Protocolo de Ligação de Dados

Relatório Laboratorial

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Grupo 3:
Carlos Miguel Lucas - ei11140
Maria Teresa Chaves - up201306842

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

10 de Novembro de 2016

Resumo

Na unidade curricular de Redes de Computadores foi desenvolvida uma aplicação de transferência de um ficheiro entre dois computadores, para o teste de um protocolo de ligação de dados, usando para isso a porta-série RS-232. A aplicação foi desenvolvida com sucesso, transferindo os dados necessários e detetando eventuais erros que possam acontecer durante a sua transmissão.

Conteúdo

1 Introdução

No âmbito da unidade curricular de Redes dos Computadores, foi-nos proposto o desenvolvimento de um protocolo de ligação de dados para o envio de um ficheiro entre dois computadores, através de uma porta de série RS-232.

Este projeto teve como principal objetivo o desenvolvimento de um protocolo de ligação de dados, de acordo com a especificação fornecida no guião do trabalho. Para além disso, teve também como objetivo testar o protocolo desenvolvido com uma aplicação simples de transferência de ficheiros, que também seguiu as especificações fornecidas.

O relatório está dividido em várias secções, sendo que a primeira se dedica a uma pequena **introdução** acerca do projeto desenvolvido. Na secção seguinte descrita a **arquitetura da aplicação** mostrando como foi feita a sua divisão nas diferentes camadas e o que contém cada uma delas. Na **estrutura do código** é detalhado o funcionamento das camadas de forma independente e entre si. Nos **casos de uso principais** é explicada qual a sequência de funções que são chamadas durante a utilização da aplicação. No **protocolo de ligação lógica** são identificados os principais aspetos funcionais, sendo feita uma descrição da sua estratégia de implementação bem como de alguns extratos de código. No **protocolo de aplicação** são indicados os principais aspetos funcionais e é descrita a estratégia de implementação destes aspetos, apresentando extratos de código. A secção **validação** contém uma descrição dos testes efetuados, apresentando resultados quantificados. Por último, nos **elementos de valorização** identificámos quais deles foram implementados, acompanhado-os de uma descrição da estratégia adotada.

2 Arquitetura

A aplicação foi desenvolvida de acordo com as especificações indicadas no guião do trabalho, no que diz respeito à divisão do código por camadas. Formamos assim as camadas *Linklayer* e *ApplicationLayer*. Inicialmente, optamos por seguir a sugestão do guião, criando uma estrutura (*struct*) para cada uma destas camadas, guardando a sua informação. No entanto, devido a múltiplos problemas que persistiram na leitura destes dados, vimonos forçados a abandonar esta metodologia. Na *LinkLayer* foram desenvolvidas as funções recomendadas `llopen`, `llclose`, `llread` e `llwrite`. Na *ApplicationLayer* foram desenvolvidas as funções `start_control_packet`, `data_packet`, `end_control_packet` e `sm_receiver`, para além de várias funções auxiliares para imprimirem o processo desta aplicação de uma forma mais clara e *user-friendly*.

3 Estrutura do código

Ao iniciar a aplicação, o utilizador deve indicar qual a porta-série a usar e também o modo de utilização do programa (emissor ou recetor). Posteriormente, são também questionados vários parâmetros como a *baudrate*, o número máximo de retransmissões em caso de erro, o *time-out* necessário, e o tamanho máximo da trama de informação a transmitir. Em ambos os casos, é chamada a função `llopen(char * port)`, necessitando apenas da porta que deverá usar para estabelecer a ligação entre os dois computadores. Depois, no emissor, começam a ser processados os pacotes necessários para o envio, usando as funções descritas na secção anterior. Por sua vez, estas funções chamam a função `llwrite(int fd, char * buf, int size)`, que necessita do descritor da porta, o conteúdo a escrever e o tamanho deste conteúdo. Do outro lado, para conseguir processar estes pacotes, o recetor usa a função `sm_receiver` para chamar `llread(int fd, char * buf)`, que precisa do descritor da porta e do *buffer* para onde será escrito este ficheiro.

4 Casos de uso principais

O programa inicia com a escolha de vários parâmetros por parte do utilizador. No caso do recetor, este irá usar a função `llopen` para abrir a porta-série, tentando de seguida ler os pacotes de dados enviados com a função `sm_receive`. Esta função inicia a máquina de estados que verifica o tipo de pacote que recebe (controlo ou de dados) e executa a função `llread` para processar a informação do pacote recebido. Para o caso do emissor, são usadas as funções `start_control_packet` e `end_control_packet` para enviar os pacotes de controlo e data_packet para enviar os pacotes de dados, sendo que todas estas invocam a função `llwrite`, para escrever o pacote respetivo para o outro computador. O programa termina imprimindo informação estatística sobre a transmissão e terminando a conexão com a função `llclose`.

5 Protocolo de ligação lógica

A camada de ligação lógica é responsável pelo envio e receção das tramas, abertura, estabelecimento e término da ligação da porta-série. A função `llopen(char * port)` estabelece a ligação entre o emissor e o recetor, dado o nome da porta-série. Esta é aberta e posteriormente é enviando uma *flag SET*, no caso do emissor. No caso contrário, é lida um `SET` para que a função retorne com sucesso.

No lado do emissor, é usada a função `llwrite(int fd, char * buf, int size)` que verifica qual a *flag* de informação a ser enviada baseado no número da trama e escreve-a usando para isso uma máquina de estados. Esta função necessita de um descritor da porta para a qual será transmitida

a informação, da própria informação a transmitir e do seu tamanho em bytes.

No lado do recetor, é usada a função `llread(int fd, char * buf)` que recebe um descritor da porta para da qual será recebida a informação, e de um *buffer* para conter a informação. Ao ler a informação é verificado se a flag do campo de controlo é um DISC ou se é um SET, pois caso o sejam significa que a transmissão terminou ou iniciou, respetivamente.

No final destes processos, em ambos os casos é usada a função `llclose(int fd)` que usa o descritor da porta para terminar a conexão entre o recetor e o emissor.

6 Protocolo de aplicação

A camada da aplicação está incumbida pelo envio dos pacotes de dados necessários para a camada lógica processar os ficheiros recebidos. Estes pacotes foram divididos pela seguinte forma:

- um pacote de controlo inicial
- pacotes de dados
- um pacote de controlo final

Inicialmente é enviado um **pacote de controlo**, composto por um octeto de controlo e por informações sobre o tamanho e nome do ficheiro, codificados na forma TLV (*Type, Length, Value*), ou seja, indicando o tipo de informação a transmitir, o tamanho em octetos desta informação, e por fim, a informação a enviar.

Em caso do pacote inicial ter sido enviado com sucesso, é enviado o **pacote de dados**. Este pacote contém um *header* que consiste: num campo de controlo; um octeto com o número da sequência da trama; dois octetos que indicam o tamanho da trama. De seguida, são enviados vários octetos (número exato indicado nos octetos anteriores) com o conteúdo da trama.

No final da transmissão do pacote de dados, é enviado um outro **pacote de controlo**, igual ao pacote de controlo inicial, com a única diferença sendo o octeto de controlo, usado para distinguir qual o pacote que é lido no momento.

7 Validação

No inicio do desenvolvimento do projeto foi criada uma aplicação em que apenas era transferida uma *string* inserida pelo utilizador, de forma a poder

testar se os cabeçalhos estavam a ser corretamente adicionados a um pacote. Assim que a aplicação transferiu com sucesso a *string* inserida, avançamos para a implementação da leitura do ficheiro e da sua divisão nos diferentes pacotes. No final desenvolvemos a parte do *stuffing* e *destuffing* dos dados. Desta forma conseguimos garantir que a aplicação transferia com sucesso o ficheiro pretendido.

8 Elementos de valorização

Os seguintes elementos foram implementados com sucesso na aplicação:

- Seleção de parâmetros pelo utilizador - o utilizador pode definir vários parâmetros, tal como explicado na secção Estrutura do código.
- Implementação do `REJ` - no caso de erro numa transmissão, são enviadas *flags* `REJ` para sinalizar que houve uma interrupção desta.
- Registo de ocorrências - todos os casos de retransmissões, *timeouts* e `REJs` enviados/recebidos são detalhados aquando do término do programa. Foram implementados contadores para todos estes casos, que são incrementados sempre que uma destas situações se verifica.
- Geração aleatória de erros em tramas de informação - os erros que introduzidos no programa foram detetados com sucesso, devido à constante verificação das *flags* recebidas aquando do envio e receção das tramas de informação.
- Verificação da integridade dos dados pela Aplicação - o programa calcula, a cada trama, uma *flag* `BCC1` para cada *header* da trama, e caso não se verifique a integridade deste, a trama será rejeitada. O mesmo é feito para o `BCC2`, em que em vez de verificar a integridade do *header*, fá-lo em relação a todos os dados contidos na trama. É verificado também se existe alguma trama duplicada, e neste caso, também será rejeitada.

9 Conclusões

Neste trabalho verificamos a importância de uma correta estruturação e organização do código. Caso não tivéssemos divido o código em camadas lógicas tornaria-se bastante complicado de desenvolver esta aplicação e ficaria mais difícil interligação entre a parte da aplicação e a parte lógica.

A implementação deste protocolo permitiu-nos alargar as nossas capacidades e conhecimento para com a área de redes de computadores.