Relatório do 1º Trabalho Laboratorial

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Grupo:
Nuno Oliveira – up201506487
Pedro Miranda – up201506574
Carolina Azevedo – up201506509

Faculdade de Engenharia da Universidade do Porto

Rua Roberto Frias, sn, 4200-465 Porto, Portugal

ÍNDICE

SUMÁRIO

Contexto

Este trabalho está a ser desenvolvido para a unidade curricular de Redes de Computadores do 3º ano do curso de Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto, sobre o contexto de tranferência de dados entre dois computadores usando uma porta série RS-232 em comunicação assíncrona.

Conclusões

Este relatório pretende esclarecer todo o processo de comunicação e transferência de dados usando uma porta série RS-232. Com especial ênfase na arquitetura e estrutura do código, nos protocolos de ligação e aplicação para a boa interpretação da informação em ambos os sistemas, os processos de validação dos dados transferidos e a eficiência dos protocolos em relação às alternativas conhecidas.

INTRODUÇÃO

Objetivos:

- Implementar um protocolo de ligação de dados, de acordo com a especificação fornecida
- Testar o protocolo com uma aplicação simples de transferência de ficheiros, igualmente especificada

Descrição:

O relatório vai primeiro especificar a estruturação do código utilizado e a sua arquitetura, seguido dos casos de uso principais. Posteriormente, estará indicada toda a informação relativa aos protocolos utilizados, de ligação de dados e de aplicação, respetivamente, assim como a sua eficiência comparativamente a outros considerados. Por fim, os testes utilizados para validação dos dados transferidos com os resultados esperados e uma breve conclusão final de todo o relatório apresentado.

ARQUITETURA

O código está dividido em 2 grandes partes, uma dedicada à camada de aplicação e a outra dedicada à camada de ligação de dados. Para a interação entre as duas é usado um "main" que é responsável de interpretar o papel de Receptor ou Emissor, dependendo da intenção do utilizador, e que trata de usar os recursos de cada uma das camadas anteriormente referidas para a transferência de dados decorrer sem problemas.

ESTRUTURA

Main:

Como responsável pela execução do programa requer o *input* do utilizador para interpretar o papel de Emissor ou Receptor:

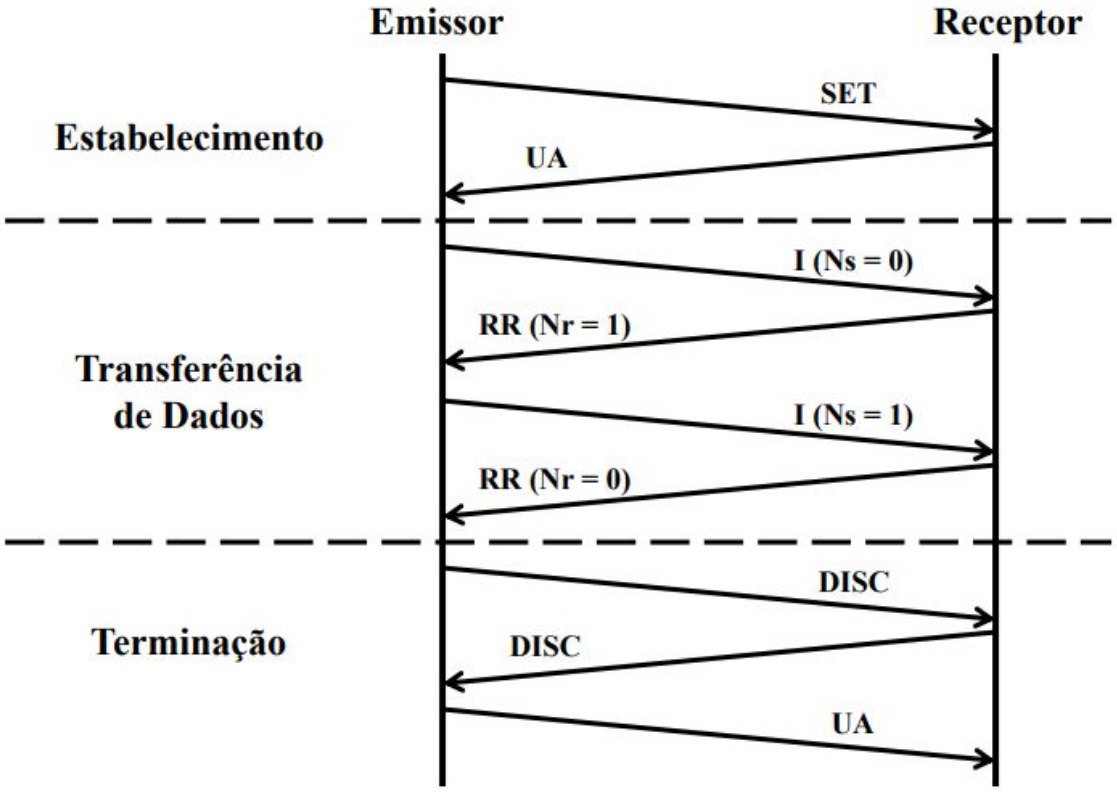
Emissor - Começa por estabelecer a ligação entre os dois computadores recorrendo à função *set_up_connection()*, após estabelecida a ligação sem problemas, inicia a transferência dos dados pretendidos utilizando a função *send_data()*. Assim que concluída a transferência, aguarda a confirmação do lado do Receptor para poder encerrar a ligação e, por conseguinte, o programa.

Receptor - Assim como o Emissor, estabelece a ligação entre os dois computadores recorrendo à mesma função *set_up_connection()*. De seguida, prepara-se para a receção dos dados por recurso à função *receive_data()*, que trata de receber, processar e guardar toda a informação recebida devidamente. Quando a transferência estiver concluída, comunica com o Emissor para terminar a ligação e prosseguir ao encerramento da sua execução.

PRINCIPAIS USOS

O caso de uso mais geral é simples. O Emissor prepara o ficheiro a ser transferido, certificando que ele encontra-se no diretório do programa e prossegue com a execução. De seguida especifica que pretende ser o Emissor e o programa trata do resto. O Receptor apenas executa o programa e específica que é o Receptor. O programa vai então prosseguir com a transferência do ficheiro e, assim que terminada a execução, uma cópia do ficheiro deverá encontrar-se no diretório do programa do lado do Receptor.

Fazendo uma análise mais específica, assim que assinaladas as funções pretendidas, *Emissor* e *Receptor*, ambos os programas vão estabelecer uma ligação (*set_up_connection()*). No caso do Emissor, a função *send_data()* será executada, no caso do Receptor, a função *receive_data()*. Ambos os programas irão terminar assim que a transferência esteja completa e tiverem enviado os sinais DISC para desconectar.



PROTOCOLO DE LIGAÇÃO

O Protocolo de ligação implementado tem como principais funções a configuração da Porta de série, estabelecimento da ligação através desta, a transferência de dados pela porta establecida (fazendo stuffing e destuffing) e a recuperação de erros durante este processo.

Para isto foram implementadas as funções *llopen()*, *llclose()*, *llwrite()* e *llread()*.

- *llopen()* e *llclose()*

Estas funções são necessárias para iniciar e terminar a ligação pela porta de série. Para isso a função *llopen()* começa por alterar as configurações da porta de série para as pretendidas. Depois, se a aplicação for transmissor, cria uma trama SET e envia-a. Para isto existe uma função *send_US_frame()* , que enquanto não recebe a resposta pretendida (UA) activa um alarme com duração pré-definida(3s) e tenta enviar a trama que lhe é passada. Se o alarme for desencadeado, conta como um timeout e tenta enviar a trama novamente. Caso o número de timeouts máximo permitido (3) for excedido, esta função termina com um estado de erro, informando a função *llopen()* que não conseguiu estabelecer a comunicação com o recetor. Se a aplicação for Recetor, fica à espera até receber uma trama SET, e quando isto acontece, responde com uma trama UA , estabelecendo assim, a ligação com sucesso. A função *llclose()*, do lado do transmissor, tenta terminar a ligação ao enviar uma trama DISC utilizando a função *send_US_frame()*, que espera pela resposta do Receptor, que terá de ser uma trama DISC. Ao recebê-la, responde com uma trama UA, informando o Recetor que recebeu a sua ordem de finalizar a ligação. Após isto, repõe as configurações que tinham sido alteradas no *llopen()*. O Receptor espera até receber uma trama DISC, respondendo com uma trama do mesmo tipo. Após a transmissão da resposta, espera pela trama UA do Transmissor e repõe as configurações, terminando com sucesso.

- *llwrite()* e *llread()*

Estas funções são responsáveis pela escrita e leitura de dados no decorrer da aplicação. A função *llwrite()* recebe um pacote e envia-o pela porta de série depois de o encapsular numa trama I. Para isto criou-se uma função, a *create_I_frame()*. Esta função trata de gerar todos os campos da trama I e de fazer o byte stuffing necessário. A função *llread()* espera até receber uma trama. Se for uma trama DISC, então procede ao fecho da ligação estabelecida, senão confere a validade do cabeçalho da trama recebida, rejeitando-a se for inválida (envia um comando REJ). Caso o cabeçalho seja válido, a função continua o processo de verificação, fazendo destuff ao pacote contido na trama e ao Block Check Character correspondente ao pacote. De seguida, confirma se o pacote é válido. Caso o

resultado seja afirmativo, envia uma trama RR, mesmo que o pacote seja duplicado. Caso contrário, transmite uma trama REJ, exceto se se tratar de um pacote duplicado, confirmando-o com RR.

Quanto à prevenção de duplicados, é comparado o nº de sequência com o byte de Controlo (C) da trama, se estes forem iguais o nº de sequência é invertido e continua a execução, caso não o seja, a trama é descartada.

```
int has_valid_sequence_number(char control_byte, int s) {
  return (control_byte == (s << 6));
}
```

PROTOCOLO DE APLICAÇÃO

O protocolo de aplicação implementado tem como funções a geração e transferência dos pacotes de dados e controlo e a leitura e escrita do ficheiro pinguim.

- *send_data()* e *receive_data()*

A função *send_data()* é a função responsável pelo comportamento principal do transmissor, e envia dados para a camada inferior enviar pela porta de série para o receptor. Para isto, a função começa por criar um pacote de controlo, o Start Packet, que contém a informação codificada em TLV (Type, Length, Value), isto é, para cada parâmetro a passar nesse pacote, é necessário passar o tipo do parâmetro, depois o seu tamanho e só depois o valor do parâmetro em si. Na aplicação desenvolvida, é passada neste pacote a informação das permissões do ficheiro, do seu nome e do seu tamanho. A função usa depois a função da camada inferior *llwrite()* para codificar e enviar este pacote. Depois, a função lê o ficheiro que se quer transferir e entra no ciclo principal do programa, em que se vai construindo pacotes de dados com os bytes lidos do ficheiro e enviando esses pacotes com a função *llwrite()*. Quando o ficheiro acaba de ser escrito, a função envia finalmente um End Packet e termina. A função *receive_data()* é a função responsável pelo comportamento principal do receptor, recebe dados da camada inferior para compor o ficheiro lido. Para isto, a função começa por ler, usando a função *llread()* da camada inferior, o pacote de controlo de onde tira o nome do ficheiro a ler, as suas permissões e o seu tamanho final. Depois, enquanto receber pacotes válidos e que não sejam o End Packet, continua a escrever os bytes de informação vindos de *llread()*, acabando quando o receber o End Packet.

VALIDAÇÃO

Para validação do Projeto foram realizados os testes:

- Envio do pinguim
- Fechar o interruptor, abrir e enviar o pinguim
- Enviar o pinguim,fechar o interruptor a meio e voltar a abri-lo
- Enviar o pinguim,introduzindo erros com o cabo

A aplicação foi capaz de ultrapassar todos estes testes,verificando-se pelo facto do pinguim estar completo e pelas mensagem de sucesso apresentadas no ecrã.

EFICIÊNCIA

O uso de um protocolo Stop & Wait é muito eficiente tendo em conta o caso de utilização, visto que a transferência é efetuada a uma distância muito pequena com recurso a um cabo de ligação direta. Desta forma, o tempo de propagação dos dados é praticamente nulo e o tempo de espera pela resposta do receptor não atrasa a transferência de uma forma significativa. Se a transferência for efetuada usando tecnologias de maior alcance, onde o tempo de propagação já seria significativo, este protocolo já não seria o indicado pela sua falta de eficiência.

Quanto a tempos de execução com diferentes tamanhos de Packet e diferentes Baudrate temos:

		Baudrate		
		B2400	B19200	B38400
Packet Size	64	63,3848 (s)	7,9577 (s)	3,9911 (s)
	128	54,3543 (s)	6,8083 (s)	3,4104 (s)
	256	50,1001 (s)	6,2693 (s)	3,1412 (s)

CONCLUSÕES

No geral,achamos que o grupo atingiu o objetivo principal do projeto. Foi um projeto trabalhoso e complexo mas fez com que todos os elementos entendessem o sistema de camadas, a independência destas, como foi provado no projeto.

Há alguns aspetos por onde o projeto poderia evoluir como a funcionalidade de poder enviar um ficheiro à escolha e não apenas o pinguim e a geração aleatória de erros.Também poderiam ser adicionados mais comentários e uma melhor estruturação do código visto este ser um pouco complexo de ler e perceber.

Contudo,como dito anteriormente,sentimos que o objetivo principal do projeto foi alcançado com sucesso pelo grupo.