Relatório do 1º Trabalho Laboratorial

Redes de Computadores

MIEIC 3º Ano

Turma: 3MIEIC01

Autores:

Daniel Santos Teixeira – ei10067@fe.up.pt

Luís Guilherme Martins – ei10105@fe.up.pt

Victor Cerqueira – ei10055@fe.up.pt

Sumário

A aplicação apresentada no relatório tem como objectivo a comunicação entre 2 computadores por porta de série RS-232 (comunicação assíncrona) usando um protocolo, implementado em linguagem C, definido no guião do trabalho com a finalidade de transmissão de uma imagem entre os computadores e a correcção dos erros de transmissão.

1. Introdução

Este projecto foi realizado no âmbito da unidade curricular de Redes de Computadores incluída no 1º semestre do 3º ano do Curso Mestrado Integrado em Engenharia Informática e Computação e consiste no desenvolvimento de uma aplicação em C ou C++, que nos permita comunicar entre computadores, por um cabo de série, mais propriamente, o objectivo será seguir enviar uma imagem de um pinguim ("pinguim.gif") tendo em conta várias os possíveis erros que possam ocorrer durante essa transmissão de informação.

O grupo tinha por objectivo preparar duas aplicações, uma a ser corrida em cada computador. Uma das aplicações (sender) teria de analisar a imagem, transformá-la e enviar os dados, enquanto a outra (receiver) tem a funcionalidade de receber esses dados, e transformá-los novamente numa imagem. Se todo esse processo funcionasse na perfeição, a imagem final teria de ser a mesma que a imagem inicial.

2. Arquitectura

O trabalho foi dividido em dois ficheiros, o Sender.c e o Receiver.c e desenvolvido em linguagem C. Isto cria duas aplicações separadas, uma em cada computador, que comunicam entre si através das funções llopen(), llread(), llwrite(), llclose().

3. Estrutura do código

A API utilizada no desenvolvimento deste trabalho foi a fornecida pelos professores e faz uso das portas de serie presentes nos computadores, e que as funções nela contidas permitem a abertura, escrita, leitura e fecho de um descritor de ficheiro (por onde a informação é transferida de um computador para outro). Esta API permite, também, alterar as suas definições bem como limpar todo o conteúdo desnecessário.

As principais funções são bastante semelhantes em ambos os ficheiros: llopen(int porta) – abre a porta de serie em ambos os ficheiros e apenas no sender, envia o SET e recebe o UA; llwrite(int fd, unsigned char *buffer, int length) – escreve na porta de serie sendo igual em ambos os ficheiros; llread(int fd, unsigned char *buffer) – esta função é mais simples no sender pois as tramas que são lidas são sempre semelhantes (RR), enquanto eu que no receiver a função tem de estar preparada para ler tramas diferentes; llclose(int fd) – no sender esta função, envia e recebe o DISC, responde o UA e fecha no fim a porta de série, enquanto que no receiver é apenas utilizada para fechar a porta de série.

4. Casos de Uso Principais

Os casos de usos identificados nesta aplicação são transferências de ficheiros por um cabo de serie entre dois PC´s com Linux. No inicio, o tamanho do campo de dados das tramas I e o nome do ficheiro são passados como argumento à aplicação Sender que inicia com a função llopen(). Esta função inicia a porta de série e envia a primeira trama SET esperando resposta do outro lado. Na aplicação Receiver não são passados argumentos e começa também com a função llopen() que, neste caso, apenas inicia a porta de série. A seguir é chamada e função estabelecimento(int fd) que fica à espera de receber a trama SET e quando finalmente a recebe, responde UA. Seguidamente entre em acção a função: int transfDados(int fd, char* buffer, char* data, int filesize) que é composta por várias funções que permitem, por exemplo, a verificação de BCC's, separar o campo de dados das tramas I do resto dos campos e verificações de erros de envio ou de recepção entre outros. No fim, na aplicação sender é chamada a função llclose() que envia o DISC, recebe DISC, responde UA e fecha a porta de série, terminando o programa. No Receiver no entanto, a função só é chamada depois de o UA ser recebido com sucesso, fechando assim a porta de série e terminando a aplicação.

5. Protocolo de ligação lógica

O protocolo de ligação lógica, como já foi visto acima, está dividido em quatro funções principais, que correspondem a abertura da ligação (llopen), envio de dados (llwrite), recepção de dados (llread), e encerramento da ligação (llclose). Todas elas com funcionamentos baseados numa máquina de estados desenvolvida para o lado do emissor e outra para o lado do receptor.

Relativamente a abertura da ligação, pelo lado do receptor, a leitura de dados funciona através do recebimento byte a byte dos mesmos, desde o momento em que se encontra uma FLAG, até o momento em que se encontra a seguinte, guardando-se todos num buffer para posterior análise. Esta, como consta das especificações passa pela verificação do BCC. Depois é feita a verificação para ver se é a trama correspondente ao SET e se assim for, envia-se a resposta com uma trama UA.

Continuando a abertura, mas agora pela perspectiva do emissor, faz-se o envio da trama SET e faz-se a recepção de dados byte a byte, já com o funcionamento do timeout iniciado. Mas aqui, a verificação da trama recebida é feita no momento da recepção de cada byte, inclusive do BCC.

Passando para o envio de dados, este funciona com a criação de uma trama com os dados recebidos da aplicação que é enviada de uma única vez através de um comando write. Então, dá-se início ao timeout e recebe-se os dados byte a byte verificando-se de que tipo de trama se trata, já que há vários tipos de resposta por parte do receptor.

Na fase de envio de dados, temos da parte do receptor a recepção das tramas mencionadas no parágrafo anterior. A operação desta recepção é a mesma da abertura de ligação. Com uma opção a mais que é a verificação de um segundo BCC relativo apenas aos dados (o primeiro é apenas relativo ao cabeçalho), já que não são tramas conhecidas previamente.

Nesta fase do protocolo de ligação, podemos ainda receber mais dois tipos de trama que são a do fim de ligação, e da confirmação de fim da ligação. Com estas diferenças podemos ir para vários estados diferentes.

Estando no estado de recepção desta trama de dados, existem uma variável booleana "isValid" que verifica a validade do I que é recebido, se este for o correcto, responde RR alterando do NR, se não for correcto, responde RR mantendo o NR anterior e permitindo assim ao sender saber que a sua trama foi mal enviada ou perdida.

Por último, temos a função de encerramento da ligação, que apenas funciona do lado do emissor, fazendo um envio byte a byte de cada uma das duas tramas necessárias ao fim da ligação. A segunda só é enviada depois da recepção de uma trama DISC que também é recebida e verificada byte a byte.

6. Protocolo de aplicação

A aplicação recorre às funções mencionadas acima, abrindo a ligação, enviando dados e fechando a ligação.

O código foi dividido em duas aplicações diferentes para o envio e para a recepção de dados, apesar de a maioria das funções implementadas nos dois serem idênticas.

Quando se corre a aplicação Sender é necessário passar com argumentos o tamanho máximo das tramas de dados e o nome do ficheiro a ser transmitio. Quando se corre a aplicação Receiver não é preciso passar nenhum argumento.

O transmissor tem como tarefas a abertura do ficheiro especificado para leitura, determinação do seu tamanho e sua divisão para envio em pacotes de dados (de tamanho definido) que seguem uma determinada configuração. Mas antes destes, deve abrir a ligação e enviar um pacote de início com características do ficheiro, como o tamanho. Depois, deve enviar um pacote e fim, que apenas difere do anterior por um primeiro byte que indica que é o de fim. Portanto, há uma chamada a llopen, depois uma a llwrite para o pacote de início, as necessárias a llwrite para o envio

dos pacotes de dados e mais uma para o pacote de fim. Para encerrar a ligação, uma chamada a llclose. Qualquer erro numa destas funções, impede a aplicação de continuar.

O receptor, cria um ficheiro e faz também uma chamada a llopen para abertura da ligação. Depois, diversas a llread para recepção dos dados, aproveitando aquilo que são as partes que interessam para escrita no ficheiro que foi previamente criado. Esta escrita respeita a verificação dos números de sequência, para verificar se os dados recebidos estão dentro da ordem necessária.

7. Validação

Foram feitos vários testes ao longo do trabalho não só para verificar a robustez do próprio código, como para fazer o próprio Debug de erros que apareciam, através de "printf()". Além disso, foi testado, não só o envio de tramas de dados com diferentes tamanhos (10, 512, 1024) como também o envio de diferentes ficheiros para além do pinguim.gif.

8. Elementos de valorização

Alguns elementos de valorização foram implementados, nomeadamente, a verificação de se o tamanho real do ficheiro é igual ao tamanho indicado nas tramas de controlo. O programa está preparado também para identificar tramas repetidas, danificadas ou perdidas, uma vez que o emissor só envia a trama seguinte se receber a confirmação (RR) de que a trama anterior foi recebida correctamente. Desta forma é eliminada a possibilidade de se perderem tramas. Para rejeitar as tramas danificadas recorre-se ao campo de controlo BCC que calcula o "ou exclusivo" de cada um dos elementos recebidos e comparado com o valor recebido pelo emissor e que resulta dos mesmo cálculos. Relativamente às tramas repetidas, o protocolo está implementado de forma a que o receptor guarde a primeira trama recebida se for válida e ignora as seguintes caso sejam iguais à anterior.

9. Conclusões

Chegado este projecto ao fim, é nos possível afirmar que foi realizado com sucesso, obedecendo a todos os objectivos e considerações que nos foram feitas com o enunciado, conseguindo implementar tudo o que era necessário para um correcto funcionamento da aplicação.

Com este projecto, conseguimos perceber a importância de estudar as bases dos temas abordados no projecto antes de proceder à sua execução bem como a enorme importância de fazer o projecto com tempo, sendo que se consegue trabalhar melhor, pensar nos objectivos do trabalho e implementa-lo com a verdadeira noção do que estamos a fazer, ao contrário de quando estamos a trabalhar em cima dos prazos.

O projecto foi bastante desafiante, tivemos que aplicar muitos conceitos novos e variados e a sua implementação repetida facilitou a aprendizagem deles, pois permitiu uma consolidação melhor dos conhecimentos que a cadeira transmite.

A. Anexo

Em anexo será enviado os dois ficheiros de código implementados:

- Sender.c;
- Receiver.c

Optamos por enviar o código todo em anexo, pois ocuparia muito espaço no relatório e ultrapassaria as 8 páginas permitidas.