

*Protocolo de Ligação de Dados*

Relatório

Francisco Teixeira Lopes

ei11056@fe.up.pt

Nelson André Garrido da Costa

up201403128@fe.up.pt

Índice

Sumário 1

Introdução 2

Arquitetura 2

Estrutura do código 3

Casos de uso principais 4

Protocolo de ligação lógica 4

Protocolo de aplicação 4

Validação 5

Eficiência do protocolo de ligação de dados 5

Conclusões 7

Sumário

Este trabalho insere-se no âmbito da Unidade Curricular Redes de Computadores, do Mestrado Integrado em Engenharia Informática e da Computação da Faculdade de Engenharia da Universidade do Porto, abordando conteúdos programáticos, abordados nas aulas teóricas.

Entre eles, destacam-se o Nível Físico – *Physical Layer* -, a Camada de Ligação de Dados - Data *Link Layer* – e a Camada de Aplicação – *Application Layer*. A proposta de trabalho teve como base a transferência, por cabo, de ficheiros entre dois computadores, através das suas portas de série.

Este relatório prende-se na consolidação do trabalho e conhecimento adquirido até à data, fazendo-se a união entre as componentes prática e teórica sobre as quais se debruça o projeto apresentado, que se pretende mostrar bem entendidas e implementadas.

**Introdução**

O projeto foi trabalhado no contexto da Unidade Curricular de Redes de Computadores, inserida no plano do 3º ano do Mestrado Integrado em Engenharia Informática e Computação.

Os objetivos principais deste trabalho prendem-se essencialmente na implementação do protocolo de ligação de dados, através de uma aplicação responsável pela transferência de ficheiros entre dois computadores - o emissor e o recetor -, recorrendo às suas portas de série e combinando características de protocolos já existentes, como a transparência e a transmissão organizada em tramas – de informação, controlo e não numeradas - e utilizando integralmente transmissão em série assíncrona.

De modo a que o processo fosse feito em conformidade com estes objetivos, foram incluídos, na aplicação, mecanismos de deteção de erros como, por exemplo, dados enviados em duplicado ou falhas de conexão entre as máquinas, e respetiva resposta, assim como suporte para dois tipos de pacotes enviados pelo emissor: os de dados (com porções do ficheiro) e os de controlo para iniciar – *start* - e finalizar a transmissão – *end*.

Este relatório, sendo o *pós-laboral* do trabalho prático, serve como suporte à sua fácil compreensão e está estruturado da seguinte forma:

* **Introdução:** indicação dos objetivos do trabalho e do relatório; descrição da lógica do relatório;
* **Arquitetura:** blocos funcionais e interfaces;
* **Estrutura do código:** APIs consultadas e utilizadas, principais estruturas de dados usadas, principais funções e sua relação com a arquitetura;
* **Casos de uso principais;**
* **Protocolo de ligação lógica:** identificação dos principais aspetos funcionais e descrição da estratégia de implementação destes aspetos;
* **Protocolo de aplicação:** tal como no ponto anterior, explana-se a maneira de pensar nesta fase;
* **Validação:** testes realizados e quantificação de resultados.
* **Eficiência do protocolo de ligação de dados:** caracterização estatística da eficiência do código desenvolvido comparativamente a uma caracterização teórica;
* **Conclusões:** considerações sintéticas finais sobre o trabalho realizado.

**Arquitetura**

A aplicação está organizada em duas camadas:

* **Camada do protocolo de ligação de dados:** funções do protocolo, especificando funcionalidades de sincronismo, numeração de tramas, controlo da transferência, tratamento de erros, abertura e definição das propriedades da porta de série – LinkLayer.c/LinkLayer.h
* **Camada da aplicação:** funções de escrita e leitura do ficheiro, através do processamento e envio de pacotes de controlo e dados, sendo fundamental para o funcionamento do programa. – AppLayer.c/AppLayer.h

Em relação à interface da aplicação, todos os valores globais às camadas e estruturas funcionais estão incluídos no ficheiro **Globals.h,** como, por exemplo, o *baudrate*, macros que definem valores predefinidos, como as *FLAGS*, tamanhos máximos para o nome e tamanho do ficheiro a transmitir, entre outros. É também apresentada, na linha de comandos, ao utilizador recetor, uma barra de progresso relativa à transferência com a respetiva percentagem. Mensagens *log* estão previstas, se o utilizador ativar a macro EN*ABLE\_DEBUG* no ficheiro suprarreferido.

**Estrutura do código**

* **Camada de ligação de dados:**

Utiliza-se uma ***struct*** para a sua representação, onde é guardada, entre outra informação, a porta de série, o ***baudrate***, um número de **sequência** e a última resposta, que determina qual a trama a esperar receber ou enviar de seguida, o tamanho, variáveis de ***timeout****,* para o contador de segundos de espera, e as configurações da ***termios***.

Das funções desta camada, destacam-se a ***llopen()*** - que tenta estabelecer a ligação -, ***llclose\_transmit()***/***llclose\_receive()*** - que tentam terminar a ligação -, ***llread()*** – que lê e valida pacotes de dados -, ***llwrite()*** – que processa e envia pacotes de dados.

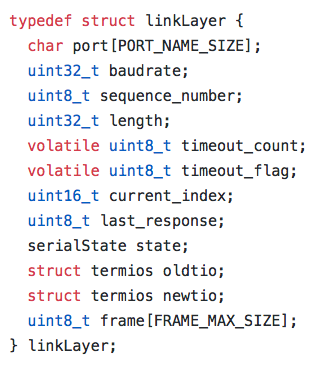
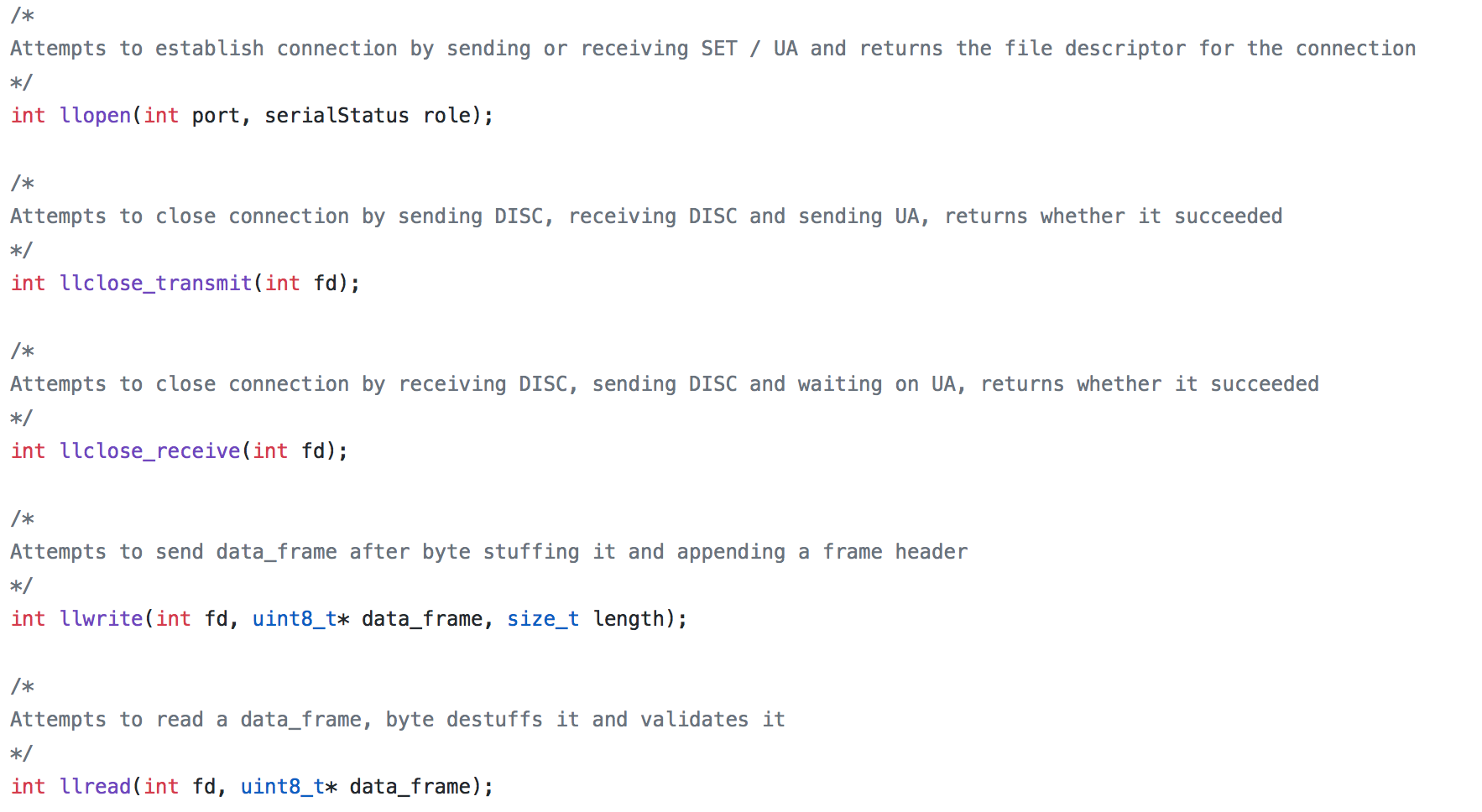


Fig1: código da *struct* da camada de ligação de dados.

Fig2: código dadeclaração das principais funções da camada de ligação de dados.

* **Camada de aplicação:**

Utiliza-se uma ***struct*** para a sua representação, onde é guardada, entre outras informações, o ***file descriptor*** da porta de série, o **tamanho** e o **nome** do ficheiro a transferir, um **contador** de envio e o **estado** do envio.

Das principais funções afetas a esta camada, destacam-se a ***alsend()*** – que abre o ficheiro a enviar e o processa em pacotes de dados-, ***alreceive()*** – que tenta ler e guardar o ficheiro no computador recetor -, ***createSEPacket()***/***createDataPacket()*** – que geram os pacotes de controlo e dados resultantes do ficheiro a enviar/receber.

Fig4: código dadeclaração das principais funções da camada de aplicação.

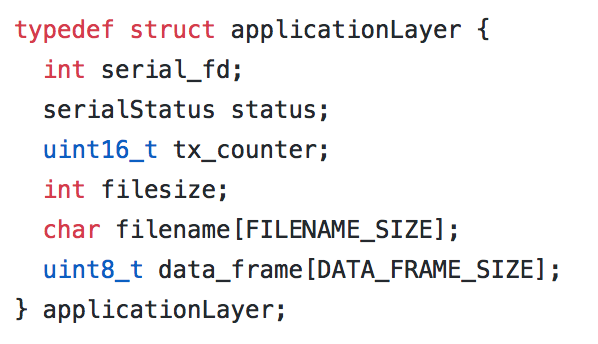
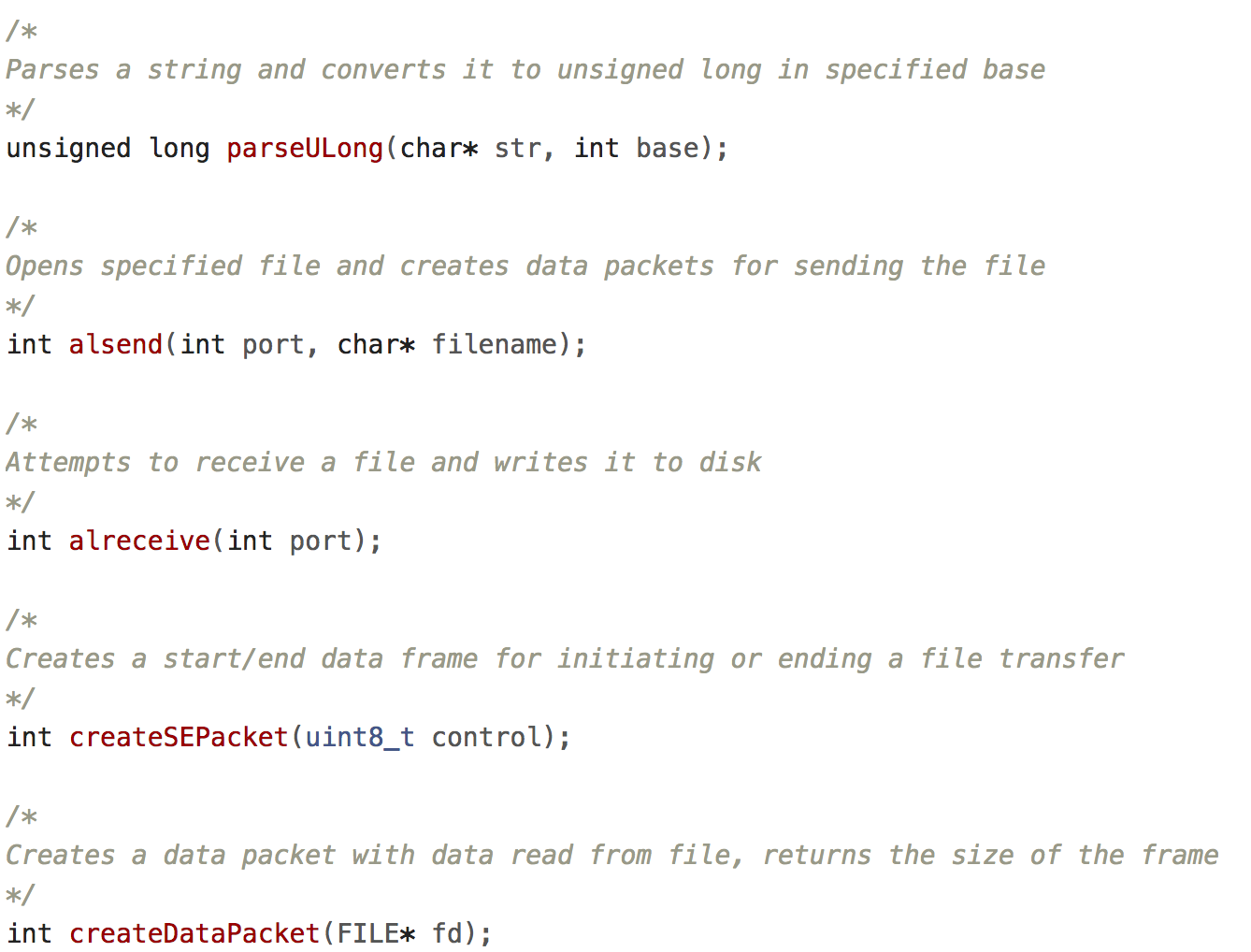


Fig3: código da *struct* da camada de aplicação.

**Casos de uso principais**

Os casos de uso do programa incluem aqueles que são responsáveis pela transferência dos ficheiros:

* Configuração da ligação com escolha do papel de emissor ou recetor e identificação do ficheiro, através do seu nome, pelo emissor.
* Estabelecimento da ligação.
* Computador emissor envia o ficheiro particionado em pacotes.
* Computador recetor recebe os dados e escreve-os no ficheiro a guardar.
* Terminação da ligação e fecho dos ficheiros até ao momento abertos.

**Protocolo da ligação lógica**

No que diz respeito ao protocolo da ligação lógica, implementa-se as principais funções relativas à camada de ligação lógica.

Ao ser invocada pela máquina emissora, *llopen()* envia o comando *SET* para tentar estabelecer a ligação, esperando pela resposta do recetor - a trama *UA* - que, se não for recebida dentro do período de tempo predefinido – 3 segundos -, o faz reenviar o *SET*. Este procedimento é replicado um número também predefinido de vezes – 3 vezes – que, se excedido, faz terminar o programa sem sucesso. Uma vez concluído este processo, a ligação é estabelecida e a execução continua.

De seguida, *llwrite()* recebe, como argumento, a informação a transmitir, que será usada para a partição em tramas, numeradas com número de sequência – 0 ou 1 – e que serão enviadas ao recetor. O programa fica, então, à espera duma resposta deste último, mais uma vez, durante um período de tempo, que findo, o faz reenviar a trama, e repetir o processo até se exceder as 3 tentativas – terminação sem sucesso. Quando se obtém, como resposta, a trama *RR*, o recetor recebeu a trama enviada sem erros e a função termina com sucesso. Se, pelo contrário, a resposta for uma trama *REJ*, o processo de envio repete-se.

A função *llread()* corre a par da *llwrite()*, na máquina recetora, e permanece em ciclo até receber uma trama, que é validada, recorrendo a uma máquina de estados que controla se os bytes recebidos são os esperados. Casos todos os campos estejam válidos, a função envia a trama *RR* e, caso contrário, envia a *REJ*, com os números de sequência da trama recebida. Por outro lado, se a função receber o comando *DISC*, a ligação deve ser terminada.

Por último, a função llclose() é invocada pelo emissor, que envia o comando *DISC*, indicando o fim da ligação, esperando receber, do recetor, um comando igual, que quando recebido, o faz enviar o comando *UA* – unumbered acknowledge.

**Protocolo de aplicação**

Relativamente ao protocolo de aplicação, é feita a leitura e a conversão dos argumentos passados na linha de comandos, nomeadamente o papel de cada computador, a porta de série e o nome do ficheiro a transferir.

De seguida, o computador emissor invoca *alsend()*, onde lê o ficheiro a enviar e o particiona em pacotes de dados.

O recetor, por sua vez, invoca *alreceive()*, que recebe os pacotes de dados, escrevendo o ficheiro internamente.

**Validação**

Para validar a eficácia do projeto desenvolvido, recorreu-se a vários métodos, que correram com sucesso:

* Interromper a ligação, enquanto a informação é transmitida.
* Geração de picos de tensão no cabo que liga as portas de série.
* Usar valores de *baudrate* e tamanho de pacotediferentes.
* Transmissão de diferentes ficheiros, com diferentes tamanhos.

**Eficiência do protocolo de ligação de dados**

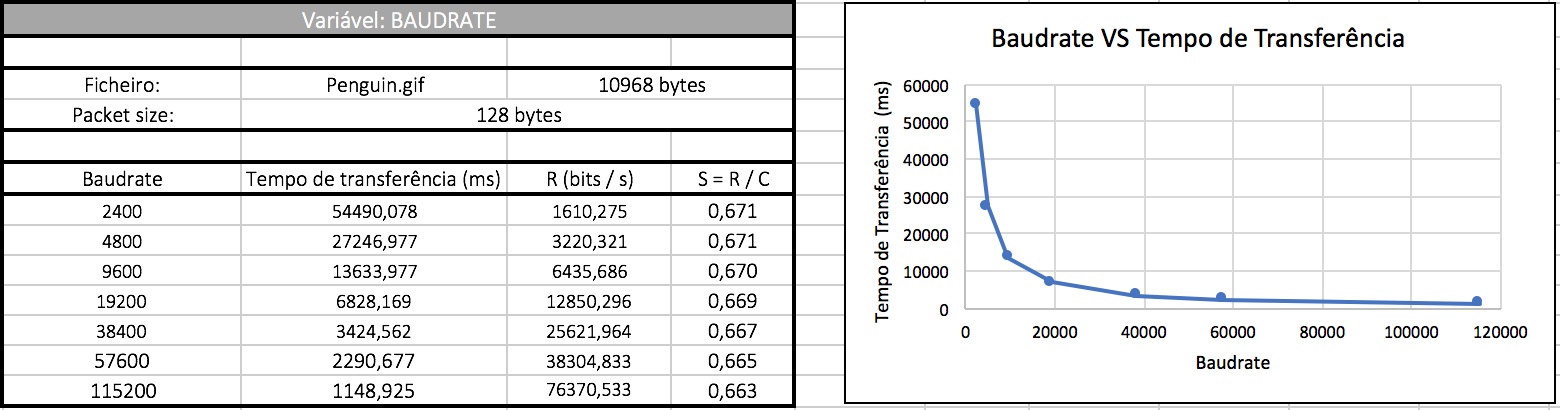


Fig5: Eficiência do programa, quando variada a taxa de transmissão, *baudrate*.

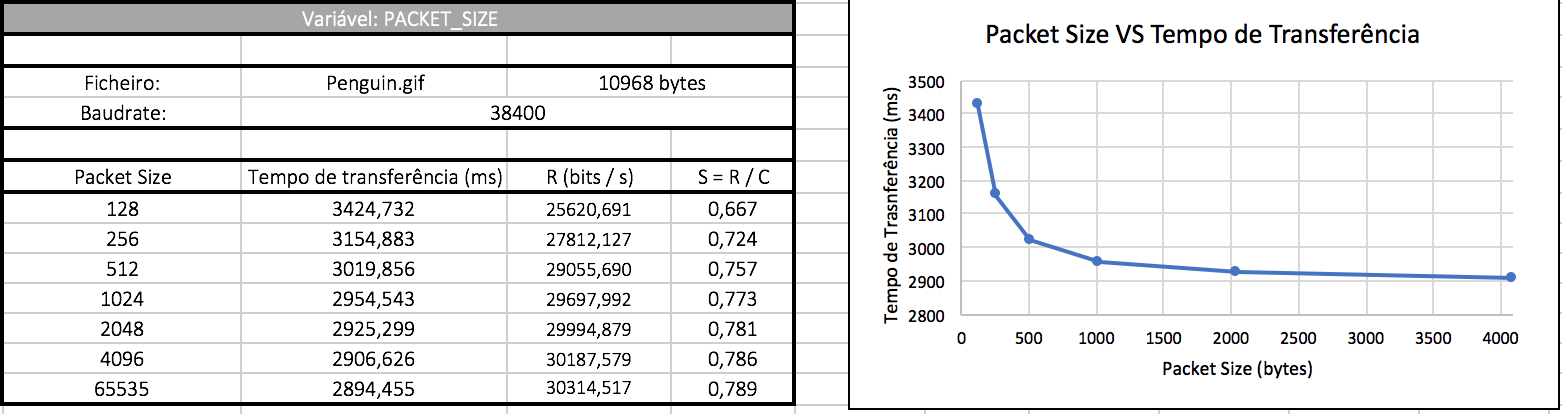


Fig6: Eficiência do programa, quando variado o tamanho dos pacotes de dados.

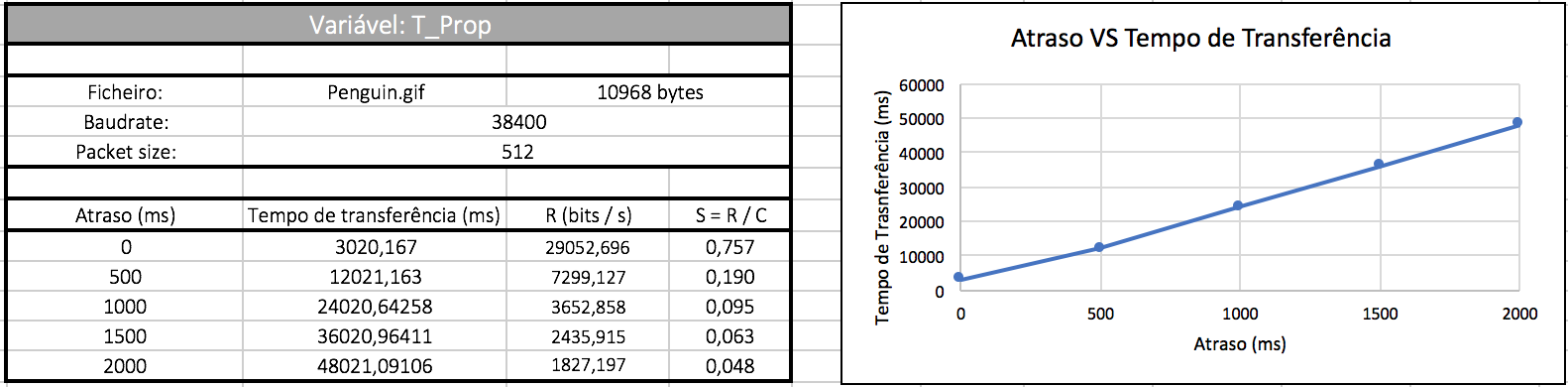
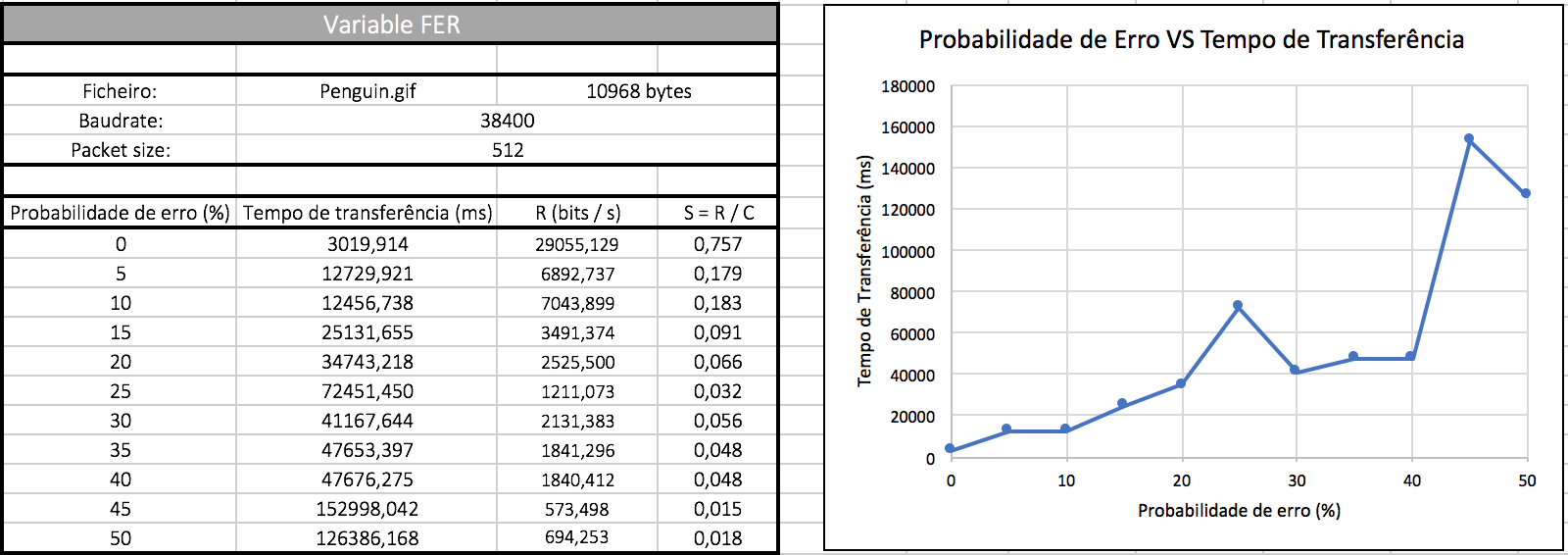


Fig8: Eficiência do programa, quando simulados erros, recorrendo a valores de probabilidade de ocorreram.

Fig7: Eficiência do programa, quando introduzido um atraso na receção das tramas, fazendo variar o tempo de propagação.

Relativamente ao teste de eficiência do programa com erros, convém referir que, para conseguirmos chegar a uma medição, alteramos o número de tentativas de *timeout*, de 3 para 10, de modo a que o programa não terminasse sem sucesso, principalmente com valores de probabilidade de erro mais elevados.

Comparando com os valores teóricos de referência relativos ao modem de linha telefónica, lecionado nas aulas teóricas, concluímos que o nosso projeto é eficiente, visto que o valor de *a* é sempre inferior a 1.

Neste contexto, não fizemos comparação teórica no teste de eficiência à T\_Prop, visto que estamos a induzir atrasos reais que afetam diretamente o tempo de execução do programa, o que resulta em valores de *a* muito díspares.

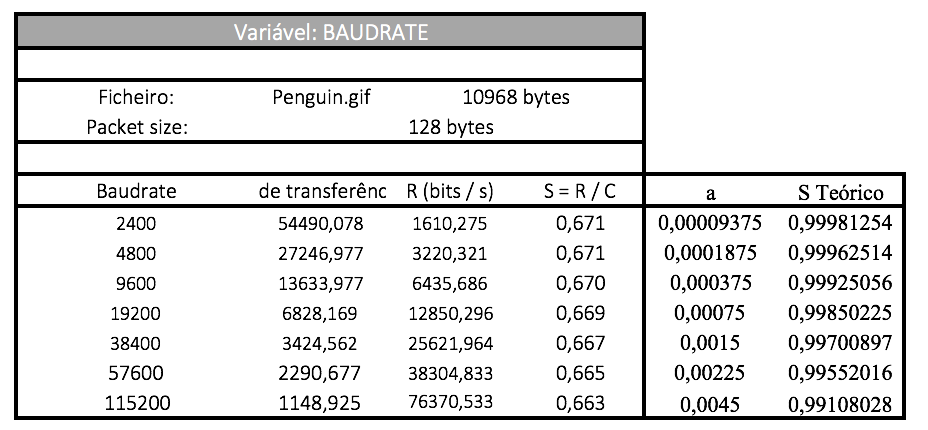
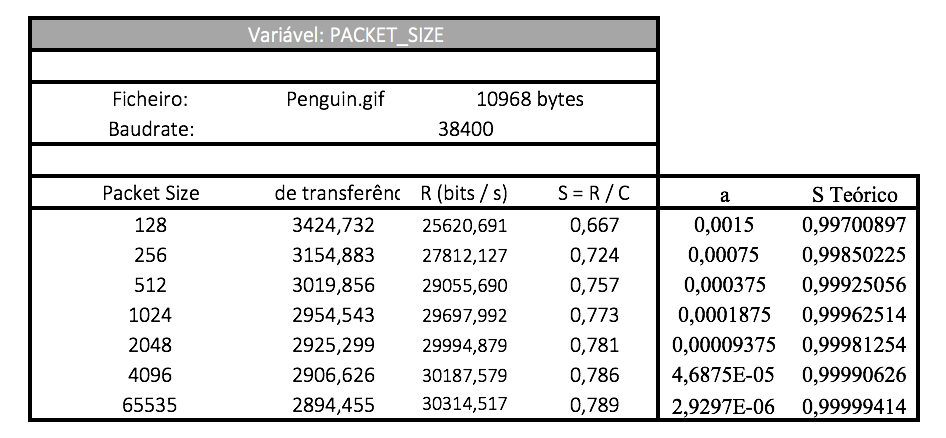
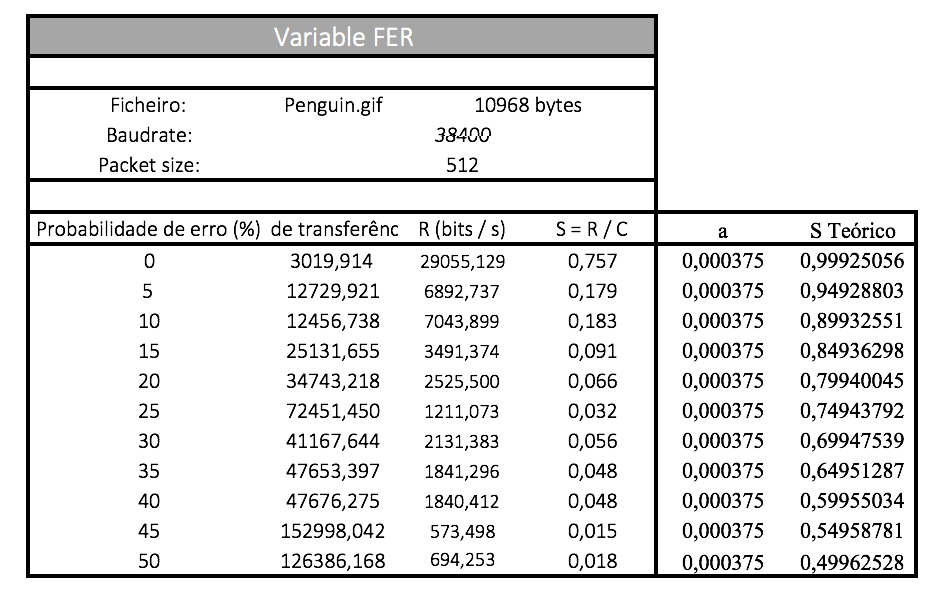


Fig9: Eficiência do programa, comparativamente aos valores teóricos.

**Conclusão**

O protocolo de ligação de dados foi correta e eficazmente implementado, tendo sido possível a transmissão resistente a erros de ficheiros entre dois computadores, utilizando as suas portas de série.

O programa possui duas camadas, que são inter-independentes, como estava previsto, sendo que na camada de ligação de dados não é feito qualquer processamento que incida sobre o cabeçalho do pacote de dados a enviar e na camada de aplicação não é feito qualquer processamento o endereçamento ao cabeçalho da trama. As componentes características de outros protocolos, como a transparência, foram implementadas.

A eficiência do programa, depois de testada e analisada pode ser justificada pelo facto de se gastar uma parte considerável do tempo a processar a informação, em detrimento da sua simples transmissão.

Para concluir este relatório, considera-se por bem evidenciar que os objetivos deste projeto foram alcançados, tendo sido feita com sucesso, a união e a consolidação da componente teórica e da prática da Unidade Curricular respetiva.