Protocolo de Ligação de Dados

Relatório do 1º trabalho laboratorial

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Grupo xx:
Francisco Rodrigues - 201305627l
João Nogueira - 201303882
Marta Lopes - 201208067

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

29 de Outubro de 2015

Conteúdo

1 Sumário

Este relatório tem como objetivo explicar o primeiro projeto, realizado para esta unidade curricular, denominado "Protocolo de Ligação de Dados". Este projeto consiste no envio de informação de um computador para outro, através do uso de porta de série. Foram assim implementados programas para ler e escrever a informação a ser enviada.

O projeto foi finalizado com sucesso, sendo que os dados foram enviados e recebidos de forma correcta. Foram também incluidos a prevenção e correção de erros ao longo da transmissão, restabelecendo a trasmissão quando os erros acontecem.

2 Introdução

O principal objetivo deste trabalho, que foi realizado ao longo das aulas laboratoriais de Redes de Computadores, consiste em implementar um protocolo de ligação de dados, de acordo com a especificação descrita no guião, e também testar o protocolo com uma aplicação simples de transferência de ficheiros, igualmente especificada. O ambiente de desenvolvimento utilizado foi em PC's com *Linux*, a linguagem de programação foi C e as portas de série existentes realizavam comunicação assíncrona.

O protocolo de ligação de dados pretende assim fornecer um serviço de comunicação de dados fiável entre dois sistemas ligados por um cabo de série. As funções utilizadas serão a de criação e sincronismo de tramas que irão organizar os dados a ser enviados (*framing*), a do estabelecimento/conclusão da ligação, a numeração de tramas, o controlo de fluxo, a confirmação de envio sem erros e o controlo dos erros que poderão ser criados por *time-outs*, tramas fora da sequência esperada ou retransmissões.

Sendo assim, este relatório servirá para descrever o nosso trabalho de uma forma mais teórica para assim poderem ser avaliados certos pormenores que não seriam possíveis de avaliar na apresentação do projeto.

O nosso relatório terá então as **secções principais** seguintes:

- **Arquitetura**: especificação dos blocos funcionais e da *interface*;
- **Estrutura do código**: API's, principais estruturas de dados, principais funções e a sua relação com a arquitetura;
- **Casos de uso principais**: identificar os principais aspectos, abordando as sequências de chamadas de funções;
- **Protocolo de ligação lógica**: identificar os principais aspectos funcionais da *linkLayer*, descrevendo a estratégia de implementação;
- **Protocolo de aplicação**: identificar os principais aspectos funcionais da *applicationLayer*, descrevendo a estratégia de implementação;
- **Validação**: testes efetuados ao programa com apresentação de resultados;
- **Elementos de valorização**: identificação dos elementos implementados, descrevendo a estratégia de implementação.

3 Arquitetura

(blocos funcionais e interfaces)

4 Estrutura do código

(APIs, principais estruturas de dados, principais funções e sua relação com a arquitetura)

5 Casos de uso principais

(identificação; sequências de chamada de funções)

6 Protocolo de ligação lógica

O protocolo de ligação lógica está implementado na camada *linkLayer*, camada da qual depende a camada *applicationLayer*.

Principais aspectos funcionais:

- Configurar a porta de série como é pretendido;
- Repôr a configuração da porta de série como originalmente após a transferência dos dados pretendidos;
- Estabelecer a ligação de dados utilizando a porta de série;
- Enviar comandos;
- Enviar/receber mensagens;
- Processo de *Stuffing* e *Destuffing* dos *packets* recebidos da camada *applicationLayer*.

Funções implementadas na *linkLayer*:

1. **llopen**

   Esta função é responsável por configurar a porta de série com as opções pretendidas (utilizando, por exemplo, o *baudRate* escolhido pelo utilizador) e por guardar numa variável que passa por argumento à função *configure*. Depois de configurar a porta de série, do lado do emissor envia o comando SET e aguarda a resposta UA do recetor que, ao ser recebida terimina a função. Utiliza o alarme para controlar os *timeouts*. Do lado do recetor aguarda a recepção do comando SET e, ao recebê-lo responde com o comando UA.

2. **llclose**

   Do lado do receptor, esta função começa por aguardar a recepção do comando DISC, respondendo com o mesmo comando. Após o envio deste comando, e imediatamente antes de reestabelecer as configurações originais da porta de série aguarda a recepção do comando UA por parte do emissor. Do lado do emissor, esta função começa por eviar o comando DISC e aguarda pela resposta com o mesmo comando por parte do receptor (implementando o alarme para manter controlo dos timeouts). Após a correta recepção de DISC evia o último comando do programa, comando UA. Termina por fazer *resetConfiguration*, função na qual é feito um *sleep* de um segundo por forma a garantir que as configurações originais da porta de série não são restabelecidas antes de que toda a informação tenha sido passada.

3. **llwrite**

   Esta função começa por alocar memória num *buffer* no qual a informação será organizada (antes do processo de *stuffing*). Após alocar a memória necessária começa por atribuir os valores de FLAG, A, C e BCC de acordo com o número de sequência da trama. A partir de BCC é copiada para o buffer toda a informação recebida por argumento desta função da *applicationLayer* correspondente ao *packet* a enviar. Antes de terminar o preenchimento do *buffer* coloca o BCC2 ('ou' exclusivo dos octetos do *packet*) e a FLAG para terminar. Depois de ter a trama preenchida e antes de começar o processo de envio, envia o *buffer* para a função *stuff* que retorna um *buffer* com a trama pronta para envio. Envia a trama através da porta de série e aguarda a resposta por parte do recetor, o que vai determinar o caminho a seguir pela função, que pode variar entre terminar a função pois houve a recepção de RR, ou re-enviar a mesma trama por ter sido rejeitada. A mesma trama pode ser re-enviada por ocorrência de *timeout*, o que também é controlado nesta função através da implementação de alarmes. Esta função pode também terminar pela ocorrência de *timeouts* maior do que o número máximo definido previamente pelo utilizador.

4. **llread**

   A função *llread* começa por alocar memória para o *buffer* que vai ser recebido, entrando imediatamente a seguir num ciclo do ual apenas sai quando algo é lido da porta de série. Após a leitura o *buffer* lido é passado para a função responsável pelo processo de *distuffing* que retorna a trama de Informação recebida "descodificada". Os valores de BCC e de BCC2 são verificados por forma a garantir que a trama foi recebida sem erros, caso o tenha sido é interpretado o número de sequência e se tudo for o pretendido é enviado o comando RR, caso contrário é enviado o comando REJ, pedindo que a mesma seja enviada novamente.

7 Protocolo de aplicação

(identificação dos principais aspectos funcionais; descrição da estratégia de implementação destes aspectos com apresentação de extractos de código)

8 Validação

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

9 Elementos de valorização

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

10 Conclusões

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

11 Anexos