Universidade de Brasília

Departamento de Ciência da Computação
Teleinformática e Redes 1

Trabalho 3

Camada de Enlace

Autor:

José Vinícius Garreto Costa 18/0123734

Luiz Carlos Schonarth Junior 19/0055171

Hyago Gabriel Oliveira Figueiredo 17/0105067

Simulador de Camada de Enlace

Brasília
22 de setembro de 2022

1 Introdução

1.1 Descrição do problema

Simular o funcionamento da camada de enlace usando os seguintes protocolos de enquadramento de dados:

- **Contagem de caracteres**
- **Inserção de bytes ou caracteres**

E usando os seguintes protocolos de detecção de erro:

- **Bit de paridade par**
- **CRC (polinômio CRC-32, IEEE 802)**

Protocolo utilizado para correção de erros: Hamming

1.2 Visão Geral do Simulador

- **O simulador recebe um input do usuário (string).**

- **Recebe também os dados relativos aos tipos de enquadramento, controle de erros e codificação**

- **Os quadros são formados e codificados**

- **O sinal elétrico é gerado**

- **Os bits são transmitidos**

- **A simulação inicia o caminho contrário, o de recepção do sinal transmitido**

- **A entrada é transformada novamente num quadro**

- **O quadro codificado é testado quanto a existência de erros**

- **Se não houver erros, o trem de bits é gerado**

- **A mensagem é, finalmente, recebida**

2 Implementação

2.1 Estrutura

A classe mais superior é a classe Pilha, onde está contido o método Simula, que faz a simulação inteira da comunicação. Essa pilha também armazena ponteiros para instancias de classes que implementam a comunicação, tais como as classes transmissor (receptor da camada de aplicação), o transmissor e receptor da camada de enlace, o transmissor e receptor da camada física e o meio de comunicação entre essas duas camadas, física e enlace.

O método simula recebe da interface gráfica a mensagem inserida pelo usuário, o tipo de enquadramento, o tipo de controle de erro e o tipo de codificação, e com isso cada camada recebe essas informações. Os métodos responsáveis para a simulação são: envia mensagem e recebe mensagem.

2.2 Camada de Aplicação Transmissora

Por meio de um laço, inserimos cada carácter, interpretado como byte, no novo vetor. Com o byte em mãos, podemos transmitir o sinal usando pouca memória e fazer operações eficientes em cada bit.
enviaMensagem() é um método que simula a transmissão da pilha inteira. Inicialmente insere a mensagem no transmissor de aplicação e chama o método transmitir, onde a mensagem se transforma em um trem de bits e é enviada para instancia do transmissor de enlace.

```
/* Método responsável por receber uma string e tipo de codificação
 * do usuário e simular a comunicação entre transmissores e
 * receptores implementados na pilha. os resultados são armazenados
 * nos atributos privados da instância de Pilha. */
void Pilha::simula(
    const std::string &msg,
    tipos_enquadramento tipo_enq
    tipos_controle_erro tipo_err,
    tipos_codificacao tipo_cod
    )
{
    input = msg;
    setEnquadramento(tipo_enq);
    setControleDeErro(tipo_err);
    setCodigo(tipo_cod);

    enviaMensagem();
    recebeMensagem();
} // fim do método Pilha::simula
```

2.3 Camada de Enlace Transmissora

Nesta camada efetuamos o enquadramento do sinal, detecção e correção de erros. Primeiramente encaminhamos a mensagem para a Pilha::simula(), onde podemos optar, através da interface gráfica, pelo tipo de enquadramento, tipo de controle de erro e o tipo de codificação que será utilizada na comunicação.

Os dados que o usuário entrou são passados para as demais instâncias de cada camada. O enquadramento é passado para os objetos da camada de enlace juntamente com o tipo de controle de erros e avisa para os transmissores e receptores da camada física qual o tipo de codificação./

Dentro do mesmo método Pilha::simula() há a chamada de outros métodos interno:/

• O transmissor de enlace gera o quadro, através do método geraQuadro()

• geraQuadroCodificado(): que internamente já enquadra e, a partir do quadro, já codifica usando o tipo de controle de erros escolhido

2.4 Camada Física Transmissora

Nesta camada efetuaremos as codificações nos bits. Vale lembrar eles estão em codificação binária, e, por isso, não será necessário quaisquer alterações. Já para as codificações -já implementadas num momento anterior- manchester e bipolar, precisaremos dobrar o número de bits, pois precisamos tornar possível representar subida e decida no caso da Manchester, e -1, 0 e 1 na implementação do caso da bipolar.

Após a codificação o sinal é enviado para a instância do meio de comunicação, chamando o método transmitir(), onde é feito o stream dos bits entre os trasnmissores

2.5 Codificação Manchester

Nessa codificação temos que o bit 0 é representado por uma subida, e o bit 1 por uma descida. Representaremos subida por 01, e descida por 10.

2.6 Codificação Bipolar

Nessa codificação, temos que o bit 0 é representado por 0, e o bit 1 pode ser 1 ou -1, dependendo de quantos numeros "1"foram encontrados no byte. Se um número par de 1's for encontrado, será 1, senão, será -1. Representaremos 0 por 00, 1 por 01 e -1 por 10.

2.7 Meio de comunicação

Da instancia do meio de comunicação, o método transmitir() é utilizado. Esse método recebe como parametro um ponteiro para a instancia do transmissor e receptor físico e faz uma string, enviando bit a bit do transmissor pro receptor.

2.8 Camada Física Receptora

O método recebeMensagem() faz o caminho contrário do que foi relatado até aqui. O receptor físico recebe os sinais e chama o método geraQuadro() que recebe o sinal e o traduz para um quadro.

2.9 Camada Enlace Receptora

O receptor de enlace chama o método receber() que recebe do receptor físico o quadro codificado, e dentro de um bloco Try Catch o método geraQuadro() é chamado e funções internas de detectar erros são chamadas, de acordo com o método escolhido pelo usuário.

Caso o controle de erro seja hamming, o quadro irá receber uma correção, caso contrário, será lançada uma excessão que será exibida na interface gráfica do usuário.

2.10 Camada de Aplicação Receptora

Um trem de bits é gerado nesta camada, a partir do método geraTremDeBits(), onde é removido dos quadros os cabeçalhos.

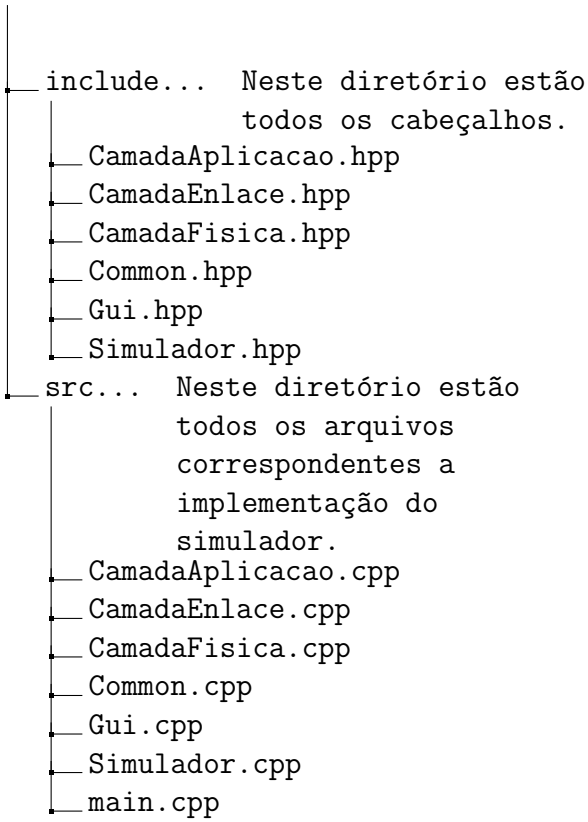
2.11 Aplicação Receptora

O receptor da aplicação utiliza o método receber() para tratar esse trem de bits e por fim, gerar a mensagem.

3 Formas de executar o programa

3.1 Introdução

Primeiramente, é importante pontuar que o programa foi dividido em diferentes arquivos que, por sua vez, representam as camadas física, de enlace e de aplicação. Foi possível, também, organizar o simulador num arquivo à parte, de modo que está presente na nossa pasta src como simulador.cpp. Basicamente temos o diretório src, contendo todos os programas, o diretório include, onde foram listadas as headers com seus respectivos .hpp



3.2 Clonagem e execução

Para executar o programa é necessário pré-instalar algumas bibliotecas, os seguintes comandos serão necessários:
• apt-get install libsdl2-dev g++ make

Após a pré-instalação o repositório deve ser clonado pelo seguinte comando:
• git clone –recurse-submodules
https:/github.comluizschonarthsimulador_TR1_2022-1.git

Para compilar e rodar o simulador os seguintes comandos devem ser utilizados:
• make
• bin/simulador

4 Conclusão

Com a implementação de todas as camadas, conseguimos simular por completo o caminho de uma mensagem, desde o emissor, até o receptor. Utilizando de métodos que simulam as camadas de transmissão é possível transformar a mensagem em um trem de bits e posteriormente em uma sinal elétrico, e com as camadas receptoras, conseguimos fazer o caminho contrário e transformar esses dados novamente na mensagem inicial.
Alguns métodos foram implementados a fim de testar a veracidade da teoria na prática, como os diferentes tipo de codificação e detector de erros, e os resultados obtidos foram satisfatórios.
Com isso conseguimos simular do início ao fim, todo o caminho de uma mensagem do seu emissor ao seu receptor.