Faculdade de Engenharia da Universidade do Porto

Ligação de Dados

RC - Relatório Projeto 1

Turma: 3LEIC04

Trabalho realizado por:
Anete Pereira (up202008856)
Isabel Amaral (up202006677)
Milena Gouveia (up202008862)

Sumário

Este trabalho laboratorial foi desenvolvido no âmbito da unidade curricular de Redes de Computadores. O objetivo foi o desenvolvimento de um protocolo de ligação de dados e o de uma camada de aplicação integral a esta ligação, os quais permitem a transferência de ficheiros entre dois computadores usando uma conexão entre portas série RS-232.

A nossa implementação do protocolo de ligação de dados é capaz de transferir corretamente um ficheiro de um computador para o outro e recuperar informação perante a ocorrência de erros durante a transmissão.

Introdução

Neste trabalho laboratorial aplicamos o conhecimento adquirido nas aulas teóricas para implementar um protocolo de ligação de dados capaz de fornecer um serviço de comunicação de dados fiável entre dois sistemas, um com o papel de transmissor e outro com o papel de recetor, ligados por meio de um cabo de série. O protocolo foi desenvolvido na linguagem de programação C e no sistema operativo Linux.

O objetivo deste relatório é examinar a componente teórica deste trabalho. O mesmo encontra-se dividido nas seguintes secções:

- **Arquitetura:** identificação e descrição dos blocos funcionais e interfaces.
- **Casos de uso principais:** identificação das principais funções e descrição da sequência de eventos.
- **Protocolo de ligação lógica:** identificação dos principais aspetos funcionais e da descrição da estratégia de implementação destes aspetos.
- **Protocolo de aplicação:** identificação dos principais aspetos funcionais e da descrição da estratégia de implementação destes aspetos.
- **Estrutura do código:** APIs, principais estruturas de dados e funções e a sua relação com a arquitetura.
- **Validação:** descrição dos testes efetuados.
- **Eficiência do protocolo de ligação de dados:** análise estatística da eficiência do protocolo, efetuada recorrendo a medidas sobre o código desenvolvido.
- **Conclusões:** reflexão sobre os objetivos alcançados.

Arquitetura

O sistema desenvolvido segue uma organização em camadas e inclui uma camada de ligação de dados, com um menor nível de abstração, e uma camada de aplicação que usa os serviços proporcionados pela ligação de dados.

A camada de ligação de dados inclui toda a lógica sobre a criação das tramas segundo a estrutura definida para cada tipo e o seu envio, bem como o envio de tramas de confirmação.

A camada de aplicação tem como objetivo o processamento dos argumentos recebidos e, dependendo do papel a ser desempenhado na transferência ser o de transmissor ou o de recetor, a leitura/criação do ficheiro e a divisão da informação do ficheiro em porções ou união das porções recebidas, respetivamente.

Deste modo, é possível atingir independência entre ambas, dado que a camada de aplicação não conhece os detalhes da implementação do protocolo de ligação de dados, tais como os mecanismos utilizados para delimitação das tramas, *stuffing*, *ARQ*, etc., e na camada de ligação de dados não é feito nada em relação ao processamento do ficheiro, ou seja não são levados em conta pormenores como a distinção entre os diferentes tipos de tramas de informação, o número de tramas a ser enviadas, a numeração das tramas, etc.

Casos de uso principais

Após o projeto ter sido compilado, deverá ser executado da seguinte forma, tendo em atenção que o recetor deverá ser inicializado em primeiro lugar:

Recetor: ./bin/main <porta> rx <ficheiro>

Transmissor: ./bin/main <porta> tx <ficheiro> ,

<porta>: porta série (/dev/ttySx)

<ficheiro>: localização relativa do ficheiro a ser enviado

A transmissão do ficheiro decorre segundo a seguinte sequência de eventos:

- é configurada a ligação entre os dois computadores
    - llopen() ao nível da camada de aplicação
- é estabelecida a ligação entre os dois computadores através do envio de tramas de supervisão que sinalizam o início da transmissão
    - llopen() ao nível da camada de aplicação
- do lado do transmissor, o ficheiro escolhido é aberto e o seu conteúdo é lido e armazenado num buffer
    - send_file() ao nível da camada de aplicação
- o transmissor divide os dados do ficheiro em porções e envia as mesmas trama a trama

    - send_data() com sucessivas chamadas a llwrite() ao nível da função send_file()

- ao mesmo tempo, o transmissor vai recebendo trama a trama as diferentes porções do ficheiro e reconstrói o conteúdo do ficheiro original armazenando-o num buffer
    - receive_file() com sucessivas chamadas a llread() ao nível da camada de aplicação

- do lado do emissor, a informação recebida é reescrita num novo ficheiro
    - receive_file() ao nível da camada de aplicação

- é terminada a ligação entre os dois computadores através do envio de tramas de supervisão que sinalizam o final da transmissão
    - llclose() ao nível da camada de aplicação

Protocolo de ligação lógica

O protocolo de ligação de dados fornece um serviço de comunicação entre dois computadores. A troca de informação entre os dois sistemas baseia-se na transmissão e receção de três diferentes tipos de tramas: informação, supervisão (SET, DISC, RR e REJ) e não-numeradas (UA). Todos estes tipos de tramas contêm um cabeçalho comum que permite identificar o tipo de trama, estando delimitadas por *flags* de 8 bits, e utilizam um mecanismo de geração de um *Block Check Character* que permite detetar erros de discrepâncias entre a informação enviada e a informação recebida. As tramas de informação, além do cabeçalho, também transportam um campo de dados com a informação do ficheiro a transmitir. Esta informação é protegida por um *Block Check Character* próprio e, de modo a evitar o falso reconhecimento de uma *flag* no interior do campo de dados, é utilizado um mecanismo de transparência baseado em *byte stuffing* antes do envio e *destuffing* após a receção.

Como mecanismo de controlo de erros, são também utilizadas diversas técnicas: retransmissões com base em timeouts, deteção de tramas duplicadas e o envio de tramas de confirmação. Perante um erro no cabeçalho da trama, esta é simplesmente ignorada. Caso a trama recebida seja duplicada, isto é com código diferente daquele que o recetor estava à espera de receber, esta é ignorada e é enviado um código de confirmação RR a informar qual é a trama que na verdade deve ser recebida. No caso de a informação do campo de dados da trama recebida ser inconsistente, ou seja não corresponder com o BCC2, esta é ignorada e é enviado um código de rejeição REJ a informar que a trama em questão deve ser reenviada.

O protocolo de ligação lógica assenta no uso de quatro funções principais: llopen(), llwrite(), llread() e llclose(), as quais funcionam como interface para a camada de aplicação aceder aos serviços da camada de ligação de dados.

int llopen(LinkLayer connection_parameters): estabelecimento da ligação, devolve o identificador da ligação de dados ou -1 em caso de erro.

int llwrite(int fd, unsigned char* packet, int packet_size): envio de uma trama de informação com **packet** no campo de dados e campo de dados de tamanho **packet_size**, devolve um valor diferente de 0 em caso de erro.

int llread(int fd, unsigned char* packet): receção do campo de dados contido numa trama de informação através de **packet**, devolve o tamanho do campo recebido.

int llclose(int fd, LinkLayer connection_parameters): terminação da ligação, devolve o identificador da ligação de dados ou -1 em caso de erro.

Protocolo de aplicação

O protocolo de aplicação utiliza dois tipos de pacotes a ser enviados pelo emissor: pacotes de controlo e pacotes de dados. Os pacotes de controlo contém um cabeçalho, com o número de sequência da trama e o tamanho total do pacote, e um campo de dados com informação lida diretamente do ficheiro. Os pacotes de controlo são enviados no início e no fim da transmissão dos pacotes de dados e transportam o nome do ficheiro que está a ser transmitido (ex: penguin.gif) e o tamanho do mesmo.

Os pacotes da camada de aplicação são enviados à camada de ligação de dados e utilizados como campo de dados das tramas de informação que são enviadas e recebidas a esse nível.

Estrutura do código

O código está dividido em cinco ficheiros principais: transmitter.c, receiver.c, link_layer.c, application_layer.c e utils.c que correspondem respetivamente ao transmissor, recetor, camada de ligação, camada de aplicação e funções auxiliares úteis para o programa.

Transmissor - transmitter.c

Contém as funções relativas ao transmissor.

Funções Principais:

- tx_start_transmission(): responsável por iniciar a transmissão do lado do transmissor enviando uma trama SET para o recetor e posteriormente verificando com auxílio da state machine tx_state_machine() se a resposta recebida corresponde a uma trama UA.

- tx_stop_transmission(): responsável por terminar a transmissão do lado do transmissor começando por enviar uma trama DISC para o recetor e, se a resposta recebida por parte do mesmo for igualmente uma trama DISC, enviando também uma trama UA.

- send_info_frame(): envia uma trama de informação e espera pela resposta relativa ao envio da mesma ou que ocorra timeout, podendo precisar de realizar um reenvio.

Recetor - receiver.c

Contém as funções relativas ao recetor.

Funções Principais:

- rx_start_transmission(): responsável por iniciar a transmissão do lado do recetor enviando uma trama UA para o transmissor após a receção de uma trama SET que é verificada com auxílio da state machine rx_state_machine().
- int rx_stop_transmission(): responsável por terminar a transmissão do lado do recetor começando por enviar uma trama DISC para o transmissor após a receção de uma trama DISC e esperando pela receção de uma trama UA.
- int receive_info_frame(): recebe uma trama de informação, verifica com auxílio da state machine info_frame_state_machine() se há erros na trama e envia uma resposta adequada através de tramas de confirmação RR ou REJ.

Camada de ligação - link_layer.c

Contém as principais funções e macros da camada de ligação.

Estrutura de Dados:

- **LinkLayerRole:** armazena os possíveis papéis que podem ser desempenhados pelos sistemas, isto é, transmissor ou recetor.
- **LinkLayer:** armazena a porta série e o papel desempenhado pelo sistema a correr o programa.

Funções Principais:

- llopen(): configura e estabelece a ligação entre o transmissor e o recetor recorrendo às funções rx_start_transmission() e rx_stop_transmission().
- llwrite(): do lado do transmissor, envia as tramas de informação uma a uma recorrendo à função send_info_frame().
- llread(): do lado do recetor, recebe as tramas de informação uma a uma recorrendo à função receive_info_frame().
- llclose(): efetua a terminação da ligação recorrendo às funções rx_stop_transmission() e tx_stop_transmission().

Camada de aplicação - application_layer.c

Contém as principais funções e macros da camada de aplicação.

Funções Principais:

- send_data(): divide o ficheiro em pacotes de no máximo 252 bytes, para cada pacote constrói o respectivo pacote de dados, com auxílio da função assemble_data_packet(), e envia-o ao recetor recorrendo à função llwrite()
- send_file(): abre o ficheiro em modo de leitura, determina o seu tamanho e guarda a sua informação num *buffer*, envia o pacote de controlo *start,* divide a informação lida do ficheiro por diferentes pacotes de dados e envia-os com auxílio da função send_data() e, por fim, envia o pacote de controlo *end.*
- receive_file(): determina o tamanho do ficheiro a ser recebido a partir do primeiro pacote de controlo recebido, recorrendo à função llread(), copia cada pacote de dados recebido para um *buffer*, por fim, é criado um novo ficheiro para onde é escrita toda a informação que foi guardada no *buffer.*
- application_layer(): controla o fluxo de execução do programa, tanto do lado do transmissor como do lado do recetor

Utils - utils.c

Contém várias funções auxiliares que consideramos úteis definir.

Funções Principais:

- stuffing(): realiza o stuffing de bytes do campo de controlo de uma trama de informação
- send_control_packet(): constrói um pacote de controlo e efetua o seu envio
- receive_control_packet(): recebe um pacote de controlo
- assemble_data_packet(): constrói um pacote de dados
- assemble_supervision_frame(): constrói uma trama de supervisão
- assemble_information_frame(): constrói uma trama de informação assegurando o *stuffing* do seu campo de dados
- create_termios_structure(): faz a configuração da porta série

Validação

Para verificar a eficiência do nosso programa realizamos os seguintes testes:

- Envio de ficheiro com variação de erros no BCC1 e BCC2 (FER) - 0%, 5%, 10%, 20%
- Envio de ficheiro com diferentes valores para o BAUDRATE - 4800, 9600, 19200, 38400, 57600
- Envio de ficheiro com pacotes de diferentes tamanhos - 256, 512, 1024, 2048 bytes

- Envio de ficheiros com simulação de diferentes tempos de propagação de tramas de informação - 0s, 0.05s, 0.1s, 0.15s, 0.2s
- Envio de ficheiro com interrupção no cabo
- Envio de ficheiros com diferentes tamanhos

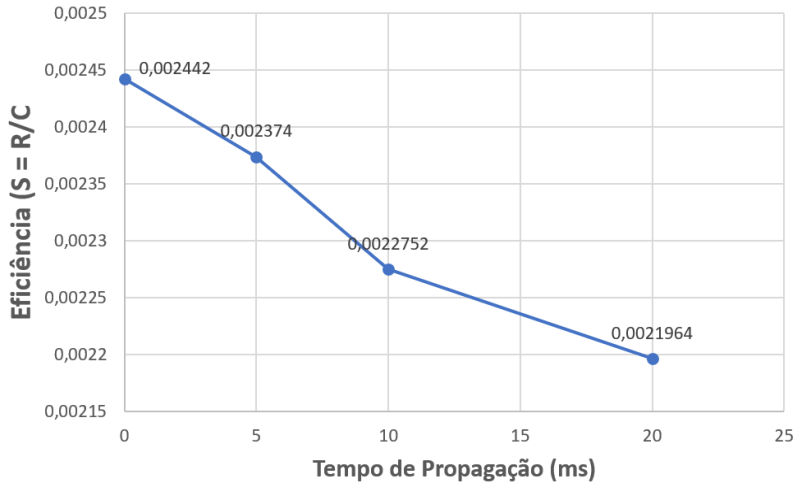
Eficiência do protocolo de ligação de dados

As análises de eficiência (S) do programa foram realizadas variando individualmente o Frame Error Ratio (FER), o tempo de propagação (T_Prop), a Capacidade de ligação (C), e o tamanho da trama de informação (I_Frame). Foi usado o ficheiro penguin.gif disponibilizado e cujo tamanho é de 10968 bytes.

Variação do FER (Anexos: Tabela 1)

Para variar o FER, simulou-se no recetor, a ocorrência de erros no cabeçalho (BCC1) e no campo de dados (BCC2), com probabilidades predefinidas.

Desenhou-se um gráfico em função do valor de S com as percentagens de erros simulados, sendo estes 0%, 5%, 10% e 20%. Através da análise do gráfico conclui-se que o FER tem um impacto significativo na eficiência do programa. Isto deve-se ao facto de um erro gerado no BCC1 implicar o reenvio da trama associado a uma operação de timeout por 3 segundos. Já um erro no BCC2 não tem tanto impacto uma vez que só causa o reenvio imediato da trama.

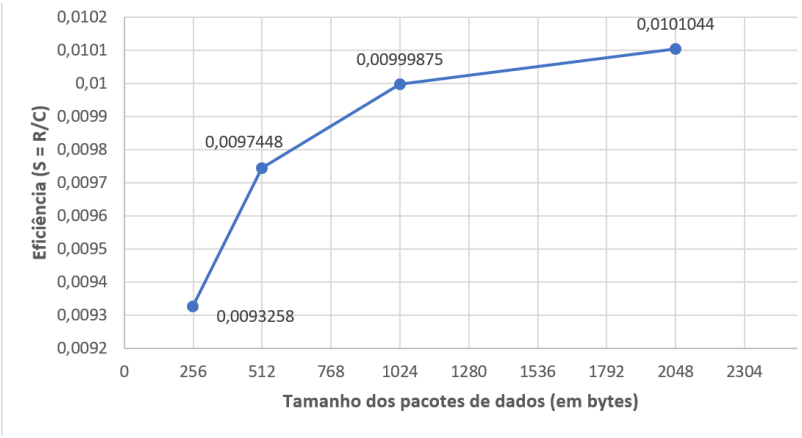


Variação do C

Para os valores testados, o aumento do Baudrate não causou uma diferença significativa na eficiência do programa.

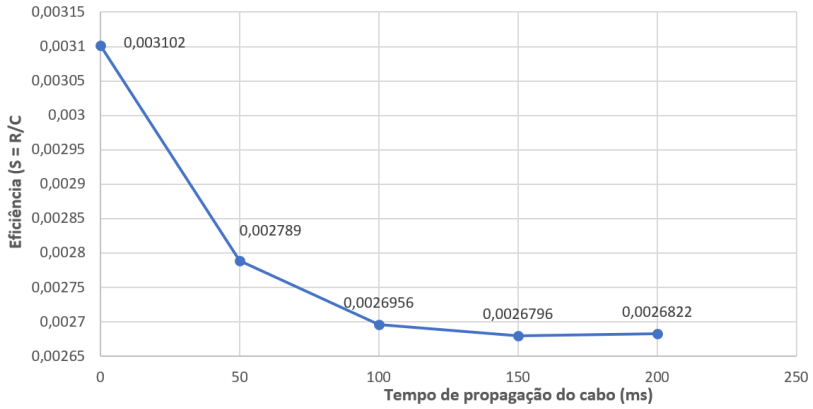
Variação do I_Frame (Anexos: Tabela 2)

Com base no gráfico abaixo verifica-se que quanto maior o tamanho dos pacotes de dados, mais eficiente é a transferência do ficheiro. Isto acontece porque, como está a ser enviada mais informação em cada trama, é necessário a transferência de menos tramas.



Variação do T_Prop (Anexos: Tabela 3)

Com base no gráfico a seguir apresentado, pode-se concluir que com a variação do tempo de propagação, a eficiência do programa diminui. Isto deve-se ao facto de, com o atraso introduzido, cada trama demorar mais tempo a ser enviada/recebida.



Conclusões

O trabalho foi concluído sendo possível o envio de ficheiros entre dois sistemas ligados através de uma porta série com sucesso. Foi possível aplicar corretamente o conceito de distinção entre camadas, atingir os objetivos relacionados com a estrutura do código e aprofundar diversos conceitos introduzidos nas aulas teóricas.

Anexo I - Estudo da Eficiência do Protocolo

FER (%)	0	5	10	20
Eficiência (S)	0,00233	0,002247	0,002123	0,002291
	0,002445	0,002589	0,002339	0,002235
	0,002473	0,002384	0,002195	0,002094
	0,002454	0,002329	0,002394	0,002133
	0,002509	0,002321	0,002325	0,002229
Média de S	0,002442	0,002374	0,002275	0,002196

Tabela 1: Cálculos de eficiência segundo variação de FER

l_Frame	256	512	1024	2048
Eficiência (S)	0,009126	0,009694	0.009424	0,010231
	0,009276	0,009904	0,009084	0,009786
	0,009826	0,009746	0,009625	0,009737
	0,009811	0,00907	0,009895	0,010304
	0,00859	0,01031	0,011391	0,010464
Média de S	0,009326	0,0097448	0,0099988	0,0101044

Tabela 2: Cálculos de eficiência segundo variação do tamanho do pacote de dados

T_Prop	0	50	100	150	200
Eficiência (S)	0,002813	0,002628	0,002728	0,002976	0,002819
	0,002921	0,002808	0,002676	0,002495	0,003267
	0,002837	0,00281	0,002628	0,00228	0,002354
	0,003295	0,002467	0,002415	0,003116	0,002406
	0,003644	0,003232	0,003031	0,002531	0,002565
Média de S	0,003102	0,002789	0,002696	0,00268	0,002682

Tabela 3: Cálculos de eficiência segundo variação do tempo de propagação