Simulador dos Protocolos Binário, Manchester e Bipolar

Diogo Luis Teixeira de Macedo Sampaio - 16/0056489 Gabriel Carvalho Moretto - 15/0154917 Pedro Henriques Nogueira - 14/0065032

Teleinformática e Redes 1 - Turma A - 2020/2



Universidade de Brasília Departamento de Ciência da Computação Professor Geraldo Pereira Rocha Filho

1 Introdução

Em uma rede de dispositivos, é necessário a utilização de protocolos de banda base, já que os dispositivos consequem interpretar informações em bits (sinal digital) e os cabos transmitem sinais analógicos, do contrário, a informação de um dispositivo não poderia ser passada para os cabos e nem a informação que chega em um dispositivo pelos cabos poderia ser interpretada.

Vamos tratar aqui de uma implementação que compreende alguns desses protocolos, na linguagem de programação C++: Binário, Manchester e Bipolar. As camadas de enlace, transporte e aplicação são mencionadas de forma breve no trabalho, porém o escopo desse trabalho compreende a camada física para o tratatamento dos protocolos que foram mencionados.

Para este relatório, utilizamos a mensagem "Oi!", para demonstrar os efeitos de cada protocolo, sendo esta representada pelo sinal abaixo:

1.1 Protocolo Binário

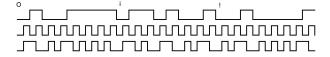
Nesse protocolo, o sinal que é mandado do dispositivo em bits é traduzido em um sinal binário onde a presença do bit 1 é correspondente a presença da voltagem no sinal, enquanto o bit 0 corresponde a ausência de voltagem.



1.2 Protocolo Manchester

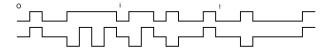
Geralmente, quando se transmite um sinal é necessário a sincronização ou temporização dos dispositivos transmissor e receptor, pois é necessário saber a extensão desse sinal tanto para envio quanto para recepção. Para corrigir esse problema sem necessitar de sincronização entre dispositivos (já que isso é extremamente dispendioso) é usado um clock em conjunto com o sinal, e para enviar essa informação do clock junto com a informação sem enviar esses dois sinais separados e desperdiçar largura de banda de sinal, nós usamos o protocolo manchester.

Esse protocolo consiste em combinar um sinal de clock com o nossa menssagem usando a operação lógica de disjunção exclusiva (ou XOR).



1.3 Protocolo Bipolar

O protocolo bipolar consiste em fazer uma melhor utilização dos componentes eletrônicos, e ele faz isso usando não somente a presença ou ausência da voltagem no sinal, como ele usa voltagem negativa, fazendo assim que o sinal fique balanceado. Sinais balanceados têm um certo apelo, pois não requerem o componente elétrico corrente contínua (CC), o que é interessante pois esse componente dificulta a comunicação de componentes mais básicos. O sinal bipolar possui voltagem positiva ou negativa na presença de um bit 1 e voltagem nula na presença de um bit zero, sendo que o valor da voltagem alterna de positivo para negativo e de negativo para positivo para toda vez que há um bit 1 sendo ele em sequência ou não.



2 Implementação

A aplicação é feita pelo terminal, e quando ela é aberta, é pedido do usuário que ele escreva uma mensagem, que representa a aplicação transmissora, depois o programa pede para que o usuário escolha um dos três tipos de codificação (Binário, Manchester e Bipolar), através de uma numeração de zero a dois. Uma vez que o tipo de codificação é escolhido, tanto a menssagem quanto a escolha da codificação é enviado para a parte do programa responsável pela camada de aplicação transmissora que, para efeitos do trabalho, transforma a menssagem de uma string para um vetor de inteiros que corresponde a mensagem escrita em ASCii.

Depois dessa conversão, o vetor é enviado para a parte do código responsável pela camada física transmissora, onde ela passa por um dos protocolos dependendo da escolha que o usuário fez na parte do código da aplicação transmissora (zero para codificação binária, um para codificação manchester e dois para codificação bipolar). Depois da codificação, a menssagem em ASCii é replicada para a parte do código que diz respeito ao meio de comunicação, para que em seguida ela seja decodificada pela parte do código responsável pela camada física receptora, que decodifica de acorda com o método que ela foi transforma, para que por sua vez ela seja convertida de ASCii para uma string na parte da camada de aplicação receptora para que finalmente ela chegue na aplicação da parte receptora no código. O resultado do código consiste na mesma mensagem que foi enviada pela aplicação transmissora, no que qualifica um sucesso, já que os protocolos não modificaram os dados.

2.1 Protocolo Binário

Para a implementação desse protocolo não necessária fazer nenhum tipo de alteração no vetor de inteiros, já que o próprio formato do código em ASCii conseque representar fielmente o resultado do protocolo binário

```
vector<int> CamadaFisicaTransmissoraCodificacaoBinaria(
vector<int> quadro) {
return quadro;
}
```

Uma vez que não houve alteração no vetor para transmissão, também não houve necessidade de decodifica-lo na recepção do sinal

```
vector<int> CamadaFisicaReceptoraCodificacaoBinaria(
    vector<int> fluxoBrutoDeBits) {
    return fluxoBrutoDeBits;
}
```

Abaixo temos a tela do terminal quando o protocolo binário é escolhido com a menssagem de exemplo "Oi!".

2.2 Protocolo Manchester

Na implementação, nesse caso usamos de uma iteração que percorre o vetor do sinal para que possamos fazer duas operações em cada bit, que é a operação XOR (ou exclusivo) com os bits zero e um vindas do clock.

```
vector<int> CamadaFisicaTransmissoraCodificacaoManchester(vector<int> quadro) {
   vector<int> manchesterBits;
   for (unsigned i = 0; i < quadro.size(); i++) {
      manchesterBits.push_back(quadro.at(i) ^ CLOCK_MANCHESTER.at(0));
      manchesterBits.push_back(quadro.at(i) ^ CLOCK_MANCHESTER.at(1));
   } // CLOCK_MANCHESTER = [0,1] e so existe para padronizar a codificacao
   return manchesterBits;
}</pre>
```

Para a decodificação do sinal usamos um método bem simples: iteramos o sinal codificado de dois em dois para recuperar o sinal original, pois foi observado que o sinal codificado possui o sinal original nesse intervalo sem qualquer tipo de perda ou modificação, já que em qualquer caso, a operação XOR de qualquer tipo com o zero resulta no mesmo bit.

```
vector<int> CamadaFisicaReceptoraCodificacaoManchester(
    vector<int> fluxoBrutoDeBits) {
    vector<int> bitsDecodificados;
    for (unsigned i = 0; i < fluxoBrutoDeBits.size(); i += 2) { // Sinal original bitsDecodificados.push_back(fluxoBrutoDeBits.at(i)); // esta na xor com 0
    }
    return bitsDecodificados;
}</pre>
```

Abaixo temos a tela do terminal quando o protocolo manchester é escolhido com a menssagem de exemplo "Oi!".

```
bj@BJ114:/mnt/c/Users/Pedro Nogueira/Documents/projects/trabalhoTR1/fonte$ make executa
./main.exe

Digite uma mensagem:
0i!

Selecione um modo de codificação:
0: Codificação Binária
1: Codificação Manchester
2: Codificação Bipolar
1

Mensagem em ASCII caracter por caracter:
0 - 01001111
i - 01101001
! - 00100001

Codificação Manchester:
011001011011010100
00111010101010100
001011001101010100
00101100101010100

Mensagem recebida: 0i!

bj@BJ114:/mnt/c/Users/Pedro Nogueira/Documents/projects/trabalhoTR1/fonte$
```

2.3 Protocolo Bipolar

Neste caso, nós percorremos o vetor de inteiros de modo a achar os bits no sinal em ASCii que sejam iguais a um e mudar o polo no nosso sinal de transmissão de modo que o sinal tenha dois polos, negativo e positivo.

```
vector<int> CamadaFisicaTransmissoraCodificacaoBipolar(vector<int> quadro) {
   bool umNegativo = false; // Meu bipolar comeca com 1 positivo
   for (unsigned i = 0; i < quadro.size(); i++) {
     if (quadro.at(i) == 1) {
        if (umNegativo) {
            quadro.at(i) = -1;
        }
}</pre>
```

Para recuperamos o nosso sinal codificado na implementação, percorremos o vetor para recuperarmos o modulo de cada bit, no que resulta no sinal original.

```
vector<int> CamadaFisicaReceptoraCodificacaoBipolar(
    vector<int> fluxoBrutoDeBits) {
    for (unsigned i = 0; i < fluxoBrutoDeBits.size(); i++) { // So faz modulo
        fluxoBrutoDeBits.at(i) = fabs(fluxoBrutoDeBits.at(i)); // de tudo
    }
    return fluxoBrutoDeBits;
}</pre>
```

Abaixo temos a tela do terminal quando o protocolo bipolar é escolhido com a menssagem de exemplo "Oi!".

```
Dj@BJ114:/mnt/c/Users/Pedro Nogueira/Documents/projects/trabalhoTR1/fonte$ make executa./main.exe

Digite uma mensagem:
Oi!

Selecione um modo de codificação:
0: Codificação Binária
1: Codificação Binária
2: Codificação Bipolar
2

Mensagem em ASCII caracter por caracter:
O - 01001111
i - 01101001
! - 00100001

Codificação Bipolar:
0 1 0 0 -1 1 -1 1
0 -1 1 0 -1 0 0 1
0 0 -1 0 0 0 0 1

Mensagem recebida: Oi!

bj@BJ114:/mnt/c/Users/Pedro Nogueira/Documents/projects/trabalhoTR1/fonte$ ■
```

3 Membros

A distribuição do trabalho se deu pela seguinte forma: Pedro implementou o código em sua completude enquanto Gabriel e Diogo ficaram responsáveis pelo relatório.

4 Conclusão

A implementação satisfez o esperado: ela recebe uma mensagem e passa pelo código de forma a codificar e decodificar uma menssagem sem modificá-la, no que configura um sucesso. De forma prática, a aplicação em C++ não possui fins práticos, porém o fim didádico por conta do trabalho feito teve um impacto extremamente positivo no processo de aprendizado, já que os membros do grupo possuem familiaridade com a área da tecnologia da informação.