

Redes de Computadores

*1º Trabalho Laboratorial*

*´7 de novembro de 2017*

Índice

|  |  |
| --- | --- |
| **Sumário** | 0 |
| **Introdução** | 2 |
| **Arquitetura** | 2 |
| **Estrutura do código** | 2 |
| **Casos de uso principais** | 2 |
| **Protocolo de ligação lógica** | 2 |
| Identificação dos principais aspetos funcionais | 2 |
| Descrição da estratégia | 2 |
| **Protocolo de aplicação** | 2 |
| **CARLOS - Validação** | 2 |
| **CARLOS - Eficiência do protocolo de ligação de dados** | 2 |
| **Conclusões** | 2 |
| **Anexo I** | 2 |

Sumário

No âmbito do primeiro trabalho laboratorial de Redes de Computadores, este relatório tem como objetivo realizar uma análise estatística ao comportamento do protocolo implementado para a transmissão e receção de ficheiros através da porta série.

Assim, o trabalho foi realizado com sucesso, uma vez que todos objetivos propostos foram cumpridos, e estando desta maneira o protocolo capaz de assegurar uma transmissão correta dos dados, mesmo que ocorram alterações nos pacotes durante a mesma.

(dois parágrafos: um sobre o contexto do trabalho; outro sobre as principais conclusões do relatório, falar sober a )

Camada de Aplicação (Application Layer), a Camada de Ligação de Dados (Data Link Layer) e a Camada Física (Physical Layer). A proposta de trabalho baseou-se em realizar uma transferência de ficheiros entre dois computadores

Introdução

Arquitetura e Estrutura do código

O trabalho está organizado em duas principais camadas lógicas que permitem uma correta organização e funcionamento do protocolo, tornando-o modular.

Camada de Ligação (Data Link Layer)

A camada de Ligação é a camada de software “mais baixa” e é responsável pela comunicação e verificação da coesão dos resultados provenientes da porta de série.

Deste modo, esta camada contém todas as funções relativas ao estabelecimento da ligação entre os terminais (LLOPEN), escrita e leitura da informação utilizando a porta série (LLWRITE / LLREAD), e respetiva análise em relação à sua coesão (state\_machine e verify\_bcc2). Esta camada é também responsável pelo fecho da ligação no fim da comunicação (LLCLOSE).

FRASE E ESTRUTUA DA STRUCT

Camada de Aplicação (Application Layer)

A camada de Aplicação é a camada “mais alta” e é responsável pela inicialização do programa (função main). É a partir desta que se desenrola o processo de transferência de dados através da porta de série. Esta camada é também responsável pela criação dos pacotes que contêm os dados a enviar, tanto as tramas de Start e End (create\_STARTEND\_packet) como as tramas de Informação (data\_package\_constructor).

Assim, esta camada contém, entre outras funções, as funções de leitura (handle\_readfile) e de criação/escrita (handle\_writefile) do ficheiro a transmitir, no caso de o programa ser executado no modo de escritor ou no modo de leitor respetivamente.

Para uma melhor estruturação esta camada contém duas estruturas de dados (structs) essenciais para guardar informação importante relativa ao ficheiro de leitura/escrita e dados a enviar para a camada de ligação relativos à comunicação com a porta de série.

ESTRUTURA DA STRUCT

Casos de usos principais

Protocolo de ligação lógica

A camada de ligação é uma das camadas implementadas neste projeto. Esta camada é responsável pela comunicação e coesão da mesma através de verificações feitas aos valores BCC1, BCC2 e C1. Assim esta camada tem as seguintes funcionalidades:

* Estabelecer e terminar uma ligação através da porta de série (Envio e receção das tramas SET e UA).
* Escrever e ler mensagens através da porta série (Envio e receção das tramas RR, REJ e I)
* Adicionar e remover os cabeçalhos necessários para o envio das tramas S e I referentes ao envio dos pacotes de informação, e respetivo stuff/destuff.
* Analisar os cabeçalhos das mensagens recebidas enviando a respetiva trama de resposta.

Deste modo as principais funções desta camada são as seguintes:

LLOPEN

*int LLOPEN(char\* port, char\* mode, char\* timeout, char\* max\_transmissions)*

A função llopen é responsável por estabelecer a ligação através da porta de série. Como tal, esta função ao ser invocada abre e configura a porta de série com os respetivos parâmetros necessários - *set\_serial\_port()* - posteriormente é enviado pelo emissor o comando SET – *set\_writer()* - e como resposta o recetor envia UA – *set\_reader()*. Em caso de sucesso a ligação fica estabelecida, e a função retorna o valor do descritor da ligacao através da porta de série, caso contrário existem as seguintes possibilidades:

* se UA não for recebido: a trama SET é reenviada passados timeout segundos até um máximo de max\_transmissions tentativas. Caso seja atingido o número máximo de tentativas a função retorna -1 indicando que a ligação não foi estabelecida.
* caso exista resposta por parte do recetor, mas esta não seja afirmativa o emissor reenvia a trama SET até que a resposta seja afirmativa.

LLWRITE

*int LLWRITE(int fd, unsigned char\* msg, int\* length);*

A função llwrite é responsável pelo envio dos pacotes de Informação. Assim esta função recebe como argumento, entre outros, a mensagem a ser enviada (pacotes de controlo + dados) e o respetivo tamanho. Deste modo, com recurso à função *create\_package()* esta cria a trama com os cabeçalhos necessário (flags, endereço, controlo(c1), bcc1, bcc2 e respetivo stuffing) e envia a mesma através da porta de série. O envio das tramas de informação é similar ao estabelecimento da ligação por parte do emissor, ou seja, a mensagem é enviada e fica-se a aguardar resposta sendo reenviada a mesma quando for atingido o tempo de timeout. Caso a resposta recebida seja o comando RR(C1 = 0x05/0x85), a mensagem foi recebida corretamente pode-se enviar o próximo pacote; caso a resposta seja o comando REJ(C1 = 0x01/0x81) o último pacote é reenviado. A função retorna TRUE em caso de sucesso e FALSE caso contrário.

LLREAD

*unsigned char\* LLREAD(int fd, int\* length)*

A função llread é responsável pela leitura dos pacotes de informação provenientes da porta de série. Como tal, esta função ao ser invocada lê a informação, um char de cada vez, verificando o início e fim da trama(FLAG) e coesão do cabeçalho (BCC1), através da chamada a função *state\_machine().* São também invocadas aqui as funções de introdução de erros aleatórios no cabeçalho e dados. Caso não ocorra nenhum erro no BCC1 esta função verifica se a trama recebida se trata de um DISC (Sinal de terminação de envio das tramas de informação). Em caso afirmativo é invocado LLCLOSE para se proceder ao protocolo de fecho da porta de série. Caso contrário, é verificado se não se trata de um duplicado (verificando o valor do campo de controlo), é feita a remoção do cabeçalho (*remove\_head\_msg\_connection()*) e é feito o destuffing e respetiva verificação do valor de BCC2(*verify\_bcc2()*). Posteriormente é enviada, através da função *send\_response(),* a resposta de acordo com os dados recebidos, segundo a seguinte lógica:

* Trata-se de um duplicado: é enviado RR
* Não é duplicado e BCC2 está correto: é enviado RR
* Não é duplicado e BCC2 está errado: é enviado REJ

Caso seja enviado RR será o valor correspondente a RR do complemento do tipo de trama recebida, ou seja, caso se receba uma trama 1 (C = 0x40) envia-se RR correspondente a RR0(0x05). Já no caso de ser enviado REJ será o valor correspondente a REJ do tipo de trama recebida, ou seja, caso se receba uma trama 1, envia-se REJ correspondente a REJ(0x81).

LLCLOSE

*void LLCLOSE(int fd, int type)*

A função LLCLOSE é responsável por terminar a ligação através da porta de série. Esta função tem comportamentos distintos caso se trate do recetor ou do emissor. Assim, no caso do recetor, esta função é chamada na função LLREAD ao ser encontrado um DISC. Após isso o recetor através da chamada à função *send\_disc()* e análise da respetiva resposta completa a confirmação de fecho e termina a ligação. Já no caso do emissor, esta função é chamada após o envio e receção da respetiva resposta da trama END. É chamada a função send\_disc() que, tal como para o recetor envia uma trama DISC e espera uma resposta. Seguidamente, e caso haja sucesso, a ligação é também terminada no lado do emissor.

Protocolo de aplicação

A camada de aplicação é a camada de mais alto nível implementada neste projeto. Esta camada é responsável pelas seguintes funcionalidades:

* Envio/receção dos pacotes de dados e de controlo
* Leitura/escrita do ficheiro a enviar/recebido

Uma vez que é nesta camada que se inicia todo o processo, a função main tem dois blocos e escolhe qual executar dependendo se se trata do emissor ou do recetor, parâmetro este que é recebido como argumento da função. Em ambos os casos vai ser executada no inicio a função LLOPEN para abrir e configurar a porta de série.

Desta forma existem funções separadas para ambos os modos, emissor e recetor.

Emissor

O emissor após a execução da função LLOPEN, preenche a estrutura file com os dados do ficheiro necessários à abertura e leitura do mesmo.

O restante funcionamento do programa divide-se em 3 partes fundamentais, e sequenciais:

* Criação e envio da trama START utilizando as funções *create\_STARTEND\_packet()* e *send\_message()*, sendo que esta ultima função envia a trama para a camada de ligação através da chamada à função LLWRITE
* Leitura e envio dos vários pacotes de dados provenientes do ficheiro. Para tal, é chamada a função handle\_readfile() que lê o ficheiro e chama a função send\_message() para envio da informação lida
* Por fim, é criada e enviada, de forma idêntica à START, a trama END

Recetor

Já no caso do recetor apenas é chamada a função get\_message() que, após executar a função LLREAD analisa os dados obtidos e verifica qual dos seguintes tipos de pacotes se trata, retornando o pacote lido:

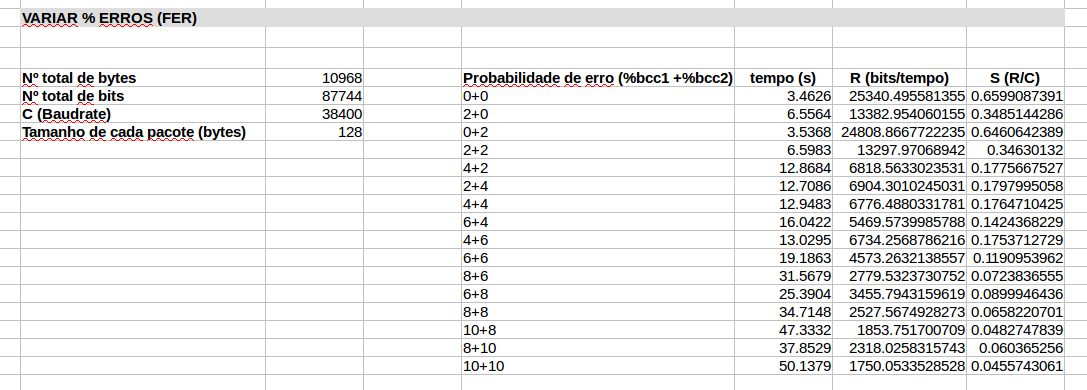
* Mensagem START: chama a função start\_message(), que preenche os dados da estrutura file com os respetivos dados para a criação do ficheiro, criando-o.
* Pacotes de Informação: chama a função get\_only\_data() que remove o cabeçalho do pacote, e posteriormente chama a função handle\_writefile() que escreve os dados para o ficheiro de destino.
* Mensagem END: chama a função verify\_end() que verifica se o tamanho do ficheiro recebido corresponde ao tamanho recebido na trama final.
* Se a trama recebida for um DISC ou nula, nada é analisado.

O recetor executa a função get\_message() enquanto esta não retornar uma trama DISC, sinal de que a execução do programa terminou e a porta e série já foi fechada.

Validação

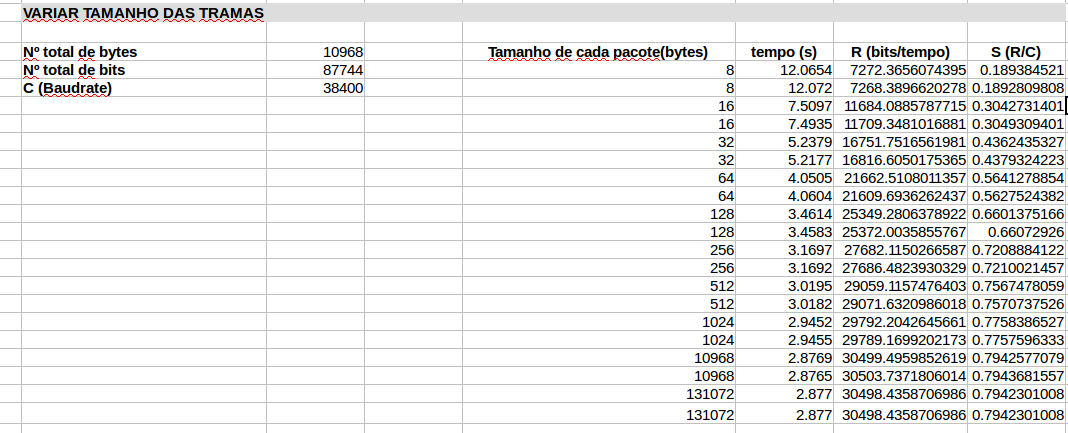
Nesta secção são abordados os diversos testes realizados. Foram testados no total três tipos de testes para o mesmo ficheiro, pinguim.gif sugerido no guião do trabalho.

O primeiro tipo de testes foram aplicados em volta da percentagem de erros do BCC1 e do BCC2.

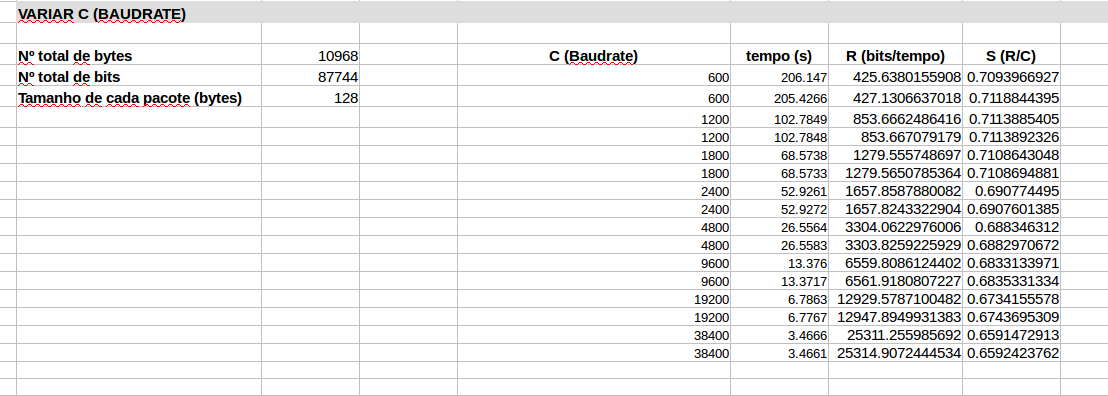


Como é possível verificar, quanto maior for a percentagem de erros dos BCCs maior o tempo de transmissão do ficheiro, sendo o BCC1 que causa maior impacto.

O segundo tipo de testes realizados tiveram em conta os tamanhos dos pacotes enviados.

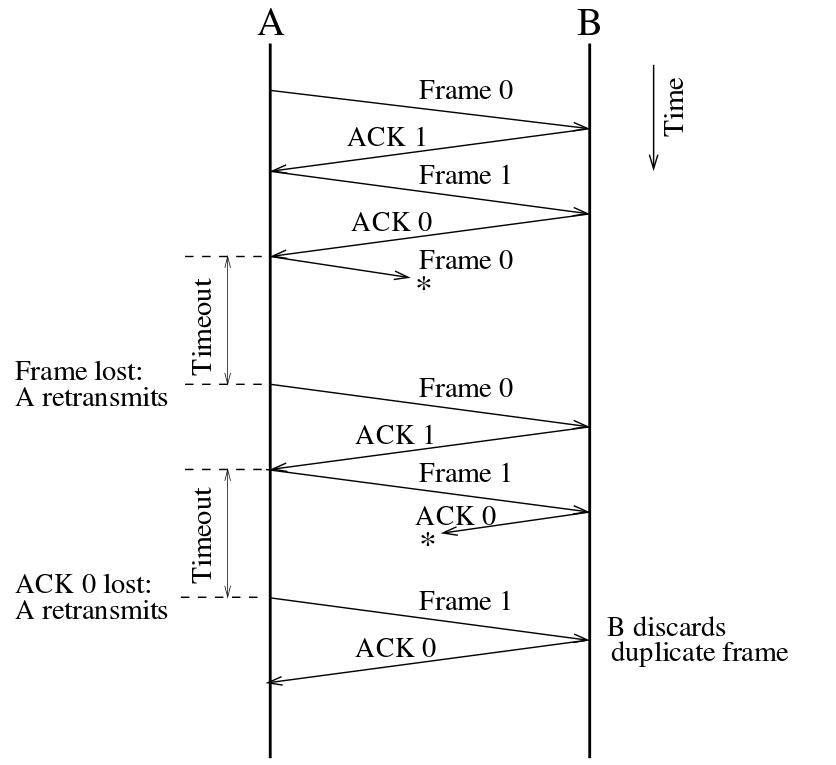
 Para diferentes tamanhos de pacotes(em bytes) verificamos que quanto maior forem os tamanhos menor o tempo de transmissão, isto porque menos pacotes são enviados.

O terceiro e último tipo de testes realizados foram aplicados a diferentes valores de BAUDRATE.



Para diferentes valores do BAUDRATE, quanto maior for menor o tempo de transmissão do ficheiro pois é permitido um maior envio de informação por segundo.

Eficiência do protocolo de ligação de dados

 No mecanismo Stop and Wait de ARQ(Automatic Repeat Request), o emissor após enviar uma trama I

espera sempre por uma resposta do recetor, se o recetor confirmar a trama enviada o emissor procede e envia uma nova trama, caso contrário o emissor é obrigado a reenviar a trama I até que esta seja aceite pelo recetor. O recetor ao receber uma trama I verifica se deteta algum erro,caso haja erros envia ao emissor mensagem de rejeição NACK, caso contrário manda confirmação ACK ao emissor.

De forma o programa reconhecer se a trama recebida é uma nova ou apenas uma retransmissão, as tramas I e ACK devem ser numeradas do estilo I(0),I(1) e ACK(0)

,ACK(1),em que ACK(i) indica ao emissor que o recetor está à espera de uma trama I(i).

Este mecanismo necessita do uso de um alarme com timeout de forma a resolver problemas como a perda das tramas I,ACK ou NACK.

O projeto realizado usa este mecanismo de ARQ, sempre que o emissor manda uma trama espera sempre uma resposta com utilização de um alarme de três segundos. Por exemplo, se o emissor enviar uma trama de informação (quer seja Start,End ou normal) espera sempre por um RR (com numeração 0 ou 1 dependendo da numeração da trama enviada) de resposta positiva ou REJ de resposta negativa, isto não se aplica somente as tramas de informação (tramas de supervisão de certa forma usam também este mecanismo) embora apenas estas necessitam necessariamente de serem numeradas.

Tendo em conta os testes efetuados, após os cálculos do débito recebido bits/s e da eficiência de cada teste podemos ver através dos gráficos apresentados