FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

1º Projeto de Redes de Computadores 22/23

3LEIC11

Diogo Tomáz Câmara - up201905166
Diogo Azevedo Lemos - up202003484

Indice

Introdução

O objetivo do trabalho realizado foi o de implementar um protocolo de ligação de dados, de acordo com a especificação descrita no guião de trabalho e testar esta mesma implementação de protocolo com uma aplicação simples de transferência de ficheiros, igualmente especificada .

Arquitetura

Foram-nos dados instruções de maneira a que o projeto fosse sido divido em 2 camadas independentes: a "application layer" e a "link layer". A "application layer" não tem acesso à "link layer" , exceto às suas funções principais. A "link layer" é constituída por 4 funções : "llopen" ,"llread" , "llwrite" e "llclose" .

Estrutura da Aplicação

O nosso código está dividido em 3 módulos, o módulo "Application layer", o módulo "link layer" e o módulo "extra functions" . O módulo "Application layer" tem a função "applicationLayer" onde é feita a leitura/escrita em ficheiros e as "calls" a funções de outro módulos mas especificamente o "link layer" . No módulo "link layer" temos as 4 funções principais usadas pela função "applicationLayer" que são "llopen" ,"llread" ," "llwrite" e "llclose". Dentro destas funções são chamadas funções de um outro módulo chamado "extraFunctions" . No módulo "extraFunctions" temos as funções utilizadas pela "llopen" como o "OpenSerialPort" , e funções usadas pelo "llclose" tais como "ClosePort". Para além destas funções, também podemos encontrar funções responsáveis por implementar máquinas de estados que são responsáveis pela verificação do conteúdo nas tramas.

No " applicationLayer" é utilizada o struct linkLayer, que guarda informações sobre o nome da serialPort, o role, a velocidade de transmissão, o número de timeouts e o número de retransmissões.

Casos de uso principais

Para se poder correr o programa, temos primeiramente de o compilar, usando o Makefile. O programa será executado de 2 maneiras diferentes ,uma com o papel de receiver que deve ser executado primeiro de maneiras a que fique à espera que o transmitter inicie a transmissão. Se executarmos primeiro o transmitter, este tentará enviar a informação ao receiver até que número de retransmissões seja excedido. É de notar que nas funções "llopen" e "llclose" a distinção de funcionalidade emissor ou receiver é feita pelo parâmetro role do "link layer".

- No caso de o programa estar a ser executado como emissor as funções mais importantes são executadas da seguinte maneira:

- **Application layer** -> **llopen** -> **set_serial_port** -> **statemachine**
- **Application layer** -> **llclose** ->**statemachine_close** -> **close_serial_port**
- **Application layer** -> **llwrite** -> **statemachineRR**

- No que diz respeito aos casos de um Recetor as funções de maior relevo são:

- **Application layer** -> **llopen** -> **set_serial_port** -> **statemachine**
- **Application layer** -> **llclose** ->**statemachine_close** -> **close_serial_port**
- **Application layer** -> **llread** -> **states**

Protocolo de ligação lógica

O "link layer" é um da camada do qual a "application layer" depende. O "link layer" é responsável pelas seguintes funcionalidades: Estabelecer e terminar uma ligação através da porta de série, criar e enviar comandos e mensagens(informação) através da porta de série aberta. Para o fazer o "link layer" e as suas respectivas funções fazem "stuff" e "destuff" de pacotes de dados que são criados /dados por informação proveniente da "application layer".

A função "llopen" é a responsável por estabelecer uma ligação à porta de série. Quando o emissor chama esta função, é enviado o comando SET através da porta de série e aguarda a resposta do "receiver"(comando UA). Se não chegar uma resposta e o tempo de time out definido for excedido, recorrendo ao alarme, é feita uma nova tentativa e o comando SET é reenviado. Este ciclo repete-se até chegarmos a um determinado limite de tentativas. Se UA for ser

recebido como resposta do envio do comando SET,então a ligação foi estabelecida.

A função "llwrite" recebe um buffer com a informação a escrever para a porta de série e fica a aguardar a receção de uma resposta. Caso uma resposta não seja recebida num intervalo de tempo pré definido em time out, é feita uma nova tentativa de envio da mensagem. Quando uma resposta for recebida, caso essa resposta seja o comando RR, a mensagem foi transmitida correctamente;

A função "llread" tenta receber uma mensagem através da porta de série. Caso a mensagem recebida seja uma mensagem de informação, essa informação é guardada e o comando RR é enviado através da porta de série.

A função "llclose" é responsável por terminar a ligação através da porta de série. Se o emissor chamar esta função, o comando DISC é enviado pela porta de série, aguarda pela receção do comando DISC, e finalmente envia o comando UA. O recetor aguarda pelo comando DISC, quando o receber reenvia-o e aguarda pela receção do comando UA. Por fim a ligação através da porta de série é terminada

- **State machines**

As máquinas de estado do nosso programa trabalham todas com um "switch case".Todas elas têm como base um "pointer" para um estado inicial , que recebem como argumento, que é alterado para o estado correto através dos "cases", dependendo do valor recebido no byte passado como argumento. Para além das óbvias diferenças no que toca aos switch, é de notar que na função "state", função auxiliar no "destuffing" ,para além das mudanças de estados, a máquina também ajuda-nos a encontrar determinados bytes de forma a nos auxiliar no "destuffing" acionado diversas flags passadas como argumentos sem que haja alteração de estados.

Protocolo de aplicação

Dentro da função "applicationlayer" guardamos os parâmetros da conexão numa variável "dl_layer" sendo esta passada para a função "llopen". Através de uma estrutura condicional if conseguimos determinar que a aplicação que está a correr é ou "transmitter" ou "receiver".

- **Receiver**

É aberto um ficheiro onde serão escritos novos dados. Também criamos um array onde serão escritos os bytes recebidos denominado por packet. Usando o packet como argumento chamamos a função da "llread". Prosseguimos com um

ciclo while, que vai lendo da “porta da serie” pacotes de controlo e de informação.À medida que vamos percorrendo este ciclo vai sendo escrito no packet os caracteres lidos da porta da série. Para terminar é chamada a função “llclose” que termina a ligação recetor e emissor.

- **Transmitter**

O transmitter ao abrir o ficheiro obtém o tamanho desse mesmo ficheiro. Enviamos informação sobre o tamanho nos pacotes de controlo que usando a função “llwrite” são enviados para o receptor. Lemos o conteúdo do ficheiro um byte de cada vez guardando num pacote de dados ,repetindo este processo num ciclo while de maneira a que toda a informação do ficheiro fique num pacote de dados que será enviado para a função “llwrite”.

Validação

Finalmente averiguamos o sistema de controlo de erros. Como tal, interrompendo a ligação porta série conseguimos verificar que ocorre um timeout do transmitter pois não é recebida a trama que confirma a receção desta pelo receiver. Assim sendo temos um print que mostra ao utilizador que foi dado um timeout. Reagindo ao timeout o transmitter tenta enviar a trama por uma segunda vez esperando uma receção da parte do receiver continuando a prosseguir da mesma maneira até atingir o número máximo de retransmissões(no nosso caso, 3).
Quanto aos testes da funcionalidade do nosso código,foi testado unicamente com o “penguim.gif” dado pelos professores. Quando terminado de executar o nosso código , executando o comando “diff” foi possível ver que o ficheiro recebido era igual ao enviado.

Conclusão

Através deste relatório descrevemos como foi feita a estrutura, organização e implementação do nosso trabalho. Este trabalho fez-nos ganhar e ter uma melhor noção de que como funcionam os sistemas de protocolos de ligação de dados, os seus maior problemas e erros.