Simulador de comunicação

Trabalho para a disciplina
Teleinformática e Redes 1

Feito por
Artur Padovesi Piratelli - 211038208
Miguel Mendes Luna - 211026501
Lucas Corrêa Boaventura - 211038262

Introdução

O trabalho se resume em:

- Simular um emissor de sinal;
- Simular um receptor de sinal;
- Acompanhar os processos com uma interface;

Nesse âmbito, devem poder ser simulados os seguintes aspectos da camada física:

- Codificação/modulação banda base:
    - NRZ-Polar;
    - Manchester;
    - Bipolar;
- Modulação por portadora:
    - ASK;
    - FSK;
    - 8QAM;

E da camada de enlace:

- Enquadramento de dados:
    - Contagem de caracteres;
    - Contagem de bits ou inserção de bytes;
- Protocolo de detecção de erros:
    - Bit de paridade par;
    - CRC (polinômio CRC-32, IEEE 802);
- Protocolo de correção de erros:
    - Hamming;

O simulador em geral, representado na interface, funciona de forma simples: o usuário, inicialmente, escolhe o tipo de codificação a ser utilizada (dentre as opções NRZ-Polar, Manchester e Bipolar), para, em seguida, digitar a mensagem a ser enviada e apertar o botão "send" que, ao pé da letra, envia a mensagem.

Então o transmissor representará o texto a ser enviado (o qual o usuário digitou há pouco), os bits enviados (após os protocolos de enquadramento, detecção de erros e correção de erros) e a codificação de tais bits conforme a opção selecionada no início. Além do transmissor, temos as modulações representadas: serão mostrados na interface os gráficos referentes às modulações do trem de bits do tipo ASK, FSK e 8QAM.

Por fim, o receptor representará os bits codificados recebidos, o trem de bits decodificado (já retirados os protocolos de enlace), e o texto representado pelo trem de bits mencionado, o qual representa a sequência de caracteres digitada inicialmente pelo usuário.

Implementação

O simulador foi produzido como uma interface web “observando” dois servidore. Nesse sentido, foi utilizada a linguagem TypeScript para desenvolver a maior parte dos protocolos e procedimentos utilizados no “back-end”. O "front-end" foi desenvolvido com HTML/CSS/JS, utilizando o framework vite para desenvolvimento.

Assim, o programa principal se divide em três pastas principais: a interface implementa a interface propriamente dita (front-end); o transmissor recebe o texto digitado pelo usuário (mensagem a ser enviada) e o transforma em uma sequência de bytes a partir da tabela ASCII, para então utilizar os protocolos cabíveis, adicionando a codificação ao final; e o receptor, como o próprio nome diz, recebe esse trem de bits codificado para então realizar o processo de decodificação, e os procedimentos de retorno aos caracteres iniciais.

Dessa maneira, as pastas transmitter e receptor possuem um arquivo server.ts cada uma, os quais são utilizados para fazer os chamamentos das funções em geral. Recebimento do trem de bits digitado pelo usuário, transformação de texto em bits, protocolo de detecção de erros, hamming, enquadramento, todos esses procedimentos, tanto de "ida" como de "volta", são chamados nos arquivos server.ts. Exemplificando, a função de codificação (NRZ-Polar, por exemplo), é chamada no arquivo server.ts da pasta transmitter, enquanto que a função de decodificação é chamada no arquivo server.ts da pasta "receptor".

Para escolher entre o controle de erro e as opções de enquadramento, o config.ts das pastas transmitter e receptor foi feito por hardcode para escolher entre estes.

Diminuindo o nível de abstração, analisaremos agora os arquivos presentes em cada uma das pastas.

Transmitter

Na sub-pasta InterfaceGUI, os arquivos sendDataToReceptor.ts e InterfaceComms.ts que fazem o envio de informações para o receptor e para a interface, respectivamente.

Na sub-pasta CamadaFisica, o encode.ts realiza a codificação do trem de bits em Manchester, Bipolar ou NRZ-Polar, enquanto que o bitsFromText.ts converte a string digitada pelo usuário em um trem de bits conforme a tabela ASCII.

Na sub-pasta CamadaFisica, o arquivo errorControl.ts é responsável pelo controle de erros, implementando a função addEDC e hamming, o bitCounting.ts implementa a contagem de bits, words e caracteres e, por fim, o arquivo byteInsertion.ts realiza a inserção de bytes (flags).

O arquivo config.ts é apenas um arquivo de configuração auxiliar.

Receptor

Por outro lado, o receptor possui arquivos análogos.

O arquivo onReceivedText.ts possui uma função que pode ser utilizada como ponto de entrada no programa, simplesmente faz "log in" no console. Por sua vez, o arquivo interfaceComms.ts é análogo aos do transmissor.

O arquivo textFromBits.ts realiza a conversão de um trem de caracteres (bytes) em uma string de letras legível ao usuário. Além disso, o arquivo decode.ts realiza o processo inverso em relação ao arquivo encode.ts.

Na sub-pasta CamadaFisica, os arquivos checkErrorControl.ts e checkBitCounting.ts fazem o procedimento inverso aos arquivos errorControl.ts e bitCounting.ts, respectivamente, verificando a existência de ruídos. Por fim, o arquivo getFramesByInsertionFlag.ts realiza a remoção das flags inseridas.

Config.ts é análogo aos do transmissor.

Interface

Finalmente, a pasta interface implementa o front-end propriamente dito.

Na sub-pasta CamadaFisica, o arquivo modulator.js tem a função responsável pelas modulações ASK, FSK e 8QAM.

Na sub-pasta Simulador, o arquivo chart.js faz a simulação das modulações ASK, FSK e 8QAM para mostrar em gráfico no Front. Já o arquivo main.js, faz o streaming de eventos entre transmissor e receptor pela web.

OBS: Vale ressaltar que foi decidido para o projeto, a utilização de um padding de zeros em:

- Antes do enquadramento, nos quadros não múltiplos de 8;
- Na modulação 8QAM, quando há trem de bits não múltiplo de 3;

Membros

Lucas:

- ★ Implementou as codificações Manchester,NRZ-Polar e Bipolar;
- ★ Modulações ASK, FSK e 8QAM;
- ★ Os protocolos de detecção de erros (Bit de paridade par e CRC);
- ★ O protocolo de detecção de erros Hamming;
- ★ Integração de tais implementações com a interface (simulador).
- ★ Produção do relatório.

Artur:

- ★ Implementou a interface (simulador), com "mock" do receptor e do transmissor;
- ★ Função de transformar o texto digitado pelo usuário em um trem de bits segundo os caracteres pela tabela ASCII;
- ★ Enquadramento de dados por contagem de caracteres e por inserção de bytes;
- ★ Revisão e aprimoramento de códigos.
- ★ Produção do relatório.

Miguel:

- ★ Implementou as decodificações Manchester, NRZ-Polar e Bipolar;
- ★ O enquadramento de dados por contagem de bits, por contagem de "words" e por inserção de bytes;
- ★ Integração de tais implementações com a interface;
- ★ Produção do relatório.

Conclusão

Neste trabalho foram implementados os protocolos de enquadramento, detecção e correção de erros; codificação, modulação banda-base e modulação por portadora, a fim de melhor entender o funcionamento das camadas de enlace e física.

Os resultados encontrados estão de acordo com o esperado, podendo ser enviada uma mensagem que é corretamente manipulada e recebida para poder ser lida sem erros.

Conforme os resultados obtidos, fomos capazes de entender na prática os protocolos e processos existentes numa rede de comunicação real.

Foram encontradas dificuldades em proteger o isolamento entre as camadas física e de enlace, porém conseguimos separar os arquivos de forma a entender as partes feitas por cada uma dessas camadas. Além disso, foram encontradas dificuldades no processo de integrar todos os protocolos, de forma que cada protocolo tratasse de forma correta o trem de bits gerado na etapa anterior, porém conseguimos chegar ao bom funcionamento do sistema.