Universidade do Porto

FEUP Faculdade de Engenharia

PROJETO 1

Redes de Computadores

Trabalho realizado por:
Pedro Valente (UP201904865)
Joaquim Oliveira (UP201908075)

Sumário

1 Introdução

O presente projeto teve como motivação a implementação de um protocolo de ligação de dados, através do uso de C em Linux, com portas-série RS-232, através de comunicação assíncrona. O objetivo foi estabelecer uma comunicação de dados fiável entre dois sistemas ligados por cabo, por meio de um canal de transmissão. Iremos testar esta implementação através da passagem de uma imagem de um dispositivo para o outro. O protocolo apresenta: Dados organizados em tramas, estabelecimento e terminação de ligação, numeração de tramas, confirmação positiva de receção de trama, controlo de erros através de mecanismos de Stop-and-Wait, Go-back-N e Selective Repeat. Também irá garantir transparência, isto é, transmissão de dados que será independete de código e os dados serão protegidos por mecânismos de byte stuffing.

2 Arquitetura

Na codificação, para as tramas, foi usado **unsigned char*** para a representação de tramas em memória.

2.1 Máquinas de Estados

Para lidar com a receção de tramas e descodificação das mesmas, tanto a nivel das tramas de supervisão pacotes de dados, e tramas de informação, foram utilizadas máquinas de estados para fazer a constante verificação se o protocolo estabelecido está a ser cumprido ou se aconteceu algum problema. No código, estão presentes nos ficheiros stateM_data.c e stateM_lib.c as respetivas máquinas de estados. Estas foram criadas como bibliotecas para facilitação de código.

2.2 Protocolo de ligação de dados

A camada de ligação de dados visa a tratar das ligações, escrita e leitura de tramas de um dispositivo para o outro e fazer o controlo se a o protocolo está a ser cumprido. O ficheiro utilizado foi o link.c.

2.3 Camada da aplicação

A camada da aplicação encontra-se presente no ficheiro app.c. Esta camada lida com a chamada de funções de inicialização, llopen(), leitura, llread(), escrita, llwrite() e de fecho de transmissão, llclose(), e não pode saber a forma como funciona o protocolo de ligação de dados, apenas podendo fazer chamadas às funções fornecidas pelo mesmo.

2.4 Byte Stuffing

O programa byteStuffing.c trata do byte stuffing, que é corrido no protocolo de ligação de dados. Está criado como uma biblioteca para facilitação de código.

3 Estrutura do código

3.1 Estruturas de dados

Para este trabalho foram criadas duas estruturas de dados, o applicationLayer, presente na biblioteca app.h, de forma a facilitar e organizar o applicationLayer. Já para o protocolo, foi criado o linkLayer, na biblioteca link.h, também para o mesmo propósito.

3.2 Código das máquinas de estados

Em relação ao programa stateM_lib.c, dispomos de duas funções importantes, stateM_SET() e stateM_UA(). A primeira, recebe um estado, que neste caso seria um byte, e vai guardando em variáveis globais os estados da máquina caso passe na mesma, o que permite percorrer para uma sequência de dados e encontrar tramas. Caso esteja a precorrer a trama e esta for inválida, esta irá dar reset. Esta função correrá para tentar perceber se temos um SET válido. Analogamente, a segunda função irá desempenhar o mesmo papel, mas para o UA em vez do SET. Já no programa stateM_data.c, está presente uma máquina de estados, função stateM_data, com o objetivo de verificar se, na informação recebida, a trama de dados está de acordo com o protocolo, através, mais uma vez, de variáveis globais e verificando em cada passo se o estado corresponde ao que é suposto receber. Se por alguma razão ocorrer um erro, a máquina de estados irá dar reset.

3.3 Código do Byte Stuffing

A função byte_stuff(), tem como objetivo alterar os bytes 0x7d e 0x5e pois isto poderia fazer com que a leitura da trama terminasse mais cedo do que aquilo que na verdade é. Esta percorre a trama inteira e vai alterando até terminar. Já a função byte_destuff() faz exatamente o contrário, para poder ser lida a informação original na receção.

4 Casos de uso principais

Como caso de uso, foi testada a transferência de um ficheiro gif de um computador para o outro, tendo um dos computadores o papel de emissor e o outro o de recetor. É necessário o emissor selecionar a porta série. Os principais passos são :

- O emissor e recetor têm de selecionar a respetiva porta série e que papel irão desempenhar;
- A função main, presente na camada da aplicação, corre e inicia a conexão com llopen;
- Os dados são enviados através da função llwrite e recebidos pelo recetor através de llread;
- O pacote de controlo é enviado seguido de vários pacotes de dados;
- A transferência é concluida com outro pacote de controlo;
- No final, a conexão é terminada pelo llclose.

5 Protocolo de ligação lógica

O protocolo de ligação é a camada em que está toda a lógica, funções e variáveis necessárias para a implementação do protocolo corretamente, esta é a responsável pela leitura e escrita de dados, gerenciação de timeout e erros. São implementadas quatro funções principais: **llopen**, **llclose**, **llread** e **llwrite**.

As funções llopen e llclose, têm o objetivo de estabelecer e terminar a ligação de dados. Sendo a llopen, a função gera uma SET e envia-a para a porta série, aguardando 3 * 3 segundos pela resposta UA, gerada pelo outro dispositivo. Caso este tempo seja ultrapassado o processo termina. Já a função llclose envia um **DISC**, informando o receiver de que a conexão será terminada, e ficando à espera de seguida de uma trama UA, terminando o processo e concluindo-se a execução do programa. Já o llwrite recebe um pacote de dados, ou de controlo de dados, e transforma-o numa trama de informação, enviando-o de seguida para a porta série, ficando à espera de uma

resposta por parte do recetor. Depois de receber a resposta, e se receber um **RR** (Receiver Ready), retorna o numero de bytes que passou. Se por outro lado receber um **REJ** (Reject), tenta enviar a mesma trama novamente. Por fim, o llread recebe uma trama da porta série. De seguida, aplica-lhe byte destuffing e verifica se esta está correta através das máquinas de estado. Se estiver, envia um **RR**. Caso se trate de uma trama duplicada, descarta a nova e volta a tentar enviar o **RR**.

6 Protocolo de aplicação

O protocolo de aplicação é a camada na qual o utilizador interage com o protocolo de ligação, o utilizador não sabe como este está implementado apenas como o utilizar. Inicialmente é feita a verificação de que tipo de modo é que estamos, t = transmissor e r = recetor, após se saber em que modo se está, é feito o llopen(), após esse retornar 0, se estivermos como emissor é aberto o ficheiro passado como argumento na inicialização do programa, se o ficheiro for aberto com sucesso então é feita a fragmentação com o tamanho defenido em DATASIZE. Antes de começar a enviar os dados do ficheiro é feita a geração de um pacote de controlo através da função controlpackagegen() em que recebe como parâmetros o nome do ficheiro, o tamanho do ficheiro e um 1(inicio), e enviada com o llwrite(). Agora é feito um ciclo para percorrer os fragmentos do ficheiros e gerar um pacote de dados individualmente, este pacote é gerado pela função datagenpackage() e enviado pelo llwrite(), após todos os dados do ficheiro terem sido mandados é feita uma trama de controlo novamente mas agora é passado um 0 em vez de um 1 para sinalizar que é uma trama de finalização. Por fim é encerrada a transmissão com o llclose(). No lado do recetor, após o llopen(), é abero um ficheiro vazio e feito um llread(), após este returnar com o número de bytes lido, ou seja, != -1, é feita um llclose() e fechada a conecção.

7 Validação

Para efeitos de validação foi feito uma ligação por ssh ás bancadas do laboratório, conforme é mostrado na imagem é possível verificar o programa a funcionar como esperado.

8 Eficiência do protocolo de ligação de dados

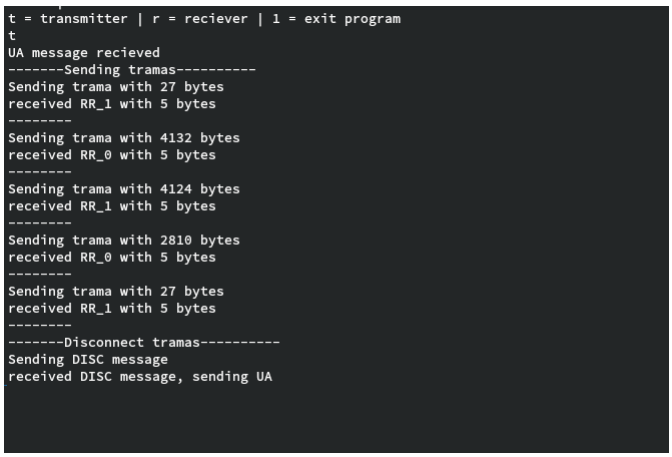


Figura 1: Emissor

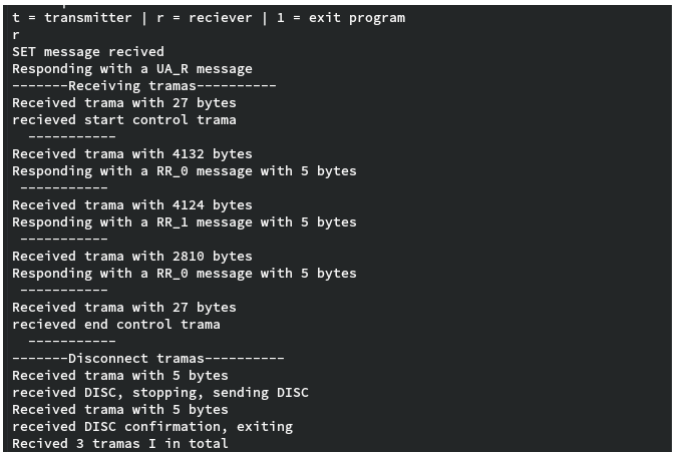
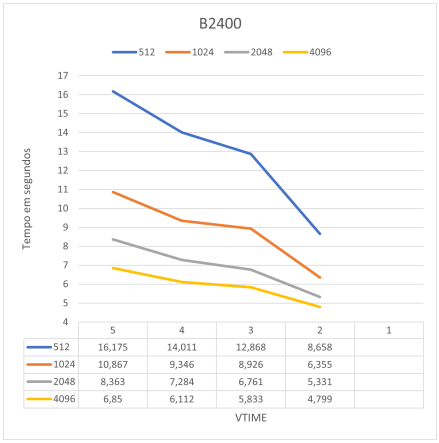


Figura 2: Receiver

9 Eficiência do protocolo de ligação de dados

9.1 Variar VTIME



	5	4	3	2	1
512	16,175	14,011	12,868	8,658	
1024	10,867	9,346	8,926	6,355	
2048	8,363	7,284	6,761	5,331	
4096	6,85	6,112	5,833	4,799	

	5	4	3	2	1
512	16,354	13,929	12,978	8,657	
1024	11,053	9,421	9,101	6,435	
2048	8,7	7,219	6,867	5,317	
4096	6,613	6,025	5,717	4,713	

	5	4	3	2	1
512	16,201	14,023	13,269	8,703	
1024	10,801	9,43	8,826	6,358	
2048	8,549	7,224	6,623	5,396	
4096	6,654	5,997	5,52	4,727	

10 Conclusões

Devido há nossa implementação não conseguiamos quantificar o tempo de execução do llread() o que levou a não descobrirmos a efeciência do protocolo.

Em suma, o objetivo do projeto foi concluido com sucesso, todos os pontos propostos conseguiram ser alcançados menos a apresentação dos gráficos de variação de parametros.

Este foi importante na nossa aprendizagem para perceber a forma como funciona a transferência de dados e aplicação de protocolos, tendo sido uma mais valia para os nossos conhecimentos teórico-práticos de redes de computadores.