Departamento de Ciência da Computação (CIC)
TELEINFORMÁTICA E REDES (204315)

Simulador da Camada Física

Trabalho nº 1

Número	**Nome completo**
16/0120781	Gabriel Correia de Vasconcelos

Prof. Geraldo P. Rocha Filho

Brasília, 4 de abril de 2021.

Introdução

Descrição do problema a ser resolvido e visão geral sobre o funcionamento do simulador.

O relatório a seguir tem o objetivo de descrever o desenvolvimento e implementação de um programa que simula a camada física de comunicação entre duas aplicações, uma atua como transmissora do sinal e a outra como receptora deste mesmo sinal. Este programa está disponível na íntegra em um repositório no Github junto com suas instruções de uso. As funções de simulação foram desenvolvidas na linguagem C++ e compiladas em seu padrão C++11. Já o ambiente de desenvolvimento e compilação escolhido foi um *container* oficial do Docker com o gcc 4.9 instalado.

A comunicação entre dois ou mais dispositivos eletrônicos, também chamada de telecomunicação, necessita de um conjunto de regras e protocolos para ditar como os dados serão transportados de um ponto a outro. Esses conjuntos de protocolos são chamados de modelos de rede de computadores e os mais conhecidos são: o modelo TCP/IP e o modelo OSI. Esses modelos dividem suas diretrizes em camadas de abstração hierárquicas que existem para modularizar suas funcionalidades. O relacionamento entre as camadas é hierárquico pois cada camada, além de ter uma função específica no processo de comunicação, depende da camada anterior a ela.

A camada física, que será implementada neste simulador, funciona como uma interface entre um dispositivo e um meio de transmissão. Esta é a abstração de mais baixo nível e descreve como um fluxo de bits bruto ou sinal físico deve ser transmitido e recebido no meio de transmissão (i.e. cabo coaxial, fibra óptica, ar etc). Como estamos no meio físico, a transmissão está sujeita a ruídos e imprevisibilidades e por isso essas informações podem ser codificadas, para aumentar sua eficiência e resiliência a erros durante a transmissão. Neste simulador estão implementadas as codificações: binária (Non Return Zero unipolar), Manchester e bipolar, que são transformam quadros em um fluxo bruto de bits e vice e versa..

O simulador além de implementar a camada física, possui uma camada de aplicação simples, que será a interface entre o ser humano e a aplicação. Ela define e trata a mensagem (entrada de dados, no caso de texto) que será transformada em quadro e enviada à camada física para ser codificada e enviada ao meio de transmissão. Esta também é responsável por receber os dados decodificados para mostrar ao usuário apropriadamente.

O meio de transmissão é implementado via software e consiste apenas na cópia dos dados de uma variável de entrada para a de saída, bit a bit, simulando uma tensão que pode ser 0, 1 Volt para o valor 1 e -1 Volt para o valor -1.

Implementação

Descrição detalhada do desenvolvimento com diagramas ilustrativos, o funcionamento dos protocolos, procedimentos utilizados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.

A implementação deste simulador seguiu a especificação do primeiro trabalho prático da disciplina de Teleinformática e Redes 1, ministrada pelo professor Geraldo Filho no semestre de 2020/2. O fluxo do simulador pode ser dividido em dois processos principais, o de transmissão e o de recepção. Esses processos serão descritos a seguir por meio de diagramas ilustrativos e explicações detalhadas das escolhas de implementação.

O processo de transmissão é composto por três funções que têm seus nomes auto explicativos, são elas:

- *AplicacaoTransmissora()*
- *CamadaDeAplicacaoTransmissora(mensagem)*
- *CamadaFisicaTransmissora(quadro)*

Este se inicia com uma mensagem de tamanho máximo de 512 caracteres, que é recebida pela *AplicacaoTransmissora*, inserida via linha de comando pelo usuário. A limitação de tamanho é definida em uma constante global e existe devido a impossibilidade de alocação de tamanho dinâmico das variáveis da biblioteca <bitset>. A *AplicacaoTransmissora* chama a função *CamadaDeAplicacao Transmissora*, que converte a mensagem recebida em bits dos caracteres da mensagem, seguindo o ASCII, chamando-o de quadro (no formato <bitset> para permitir operações lógicas bit a bit), e envia esse quadro para função *CamadaFisicaTransmissora*. Esta função, a partir da opção inserida pelo usuário, escolhe o protocolo de codificação que será realizado para codificar o quadro em um fluxo bruto de bits (no formato vector<int> para permitir valores positivos e negativos de tensão) e enviada a função que simula o *MeioDeComunicacao*. As opções do usuário foram implementadas em funções auxiliares e equivalem a um inteiro valendo:

- 1 para *CamadaFisicaTransmissoraCodificacaoBinaria***:** Non return Zero unipolar (NRZ unipolar), onde cada bit 1 do quadro equivale ao valor 1 V de tensão e cada bit 0 equivale ao valor 0 V de tensão, implementado com uma condicional que checa o valor de cada bit.
- 2 para *CamadaFisicaTransmissoraCodificacaoManchester:* Aqui existe um sinal de relógio (clock) que sempre começa do 0 e alterna entre 1 e 0 ao longo do tamanho em bits do quadro. Bit a bit do quadro, cada bit que for diferente do clock vale 1 V e cada bit igual vale 0 V, que é implementada pela operação lógica OU exclusivo (XOR).

- 3 para *CamadaFisicaTransmissoraCodificacaoBipolar:* Cada bit 1 vale alternadamente 1 V e -1 V de tensão e cada bit 0 equivale a 0 V de tensão. Implementado com uma condicional e uma variável auxiliar que invertia seu valor toda vez que o bit era igual a 1.

A função *MeioDeComunicacao,* como explicado anteriormente, apenas retorna a cópia do fluxo bruto de bits da entrada e inicia o processo de recepção da mensagem. Este processo é composto também por três funções principais e que funcionam de forma oposta às da aplicação de transmissão. São elas:

- *CamadaFisicaReceptora(fluxoBrutoDeBits)*
- *CamadaDeAplicacaoReceptora(quadro)*
- *AplicacaoReceptora(mensagem)*

A *CamadaFisicaReceptora* recebe o fluxo bruto de bits e o decodifica, com o mesmo tipo de codificação selecionado pelo usuário e salvo em uma variável global anteriormente, retornando um quadro que é enviado a função *CamadaDeAplicacaoReceptora.* Esta por sua vez converte os bits do quadro na mensagem de texto em ASCII e a envia para a *AplicacaoReceptora* que finalmente mostra ao usuário a mensagem recebida.

Na figura 1 está presente o diagrama que ilustra todo o fluxo destes dois processos descritos acima. Mais detalhes da implementação estão demonstrados no próprio código, que foi desenvolvido com variáveis mnemônicas e exemplos de entrada e saída podem ser encontrados nos comentários das funções.

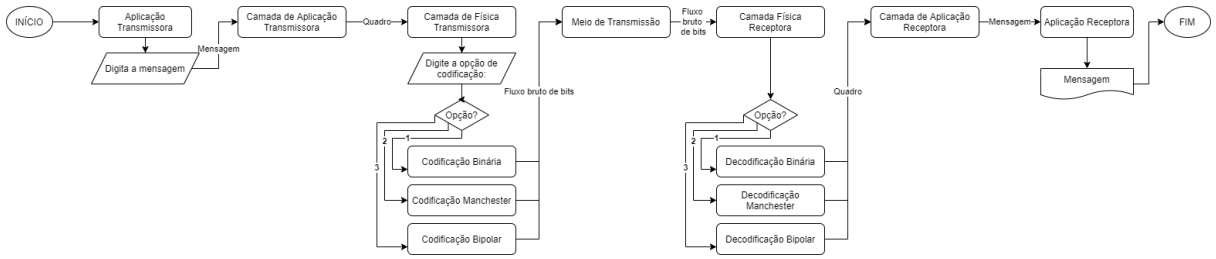

Figura 1: Fluxograma do simulador de Camada Física

Para garantir o bom funcionamento do simulador foram realizados alguns testes manuais para que se garantisse que todas as opções de codificação estivessem funcionando corretamente. Também foi rodado no código o analisador estático CppCheck (“cppcheck --enable=all --suppress=missingIncludeSystem .”) e o analisador dinâmico Valgrind (“valgrind --leak-check=full --track-origins=yes --verbose ./Simulador”) para a detecção de possíveis erros. O simulador ainda permite a opção de ver os logs de execução para acompanhar o processamento camada a camada.

Para compilar o código basta rodar o comando “g++ -std=c++11 *.cpp -o Simulador” e para executar o código basta rodar “./Simulador”.

Membros

Toda a implementação e relatório foi desenvolvido pelo aluno Gabriel Vasconcelos.

Conclusão

A maior dificuldade encontrada foi na readaptação à linguagem definida na especificação do trabalho, o C++. A falta de experiência em desenvolvimento nesta linguagem aliada a alguns alguns limites da própria (por ser uma linguagem de programação de nível mais baixo) levaram ao programador a se preocupar com alguns fatores (i.e. alocação de memória, definição de tipo e estruturas de dados mais simples) que não seriam um problema em linguagens mais modernas como Python ou Ruby.

Esse desafio é evidenciado também na entrada e saída de dados, visto que o simulador atual tem um limite de entrada tanto numérico, de 521 caracteres, como de codificação (ASCII) por não aceitar palavras com cedilha, til e acentos.

Outrossim, o desenvolvimento do trabalho contribuiu muito para fixar os conceitos aprendidos nas aulas assíncronas e visualizar como a teoria é implementada na prática, principalmente na relação entre camadas e o papel de cada uma encapsulado em funções.