Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Informática e Computação 2015/2016

Protocolo de Ligação de Dados

Redes de Computadores

Relatório do Trabalho Laboratorial 1

Elementos do grupo:

Ana Catarina Dias Amaral, up201303169@fe.up.pt

Pedro Miguel Vieira da Câmara, up201304073@fe.up.pt

Pedro Oliveira da Silva, up201306095@fe.up.pt

Sumário

O trabalho realizado ao longo das aulas laboratoriais tem como objetivo a implementação um protocolo de ligação de dados, que permite a transmissão de ficheiros entre computadores, com o uso de um cabo de série.

O seguinte relatório tem como objetivo a consolidação do trabalho realizado ao longo desta primeira metade de semestre. Neste contexto podemos afirmar que é necessário que haja uma ligação entre as partes prática e teórica que envolvem o projeto, consolidando assim a matéria entre ambas.

Introdução

O trabalho realizado ao longo das aulas laboratoriais teve como objetivo a implementação de um protocolo de ligação de dados, de acordo com a especificação descrita pelo guião. Nesta, era pedida a combinação de características de protocolos de ligação de dados existentes, entre os quais a transparência na transmissão de dados e uma transmissão organizada em diferentes tipos de tramas (tipo I, S ou U). O tipo de transmissão utilizado em todo o projeto é assíncrona. Outro objetivo do trabalho consistia em testar o protocolo com uma aplicação simples de transferência de ficheiros.

O relatório final deverá servir, assim, para explicar de uma maneira mais aprofundada o trabalho desenvolvido, de modo a esclarecer qualquer dúvida acerca da nossa implementação, encontrando-se o código relativo a esta em anexo.

Em seguida, irá ser apresentado um conjunto de tópicos que permitem explicar como está organizado o código.

Arquitetura

Blocos funcionais

O trabalho encontra-se divido em duas camadas (camada da ligação de dados e camada de aplicação), de modo a ter uma estrutura organizada e de fácil interpretação.

A camada de ligação de dados disponibiliza funções genéricas do protocolo, nas quais estão incluídas a especificação das funcionalidades de sincronismo, de controlo de erros e numeração de tramas. Podemos dizer que a camada de ligação de dados é a responsável pela transferência dos ficheiros pois é nesta que são executadas as funções de receção e emissão de tramas. Todos os aspetos relacionados com a ligação à porta série estão também aqui incluídos, como a sua abertura, fecho e definição das suas propriedades. Esta camada encontra-se definida nos ficheiros “linklayer.c” e “linklayer.h”.

A camada de aplicação não é independente da camada da ligação de dados, já que tira proveito das suas propriedades para a implementação das suas funções. Esta é responsável pela ligação entre o utilizador e o programa em si e pela transferência dos ficheiros, pois é nesta que são executadas as funções de receção e emissão de pacotes. Uma das suas principais funções é então obter as informações necessárias para a configuração do protocolo, das quais são exemplo, o número de tentativas no caso de uma falha de leitura, o intervalo de *timeout* para ativar o alarme ou até mesmo a ativação da simulação de erros. Os pacotes de controlo e de dados também são processados e enviados a partir desta camada, tornando-a uma peça fulcral no comando da aplicação em si. Esta camada encontra-se definida nos ficheiros “application.c” e “application.h”, utilizando ainda funções definidas nos ficheiros “utilities.c” e “utilities.h” somente para não repetir linhas de código.

Interface

No que toca à interface, esta encontra-se implementada nos ficheiros da camada da aplicação.

Primeiramente são mostrados ao utilizador os valores da configuração. Deste modo é mais fácil verificar se se encontram corretos.

```
************ Configuration ************
       Baud Rate: 9600
       Packet Data Length: 100
       Max Tries per Packet: 5
       Timeout Interval: 1
       Simulate Errors: no
***************************************
```

Figura 1 - Valores configuração por defeito

O utilizador pode configurar o *baud rate*, o tamanho máximo da mensagem, o número de tentativas em caso de falha de comunicação, o intervalo entre cada tentativa e até ativar a simulação de erros.

```
Usage:  ./application <serial port number> [file to send] [optional arguments]
Arguments
  -b INT         Baud rate to use
  -l INT         Packet data length
  -t INT         Max tries per frame
  -i INT         Timeout interval
  -e             Simulate errors
```

Figura 2 - Elementos configuráveis pelo utilizador

Ao longo de toda a transferência, o **recetor** é informado da percentagem que já foi recebida do ficheiro e de uma estimativa do tempo restante para finalização da transferência.

```
Received 95%, remaining time: 1 seconds
Received 96%, remaining time: 1 seconds
Received 97%, remaining time: 0 seconds
```

Figura 3 - Percentagem transferência concluída/Tempo que falta para o final da transferência

No final, é ainda mostrado a informação acerca das estáticas, em ambos os lados.

```
****************** Statistics ******************
       Unique I frames sent: 112
       I frames re-sent: 0
       I Frames received: 0
       Timeouts occurrences: 0
       REJ frames sent: 0
       REJ Frames received: 0
************************************************
```

Figura 4 - Emissor

```
****************** Statistics ******************
       Unique I frames sent: 0
       I frames re-sent: 0
       I Frames received: 112
       Timeouts occurrences: 0
       REJ frames sent: 1
       REJ Frames received: 0
************************************************
```

Figura 5 – Recetor

Estrutura do código

Camada da Aplicação

A camada em si é representada por diversas constantes e por uma estrutura de dados onde é guardado o descritor do ficheiro da porta de série e a indicação de o utilizador ser recetor ou transmissor, implementado no ficheiro “application.h”.

```
struct {
        int fd; /*Descritor correspondente à porta série*/
        int oflag; /*TRANSMITTER | RECEIVER*/
} appLayer;
```

Figura 6 - Estrutura de dados "appLayer", pertencente à camada da Aplicação

As funções principais desta camada (implementadas no ficheiro "application.c") são a **send_file** e a **receive_file** (que se encontram explicadas com maior detalhe mais à frente), que utilizam as funções **send_data_packet()** (função para enviar pacote de dados), **send_control_packet()** (função para enviar pacotes de controlo) e **processControlPacket()** (função para extrair os argumentos de um pacote de controlo recebido), de relevante importância também.

Camada da Ligação de Dados

No caso da camada da ligação de dados, foi criada uma estrutura de dados para gerir as estatísticas, outra para as configurações definidas e uma última para representar a informação relativa à trama que estiver a ser recebida.

Para além destas estruturas existe um enumerável que representa os estados possíveis na maquina de estados, diversas constantes, desde *flags* usadas a valores guardados por defeito no caso de o utilizador não os inserir ou mesmo códigos para possíveis erros. As estruturas de dados desta camada encontram-se implementadas no ficheiro "linklayer.h".

```
struct{
    char port[20]; // Dispositivo /dev/ttySx, x = 0, 1
    unsigned int sequenceNumber; // Número de sequência da trama: 0,1
    volatile bool timeout; // indica se occoreu timeout
        struct termios oldtio; // configuração anterior
        bool disconnected; // Indica se a ligação foi desligada
        int oflag; // Transmissor ou receptor
        int closed; // Indica se a porta série foi fechada
        unsigned int baudrate; // Baudrate usado (valor de configuração de acordo com o header termios)
        unsigned int data_length; // Número de bytes de dados (antes do stuffing) a enviar por pacote de dados
        unsigned int max_retries; // Número máximo de tentativas em caso de falha
        unsigned int timeout_interval; // Intervalo de timeout
        bool simulate_errors; // Simulação de erros
} linkLayer;

struct statistics_t {
        unsigned int sent_i_counter; // Número de tramas I únicas recebidas
        unsigned int retry_i_counter; // Número de tramas I reenviadas
        unsigned int received_i_counter; // Número de tramas I recebidas
        unsigned int timeout_counter; // Número de ocorrências de timeout
        unsigned int sent_rej_counter; // Número de tramas REJ enviadas
        unsigned int received_rej_counter; // Número de tramas REJ recebidas
} statistics;

typedef enum {START=0,FLAG_RCV,A_RCV,C_RCV,DATA,DATA_ESCAPE,STOP} State; // Estados possíveis na máquina de estados
typedef struct {
        State state; // Estado atual
        int fd; // File descriptor da porta série
        bool receive_data; // Indica se é para receber dados
        bool use_timeout; // Indica se é para usar timeout
        unsigned char bcc2; // BCC2 calculado
        unsigned char previous_char; // Caracter recebido anteriormente
        bool use_previous; // Indica se já foi recebido um caracter anteriormente
        unsigned char received; // Caracter recebido
        unsigned char received_control; // Controlo recebido
        unsigned char expected_control; // Controlo esperado
        bool reset; // Indica se é para reeniciar a máquina de estados
        int i; // Índice a escrever no buffer
        int s; // Valor de s, para por exemplo N(s)
        int r; // Valor de r, para por exemplo RR(r)
        char* buffer; // Buffer de dados
        int buffer_size; // Tamanho do buffer de dados
} FrameInfo;
```

Figura 7 – Estruturas de dados da camada da Ligação de Dados

As funções principais desta camada (implementadas no ficheiro "linklayer.c") são as funções **llopen**, **llwrite**, **llread** e **llclose** (que constituem a API da porta de série, que se encontram explicadas com maior detalhe mais à frente). Estas função utilizam as funções **receive_frame()** (função para receber uma trama) e **create_i_frame()** (Cria uma trama de dados (I) para um dado buffer de dados), de importância também relevante para esta camada.

Casos de uso principais

No começo do programa é necessário passar-lhe, no mínimo, o número da porta série e no caso do transmissor, o nome do ficheiro a enviar. Para além disso, podem ser definidos o *baudrate*, o número máximo de tentativas, o tempo para ativar o alarme, o comprimento dos dados enviados por pacote e a ativação do modo de simulação de erros.

É chamada a função **llopen()** para gerar o descritor da porta série utilizada. Em seguida, dependendo de ser emissor ou recetor, é chamada a função **send_file()** ou **receive_file()**. No caso do emissor, é enviado um pacote de controlo através da função **send_control_packet()** e a partir daí são criadas tramas I com o uso da função **create_i_frame()** para depois serem enviadas pelo método **llwrite()**. No final, é novamente chamada a função **send_control_packet()**. Cada pacote tem a quantidade de dados definida. Em cada pacote, que por sua vez é colocado numa trama, são colocados diversos bytes extra para garantir a fiabilidade do protocolo.

Após o envio espera-se uma resposta por parte do recetor com a função **receive_RR_frame()**. Isto repete-se num ciclo até serem enviados todos os dados. No fim, é enviado novamente um novo pacote de controlo a sinalizar o fim do envio dos dados.

Para o caso do recetor, o processo é semelhante. Vão-se lendo os bytes recebidos na função **receive_file()**, recebidos na função **receive_frame()**. Esta função chama a **llread(),** que por sua vez confirma a receção da trama através da **send_rr_frame().**

Quando todas as tramas são enviadas com sucesso, é chamada a função **llclose()** para sinalizar o fim do envio com o uso da **flag** DISC de *disconnect*.

Protocolo de ligação lógica

A camada de ligação de dados é responsável pelas seguintes funcionalidades:

- **Estabelecer e terminar uma ligação através da porta de série**
- **Enviar e receber mensagens através da porta de série**

As estruturas de dados relativas a esta camada encontram-se mostradas anteriormente.

As principais funções que garantem o correto funcionamento destas funcionalidades encontram-se explicadas abaixo:

Função llopen():

Esta função estabelece a ligação entre os dois computadores. Ao ser invocada pelo emissor, este abre a porta, envia o comando SET e aguarda pela resposta do recetor (UA), isto repete-se até o número de tentativas de transmissões ser excedido. Caso receba o UA, a ligação foi corretamente estabelecida e pode continuar com o programa. Ao ser invocado pelo recetor,

este abre a porta e espera pela receção do comando SET, ao receber envia a resposta UA e a ligação é estabelecida.

Função llwrite():

Esta função envia uma trama de dados e espera por uma resposta positiva, fazendo várias tentativas se necessário. Quando é invocada pelo emissor e recebe um buffer, é criada uma trama com a função **create_i_frame(),** que por sua vez é enviada por porta de série. Após o envio, a função fica à espera da resposta. Se acontecer *timeout*, é feita uma nova tentativa de escrita do buffer. Este ciclo acontece até ao número máximo de transmissões definido. A função **create_i_frame(),** para além de adicionar as flags necessárias, faz o *stuffing* da trama.

Função llread():

Esta função recebe uma trama de dados e envia uma resposta. Quando é invocada, tenta receber uma mensagem pela porta de série a partir da função **receive_i_frame().** Caso a função retorne UNEXPECTED_N, significa que foi recebido uma trama com número inesperado e deve ser enviado o respetivo comando RR para que o emissor continue a normal transmissão do ficheiro. Caso receba uma mensagem válida, envia o comando RR correto. Esta função utiliza a função **receive_frame**(), na qual se encontra a nossa maquina de estados e que faz o *destuffing* da trama.

Função llclose():

Esta função é responsável por terminar a ligação. Ao ser invocada pelo emissor, envia o comando DISC e aguarda pela resposta do recetor (DISC). Ao receber esta resposta, é enviado o comando UA e fecha a porta de série. Ao ser invocado pelo recetor, espera pela receção do comando DISC. Ao receber, envia uma resposta DISC e fecha a porta de série.

Protocolo de aplicação

A camada da aplicação é responsável pelas seguintes funcionalidades:

- **Envio/receção de pacotes de controlo**
- **Envio/receção de pacotes de dados**
- **Envio/receção do ficheiro especificado**
- **Interface**

As estruturas de dados relativas a esta camada encontram-se mostradas anteriormente.

As principais funções que garantem o correto funcionamento destas funcionalidades encontram-se explicadas abaixo:

Função send_file():

Esta função é utilizada pelo emissor para enviar um ficheiro. Recebe como parâmetros o nome do ficheiro e o tamanho dos dados por pacote. Envia o primeiro pacote de controlo (**START_PACKET**) com recurso à função **send_control_packet()**. Caso o pacote tenha sido enviado com sucesso é enviado o pacote de dados com recurso à função **send_data_packet()** e no final é enviado o segundo e ultimo pacote de controlo (**END_PACKET**) com recurso novamente à função **send_control_packet()**.

Função receive_file():

Esta função é usada pelo recetor para receber um ficheiro e recebe como parâmetros o tamanho dos dados por pacote. Até receber um pacote **END_PACKET**, lê da porta de série através da função **llread()** . Seguidamente, se o primeiro byte do que leu for 1 (**START_PACKET**), cria um ficheiro de *output*, caso seja 2 (**END_PACKET**), termina o ciclo e caso seja 0, significa que é um pacote de dados e escreve o que leu para o ficheiro de output. O processamento dos pacotes de controlo é feito com a chamada à função **processControlPacket**().

É na função **main()** que é chamada a função **llopen()** para abrir a porta de série. De seguida, se for o emissor, envia o ficheiro (função **send_file()**) e se for o recetor, recebe o ficheiro(função **receive_file()**). No final é chamada a função **llclose()** para terminar a ligação.

Validação

Para garantir que o trabalho foi bem feito, foram feitos diversos testes.

O principal de todos os testes foi o envio sem interrupções do ficheiro "pinguim.gif". Após garantir que o envio estava igual ao enviado, foi testado o envio com a remoção do cabo da porta de série durante o processo e foi também testada a introdução de informação errada com o raspar de uma chave nos pinos da porta série. O último foi testar o exceder de tentativas e de *timeout*.

De seguida podemos ver as estatísticas resultantes com a opção de simulação de erros e *timeout* excedido (lado esquerdo - **emissor**, lado direito - **recetor**):

```
****************** Statistics ******************
        Unique I frames sent: 112
        I frames re-sent: 18
        I Frames received: 0
        Timeouts occurrences: 2
        REJ frames sent: 0
        REJ Frames received: 16
************************************************
```

Figura 6 - Estatísticas Emissor

```
****************** Statistics ******************
        Unique I frames sent: 0
        I frames re-sent: 0
        I Frames received: 112
        Timeouts occurrences: 0
        REJ frames sent: 16
        REJ Frames received: 0
************************************************
```

Figura 7 - Estatísticas Recetor

Relativamente aos testes executados com a extração do cabo da porta de série, não possuímos imagens, porém estes testes foram comprovados pelo docente aquando da apresentação do projeto e todos eles foram bem-sucedidos.

Elementos de valorização

Da lista apresentada no guião do trabalho foi possível implementar todos os elementos de valorização com sucesso, sendo que em vez da opção de continuar o envio do ficheiro após um DISC, fizemos uma opção para continuar o envio do ficheiro após exceder o número máximo de tentativas permitidas, através de uma pergunta ao utilizador.

Seleção de parâmetros pelo utilizador e geração aleatória de erros em tramas de Informação

Ao inicializar o programa, o utilizador pode configurar o *baud rate*, o tamanho máximo da mensagem, o número de tentativas em caso de falha de comunicação, o intervalo entre cada tentativa. O utilizador pode ainda ativar a simulação de erros com "**-e**". A configuração destes elementos faz-se na função **configWithArguments()** implementada no ficheiro "application.c" que por sua vez, chama a função **setConfig()** implementada no ficheiro "linklayer.c".

Caso algum destes parâmetros introduzidos pelo utilizador sejam inválidos é mostrada a mensagem de ajuda ilustrada na **Figura 2**.

Implementação de REJ

Sempre que na receção de dados, existe um erro no BCC ou no BCC2, o que significa que os dados não foram corretamente recebidos, o comando REJ é enviado para o emissor, para que este volte a reenviar a mensagem.

Verificação da integridade dos dados pela Aplicação

A aplicação verifica que o tamanho do ficheiro recebido é igual ao tamanho indicado nos dois pacotes de controlo. A informação dos pacotes de controlo, é também verificada e deve coincidir. Também é verificada a numeração de cada pacote de forma a garantir que pacotes duplicados são ignorados.

Registo de ocorrências

Durante a execução do programa, várias ocorrências vão sendo registadas para que no final da execução sejam apresentadas as estatísticas. As ocorrências são guardadas na estrutura "statistics", definida no ficheiro "linklayer.h", onde são registadas o número de tramas I enviadas, recebidas e reenviadas, o número de ocorrências de *timeout* e o numero de tramas REJ enviadas e recebidas.

Conclusões

Relativamente ao trabalho, o grupo compreendeu bem todos os pontos pedidos na realização deste trabalho, estando este divido em duas grandes camadas como foi abordado na secção da Arquitetura deste relatório. As camadas possuem uma ligação unidirecional, sendo a camada da aplicação a exercer o controlo perante a de ligação de dados, não acontecendo o contrário. O cabeçalho dos pacotes a transportar nas tramas de informação não é processado na camada de ligação de dados, sendo invisível para esta. Nesta camada, de ligação de dados, não existe distinção entre pacotes de dados nem de controlo e as numerações dos pacotes são dispensáveis. No que diz respeito à camada da aplicação, esta não conhece as propriedades da camada de ligação de dados porém acede aos serviços por ela fornecidos, como é exemplo a configuração de argumentos, realizada na camada da aplicação (função **configWithArguments()**), e que utiliza a função **setConfig()** da camada de ligação de dados.

A realização deste trabalho contribuiu para consolidar os conceitos aprendidos nas aulas teóricas e teórico-práticas e também para aumentar o conhecimento sobre o funcionamento da porta de série.