

Teleinformática e Redes 1 - 2022/2

Simulação da camada de Enlace de dados

Nome : Carlos Eduardo da Silva Almeida

Matrícula : 180014625



Universidade de Brasília

Introdução

Como todo meio de transmissão de dados tem limitações físicas e estão sujeitos a erro, se faz necessário um processamento dos bits enviados e, se houver erros nestes, sua correção. A camada de enlace de dados implementada nessa parte do trabalho é responsável por essas duas funções. A IDE usada foi o Code::Blocks 20.03 e o compilador foi o GCC e é necessária a instalação do MinGw 18.0 para suportar os "vector<int>" do C++17.

A camada de enlace de dados faz a partição dos bits e a essa função se dá o nome de enquadramento. Como nas definições do trabalho e no livro texto não está à forma como se deve decidir com qual tamanho deve-se enquadrar assumir que o simulador envia apenas um quadro. Tendo isso em mente a contagem de bytes, inserção de flags e bit de paridade tornam-se triviais. O código Hamming, que consiste em transmitir um quadro com n bits sendo $n = \text{quantidades de bits de dados} + \text{quantidade de bits redundantes (bits estes postos nas posições que têm índices que são potências de 2)}$, foi implementada para a transmissora porém o aluno autor desse projeto não conseguiu codificar a receptora e nem o CRC. Porém foi tentado com afínco como fica claro pelo código fonte do simulador.

Implementação

A execução do programa começa na main.cpp onde a função "main" chama a "AplicacaoTransmissora".



```
main.cpp X CamadaAplicacao.cpp X CamadaFisica.cpp X CamadaFisica.hpp X CamadaAplicacao.hpp X
1      #include "CamadaAplicacao.hpp"
2
3      using namespace std;
4
5      int main(void){
6          AplicacaoTransmissora();
7          return 0;
8      }
9
```

O programa então vai para a Camada de aplicação onde a função "AplicacaoTransmissora" pede que o usuário entre com uma mensagem e transmite a

mesma para a "CamadaDeAplicacaoTransmissora". Nela o simulador chama a função "StringParaASCII" para transformar a mensagem em bits (quadro) e passa estes para a Camada de Enlace de Dados pela função "CamadaEnlaceDadosTransmissora".

A função "CamadaEnlaceDadosTransmissora" encaminha os quadros recebidos para a "CamadaEnlaceDadosTransmissoraEnquadramento" para o usuário escolher qual método de enquadramento ele quer.

```
main.cpp X CamadaAplicacao.cpp X CamadaFisica.cpp X CamadaFisica.hpp X CamadaAplicacao.hpp X CamadaEnlace.hpp X CamadaEnlace.cpp X
23 void CamadaEnlaceDadosTransmissoraEnquadramento(vector<int>quadro){
24
25     int tipoDeEnquadramento = 0;
26     vector<int> quadroEnquadrado;
27
28     cout << "Selecione o tipo de Enquadramento:" << endl;
29     cout << "0: Contagem De Caracteres." << endl;
30     cout << "1: Insercao De Bytes." << endl;
31
32     cin >> tipoDeEnquadramento;
33
34     switch(tipoDeEnquadramento){
35         case 0:
36             quadroEnquadrado = CamadaEnlaceDadosTransmissoraEnquadramentoContagemDeCaracteres(quadro);
37             break;
38         case 1:
39             quadroEnquadrado = CamadaEnlaceDadosTransmissoraEnquadramentoInsercaoDeBytes(quadro);
40             break;
41     }
42
43     cout << "Quadro Enquadrado: " << endl;
44
45     for(int i = 0; i < quadroEnquadrado.size(); i++){
46         cout << quadroEnquadrado.at(i);
47     }
48 }
```

A Partir dela o fluxo do programa vai para um dos tipos de enquadramento. O primeiro é o de Contagem de Caracteres. Ele consiste em dividir o tamanho do quadro por 8 para obter quantos bytes o mesmo tem e em seguida colocar essa quantidade no quadro, só que na notação binária. Devido a isso o simulador só pode transmitir quadros com até 255 bits, contando com o byte de quantidade de caracteres.

```
main.cpp X CamadaAplicacao.cpp X CamadaFisica.cpp X CamadaFisica.hpp X CamadaAplicacao.hpp X CamadaEnlace.hpp X CamadaEnlace.cpp X
79 vector<int> CamadaEnlaceDadosTransmissoraEnquadramentoContagemDeCaracteres(vector<int>quadro){
80     int caracteresQuadro;
81     vector<int> quadroAux;
82
83     if(quadro.size() % 8 == 0){
84         caracteresQuadro = quadro.size()/8;
85     }else{
86         caracteresQuadro = (quadro.size()/8) + 1;
87     }
88
89     std::string binary = std::bitset<8>(caracteresQuadro + 1).to_string();
90     for (int i = 0; i < binary.length(); i++){
91         if (binary[i] == '0'){
92             quadroAux.push_back(0);
93         }
94         else{
95             if (binary[i] == '1'){
96                 quadroAux.push_back(1);
97             }
98         }
99     }
100
101     for(int i = 0; i < quadro.size(); i++){
102         quadroAux.push_back(quadro.at(i));
103     }
104 }
```

Outra forma de enquadramento é a Inserção de Bytes que consiste em colocar a flag de um byte = {00001111} no começo e no final do quadro e, se porventura a flag estiver no meio do quadro, insere-se a flag ESC = {00011011} antes do byte de flag. E se ainda ocorrer a flag ESC no campo de dados insere-se outra flag ESC no byte anterior. Isso foi implementado usando-se o fato que eu assumi que o simulador só transmite um quadro por vez e a cada 8 bits à condição de que o byte anterior for igual à flag ou igual a um ESC coloca-se um ESC. O aluno responsável pelo trabalho, Carlos Eduardo, tem déficit de atenção e outros transtornos e só percebeu agora, no momento da escrita do relatório, faltando menos de uma hora para encerrar a entrega, que colocou as flags ESC a posteriori e não a priori. Mas o método de enquadramento foi entendido.

```

122
123 vector<int> CamadaEnlaceDadosTransmissoraEnquadramentoInsercaoDeBytes(vector<int> quadro){
124     vector<int> quadroAux;
125
126     for(int i = 0; i < quadro.size(); i++){
127         if((i == 0) || (i == (quadro.size() - 1))){
128             quadroAux = ColocaFlagESC(0, quadroAux);
129         }
130         if((i % 8 == 0) && (0 < i)){
131             if((quadro.at(i - 8) == 0) && (quadro.at(i - 7) == 0) && (quadro.at(i - 6) == 0) && (quadro.at(i - 5) == 0) && (quadro.at(i - 4) == 0) && (quadro.at(i - 3) == 0) && (quadro.at(i - 2) == 0) && (quadro.at(i - 1) == 0)){
132                 quadroAux = ColocaFlagESC(1, quadroAux);
133             } else if((quadro.at(i - 8) == 0) && (quadro.at(i - 7) == 0) && (quadro.at(i - 6) == 0) && (quadro.at(i - 5) == 1) && (quadro.at(i - 4) == 0) && (quadro.at(i - 3) == 0) && (quadro.at(i - 2) == 0) && (quadro.at(i - 1) == 0)){
134                 quadroAux = ColocaFlagESC(1, quadroAux);
135             }
136         }
137     }
138
139     return quadroAux;
140 }
141
142 vector<int> CamadaEnlaceDadosTransmissoraControleDeErroBitParidadePar(vector<int> quadro){
143     int contadorBitsUm = 0;
144
145     for(int i = 0; i < quadro.size(); i++){
146         if(quadro.at(i) == 1){
147             contadorBitsUm++;
148         }
149     }
150
151     if(contadorBitsUm % 2 == 0){
152         return quadro;
153     } else {
154         return ColocaBitParidadePar(quadro);
155     }
156 }

```

Depois o quadro, agora enquadrado, é enviado à controladora de erros. Como o CRC não foi implementado e o método de controle de erro por código Hamming não está funcionando na Receptora é melhor focar no controle de erro por bit de paridade.

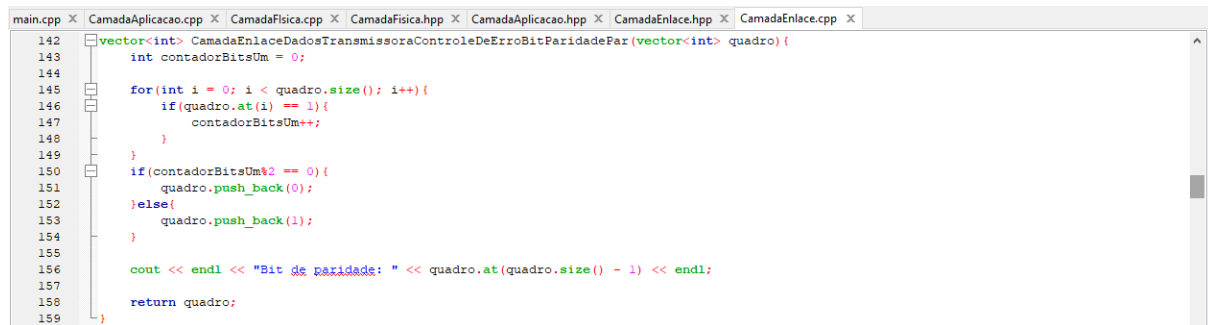
```

53
54 void CamadaEnlaceDadosTransmissoraControleDeErro(vector<int> quadro){
55     vector<int> quadroAux;
56     int tipoDeControleDeErro = 0;
57     cout << "Selecione um tipo de controle de erro: " << endl;
58     cout << "0 - Bit de Paridade Par." << endl;
59     cout << "1 - CRC." << endl;
60     cout << "2 - Hamming." << endl;
61
62     cin >> tipoDeControleDeErro;
63
64     switch(tipoDeControleDeErro){
65         case 0:
66             quadroAux = CamadaEnlaceDadosTransmissoraControleDeErroBitParidadePar(quadro);
67             break;
68         case 1:
69             quadroAux = CamadaEnlaceDadosTransmissoraControleDeErroCRC(quadro);
70             break;
71         case 2:
72             quadroAux = CamadaEnlaceDadosTransmissoraControleDeErroCodigoDeHamming(quadro);
73             break;
74     }
75
76     CamadaFisicaTransmissora(quadroAux);
77 }

```

O método de inserção de bit de paridade é muito simples, ele consiste em sua essência contar quantos bits 1 o quadro possui e, se tal quantidade tiver resto 0 quando dividido por 2, insere-se o bit 0 no final do quadro. Caso contrário, insere-se o bit 1. Isso foi

feito fazendo-se um laço que varre o vetor do quadro e incrementa uma variável chamada "contadorBitsUm" a cada bit 1 que encontra. Depois o valor dessa variável tem seu módulo por 2 tomado e, se for 0, é acrescentado o bit 0 ao final do quadro, caso contrário o bit 1.



```
142 vector<int> CamadaEnlaceDadosTransmissoraControleDeErroBitParidadePar(vector<int> quadro){
143     int contadorBitsUm = 0;
144
145     for(int i = 0; i < quadro.size(); i++){
146         if(quadro.at(i) == 1){
147             contadorBitsUm++;
148         }
149     }
150     if(contadorBitsUm%2 == 0){
151         quadro.push_back(0);
152     }else{
153         quadro.push_back(1);
154     }
155
156     cout << endl << "Bit de Paridade: " << quadro.at(quadro.size() - 1) << endl;
157
158     return quadro;
159 }
```

Após passar pelos métodos de detecção erro, o quadro é encaminhado para a camada física para ser transmitido. Após ser recebido pela receptora da camada física é encaminhada para a camada de enlace para ver se houve algum erro enquanto passava pelo canal de transmissão e, se não houve erro, ser desenquadrado e mandado para a camada de aplicação.

Membros : Carlos Eduardo da Silva Almeida - 18/0014625. O que fez: Código e relatório completos.

Conclusão

Para os propósitos didáticos considero o simulador um sucesso pois, aprendi realmente como os métodos de enquadramento, detecção e correção de erros funcionam. Eu só não fiz o simulador completo e totalmente funcionando corretamente pois, como disse anteriormente, sou portador de 6 transtornos mentais e troquei de medicamentos recentemente que não estão me fazendo bem. Se tivesse mais 1 ou 2 dias conseguiria entregá-lo completo. Também não possuo domínio sobre a linguagem C++ pois fiz todas as matérias de programação em Python.

Mesmo assim, acredito que a realização do trabalho foi muito enriquecedora para meu aprendizado.