



|  |  |
| --- | --- |
| Rede de Computadores  **Protocolo de Ligação de Dados**    3MIEIC05 - GRUPO 206 |  |
| Isla Patrícia Loureiro Cassamo | 201808549 |
| Iohan Xavier Sardinha Dutra Soares | 201801011 |

**Índice**

[Sumário 3](#_Toc56350364)

[Introdução 3](#_Toc56350365)

[Arquitetura 3](#_Toc56350366)

[Estrutura do código 3](#_Toc56350367)

[Principais interfaces 3](#_Toc56350368)

[Estruturas de dados 4](#_Toc56350369)

[Funções Importantes 4](#_Toc56350370)

[Casos de uso principais 5](#_Toc56350371)

[Protocolo de Ligação Lógica 5](#_Toc56350372)

[Protocolo de Aplicação 7](#_Toc56350373)

[Validação 8](#_Toc56350374)

[Eficiência do protocolo de Ligação de dados 9](#_Toc56350375)

[Conclusão 9](#_Toc56350376)

# Sumário

O projeto “Protocolo de Ligação de Dados” procura pôr em prática uma possível abordagem à forma como é efetuada a troca de informação entre dois computadores ligados por cabo série com o uso de protocolos para garantir uma transferência segura e confiável. Para isso foi desenvolvida uma aplicação em C que transferiria arquivos entre dois terminais conectados através de um cabo serial.

//Principais conclusões do relatório.

# Introdução

O projeto teve como objetivo implementar um protocolo de ligação de dados baseado em protocolos existentes, de modo a que mensagens assíncronas sejam recebidas e tratadas de forma fiável, de forma a contornar as limitações do cabo série quanto a confiança dos dados transferidos.

O relatório foi concebido de modo a complementar o projeto elaborado e melhor explicar o funcionamento do mesmo, após ser terminado com sucesso em ambiente real. Assim como detalhar funcionalidades e estatísticas do projeto que podem não ser percetíveis em uma primeira análise de uso.

// descrição da lógica do relatório com indicações sobre o tipo de informação que poderá ser encontrada em cada uma secções seguintes

# Arquitetura

A aplicação de transferência de dados foi organizada de modo a representar o princípio de dependência entre as duas camadas: **Aplicação** e **Ligação de Dados.**

A camada da **Aplicação** é a camada de alto nível que serve de interface com o utilizador. É nesta camada que são implementados os métodos que operam com os ficheiros de de dados tendo esta nenhum conhecimento sobre detalhes dos processos, e apenas usufrui dos serviços fornecidos pela camada de Ligação de Dados.

A camada de **Ligação de Dados** é a de mais baixo nível, onde estão implementadas as funções genéricas que realmente tratam do envio e receção de informação, pelo que é efectivamente é quem segue o protocolo de ligação de dados.

# Estrutura do código

## Principais interfaces:

O código se divide, então, em sete interfaces, por motivos de organização e simplificação, que se encaixam cada uma em uma das categorias citadas assim. Estas interfaces são:

1. main – Relativa à camada de aplicação, faz a detecção do das entradas do usuário para saber como o programa deve se comportar e chamando as demais interfaces de maneira adequada.
2. transmitter – Relativa à camada de aplicação, chamada pela main, faz as chamadas exclusivas do transmissor. Por questões de organização, para separar do receptor.
3. reciever - Relativa à camada de aplicação, faz as chamadas exclusivas do receptor.
4. packet – Relativa à camada de aplicação, onde se faz a criação e manutenção dos pacotes de dados.
5. ll - Relativa à camada de ligação de dados, faz as chamadas para o envio das tramas de forma a realizar o devido estabelecimento da conexão entre as partes, transferência de dados e desconexão e na ordem correta.
6. frame – Relativa à camada de ligação de dados, onde são efetivamente construídas, enviadas e recebidas as tramas.
7. utils – Tecnicamente não se encaixa em nenhuma das camadas, pois não possui código executado por si só, apenas funções e macros úteis que são utilizadas por ambas as camadas.

## Estruturas de dados:

Quanto aos dados da aplicação, são utilizados *enum’s* para guardar as máquinas de estado para o recebimento de dados, já os valores mutáveis de *time-out*, número máximo de tentativas e modo de *debug* ativado, que precisam ser acessados por diferentes partes independentes do código, são guardados em variáveis globais. Todas as outras informações importantes são passadas através de parâmetros de funções.

## Funções Importantes:

Quanto a camada da Aplicação destaca-se:

* recieverMain – Realiza o ciclo principal do receptor, enviando os pacotes de controlo e parseSendPacket para fazer a inscrição em arquivo dos dados recebidos.
* transmitterMain – Realiza o ciclo principal do transmissor, efetuando o envio dos pacotes de controlo e chamando transmitData para fazer a transmissão dos dados.
* transmitData – Lê os dados do arquivo, dividindo-os em pacotes para fazer o envio.

send\_controll\_packet – Faz a criação do pacote de controlo dados as informações quanto ao arquivo, recebidas como parâmetro. E o envia.

* parseSendPacket – Verifica os dados recebidos do pacote para fazer as operações necessárias, seja de escrever no arquivo os dados recebidos, seja de abrir ou fechar o arquivo recebido um pacote de controlo.

Quanto a camada de Ligação de Dados

* llopen – Faz os envios e recepção de tramas, de acordo com o papel de cada terminal, para garantir que a conexão entre os computadores está estabelecida.
* llwrite – Envia uma trama de dados, com os dados recebidos como parâmetro.
* llread – Faz a correta leitura dos dados de uma trama de dados, retornando por referência os dados lidos.
* llclose – Faz os envios e recepção das tramas, de acordo com o papel atual, para fazer a desconexão dos computadores de maneira adequada.

# Casos de uso principais

Os principais casos de uso são aqueles que incluem transferência de dados de um computador transmissor e um receptor e a interface que possibilita a escolha do ficheiro a ser transferido.

O fluxo da transmissão de dados e as funções principais associadas pode se verificar abaixo:

1. main e validateArgs - O utilizador escolhe o ficheiro a ser enviado, e a porta por onde quer enviar, o papel do computador (transmissor ou receptor) e seus parâmetros opcionais.
2. main - Configuração de uma ligação entre receptor e transmissor e sua respectiva verificação através da função llopen.
3. main - Transferência dos dados, esta etapa depende do papel do computador:
   1. Transmissor usando llwrite:
      1. transmitterMain: Envia o pacote de controlo START
      2. transmitData: Envia os dados do arquivo divididos em pacotes
      3. transmitterMain: Envia o pacote de controlo END
   2. Receptor usando llread:
      1. recieverMain: Lê os pacotes recebidos.
      2. parseSendPacket: Abre o arquivo com os dados recebidos do pacote de controlo START
      3. parseSendPacket: Recebe pacote a pacote e escreve os dados no arquivo.
      4. parseSendPacket: Fecha o arquivo.
4. main - Término da ligação pela função llclose.

# Protocolo de Ligação Lógica

O protocolo de ligação lógica pode ser exemplificado através das tramas de comunicação que representam cada etapa de seu processo como será demonstrado a seguir acompanhado de fragmentos de código. Essas tramas podem ser do tipo de supervisão e informação, essa segunda carregando um campo de dados. Todas elas possuem um campo de controlo que indica qual a função dessa trama, podendo esses campos serem: SET, UA, DISC, RR e REJ.

Para iniciar é enviado do transmissor uma trama de supervisão SET e então este ficará esperando uma resposta do tipo UA. Já o receptor aguardará a recepção da trama SET, fazendo a verificação da sua paridade para ter certeza de que o envio não teve comprometimentos, e recebendo-a enviará a resposta UA. Caso a resposta não chegue depois de passado um tempo pré-definido, de qualquer que seja os lados, é enviada uma resposta negativa a camada de aplicação, comunicando que não foi possível estabelecer a comunicação. Esta etapa garante que a porta série está devidamente funcionando e que há comunicação entre as partes.

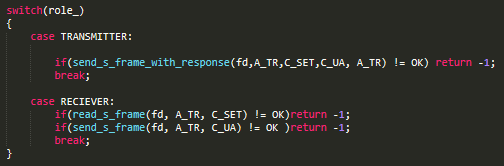


Figura 1- Envio e recepção de tramas de supervisão para estabelecimento da ligação

Uma vez estabelecida as conexões são enviadas as tramas de informação da parte do transmissor e recebidas da parte do receptor. Cada trama enviada tem um número de sequência associado, que é incrementado no caso da trama ser aceite. O receptor vai ler a trama, fazer a verificação de paridades para garantir que os dados estão íntegros, enviando sempre uma resposta adequada, REJ – se a trama contém erros ou RR – se a trama foi aceita. O número de sequência também é contado do lado do receptor e aumenta quando a trama é aceita, e a resposta RR sempre tem este número já atualizado, associa em seu envio. Quando o transmissor recebe uma resposta REJ ou nenhuma resposta ele reenvia a trama já enviada.

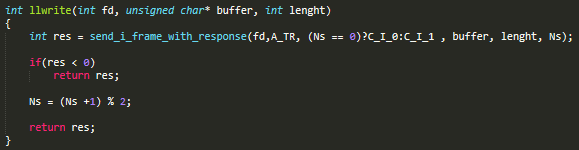


Figura 2- Envio das tramas de informação por parte do transmissor

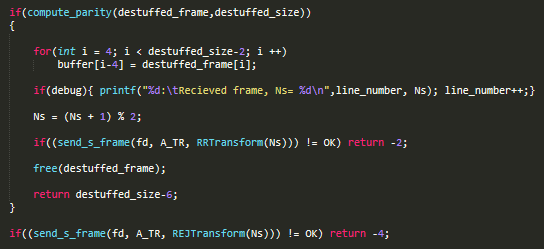


Figura 3- Envio das respostas por parte do recptor

Para encerrar é feito o envio de uma trama DISC pelo transmissor que é respondida pelo receptor com outro DISC, uma vez recebida a resposta o transmissor responde com UA enquanto o receptor aguarda essa reposta para ter certeza que a desconexão foi feita devidamente da sua parte.

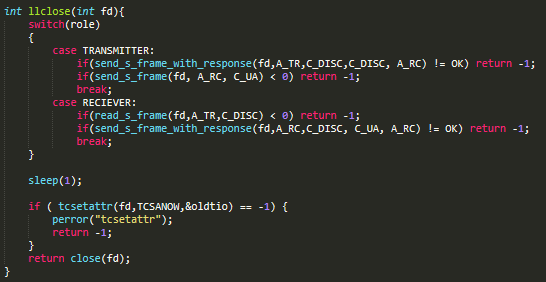


Figura 4 – Envio e recepção das tramas de supervisão para desconexão

# Protocolo de Aplicação

O protocolo de aplicação pode ser exemplificado através dos dois tipos de pacotes e sua composição:

Pacotes de controlo, os quais carregam as informações do ficheiro, para poder abri-lo apropriadamente o receptor. E seu campo de controlo possui a indicação se está a iniciar ou terminar a transferência, START ou END.

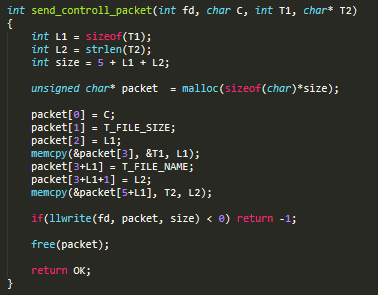


Figura 5- Envio dos pacotes de controlo

Pacotes de dados, possuem além do campo de dados e controlo, este último que vai ser sempre DATA, indicando que aquele pacote é de dados, possui o número de sequência do envio do ficheiro, e campos indicando o tamanho (número de octetos) do campo de dados.

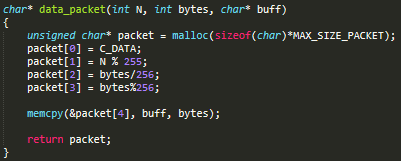


Figura 6 - Contrução do pacote de dados

# Validação

Para garantir que os protocolos funcionavam devidamente foram efetuados testes tanto com os cabos séries quanto com função que simulam erros nestes. Os possíveis erros são: Corrupção dos dados, os cabos série não são confiáveis e podem enviar os dados errado, e o código deveria conseguir reagir a este erro. Interrupções na transmissão, caso houvesse algum tipo de congestionamento no cabo, ou se por algum motivo ele seja desconectado e rapidamente reconectado, em ambos os casos o código deveria conseguir responder corretamente e voltar e enviar dados quando houver reconexão.

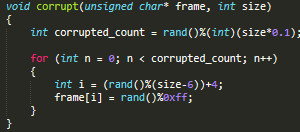


Figura 7- Função que simula a corrupção de dados

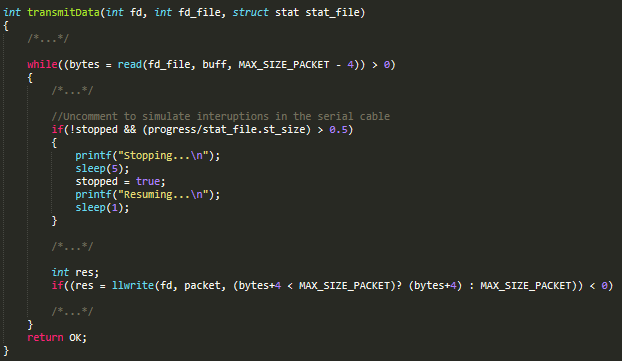


Figura 8- Função transmitData resumida, com o código que simula uma interupção no cabo série

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Time-out emissor | Time-our receptor | Número de tentarivas até retomar | Tempo até retomar funcionamento |
| Interrupção |  |  |  |  |
| Corrupção |  |  |  |  |

# Eficiência do protocolo de Ligação de dados

//TODO

# Conclusão

//TODO