



Trabalho 2: Simulador da camada de enlace

Aluna: Mariana Borges de Sampaio
Matrícula: 180046926

15 de maio de 2021

1. Introdução

- **Camada Física**

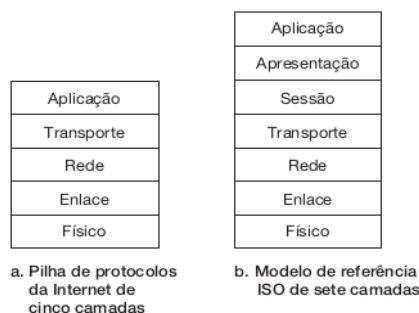
Atualmente uma das formas mais comuns de comunicação ocorre por meio do envio de mensagens pelo telefone, como whatsapp, facebook, instagram dentre outros meios. Porém o que realiza essa comunicação é a Internet. A internet pode ser descrita como “*componentes de software e hardware básico ou ela pode ser descrita em nível de infraestrutura de redes que fornece serviço para aplicações distribuídas*”. De maneira geral a internet pode ser descrita como uma rede de computadores que consegue conectar milhões de dispositivos de computação ao redor do mundo. Os componentes que são ligados a internet são chamados de sistemas finais ou hospedeiros. Esses sistemas finais são conectados entre si por meio de enlaces (links) de comunicação e comutadores (switches) de pacotes.

A comutação de pacote ocorre justamente quando sistemas finais trocam mensagens entre si, essas mensagens podem conter qualquer coisa. Dessa forma para que o sistema final de origem envie a mensagem para o sistema final de destino o originador deve fragmentar a mensagem de forma que ela seja dividida em fragmentos menores, sendo estes conhecidos como **pacotes**. Esses pacotes que são enviados têm que seguir um protocolo, de forma que seja permitida a comunicação entre os sistemas finais. Todas as atividades na Internet que envolvem duas entidades ou mais devem seguir uma série de protocolos. Por definição, tem-se que protocolo pode ser definido como:

“ *Um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento.*”

Como um modelo de referência , tem-se o modelo de camada OSI que é composto por sete camadas, cada camada possui uma função a ser desenvolvida. A camada de aplicação tem por finalidade distribuir a mensagem para os sistemas finais e a aplicação em um sistema final utiliza protocolo para trocar pacotes de informação com a aplicação em outro sistema final. Essa comunicação resulta na troca de mensagens entre os sistemas. Segue abaixo uma imagem que ilustra o modelo de referência OSI e a pilha de protocolos da Internet em cinco camadas. Essa diferença ocorre pois nem todos os sistemas usam as sete camadas exatamente, o modelo OSI é um modelo de referência.

FIGURA 1.23 A PILHA DE PROTOCOLOS DA INTERNET (A) E O MODELO DE REFERÊNCIA OSI (B)



Em cada camada ocorre a transmissão da mensagem , na imagem seguinte pode ser vista o que deve ser transferido em cada etapa. Pode ser notado que é necessário que se tenham dois sistemas como mesmas camadas para que ocorra essa troca de mensagem, isso ocorre pois a troca

de mensagens ocorre entre entidades pares, que são entidades de mesma camada em diferentes máquinas.



Seguindo o modelo de referência tem-se que a camada física é aonde tem-se o início da comunicação, sendo assim tem-se a camada física transmissora e a camada física receptora. A camada física tem como função movimentar os bits individuais que estão dentro de um **quadro**. Quadro é a definição que se tem para os pacotes na camada de enlace. Ou seja, a camada física deve realizar essa movimentação de bits individuais de forma que quando esse pacote seja passado para a camada de enlace, ele tenha formado os quadros. Visto que a camada de enlace tem por finalidade movimentar os quadros inteiros de um elemento de rede até outro elemento.

Sendo assim, com base no que foi estudado, foi desenvolvido um programa que desenvolva a função de um simulador da camada física, de forma que este simulador tenha implementado os seguintes protocolos:

1. Binário
2. Manchester
3. Bipolar

Diante disso, nesta primeira etapa este trabalho tem como intuito realizar a codificação do protocolo da camada física. De forma que seja estabelecida a codificação da mensagem que ocorre na camada física transmissora e na decodificação que ocorre na camada física receptora.

- **Camada de Enlace**

Para a segunda etapa deste trabalho tem-se o desenvolvimento que deve ocorrer para a camada de enlace, ela deve realizar a continuação da camada de protocolo. Esta camada tem como objetivo permitir que a comunicação de forma eficiente e confiável entre as unidades de informação inteira, esta unidade são os **quadros**. Diante disso a camada de enlace de dados deve usar os serviços da camada física para que seja feito o envio e o recebimento de bits através do canal de comunicação. Sendo assim a camada de enlace recebe os pacotes da camada de rede e os encapsula em quadros para que seja feita a transmissão. Cada quadro possui o cabeçalho (header), um campo de carga útil que deve conter o pacote e um final (trailer) de quadro.

Como a camada física recebe o fluxo de bits brutos e deve enviar esse fluxo para a camada de enlace, esta deve verificar se existe algum erro no fluxo de bits para que após isso é que ele seja

enviado para a camada de rede. Caso exista algum erro a camada de enlace deve ser capaz de corrigi-los. Para que isso seja feito, a camada de enlace divide o fluxo de bits que é recebido pela camada física, essa divisão é feita em quadros. Logo, nota-se que a string é o quadro para a implementação.

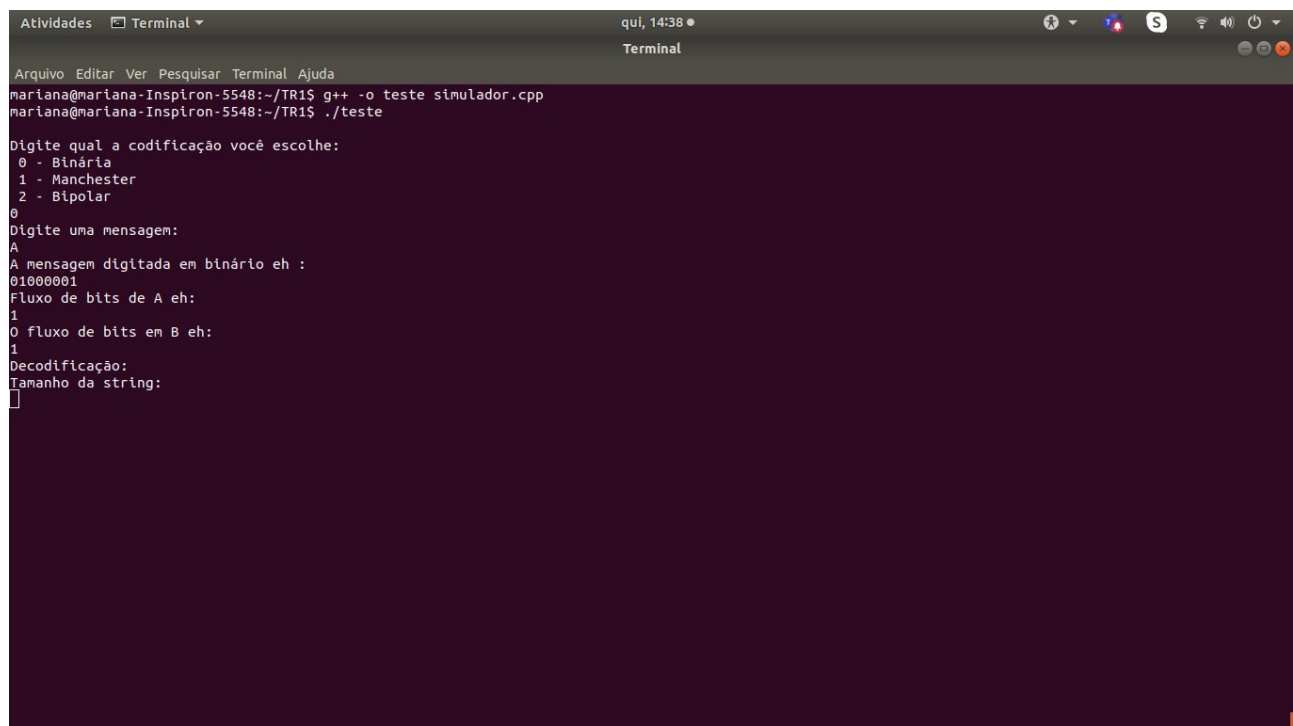
Diante disso, para a segunda etapa deve ser desenvolvido os protocolos para a implementação do enquadramento na camada de enlace.

2. Implementação da Camada Física

Segue a descrição de como cada codificação deve ocorrer e a explicação do seu funcionamento. Segue também como ocorreu a implementação em cada protocolo.

2.1) Conversor de string para binário

Como o transmissor e o receptor podem receber quaisquer tipos de mensagem, neste trabalho foi estabelecido que a mensagem poderia ser um texto, para isso é necessário realizar a conversão de texto para binário. Está será a base para o trabalho, visto que está sendo tratado sobre a camada física, está que é responsável por lidar com o bit individualmente. Como base foi tomado o código ASCII para realizar a transformação do texto para um código binário. Como teste para verificar o funcionamento da conversão de string para binário, foram inseridas as letras A e Z que têm como saída binária os valores '01000001' e '01011010', para verificar se ele converteria uma string e não só o carácter foi inserida a string AZ que tem como saída '01000001 01011010'.



```
Atividades Terminal
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
mariana@mariana-Inspiron-5548:~/TR1$ g++ -o teste simulador.cpp
mariana@mariana-Inspiron-5548:~/TR1$ ./teste

Digite qual a codificação você escolhe:
0 - Binária
1 - Manchester
2 - Bipolar
0
Digite uma mensagem:
A
A mensagem digitada em binário eh :
01000001
Fluxo de bits de A eh:
1
O fluxo de bits em B eh:
1
Decodificação:
Tamanho da string:
1
```

```
Atividades Terminal
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
mariana@mariana-Inspiron-5548:~/TR1$ g++ -o teste simulador.cpp
mariana@mariana-Inspiron-5548:~/TR1$ ./teste

Digite qual a codificação você escolhe:
0 - Binária
1 - Manchester
2 - Bipolar
0
Digite uma mensagem:
Z
A mensagem digitada em binário eh :
01011010
Fluxo de bits de A eh:
1
O fluxo de bits em B eh:
1
Decodificação:
Tamanho da string:

```

```
Atividades Terminal
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
mariana@mariana-Inspiron-5548:~/TR1$ g++ -o teste simulador.cpp
mariana@mariana-Inspiron-5548:~/TR1$ ./teste

Digite qual a codificação você escolhe:
0 - Binária
1 - Manchester
2 - Bipolar
0
Digite uma mensagem:
AZ
A mensagem digitada em binário eh :
01000001 01011010
Fluxo de bits de A eh:
1
O fluxo de bits em B eh:
1
Decodificação:
Tamanho da string:

```

2.2) Codificação Binária

A codificação binária tem como principal intuito a transformação de um número decimal sendo representado por um número binário, ou seja, em 0 ou 1. A codificação binária também é conhecida pela codificação BCD , Binary coded decimal. Tendo isso como base observa-se que a conversão de string para bit corresponde a codificação binária, visto que a string é convertida em binários respeitando a tabela ASCII. A codificação binária foi implementada na função: `CamadaFisicaTrasmissoraCodificacaoBinaria()` e sua decodificação `CamadaFisicaReceptoraDecodificacaoBinaria()`.

2.3) Codificação Manchester

A codificação Manchester tem como intuito permitir que o transmissor e o receptor tenham a mesma sincronia. Essa sincronização ocorre a partir de um sinal pulsado que é utilizado através do sinal de dados junto com o sinal de clock. O clock corresponde ao sinal de relógio, é através do clock que tem-se o sinal que é usado para descrever qual a ação que o sinal deve tomar. No caso deste trabalho, o clock indicará o que o programa deve fazer de acordo com cada fase que ele esteja executando.

Para conseguir realizar a combinação entre o sinal de clock com o sinal de dados, é feita a operação XOR. De modo que ele respeite a combinação entre as operações.

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

O NRZ, Non return to zero, é um código banda base que pode realizar dois tipos de modulação, o unipolar e o polar. No caso do NRZ – Unipolar o que é feito é que quando o sinal recebe o bit 1, ocorre o pulso do sinal e quando o sinal recebe o bit 0 o sinal “perde” o pulso, indo para zero. E o NRZ polar corresponde a quando o bit recebe valor de 1 e recebe um pulso e quando o bit é 0 ele tem um pulso de -1. Para a codificação Manchester o que será utilizado é o fluxo de bits unipolar.

A codificação Manchester através da união do conjunto de bits NRZ com o Clock e entre esses sinais é realizada a operação XOR, ao realizar esta operação é obtida a codificação Manchester. Ela garante que o tanto o transmissor quanto o receptor estejam sincronizados a partir de um sinal pulsado junto com o sinal de clock.

Na implementação foi utilizado a operação XNOR visto que ele é identificado como o inverso da XOR, ao negar ele tem-se a execução da XOR. A mensagem recebida é codificada pelo clock que foi colocado como começando em zero. Ao inserir as variações do clock a operação XNOR é realizada na função. Dessa forma o quadro tem valor duplicado em relação ao que ele recebe.

A	B	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

2.4) Codificação Bipolar

A modulação NRZI tem como função realizar a troca de bits de sinais de acordo com que o sinal ocorre, sendo ela caracterizada por ocorrer de acordo com uma transição que ocorre no sinal. Sendo assim, supondo o sinal: 01010011 em uma transição para NRZI ele seria:

NRZ	NRZI
-----	------

0	1
1	0
0	0
1	1
0	1
0	1
1	0
1	1

Quando o sinal está em 0 ele continua em 1, visto que quando o bit permanece em 0 ele não indica nenhuma transição. Quando o sinal passa para 1 ocorre uma transição, nesse caso em NRZI que estava em 1 ele receberá um desnível indo para 0. Como na sequência ele recebe um valor de 0 que não indica uma mudança do sinal o NRZI continua em 0. Seguindo este raciocínio nota-se que a modulação em NRZI ocorre a partir de um desnível que ocorre no sinal.

Na codificação bipolar também conhecida por Alternate Mark Inversion (AMI) tem o comportamento do sinal difere dos demais, visto que quando o sinal recebe o bit 1 o sinal recebe um pulso, isso pode ser dito que a sua modulação ocorre de acordo com os valores 1 e possui voltagem nula quando o sinal possui bit 0. Para uma interpretação que facilite o entendimento da codificação pode ser analisada a seguinte tabela que representa um fluxo de bits NRZ e como seria a sua saída em uma codificação bipolar.

Supondo a entrada de bits como '01010011'.

Fluxo de Bits	Codificação Bipolar
0	0
1	1
0	0
1	-1
0	0
0	0
1	1
1	-1

Nesse caso da codificação bipolar o que deve ser reparado é que ela também é uma modulação por desnível, mas a sua voltagem só irá ser alterada através de 1 e -1.

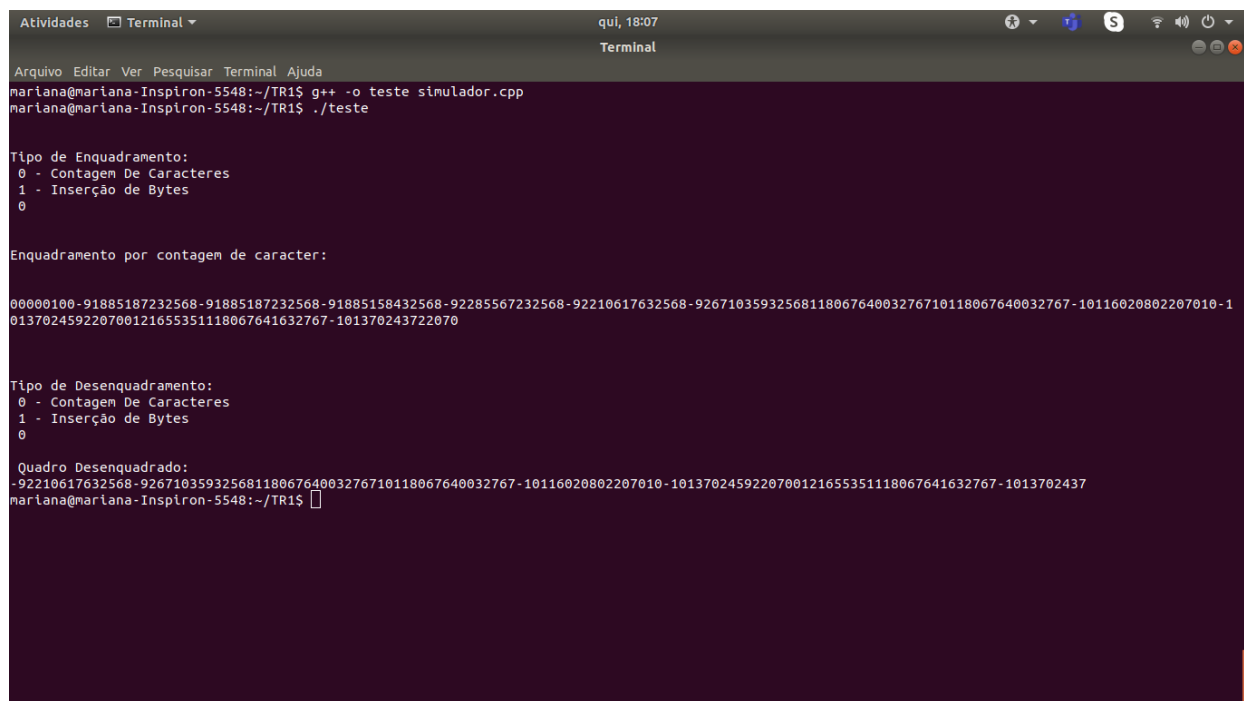
Na implementação da codificação bipolar foi considerado que como ocorre uma oscilação de desnível entre os bits, foi considerado que ao comparar dois bits de entrada ele deve ter como saída o que é esperado. No caso desenvolvido ao apresentar um desnível, indo de 0 para 0 a saída é 0, e de 1 para 0 teria uma saída de 1. No caso do desnível negativo deve ocorrer quando o bit varia de 1 para 1. Para a decodificação foi desenvolvido de forma inversa, quando o bit vai de 0 para 0 a saída é 1, de 1 para 1 a saída é 0.

3. Implementação da Camada de enlace

3.1) Contagem de caracteres

Como esse método utiliza um campo no cabeçalho para que seja possível determinar o número de caracteres o quadro total possui. Dessa forma quando a camada de enlace do receptor sabe quantos caracteres devem ser recebidos e aonde está o fim do quadro.

Para essa implementação foi considerado que o quadro tivesse um tamanho de 32 e a quantidade de divisões fossem o valor do tamanho dividido por 8. Dessa forma enquanto a quantidade de sub quadros fosse maior do que zero o enquadramento continuaria para valores menores do que 8. E para realizar o desenquadramento foi feito um laço de repetição que transfira o valor do novo quadro para o quadro inicial, dessa forma tem-se o quadro desenquadrado.



```
Atividades Terminal
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
mariana@mariana-Inspiron-5548:~/TR1$ g++ -o teste simulador.cpp
mariana@mariana-Inspiron-5548:~/TR1$ ./teste

Tipo de Enquadramento:
0 - Contagem De Caracteres
1 - Inserção de Bytes
0

Enquadramento por contagem de caracter:

00000100-91885187232568-91885187232568-91885158432568-92285567232568-92210617632568-9267103593256811806764003276710118067640032767-10116020802207010-1013702459220700121655351118067641632767-101370243722070

Tipo de Desenquadramento:
0 - Contagem De Caracteres
1 - Inserção de Bytes
0

Quadro Desenquadrado:
-92210617632568-9267103593256811806764003276710118067640032767-10116020802207010-1013702459220700121655351118067641632767-1013702437
mariana@mariana-Inspiron-5548:~/TR1$
```

3.2) Inserção de bytes ou caracteres

Para realizar a implementação da inserção de bytes foi necessário inicialmente como é a teoria para essa implementação, logo, o que se tem é que cada quadro deve começar com um byte especial, esse byte é o byte de flag. Esse byte de flag que é inserido auxilia para resolver o problema de resincronização após um erro, fazendo dessa forma que cada quadro comece e termine com bytes especiais. Quando o receptor perder a sincronização ele procura a mensagem perdida através do byte de flag a fim de descobrir o fim do quadro atual. Quando encontrado dois bytes de flag consecutivos isso indica o fim do quadro atual e início do quadro seguinte. Na sua implementação foi declarada um byte de flag e um byte de escape, eles são inseridos no momento do enquadramento e são retirados do quadro no momento do desenquadramento.


```
Atividades Terminal sex, 01:06
Arquivo Editar Ver Pesquisar Terminal Ajuda
mariana@mariana-Inspiron-5548:~/TR1$ g++ -o teste simulador.cpp
mariana@mariana-Inspiron-5548:~/TR1$ ./teste

Tipo de Enquadramento:
0 - Contagem De Caracteres
1 - Inserção de Bytes
1
Inserção de bytes
Flag:
00100011
Quadro:
0000000020000000140731929632649-9557570723276700140448949832649140731929632649-95575707232767140731929632649-1014073236803264914045102533264910100-858
993460214748364-95575725432767-95575704032767100-12147483647-95575708832767100-858993460214748364-9557569323276700140732960032649-9557569283276700-101
40731929632649-10

Tipo de Desenquadramento:
0 - Contagem De Caracteres
1 - Inserção de Bytes
1
Inserção de bytes
Flag:
00100011
Quadro Desenquadrado:
140732368032649140451025332649101032649-858993460214748364-95575776632767-95575758432767100-12147483647-95575763232767100-858993460214748364-955757476
32767140329625632649140732960032649-9557574723276700-10140731929632649-10
mariana@mariana-Inspiron-5548:~/TR1$
```

Para a implementação do tipo de controle de erro na camada de enlace foram desenvolvidas as implementações para o bit de paridade e o CRC. Na implementação foi adicionada o código de controle de erro depois do enquadramento.

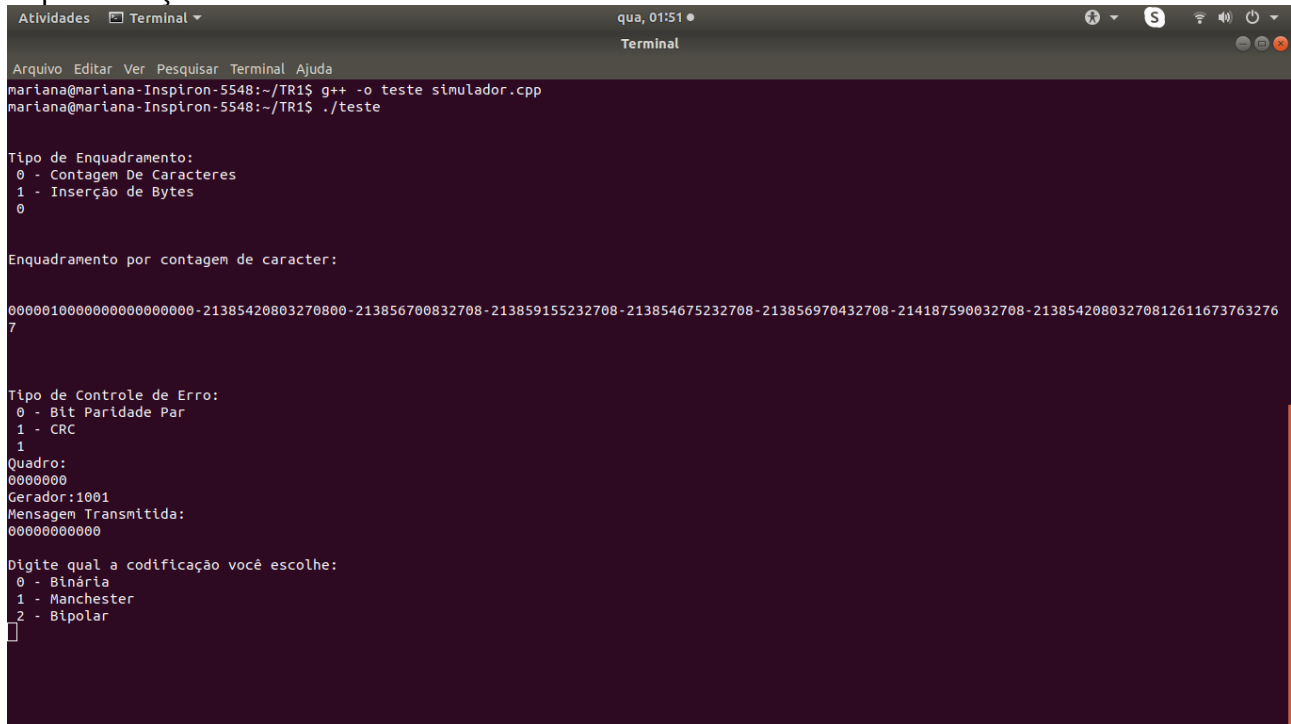
3.3) Bit de Paridade Par

Seguindo a base teórica da definição do bit de paridade, tem-se que esse tipo de controle de erro permite a detecção de erros individuais, logo, tem-se como princípio desse método a inserção do bit de paridade ao final do fluxo de bits. Para o trabalho foi implementada a função de bit de paridade par para controle de erro. O bit de paridade deve ser escolhido de forma que o número de bits transmitidos seja par ou seja ímpar. Logo, caso o seguinte fluxo de bits seja 1010101, tem-se 4 bits 1 para que o resultado possua paridade ímpar, deve ser adicionado o bit 1 ao final do fluxo de bits, logo seria: 10101011 e para que ele tenha paridade par deve ser inserido o bit 0, assim o fluxo seria 10101010, sendo assim teríamos 4 bits 1, sendo esta a paridade par. Logo na implementação foi considerado que caso seja de paridade par, nada seria alterado na codificação e caso o bit fosse 0 e caso o bit fosse 1 o bit alterna entre 0 e 1. A implementação foi feita considerando a combinação lógica xor.

3.4) CRC

O CRC, refere-se a verificação cíclica de redundância, também conhecida como código polinomial, sendo este um método utilizado para a detecção de erro para identificar quando existe alteração em um fluxo de dados. Quando o CRC é empregado o transmissor e o receptor devem concordar em relação ao grau do polinômio gerador. Para realizar um envio de mensagem como o CRC é necessário considerar que existe uma mensagem a ser enviada e existe uma flag que é apontada como o gerador. Para encontrar o CRC é necessário dividir a mensagem a ser enviada pelo gerador, no processo da divisão é utilizada a operação xor para realizar a subtração dos termos. Ao final da operação, quando no resto tem como resultado 000 isso indica que não há erros na execução do código, quando diferente de 000 isso indica que ocorre erro no código. A mensagem final a ser transmitida corresponde ao fluxo de bits acompanhado do CRC ao final do código. Para a implementação foi definido um gerador, sendo ele de 1001. A mensagem transmitida como 00000000. Ao realizar a divisão o CRC é de 0000, logo a mensagem transmitida é a mensagem de origem concatenada ao CRC, logo 00000000 – 0000. Nesse caso é indicado que não tenha nenhum erro na transmissão da mensagem. Para esse caso o meio de comunicação é apresentado como na

implementação da parte 1 do trabalho, ou seja, nenhum erro foi provocado no momento dessa implementação.



```
Atividades Terminal
qua, 01:51
Terminal

Arquivo Editar Ver Pesquisar Terminal Ajuda
mariana@mariana-Inspiron-5548:~/TR1$ g++ -o teste simulador.cpp
mariana@mariana-Inspiron-5548:~/TR1$ ./teste

Tipo de Enquadramento:
0 - Contagem De Caracteres
1 - Inserção de Bytes
0

Enquadramento por contagem de caracter:

000001000000000000000000-21385420803270800-213856700832708-213859155232708-213854675232708-213856970432708-214187590032708-21385420803270812611673763276
7

Tipo de Controle de Erro:
0 - Bit Paridade Par
1 - CRC
1

Quadro:
00000000
Gerador:1001
Mensagem Transmitida:
000000000000

Digite qual a codificação você escolhe:
0 - Binária
1 - Manchester
2 - Bipolar
1
```

3.5) Código de Hamming

4. Simulador

No simulador foi deixada a execução mais básica, de forma que ao compilar o arquivo do simulador.cpp tem-se a união de todos os arquivos, sendo eles a camadafisica.cpp e a camadafisica.h. Foram inseridas os arquivos com camadaenlace.cpp e camadaenlace.h, no simulador.cpp a chamada foi alterada para trabalhar diretamente com a implementação da camada de enlace.

Para a compilação foi utilizada a o terminal do Linux que tem as seguintes configurações seguindo o comando:

```
> cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.5 LTS"
```

No simulador para os testes iniciais de implementação foi feita a chamada direta da CamadaEnlaceDadosTransmissora, ela realiza a chamada para as demais funcionalidades da camada de enlace, logo ela realiza a chamada para a CamadaEnlaceDadosTransmissoraEnquadramento que conforme o usuário seleciona o tipo de enquadramento o código segue seu fluxo para enquadrar e desenquadrar a mensagem.

Para a compilação, é necessário estar no diretório com os arquivos e abrir a chamada `g++ -o teste simulador.cpp` e sequencialmente chamar `./teste`.

```
Atividades Terminal
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
mariana@mariana-Inspiron-5548:~/TR1$ g++ -o teste simulador.cpp
mariana@mariana-Inspiron-5548:~/TR1$ ./teste

Tipo de Enquadramento:
0 - Contagem De Caracteres
1 - Inserção de Bytes
0
```

Com o decorrer da implementação foram feitas algumas alterações para que fossem chamadas as funções seguindo a ordem da camada de aplicação com a camada de enlace e a camada física. Logo na função simulador.cpp ele chama a camada de AplicacaoTransmissora() que agora chama diretamente a camada de enlace para que seja escolhido o tipo de enquadramento e sequencialmente a correção de erro é demonstrada e posteriormente é feita a chamada da camada física que foi implementada na parte 1 do trabalho, dessa forma é seguida a sequência de implementação das camadas que estão sendo trabalhadas.

```
Atividades Terminal
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
mariana@mariana-Inspiron-5548:~/TR1$ g++ -o teste simulador.cpp
mariana@mariana-Inspiron-5548:~/TR1$ ./teste

Tipo de Enquadramento:
0 - Contagem De Caracteres
1 - Inserção de Bytes
0

Enquadramento por contagem de caracter:

0000010000000000000000001338359808326140013383348803261411338310336326141133835513632614113383321843261411335025988326141133835980832614-189454145632764

Tipo de Controle de Erro:
0 - Bit Paridade Par
1 - CRC
0
Bit Paridade par

Digite qual a codificação você escolhe:
0 - Binária
1 - Manchester
2 - Bipolar
0
Digite uma mensagem:

```

5. Membros

- **Mariana:**

- **Implementação de todas as camadas**

- Para a camada física:

CamadaFisicaReceptoraDecodificacaoBinaria,CamadaDeAplicacaoReceptoraDecodificacaoManchester,CamadaDeAplicacaoReceptoraDecodificacaoBipolar,CamadaFisicaReceptora,MeioDeComunicacao,CamadaFisicaTransmissoraCodificacaoBinaria,CamadaDeAplicacaoTransmissoraCodificacaoManchester,CamadaDeAplicacaoTransmissoraCodificacaoBipolar,CamadaFisicaTransmissora,CamadaDeAplicacaoTransmissora,AplicacaoTransmissora,AplicacaoReceptora e CamadaDeAplicacaoReceptora.

- Para a camada de enlace:

CamadaDeEnlaceTransmissoraEnquadramentoContagemDeCaracteres,CamadaEnlaceDadosTransmissora,CamadaEnlaceDadosTransmissoraEnquadramento,CamadaEnlaceDadosReceptoraDesenquadramento,CamadaEnlaceDadosReceptoraDesenquadramentoContagemDeCaracteres,CamadaDeEnlaceTransmissoraEnquadramentoInsercaoDeBytes,CamadaEnlaceDadosReceptoraDesenquadramentoInsercaoDebytes,CamadaEnlaceDadosTransmissoraControleDeErroBitParidadePar, CamadaEnlaceDadosTransmissoraControleDeErroCRC, CamadaDeEnlaceTransmissoraControleDeErro,

6. Conclusão

O trabalho que foi desenvolvido teve seus objetivos atingidos, sendo possível realizar a codificação e decodificação de mensagens utilizando o simulador que foi desenvolvido. A partir deste trabalho foi possível ter um maior entendimento de como a camada física deve se comportar de forma que é possível iniciar o sistema de comunicação entre sistemas finais. Dentre as dificuldades que foram encontradas ao desenvolver o trabalho tem-se o entendimento e como deve ser implementado em cada protocolo a codificação para obter o resultado esperado e identificar como seria o resultado esperado.

Além disso foi possível reforçar e melhorar o entendimento sobre camada física e os protocolos que podem ser implementados e como deve ser feita a comunicação do transmissor para o receptor, sem que se tenha a perda de informações da mensagem.