais funções, responsáveis pelo envio/receção de tramas e pelo cálculo do BCC2. Já os ficheiros “state\_machines.h” e “state\_machines.c” contêm o código que simula as máquinas de estados necessárias para a receção.

No ficheiro “const\_defines.h” estão definidas todas as estruturas de dados utilizadas, e definidas as principais macros usadas, que enumeramos de seguida.

Estruturas de Dados:

Recorremos à struct applicationLayer para armazenar a informação relativa à camada da aplicação, nomeadamente o nome da porta série a utilizar, o seu descritor de ficheiro, e um inteiro que indica se o programa está a fazer o papel de Emissor ou Recetor.

struct applicationLayer {

char port[20]; // Dispositivo /dev/ttySx, x = 0, 1

int fileDescriptor; // Descritor correspondente à porta série

int status; // TRANSMITTER | RECEIVER

};

Recorremos também a uma enum, current\_state, que é usada pelas máquinas de estados para identificar o seu estado atual, na receção das várias tramas.

enum current\_state {start, flag\_rcv, a\_rcv, c\_rcv, bcc\_ok, data\_rcv, bcc2\_ok, stop, finished};

Funções mais importantes:

int llopen(struct applicationLayer application);

int llwrite(int fd, unsigned char buffer, int length);

int llread(int fd, unsigned char\* buffer);

int llclose(struct applicationLayer \*application);

Macros mais importantes:

Para além das macros referentes aos bytes especiais que compõem as diversas tramas, destacamos as seguintes:

- BAUDRATE: usado na struct termios, durante o llopen(), que indica a capacidade da ligação.

- RECEIVER a 0, e TRANSMITTER a 1: para efeitos de distinção do papel da aplicação.

- TIMEOUT: que indica o tempo, em segundos, que o Emissor deve esperar resposta do Recetor, antes de reenviar a trama.

- MAX\_SIZE: indica o tamanho máximo, em bytes, de informação do ficheiro que cabe num pacote da aplicação.

Para melhor compreensão das macros, e da estrutura das tramas, detalhamos esse ficheiro, com explicações mais profundas.

**Casos de uso principais**

O Makefile por nós criado efetua a compilação do programa, resultando em dois executáveis diferentes, um para o Emissor (writenoncanonical) e outro para o Recetor (noncanonical).

*Emissor:*

O executável relativo ao Emissor exige 2 argumentos: o nome da porta série (por exemplo /dev/ttyS1), e o nome do ficheiro que vai ser transmitido (exemplo: pinguim.gif)

A sequência de chamadas efetuada por este executável é a seguinte:

- llopen: Configura a ligação entre os dois computadores, abrindo a porta série em modo de escrita e leitura. Esta configuração decorre de uma troca de tramas, a trama SET enviada pelo Emissor e a trama UA enviada pelo Recetor. Apesar de a função ser comum aos dois programas, há nela uma distinção das ações conforme o parâmetro status da struct referida no tópico anterior, recebida como parâmetro.

- Leitura do ficheiro e armazenamento da sua informação num array de unsigned chars.

- Criação do pacote de controlo Start, seguido do seu envio, já recorrendo à função llwrite().

- Criação dos pacotes de Informação, que resulta de uma divisão do array referido no segundo passo, e o seu respetivo envio, recorrendo também a llwrite().

- Criação do pacote de controlo End, seguido do seu envio, recorrendo à função llwrite().

- llclose(): Encerramento da ligação entre os dois computadores, através de uma troca de tramas. Neste caso, o Emissor receberá uma trama DISC, enviando como resposta outra trama DISC, e para terminar receberá uma trama UA.

*Recetor:*

O executável relativo ao Recetor exige também 2 argumentos: o nome da porta série (por exemplo /dev/ttyS0), e o nome do ficheiro que vai ser transmitido (exemplo: novo\_pinguim.gif)

A sequência de chamadas efetuada por este executável é a seguinte:

- llopen: Configura a ligação entre os dois computadores, abrindo a porta série em modo de escrita e leitura. Esta configuração decorre de uma troca de tramas, a trama SET enviada pelo Emissor e a trama UA enviada pelo Recetor. Apesar de a função ser comum aos dois programas, há nela uma distinção das ações conforme o parâmetro status da struct referida no tópico anterior, recebida como parâmetro.

- Receção do pacote de controlo Start, recorrendo à função llread().

- Processamento do pacote de controlo Start recebido, de modo a receber corretamente o nome e o tamanho do ficheiro que vai ser copiado, para efeitos de apenas informar o utilizador destes dados.

- Receção dos pacotes de Informação, recorrendo à função llread(). A cada pacote lido, a sua informação é processada (de modo a ficar apenas com os bytes de informação do ficheiro), e esta informação é logo de seguida escrita para o novo ficheiro, criado imediatamente antes desta receção.

- Quando recebe o pacote de controlo End, o loop de receção de tramas termina.

- llclose(): Encerramento da ligação entre os dois computadores, através de uma troca de tramas. Neste caso, o Recetor enviará uma trama DISC, recebendo como resposta outra trama DISC, e para terminar enviará uma trama UA.

**Casos de uso principais**