

# Faculdade de Engenharia da Universidade Do Porto

## Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

1º Trabalho Laboratorial

*Alunos:*

João Gama Amaral

João Nuno Ferreira

Duarte Carvalho

25 de Outubro, 2018



## Índice

1. Sumário
2. Introdução
3. Arquitetura
4. Estrutura do Código
5. Casos de uso principais
6. Protocolo de ligação lógica
  - llopen
  - llwrite
  - llread
  - llclose
7. Protocolo de aplicação
  - Envio e receção de pacotes de controlo
  - Envio e receção de pacotes de dados
8. Validação
9. Eficiência do protocolo de ligação de dados
  - Análise da variação do tamanho da trama e capacidade de ligação
  - Análise do impacto da FER na eficiência e capacidade de ligação
  - Análise da variação do tempo de propagação na eficiência
10. Conclusões

## 1. Sumário

O trabalho laboratorial tem como objetivo aplicar na prática todos os conhecimentos teóricos lecionados. O trabalho consiste no envio de dados de um computador para outro através do uso de uma porta de série.

## 2. Introdução

O objetivo do trabalho laboratorial é conseguir implementar um mecanismo de transferência de dados, recorrendo a uma porta de série. É utilizada uma porta de série, visto que é dos métodos mais básicos, permitindo assim compreender como é realizada a transferência de dados.

## 3. Arquitetura

O projeto segue uma arquitetura onde nos baseamos no princípio da independência entre camadas. Ao nível da camada de aplicação é realizada a leitura e escrita no ficheiro. Ao nível da camada de ligação de dados implementámos o protocolo de ligação de dados. As camadas estão relacionadas, pois a camada de aplicação depende da ligação de dados.

## 4. Estrutura de código

Na camada de ligação existem quatro funções principais, sendo elas: **llopen**, **llwrite**, **llread**, **llclose**. A função **llopen** é responsável pela ligação entre os dois computadores, enviando as mensagens de SET e UA. A função **llwrite** trata do envio dos pacotes de dados ou de controlo que são passados como argumento, efetuando **stuffing** e colocando-os em tramas de informação. Após envio, aguarda uma mensagem de sucesso. Caso contrário, reenvia a informação. A função **llread** aguarda uma trama de informação. Após receber a trama, realiza **destuffing** e lê os dados que constituem a trama, escrevendo a informação no ficheiro.

Na camada de aplicação existem 3 funções principais, sendo elas: **sendFileInfo**, **sendFileData** e **readDataPackets**. As funções **sendFileInfo** e **sendFileData** são responsáveis pela organização dos dados em pacotes de controlo (**sendFileInfo**) e pacotes de dados (**sendFileData**). Os pacotes de controlo criados são de início (start) e fim (end), permitindo ao programa saber quando está a iniciar a transmissão de dados e quando está a terminar. Os pacotes de dados é toda a informação que vai ser enviada entre o pacote de controlo inicial e final, sendo enviados pela função **sendFileData**.

A função **readDataPackets** chama a função **llread** para poder iniciar o tratamento dos dados recebidos e escrever no ficheiro.

## 5. Casos de uso principais

Neste trabalho laboratorial, existem dois casos de uso, sendo estes a escolha do ficheiro a enviar, incluindo o tempo de timeout e número máximo de tentativas até dar timeout, e a transmissão de um ficheiro entre os dois computadores.

Ordem:

1. Escolha do valor de timeout e número de tentativas no transmissor e receptor.
2. Escolha do ficheiro a enviar, seleccionando o seu path no transmissor.
3. Escolha do tamanho das tramas de informação a serem enviadas no transmissor.
4. Estabelecer ligação.
5. Transmissor envia os dados.
6. Recetor recebe os dados.
7. Recetor envia a resposta ao transmissor do sucesso ou insucesso da transmissão
8. Em caso de sucesso, o recetor escreve no ficheiro de output os dados recebidos.
9. Terminação da ligação.

## 6. Protocolo de ligação lógica

A camada de ligação lógica de dados é responsável por várias funcionalidades, tais como:

- Estabelecer e terminar uma ligação.
- Criar e enviar comandos pela porta de série.
- Criar, enviar e receber mensagens pela porta de série.
- Stuffing e destuffing de pacotes de dados.

### llopen

```
int llopen(int path, int mode) {
```

A função llopen é responsável por estabelecer a ligação entre os computadores através da porta de série. O emissor invoca esta função, e com a ajuda de uma **StateMachine**, envia o comando SET, aguarda resposta do recetor, que envia o comando UA, com a ajuda da **StateMachine**. Caso a resposta não chegue e o tempo de timeout seja excedido, utilizando um alarme, é feita uma nova tentativa e o comando SET é reenviado, repetindo este ciclo até ao número de tentativas máximo ser excedido ou até ser recebido o comando UA. O recetor aguarda a receção do comando SET e após a receber, envia o comando UA, sendo a ligação finalmente estabelecida.

## llwrite

```
// Cria uma trama de informacao
int llwrite(int fd, unsigned char* buffer, int length){
```

A função **llwrite** recebe um buffer que corresponde aos pacotes de dados ou de controlo. Após receber este buffer, é efetuado o cálculo do BCC2, feita a operação de **stuffing** e criação da trama de informação. Depois de a trama estar criada, é enviada, recorrendo à função **write\_frame**. Após envio, aguarda uma resposta que, se não for recebida no intervalo de tempo timeout, realiza-se feita uma nova tentativa de envio. Quando uma resposta é recebida, existem duas possibilidades, RR ou REJ. Se for RR, então a mensagem foi transmitida corretamente, caso seja REJ a mensagem não foi transmitida corretamente, sendo, então, retransmitida.

## llread

```
unsigned char * llread() {
```

A função **llread** lê, através de uma chamada à função **read**, toda a trama que lhe é enviada, byte a byte. Cada um deste é depois enviado para a máquina de estados que trata dos dados recebidos. Esta máquina interpreta cada byte, confirmando se está correto, e, caso esteja, armazena-o. Para além disto, a máquina vai calculando o BCC2 e deteta caracteres de escape, alterando-os.

Voltando à função **llread**, esta compara o valor determinado para o BCC2 e compara-o com aquele que era esperado. Caso sejam encontrados erros no cabeçalho ou no BCC2, esta é logo ignorada e envia-se uma trama de supervisão, informando que a trama foi rejeitada, o que causa um reenvio desta.

Se toda a trama estiver correta, a **llread** envia uma trama de supervisão onde indica que a trama estava bem e pode ser enviada a próxima. Por fim, é retornada apenas a parte da trama que possui a informação, que foi armazenada na **StateMachine** do recetor.

## llclose

```
// Retorna 1 em caso de sucesso
// Retorna -1 em caso de erro
int lllclose(int path, int mode) {
```

A função **llclose** termina a ligação entre os dois computadores. O recetor aguarda o comando DISC, utilizando uma **StateMachine**. Ao receber o comando este é reenviado e aguarda pela receção do comando UA. O emissor envia o comando DISC, aguarda a receção do comando DISC e, finalmente, envia o comando UA. Após estes comandos serem corretamente enviados e recebidos, a ligação é finalmente terminada.

## 7. Protocolo de aplicação

A application layer é a camada de mais alto nível do programa, responsável pelas seguintes funcionalidades:

- Envio/receção de pacotes de controlo
- Envio/receção de pacotes de dados

Os pacotes do protocolo de aplicação podem ser divididos em dois tipos: Controlo e Dados. Sendo que os de controlo podem ser ainda caracterizados como pacotes de controlo inicial e final.

Para enviar os pacotes de controlo e dados foi usada uma função mais pequena, **sendFile()**.

```
void sendFile() {  
    int sizeFile;  
    sizeFile = sendFileInfo(getFileName(app), START_C);  
    sendFileData(sizeFile, getSelectedFrameSize(app));  
    sizeFile = sendFileInfo(getFileName(app), END_C);  
}
```

### Envio e receção de pacotes de controlo

Os pacotes de controlo são gerados pela função **sendFileInfo**. Esta função é responsável pela leitura de informação do ficheiro como: nome do ficheiro e tamanho do ficheiro. Os pacotes de controlo de início e fim apenas diferem no Byte **C**, onde o início corresponde a 0x02 e o fim corresponde a 0x03.

Após o pacote de controlo ser gerado, byte a byte, recorreremos à função **llwrite** da camada inferior para poder ser gerada a trama de informação com o pacote de controlo e ser enviada para leitura. A sua leitura é efetuada na função **llread** da camada inferior, que identifica se o pacote de controlo é de início ou de fim.

### Envio e receção de pacotes de dados

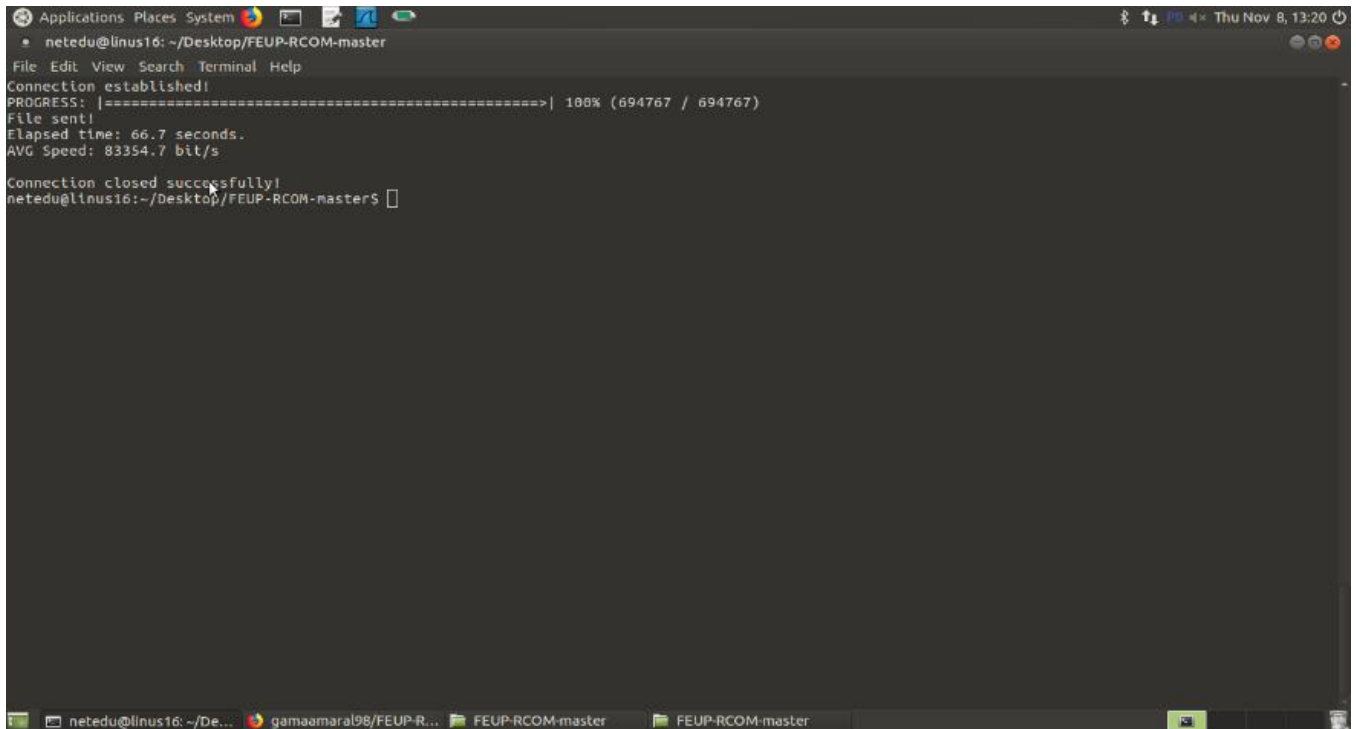
Os pacotes de dados são preparados pela função **sendFileData**. Esta função é responsável pela leitura dos bytes do ficheiro, sendo possível variar o tamanho dos bytes lidos para reduzir ou aumentar o tamanho das tramas. A função **sendFileData** recebe o tamanho do ficheiro e o tamanho das tramas que pretendemos enviar.

Após definir os bytes que constituem o pacote de dados e juntar com os bytes lidos num array, é chamada a função **llwrite** da camada inferior que é responsável pela colocação do pacote de dados numa trama de informação. Para a leitura depois do envio, é utilizada a função **llread** que identifica o pacote de dados a partir do byte **C**, processando assim os dados que foram enviados a partir da leitura do ficheiro para reconstruir este mesmo.

## 8. Validação

Para testar o programa, efetuaram-se transferências com ficheiros diferentes e com tramas de informação de diferentes tamanhos. Todas as transferências foram concluídas com sucesso, mesmo quando se interrompia a porta de série ou quando se interferia criando ruído a partir do envio de dados.

De seguida mostramos um exemplo de uma transmissão de uma imagem de 4K com tramas de 256 bytes:



```
netedu@linus16: ~/Desktop/FEUP-RCOM-master
File Edit View Search Terminal Help
Connection established!
PROGRESS: |=====| 100% (694767 / 694767)
File sent!
Elapsed time: 66.7 seconds.
AVG Speed: 83354.7 bit/s
Connection closed successfully!
netedu@linus16:~/Desktop/FEUP-RCOM-master$
```

Fig. 1 - Exemplo do estado do terminal após a transferência de uma imagem.



Fig. 2 - Figura 4K transferida.

## 9. Eficiência do protocolo de ligação de dados

**Nota:** A tabela com os valores gerais pode ser encontrada em anexo (tabela\_Geral.png).

### Análise da variação do tamanho da trama e capacidade de ligação.

Análise dos resultados obtidos apenas mudando o tamanho das tramas de informação e da capacidade de ligação.

**Nota:** A tabela com estes valores pode ser encontrada em anexo (tabela\_C.png).

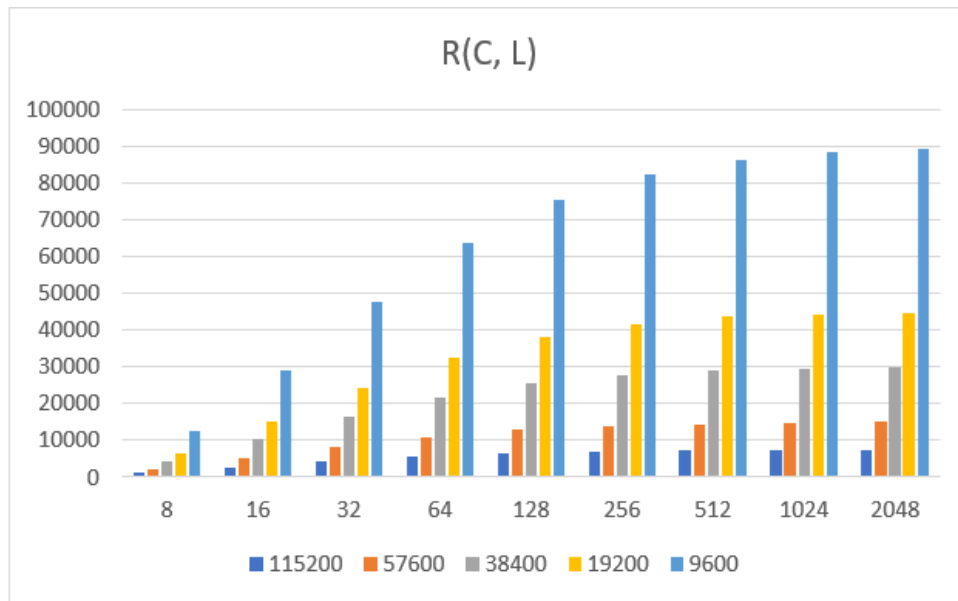


Fig. 3 –  $R(C, L)$ .

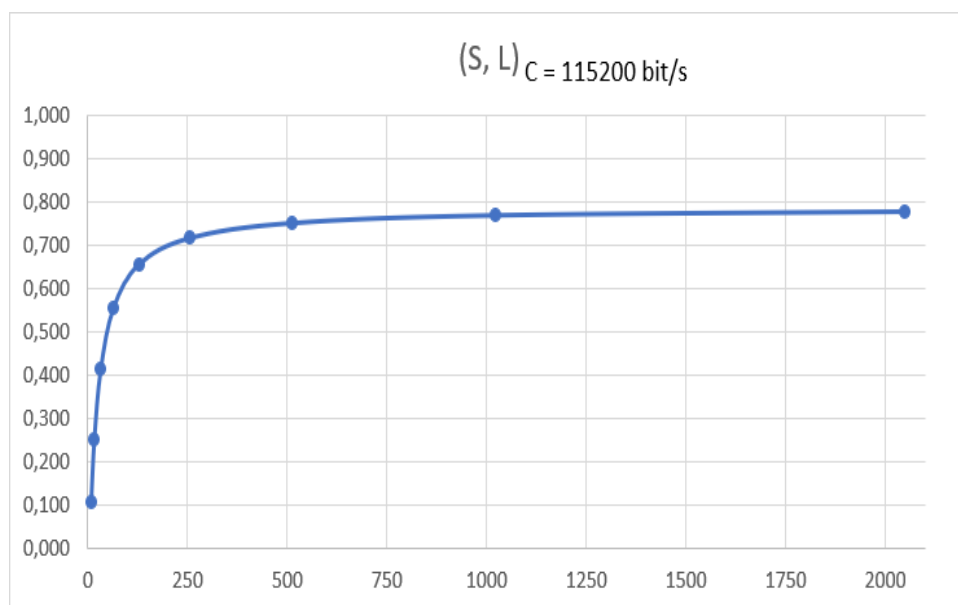


Fig. 4 –  $(S, L)$ .



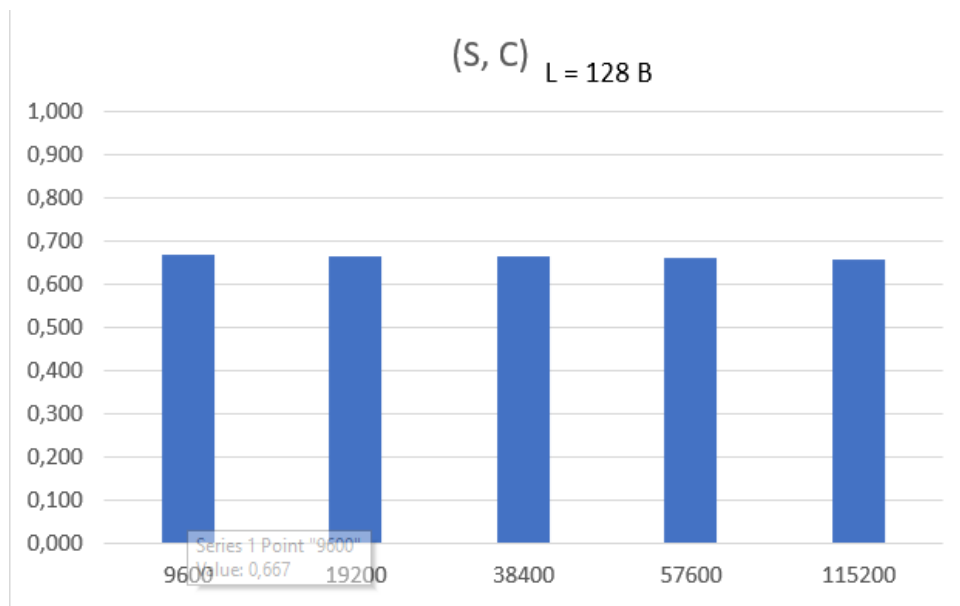


Fig. 5 – (S, C).

### Análise do impacto da FER na eficiência do protocolo.

**Nota:** A tabela com estes valores pode ser encontrada em anexo (tabela\_FER.png).

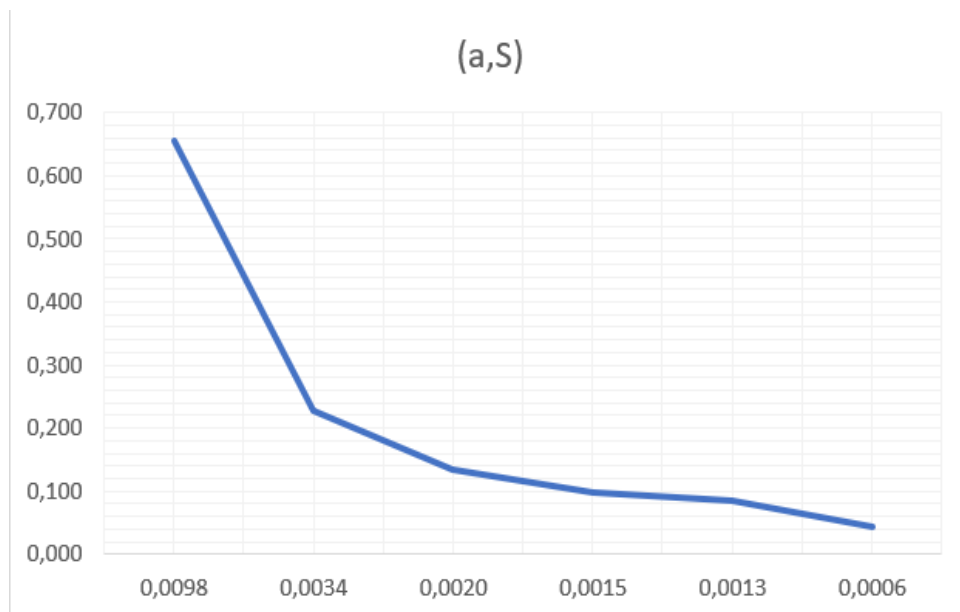


Fig. 6 – (a, S).

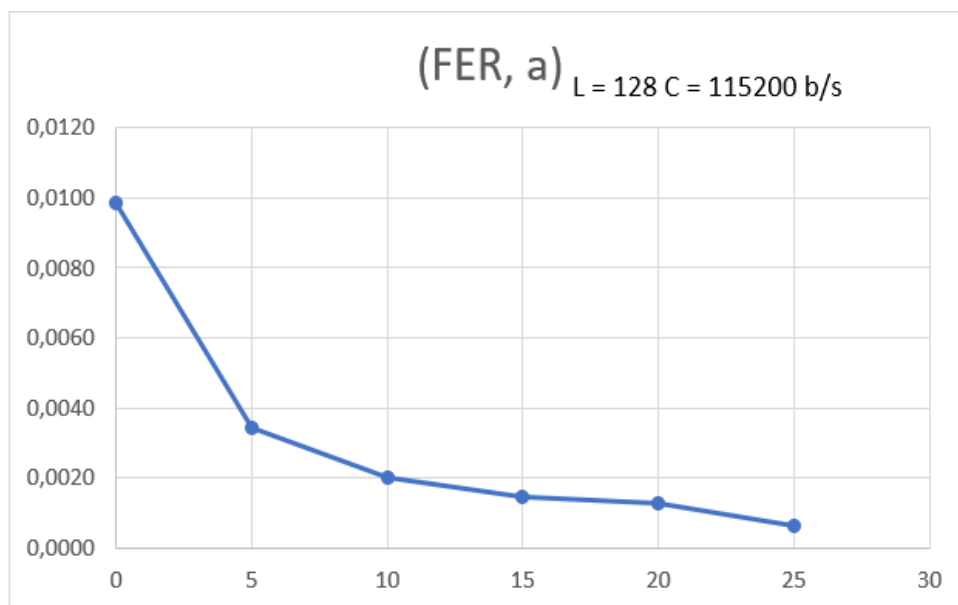


Fig. 7 – (FER, a).

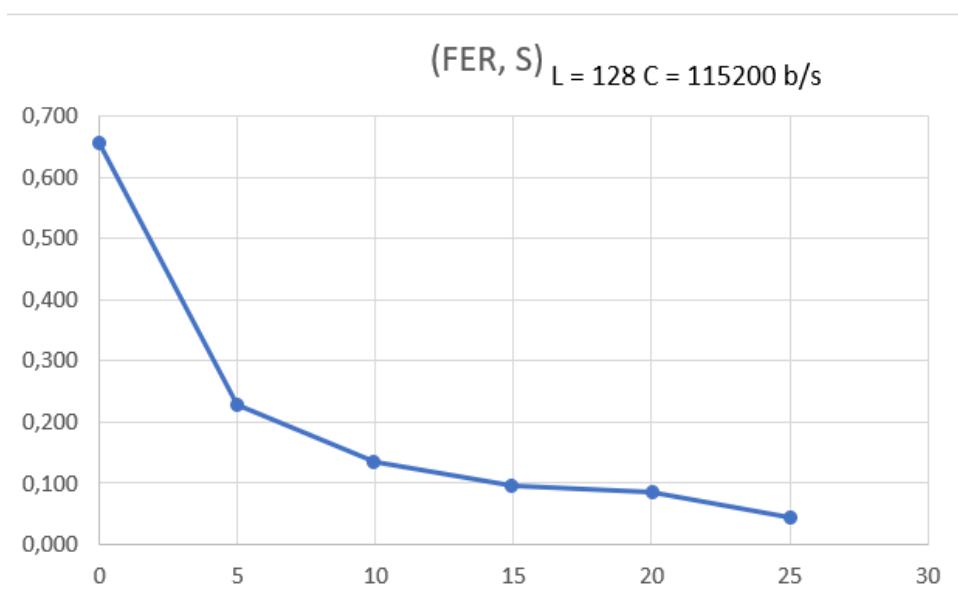


Fig. 8 – (FER, S).

## Análise da variação do tempo de propagação na eficiência.

**Nota:** A tabela com estes valores pode ser encontrada em anexo (tabela\_T\_PROP.png).

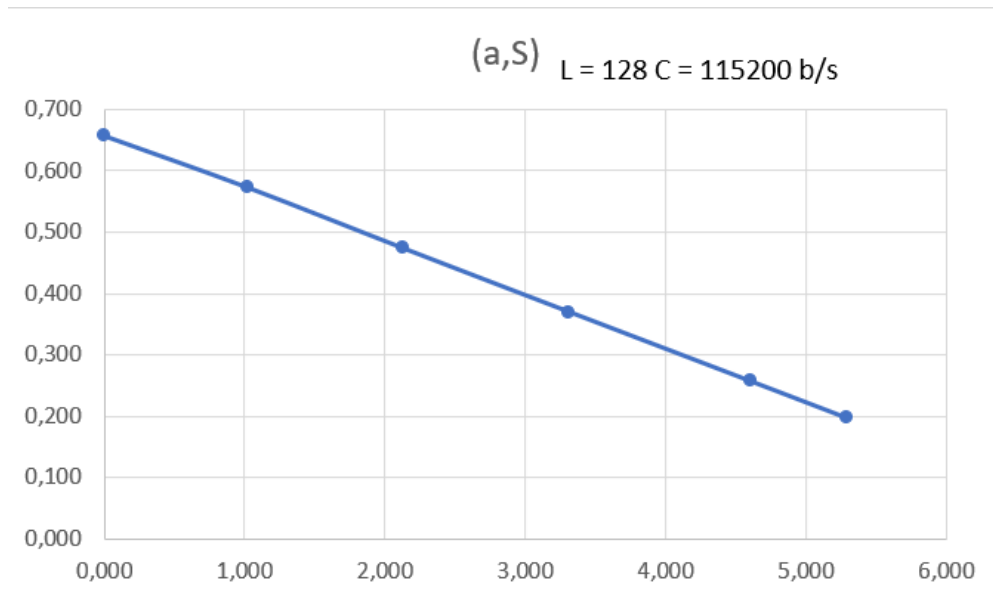


Fig. 9 –  $(a, S)$ .

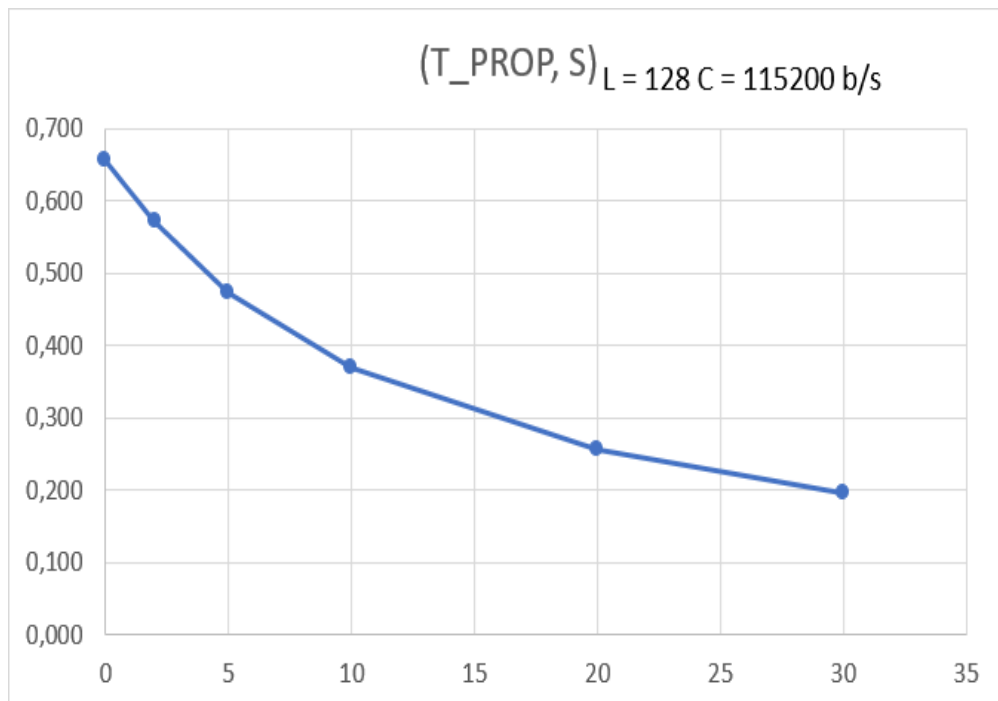


Fig. 10 –  $(T\_PROP, S)$ .

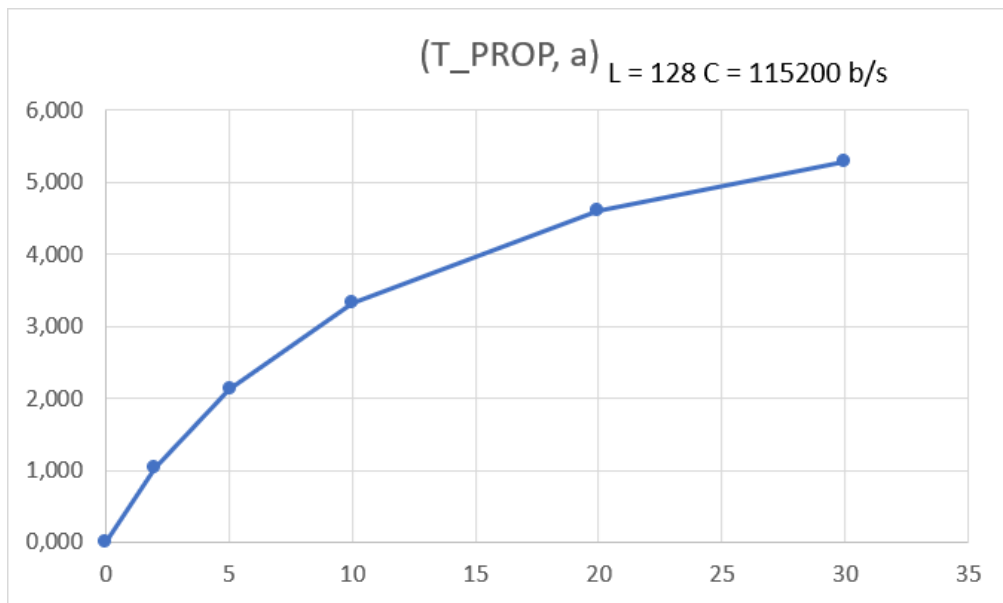


Fig. 11 – (T\_PROP, a).

## 10. Conclusões

Com base neste trabalho laboratorial, podemos concluir que, com o aumento do tamanho das tramas de informação enviadas pelo transmissor, a eficiência da transferência de dados também aumenta.

Podemos também constatar que, assumindo um tempo de propagação constante, apenas o tempo de processamento da trama é que irá fazer variar o valor de  $a$ . Quanto menor for o valor deste parâmetro, maior será a eficiência do protocolo.

Para além disto, conseguimos entender que, aumentando o  $c$ , que é a capacidade da ligação (**baudrate**), apesar de aumentar a velocidade de transmissão dos pacotes, não causa alterações significativas na eficiência do processo.

Por fim, e um pouco como seria de esperar, o rendimento do programa é claramente reduzido conforme se vai aumentando o tempo de propagação dos dados ou ainda a percentagem de erros por trama.