Protocolo de Ligação de Dados

Relatório do 1º trabalho laboratorial

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Grupo xx:
Francisco Rodrigues - 201305627l
João Nogueira - 201303882
Marta Lopes - 201208067

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

30 de Outubro de 2015

Conteúdo

1 Sumário

Este relatório tem como objetivo explicar o primeiro projeto, realizado para esta unidade curricular, denominado "Protocolo de Ligação de Dados". Este projeto consiste no envio de informação de um computador para outro, através do uso de porta de série. Foram assim implementados programas para ler e escrever a informação a ser enviada.

O projeto foi finalizado com sucesso, sendo que os dados foram enviados e recebidos de forma correcta. Foram também incluídos a prevenção e correção de erros ao longo da transmissão, restabelecendo a transmissão quando os erros acontecem.

2 Introdução

Iremos então descrever o nosso trabalho, que consiste em implementar um protocolo de ligação de dados, de acordo com a especificação descrita no guião, de uma forma mais teórica e pormenorizada para assim poderem ser avaliados certos aspectos que não seriam possíveis de avaliar na apresentação na aula. O ambiente de desenvolvimento utilizado foi em PC's com *Linux*, a linguagem de programação foi C e as portas de série existentes realizavam comunicação assíncrona.

O protocolo de ligação de dados pretende assim fornecer um serviço de comunicação de dados fiável entre dois sistemas ligados por um cabo de série. As funções utilizadas serão a de criação e sincronismo de tramas que irão organizar os dados a ser enviados (*framing*), a do estabelecimento/conclusão da ligação, a numeração de tramas, o controlo de fluxo, a confirmação de envio sem erros e o controlo dos erros que poderão ser criados por *timeouts*, tramas fora da sequência esperada ou retransmissões.

O nosso relatório terá então as **secções principais** seguintes:

- **Arquitetura**: especificação dos blocos funcionais e da *interface*;
- **Estrutura do código**: API's, principais estruturas de dados, principais funções e a sua relação com a arquitetura;
- **Casos de uso principais**: identificar os principais aspectos, abordando as sequências de chamadas de funções;
- **Protocolo de ligação lógica**: identificar os principais aspectos funcionais da *linkLayer*, descrevendo a estratégia de implementação;
- **Protocolo de aplicação**: identificar os principais aspectos funcionais da *applicationLayer*, descrevendo a estratégia de implementação;
- **Validação**: testes efetuados ao programa com apresentação de resultados;
- **Elementos de valorização**: identificação dos elementos implementados, descrevendo a estratégia de implementação.

3 Arquitetura

3.1 *Application Layer* e *Link Layer*

O projeto está organizado em duas *layers* que vão ser responsáveis pela correta funcionalidade do mesmo. A *layer* que vai servir para a aplicação tem os ficheiros *applicationLayer.c* e *applicationLayer.h* e a ligação lógica está representada nos ficheiros *linkLayer.c* e *linkLayer.h* com um ficheiro *.h auxiliar, o *linkLayerAux.h*.

A **camada de ligação lógica**, contém várias funções, servindo estas para a configuração e abertura da porta de série, o envio de comandos, envio e receção de mensagens e para a realização dos processos de *stuffing* e *destuffing*.

A **camada de aplicação**, que vai depender da camada de ligação lógica utilizando algumas das suas funções, disponibilizará funções onde serão especificados o envio e receção de pacotes de dados, o envio e receção de pacotes de controlo e finalmente a leitura e a escrita do ficheiro a enviar.

3.2 *Interface*

A interface na linha de comandos está implementada nos ficheiros *cli.c* e *cli.h* que têm as funções que vão permitir a escolha de valores de alguns parâmetros referentes à transferência do ficheiro, existindo limites impostos de acordo com o guião. Os parâmetros a serem especificados pelo utilizador serão a *baudrate*, o tamanho máximo para a mensagem, o número de *timeouts* máximo no caso de falha de comunicação e a duração de cada *timeout*. Além disso, o utilizador terá também de dizer se será o transmissor ou o recetor e qual é a sua porta de série. Depois de o utilizador especificar cada um destes valores a aplicação vai iniciar, sendo que terá uma informação inicial mostrando os parâmetros com a escolha feita pelo utilizador.

4 Estrutura do código

(APIs, principais estruturas de dados, principais funções e sua relação com a arquitetura)

5 Casos de uso principais

(identificação; sequências de chamada de funções)

6 Protocolo de ligação lógica

O protocolo de ligação lógica está implementado na camada *linkLayer*, camada da qual depende a camada *applicationLayer*.

6.1 Principais aspectos funcionais

- Configurar a porta de série como é pretendido;
- Repor a configuração da porta de série como originalmente após a transferência dos dados pretendidos;
- Estabelecer a ligação de dados utilizando a porta de série;
- Enviar comandos;
- Enviar/receber mensagens;
- Processo de *Stuffing* e *Destuffing* dos *packets* recebidos da camada *applicationLayer*.

6.2 Funções implementadas na *linkLayer*

6.2.1 ll_open

Esta função é responsável por configurar a porta de série com as opções pretendidas (utilizando, por exemplo, o *baudRate* escolhido pelo utilizador) e por guardar numa variável que passa por argumento à função *configure*. Depois de configurar a porta de série, do lado do emissor envia o comando SET e aguarda a resposta UA do recetor que, ao ser recebida termina a função. Utiliza o alarme para controlar os *timeouts*. Do lado do recetor aguarda a recepção do comando SET e, ao recebê-lo responde com o comando UA.

6.2.2 ll_close

Do lado do receptor, esta função começa por aguardar a recepção do comando DISC, respondendo com o mesmo comando. Após o envio deste comando, e imediatamente antes de reestabelecer as configurações originais da porta de série aguarda a recepção do comando UA por parte do emissor. Do lado do emissor, esta função começa por enviar o comando DISC e aguarda pela resposta com o mesmo comando por parte do receptor (implementando o alarme para manter controlo dos timeouts). Após a correta recepção de DISC evia o último comando do programa, comando UA. Termina por fazer *resetConfiguration*, função na qual é feito um *sleep* de um segundo por forma a garantir que as configurações originais da porta de série não são restabelecidas antes de que toda a informação tenha sido passada.

6.2.3 llwrite

Esta função começa por alocar memória num *buffer* no qual a informação será organizada (antes do processo de *stuffing*). Após alocar a memória necessária começa por atribuir os valores de FLAG, A, C e BCC de acordo com o número de sequência da trama. A partir de BCC é copiada para o buffer toda a informação recebida por argumento desta função da *applicationLayer* correspondente ao *packet* a enviar. Antes de terminar o preenchimento do *buffer* coloca o BCC2 ('ou' exclusivo dos octetos do *packet*) e a FLAG para terminar. Depois de ter a trama preenchida e antes de começar o processo de envio, envia o *buffer* para a função *stuff* que retorna um *buffer* com a trama pronta para envio. Envia a trama através da porta de série e aguarda a resposta por parte do recetor, o que vai determinar o caminho a seguir pela função, que pode variar entre terminar a função pois houve a recepção de RR, ou re-enviar a mesma trama por ter sido rejeitada. A mesma trama pode ser re-enviada por ocorrência de *timeout*, o que também é controlado nesta função através da implementação de alarmes. Esta função pode também terminar pela ocorrência de *timeouts* maior do que o número máximo definido previamente pelo utilizador.

6.2.4 llread

A função *llread* começa por alocar memória para o *buffer* que vai ser recebido, entrando imediatamente a seguir num ciclo do ual apenas sai quando algo é lido da porta de série. Após a leitura o *buffer* lido é passado para a função responsável pelo processo de *destuffing* que retorna a trama de Informação recebida "descodificada". Os valores de BCC e de BCC2 são verificados por forma a garantir que a trama foi recebida sem erros, caso o tenha sido é interpretado o número de sequência e se tudo for o pretendido é enviado o comando RR, caso contrário é enviado o comando REJ, pedindo que a mesma seja enviada novamente.

7 Protocolo de aplicação

O protocolo de aplicação está implementado na camada applicationLayer, camada esta que depende, como dito anteriormente, da camada linkLayer.

7.1 Principais aspectos funcionais

- Envio e recepção de pacotes de dados;
- Envio e recepção de pacotes de controlo;
- Leitura do ficheiro a enviar;
- Escrita do ficheiro a enviar.

7.2 Funções implementadas na *applicationLayer*

main
Chama as diversas funções da *Interface* do utilizador e recebe o seu *input* por forma a inicializar o programa com as configurações desejadas pelo utilizador. Chama as funções *ll_open, ll_close*, *readFile* e *sendFile*.

initStat
Função responsável por inicializar os valores da estrutura de estatística. Esta estrutura é utilizada para que, no final da execução do programa, o utilizador esteja a par do que aconteceu no desenrolar do programa.

fillLinkLayer
Função responsável por inicializar a estrutura *linkLayer*, estrutura essa passada para muitas das funções utilizadas ao longo do programa. Nesta estrutura estão atributos como por exemplo: *baudRate, timeout*, etc.

createFirstControlPacket
Função responsável por gerar o primeiro pacote de controlo. É colocado neste pacote o tamanho do ficheiro a enviar bem como o nome do mesmo.

getNameAndSizeOfFile
Esta função é responsável por, dado o primeiro pacote de controlo recebido, interpretar qual o nome e tamanho do ficheiro a receber.

sendFile
Esta função começa por ler toda a informação do ficheiro a enviar e guarda-a num *buffer* de chars, *buffer* a partir do qual a informação será lida para envio. Após a leitura chama a função responsável pelo envio do primeiro pacote de controlo e envia-o para o recetor chamando *llwrite* (função do pacote de ligação). Depois do envio do primeiro pacote de controlo é

calculado o número de pacotes necessários a enviar tendo em conta o tamanho máximo definido previamente pelo utilizador, e entra num ciclo que é percorrido o número de vezes igual ao número de pacotes de dados. Neste ciclo é, inicialmente criado o pacote de dados, com a informação necessária (C, número de sequência, número de octetos do campo de dados) e com os dados em si. É chamada a função *llwrite* para enviar o pacote e repete-se o ciclo até ao final do envio dos dados (atualizando sempre a barra de progresso no ecrã).

readFile

Esta é a função responsável pela recepção de todo o ficheiro e pela sua escrita em disco. Começa por entrar num ciclo que termina quando o último pacote de controlo é lido. Dentro do ciclo o primeiro passo é chamar a função *llread* que retorna, após a leitura, o pacote a interpretar. São então tomadas medidas diferentes nos casos de ser o primeiro pacote de controlo, o último pacote de controlo ou um pacote de dados. No caso de ser o primeiro pacote de controlo é guardada a informação do nome e tamanho do ficheiro. No caso de ser o último pacote de controlo é criado o ficheiro e toda a informação lida, até agora guardada num *buffer*, é escrita em disco. No caso de ser um pacote de dados a informação é adicionada no final do *buffer* de leitura. Sempre que um pacote é lido é atualizada a barra de progresso.

8 Validação

Para verificar se a transferência do ficheiro *pinguim.gif* foi efetuada com sucesso decidimos colocar uma verificação de inicio e fim de envio, durante o envio do ficheiro adicionamos uma barra de progresso que iria sendo actualizada à medida que os packets iam sendo enviados.

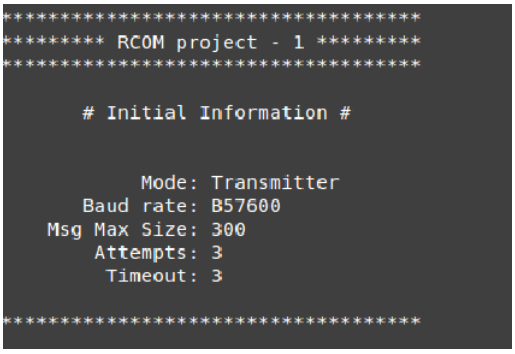

Figura 1: Informações iniciais - Modo Transmissor

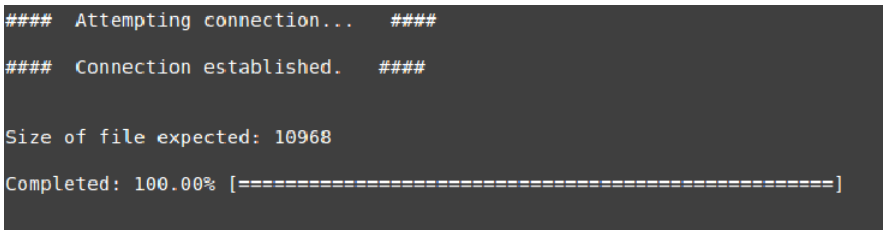

Figura 2: Inicio do envio e barra de progresso - Modo Transmissor

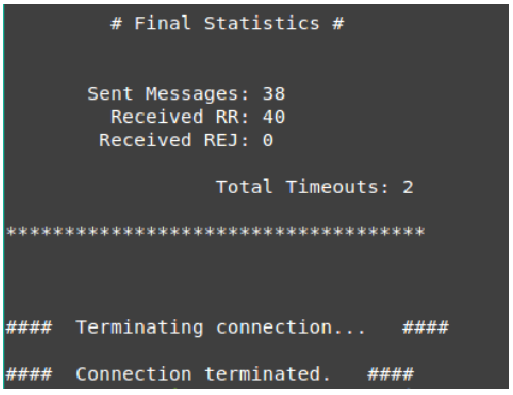

Figura 3: Fim do envio e estatisticas - Modo Transmissor

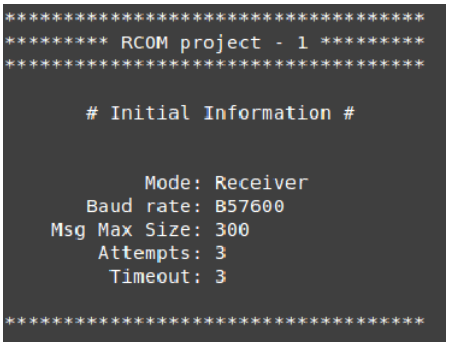

Figura 4: Informações iniciais - Modo Recetor

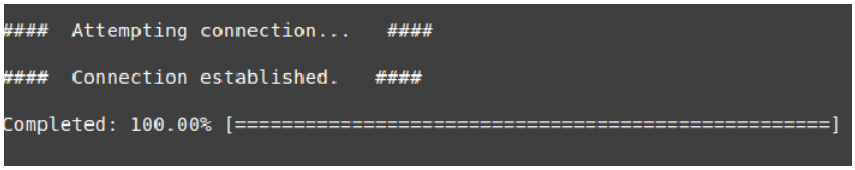

Figura 5: Inicio do envio e barra de progresso - Modo Recetor

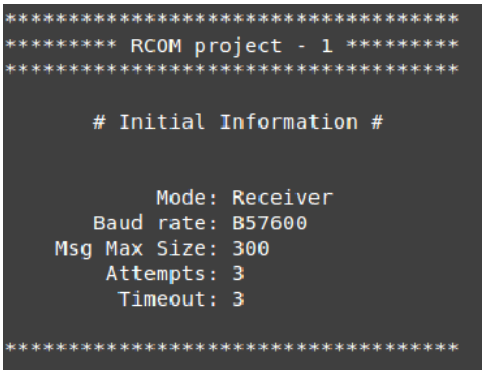

Figura 6: Fim do envio e estatisticas - Modo Recetor

Adicionamos também o modo *debug* em que iriam ser feitos mais *prints* à medida que o ficheiro ia sendo enviado, sendo também possível verificar quando houvesse *timeouts* ou envio/receção de REJ's.

9 Elementos de valorização

(identificação dos elementos de valorização implementados; descrição da estratégia de implementação com apresentação de pequenos extratos de código)

10 Conclusões

(síntese da informação apresentada nas secções anteriores; reflexão sobre os objectivos de aprendizagem alcançados)

11 Anexos