Faculdade de Engenharia da Universidade do Porto

Data Link Layer Protocol



Projeto 1 – Grupo F

Redes de computadores

Maria Pimentel Pestana Cardoso Ribeiro – up201703844

Miguel Filipe Santos Silva – up201904672

Índice

[Índice 2](#_Toc812311022)

[Sumário 3](#_Toc272900014)

[Introdução 3](#_Toc1537705396)

[Arquitetura e Estrutura do código 3](#_Toc376925325)

[Casos de uso principais 4](#_Toc1847984787)

[Protocolo de ligação lógica 4](#_Toc1816162493)

[Validação 6](#_Toc313321057)

[Elementos de valorização 6](#_Toc2051676908)

[Conclusões 6](#_Toc1942110073)

[Anexos 6](#_Toc2118850478)

# Sumário

No âmbito do primeiro trabalho laboratorial de Redes de Computadores foi realizada a transferência de dados de um computador para outro através de uma porta de série assíncrona.

Com este trabalho foi possível realizar uma transmissão de dados e a utilizar mecanismos de deteção e controlo de erros de modo a parar a transmissão se forem detetados erros ou realizar as ações necessárias.

# Introdução

Neste trabalho foi pedido para implementar um protocolo de ligação de dados através da porta de série utilizando o protocolo Stop & Wait de modo a detetar erros e anomalias. Desenvolvemos diversas funções de deteções de erros para que seja possível evitar falhas de receção de tramas.

# Arquitetura e Estrutura do código

No nosso código temos uma única API para criar a LinkLayer, que está definida numa única biblioteca, linklayer.c. Nesta biblioteca temos implementadas 4 funções, llopen(), llwrite(), llread() e llclose(), que nos permite comunicar entre dois nós através da camada de aplicação.

A função de abertura e configuração da ligação é realizada na primeira função, llopen(), onde inicialmente alteramos as configurações da porta série pretendida. Esta função pode ser chamada de duas formas: “TRANSMITTER” e “RECEIVER”.

Implementamos também as funções llread() e llwrite() que realizam a leitura e a escrita da informação. No caso da primeira inicialmente descodifica a informação presente nas tramas e envia uma confirmação dessa receção ao contrário da llwrite() que codifica a informação e posteriormente escreve e envia a trama.

Todas as funções descritas anteriormente utilizam a Struct linkLayer, que nos permite receber valores cruciais definidos pelo main.c, tais como a porta de série, número de tentativas, timeout e a role do programa.

Como auxiliar às 4 funções principais, temos uma função wait\_for\_answer(), que é responsável por ler os dados iniciais da ligação (HEADER) e retornar o valor do byte de controlo lido.

# Casos de uso principais

Os casos de uso principais são utilizados na inserção de dados pelo utilizador e transmissão de informação que diferem no caso do emissor e do recetor.

No caso do emissor:

* Inicia a ligação entre emissor e recetor através da função llopen(), que envia trama de início de ligação SET e espera receção da confirmação ao receber a trama UA.
* Se a ligação for iniciada corretamente, o emissor passa à função llwrite() para enviar as tramas de informação com a mensagem desejada.
* Quando terminar de enviar, fecha a ligação através do envio de uma trama DISC.

O emissor terá um timeOut, que ativa quando não recebe confirmação do recetor.

No caso do recetor:

* Inicia ligação através do llopen(), ficando à espera da receção da trama de início de ligação SET, e envia a trama UA, como confirmação de receção.
* Se a ligação for iniciada corretamente, passa à função llread(), que espera receção da informação e retorna à aplicação.
* Quando terminar de receber a informação, recebe uma trama DISC, envia outra trama DISC de volta e recebe UA como confirmação, para poder fechar a ligação.

# Protocolo de ligação lógica

O nosso protocolo de ligação, tem várias funções:

* Iniciar uma ligação, com o uso de tramas de Supervision e Unnumbered através de tramas SET e UA;
* Enviar e receber informação através de tramas de Informação;
* Usar bits de paridade, BCC2, para verificação de erros na receção;
* Usar stuffing/destuffing, para sabermos quando recebemos o fim da mensagem;
* Reenvio de mensagem em caso de TimeOut ou erro.

Estando estes objetivos divididos pelas seguintes funções:

**Wait\_for\_answer():**

Esta função é uma função genérica de receção do HEADER. Esta função pode ser chamada de duas formas distintas. Se for como Transmitter, iniciamos o timer e depois passamos à receção de dados. Para verificar a receção dos dados, verificamos a primeira FLAG, verificamos o BCC1 e verificamos o Address.

No fim da função retornamos o valor do Control se correr tudo bem, e se for erro retornamos -1.

**Llopen():**

A função é responsável por abrir a conexão usando parâmetros “port” definidos na linkLayer. Inicialmente colocamos os connection parameters em “ll” pois permitem usar o ladder ao longo de todo o programa. Posteriormente procedemos à verificação da abertura da porta que, caso não ocorra, é necessário chamar a função llopen() novamente. Também é nesta fase do programa que ocorre o pedido do BaudRate.

Utilizámos a variável role para definir se se trata do “TRANSMITTER” ou “Receiver”. No primeiro caso definimos um vetor??? onde colocamos a trama a enviar para criar a ligação que será comprovada pela recessão de flags com funcionalidades distintas. Após a definição de todos os parâmetros e variáveis procedemos à implementação de um código sequencial que limita a 3 retransmissões da trama. Usamos a função wait\_for\_answer() para ler o header e se receber a trama “UA”, lemos a última FLAG. Caso haja erro, retransmitimos. No segundo caso, o receiver vai estar sempre apto a ler o próximo independentemente da ocorrência de erros. Ao ser verificada a trama de recessão “Set” podemos concluir que o header foi bem recebido e procede à leitura da informação até que a última Flag seja lida, reenviando a confirmação caso esta esteja correta. Tal deteção de erros é realizada pelo BCC1.￼

**LLREAD()**

Utilizamos esta função para realizar a leitura das tramas.

A sua funcionalidade é ler a informação recebida onde também realiza verificações de erros na data utilizando o BCC2. Inicialmente esta função permanece num ciclo enquanto espera pela receção da trama que a função read() irá retornar. Após esta recessão irá ser descodificada a informação presente na trama e aguarda a receção da última flag, a primeira que indica o início da receção e a segunda define o final. Também verifica o comando SET e envia a trama UA.?????

Enviamos uma trama de controlo RR com o número 1 em caso de sucesso da transmissão, isto é, sem erros e permite então a saída do ciclo. No caso de existirem erros é enviada a trama REJ com esta informação para que a informação seja enviada de novo.

**LLWRITE()**

Esta função é utilizada para escrever informação (data).

Enviamos uma trama com controlo 0 ou 1 e recebemos uma trama RR ou REJ, com controlo 1 ou 0, respetivamente.

Depois definimos o bit de paridade da informação a enviar, e fazemos byte stuffing para uma variável *stuffed[].* Depois, escrevemos os dados e esperamos a confirmação.

Na receção, se for detetado algum erro, ou recebermos REJ, temos de reenviar a trama. Caso recebamos o RR, sem erros, a função foi bem concluída.

**Llclose():**

No final é necessário encerrar a conexão. Para isto é necessário verificar, como anteriormente, se se trata do “Transmitter” ou “Receiver”. O transmitter verifica o número de retransmissões. Verifica se recebemos o header corretamente e ao confirmar a recessão da última Flag podemos proceder à escrita dessa confirmação. Para o Receiver verificamos também a receção correta do header, Disc, e da última trama, UA.

Após todas as verificações descritas encerramos a conexão e imprimimos as estatísticas do programa.

# Validação

(descrição dos testes efetuados com apresentação quantificada dos resultados, se possível)

Ver no domingo

# Elementos de valorização

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

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

# Conclusões

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

O objetivo principal deste trabalho é que a implementação do protocolo de Ligação de dados fosse independente da camada de aplicação para que, ao alterar uma das camadas, não altere a outra ou que as alterações sejam mínimas. Tal objetivo foi atingido com sucesso. Foi por isso mantida a independência de camadas entre a camada de aplicações e a camada de ligações sendo que a camada de aplicação é responsável pela criação, interpretação de dados e o cabeçalho do pacote de dados. Já a camada de ligação é apensas responsável pela coesão das tramas e o envio das mesmas.

Tendo isto em conta, a realização deste trabalho permitiu que aprofundássemos e puséssemos em prática os conhecimentos adquiridos na unidade curricular

# Anexos

- Anexo I - Código fonte  
- Outros anexos, se desejável