FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

1º Trabalho Laboratorial

Protocolo de Ligação de Dados

João Duarte	up201707984
Diogo Nunes	up202007895
Nuno Penafort	up202008405

L.E.I.C - Redes de Computadores

Índice

Sumário

No âmbito da unidade curricular de Redes de Computadores, foi-nos proposto como primeiro trabalho a implementação de um protocolo de ligação de dados com uma aplicação que permitisse a transferência de ficheiros entre dois computadores de forma assíncrona, através de uma porta série.

1 - Introdução

Este trabalho teve como objetivo implementar um protocolo de ligação de dados, de modo que fosse possível realizar uma comunicação estável entre dois sistemas ligados por um meio de transmissão sendo, neste caso, um cabo de série.

O relatório apresenta a seguinte estrutura:

- **Arquitetura** - blocos funcionais e interfaces
- **Estrutura do código** - APIs, principais estruturas de dados, principais funções e a sua relação com a arquitetura
- **Casos de uso principais** - identificação; sequências de chamada de funções
- **Protocolo de ligação lógica** - identificação dos principais aspetos funcionais
- **Protocolo de aplicação** - identificação dos principais aspetos funcionais
- **Validação** - descrição dos testes efetuados
- **Eficiência do protocolo de ligação de dados** - caraterização estatística da eficiência do protocolo
- **Conclusões** - síntese da informação apresentada nas secções anteriores

2 - Arquitetura

O trabalho é composto por duas camadas (link_layer.c e application_layer.c) que funcionam de forma independente uma da outra e um header que define os macros necessários (macros.h).

- **link_layer** - camada responsável pelas interações com a porta série e por assegurar que a informação está a ser transmitida corretamente ou avisar quanto há existência de possíveis erros com os não conseguiu lidar.
- **application_layer.c** – camada responsável pela interação com o utilizador e com o ficheiro que foi recebido/será enviado. É nesta camada que os ficheiros são preparados para ser enviados e vice-versa.

3 - Estrutura do Código

O código está divididos em dois programas principais (link_layer.c e application_layer,c), os seus respetivos headers e mais um header que contém os macros

link_layer.c – Main Functions and Data Structures

- **llopen** – Função responsável por abrir a conexão porta série para transmissão de tramas
- **llwrite** – Função responsável por escrever a trama para a porta de série
- **llread** – Função responsável por ler a trama recebida da porta série e averiguar se esta foi recebida corretamente
- **llclose** – Função responsável por fechar a conexão com a porta série
- **state**_handler – Função responsável por
- **State** – Estrutura que contém os possíveis estados da aplicação
- **buildFrame** – Função que cria frames
- **buildDataFrame** – Função que cria frames para tramas de tipo I

application_layer – Main Functions

- **get_type_length_value** – Função que verifica os parâmetros do ficheiro
- **applicationLayer** – Função responsável por todas as interações realizadas no application_layer

4 - Casos de uso principais

Para utilizar o programa o utilizador deve primeiro compilá-lo. Foi criado um ficheiro MakeFiles que facilita o uso do programa.

1 – Usar o comando “make” para compilar o código

2 – Usar o comando “make run_cable” para simular uma ligação por porta série

3 – No terminal do transmissor, usar o comando “make run_tx”

4 – No terminal do recetor, usar o comando “make run_rx”

5 – Usar o comando “make check_files” para averiguar se o ficheiro foi transmitido corretamente

5 - Protocolo de ligação lógica

Este protocolo é responsável pela abertura, fecho, leitura e escrita de dados e pelo tratamento de tramas.

Principais funções da camada :

llopen - Função responsável por iniciar a comunicação entre dois computadores através da porta-série. Considerando que uma conexão só pode estar “aberta” quando o recetor está preparado para receber informação e o emissor preparado para transmitir informação, é necessário a implementação de um protocolo que seja capaz de indicar a disponibilidade do recetor e do transmissor. Para tal, deve-se distinguir se se trata de um transmissor ou de um recetor.

No caso do recetor, este deve enviar uma trama SET, indicando assim que está preparado para receber informação e aguardar pela UA. Já no caso do transmissor acontece

o oposto, é enviada uma trama UA e espera-se pela trama SET.

llwrite – Função responsável por “escrever” um dado packet de informação de modo que este possa ser transmitido pela porta-série.

llread - A função lê a mensagem, de seguida verifica a existência de erros no BCC1 e no BCC2, que indicam se a transmissão foi realizada de forma correta. De seguida, o destuffing ocorre, a frame de supervisão é removida e são enviadas as mensagens RR ou REJ. RR no caso de sucesso e REJ no caso de haver um erro na transmissão.

llclose - A função, no caso de ser um transmissor, recebe a mensagem DISC e escreve a mensagem UA para avisar o recetor de que a conexão foi terminada. No caso de ser um recetor envia a mensagem DISC e espera pela mensagem UA. No final é invocada a função close_serial_port do ficheiro physical_layer que fecha a porta-série.

6 - Protocolo de aplicação

O protocolo de aplicação é responsável pelo envio e receção de ficheiros, processamento do serviço (tratamento de cabeçalhos e distinção entre pacotes de controlo e de dados).

Assim podemos verificar que foram enviadas tramas de controlo de inicio e fim, com o tamanho do ficheiro e nome.

7 – Validação

Para testar o funcionamento do programa, este fio sujeito a diferentes testes, tais como:

- Envio de ficheiros de tamanhos diferentes;
- Envio do mesmo ficheiro com pacotes de tamanhos diferentes
- Envio do mesmo ficheiro com time outs diferentes
- Envio do mesmo ficheiro com retransmissões diferentes
- Interrupção da ligação da porta-série durante o envio do ficheiro

- Introdução de ruido na porta-série durante o envio do ficheiro

O nosso programa concluiu apenas alguns testes descritos acima com sucesso, havendo problemas no que diz respeito á introdução de ruido e interrupção da ligação da porta-série.

8 – Eficiência do protocolo de ligação de dados

De modo a perceber como cada um dos valores influencia a eficiência foram realizados alguns testes com a finalidade de estudar os respetivos cenários.

Os testes foram executados com o ficheiro de imagem disponibilizado (penguin.gif)

- **Variação do FER**

  Podemos observar que o FER tem um impacto significativo na eficiência do programa, isto deve-se ao facto de que ao gerar mais erros também acabamos por gerar mais time outs, o que leva a um maior tempo de execução do programa e, por consequência, uma diminuição na eficácia

- **Variação do baudrate**

  Podemos observar que o aumento do baudrate (capacidade de ligação) causa uma diminuição na eficiência

- **Variação do tamanho das I-frames**

  Podemos observar que, até certo ponto, quanto maior o tamanho de cada pacote de dados mais eficiente será o programa

- **Variação do tempo de propagação**

  Podemos observar que quanto maior for o tempo de propagação de cada trama (T_prop), menos eficiente será o programa

Stop and Wait é um dos protocolos usados para controlo de erros. Neste protocolo, inicialmente o transmissor envia tramas de informação I e espera por uma confirmação positiva **ACK** que deve ser enviada pelo recetor. Após o recetor receber a trama **I**, este averigua se houve algum erro na transmissão e, caso não seja encontrado nenhum erro, envia uma confirmação positiva **ACK** para o transmissor, caso contrário, envia uma

confirmação negativa **NACK**. De seguida, o recetor continua o processo de envio de novas tramas **I** no caso de ter recebido **ACK** e, no caso de ter recebido **NACK**, reenvia a trama **I** que não foi recebida corretamente. Existe também casos em que **I**, **ACK** ou **NACK** podem ser perdidos. Para evitar que casos como estes interrompam completamente o funcionamento do programa deve ser implementado um time out que ocorre no caso de haver uma espera demasiado elevada por **I**, **ACK** ou **NACK**. Finalmente, de modo a parear corretamente as tramas **I** com as respetivas tramas **ACK** e identificar se se trata de uma trama duplicada, deve-se numerá-las (0 ou 1) permitindo assim que, por exemplo, **ACK(0)** indica que o recetor está à espera da trama **I(0)**.

Nesta aplicação foi implementado o protocolo Stop and Wait. As tramas **I**, neste caso denominadas por CTRL_DATA, são identificadas com um 0 ou 1. Estas tramas serão enviadas pelo transmissor que por sua vez aguarda pela trama **ACK** ou **NACK**, neste caso denominadas por CTRL_RR e CTRL_REJ respetivamente, que também se encontram denominadas por 1 ou 0 dependendo da trama **I** recebida.

9 – Conclusões

O trabalho laboratorial teve como objetivo a implementação de um protocolo de ligação de dados para a transferência de dados entre dois computadores através de uma porta serie. Após a sua implementação, este foi testado e avaliado relativamente à sua eficiência.
Em suma, a criação do protocolo de ligação de dados foi praticamente bem sucedida, com a maior parte dos objetivos do guião cumpridos.

Ao longo da realização deste trabalho fomos capazes de aprofundar conhecimentos teóricos relativamente à independência de camadas e o protocolo Stop and Wait.