Universidade de Brasília

Simulador de Camada de Enlace

Teleinformática e Redes 1

Prof. Geraldo Pereira

João Pedro Assis dos Santos (17/0146367)
Pedro Augusto Ramalho Duarte (17/0163717)
Waliff Cordeiro Bandeira (17/0115810)

Brasília
2020

Introdução

Os meios de comunicação com e sem fio precisam ter a capacidade de propagar sinais analógicos e por meio desses sinais analógicos sejam encaminhadas informações digitais através da modulação digital que por sua vez utiliza conceitos de canal de banda base e canal de banda passante.

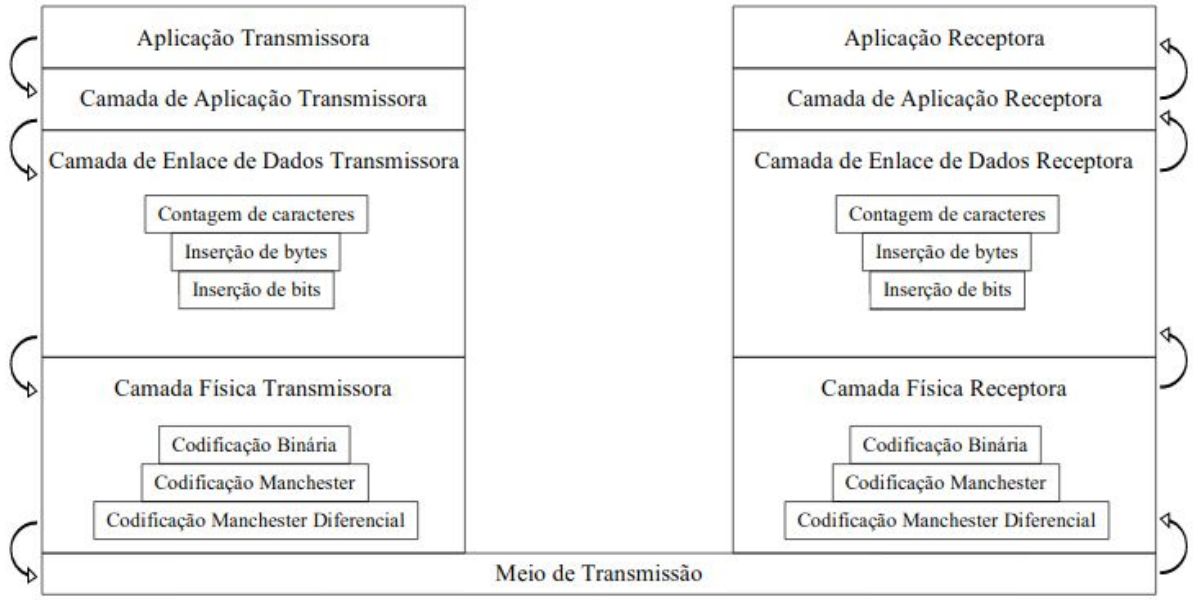
No primeiro trabalho da disciplina Teleinformática e Redes 1 foi abordado os conceitos de canal de banda base. O canal recebe esse nome pois a própria frequência está simétrica ao eixo de frequência 0, ou seja, a base do espectro de frequência será $f = 0$ Hz. A base por sua vez vai desde $f = 0$ até f_c (uma frequência de corte) (no qual todo o sinal estará contido nessa faixa de frequência).

Temos então que a banda base é a banda de frequência responsável pelos sinais transmitidos usados para modular uma porta para transmissão. Nos aparelhos celulares, por exemplo, é responsável por administrar as comunicações de rádio e trocas de informações com a antena dos dispositivos; no iPhone, também comanda o wi-fi e bluetooth.

A transmissão banda base é conhecida também como transmissão sem modulação, pois utiliza apenas um processo de codificação banda base (codificação de linha). A codificação de banda base é baseada na temporização (clocking), que determina o início e fim da transmissão dos bits para que possam ser decodificados, ou seja, determina o sinal e o nível do sinal (0 ou 1). Elas podem ser categorizadas em três distintivos níveis: curta distância (sensível ao sinal), médias distâncias (sensível à fase) e longas distâncias (códigos de blocos).

Para codificação de bits em símbolos é de extrema importância que o receptor consiga definir bem o momento que um símbolo termina e outro começa para que os bits sejam decodificados coerentemente. No primeiro trabalho prático foi desenvolvido o funcionamento do enlace físico através da implementação das seguintes codificações: Binária, Manchester e Manchester Diferencial.

Para o segundo trabalho, seguiremos a implementação da camada de enlace como descrita na imagem a seguir



Para a Camada de Enlace de Dados Transmissora serão explorados três diferentes algoritmos: Contagem de caracteres, inserção por bytes e inserção por bits. A contagem de caracteres consiste em colocar a quantidade de bytes que contém a mensagem no cabeçalho (primeiro byte) do quadro. Note que por definirmos o tamanho do quadro como um byte, o cabeçalho sempre será igual a 1. A inserção por bytes conta com a inserção de um byte de flag antes e após o campo de carga útil, caso essa flag esteja no campo de carga útil (seja um dado), deve-se colocar um byte de esc (escape) para indicar que aquela flag é um dado da mensagem, o mesmo ocorre caso o esc seja um dado do campo de carga útil, outro esc deve ser precedido. No caso da inserção de bits, quando temos um bit ‘0’ e cinco bits ‘1’ consecutivos no campo de carga útil, acrescenta-se um bit ‘0’ para sabermos que trata-se do campo de carga útil. Esses conceitos e sua implementação serão melhor abordados nos tópicos abaixo.

Implementação

A implementação do projeto utilizou práticas comuns no design orientado a objetos, explorando funcionalidades do C++ que iriam agilizar o desenvolvimento e torná-lo mais dinâmico. A arquitetura do sistema seguiu o padrão Façade, que foi de alta utilidade para uma implementação mais limpa e modular.

O sistema tem como base a classe **Simulação,** que anteriormente guardava ponteiros para duas interfaces, a **CamadaFisicaTransmissora** e a **CamadaFisicaReceptora** e nesse trabalho foram adicionadas mais duas camadas **CamadaEnlaceTransmissora** e **CamadaEnlaceReceptora**. Durante sua

inicialização, o construtor recebe o tipo de codificação e o tipo de enlace que será realizada e, com base nisso, inicializa as interfaces de acordo com o método desejado.

```
Simulacao::Simulacao(int tipoCodificacao, int tipoEnlace) {

    // Inicializa a camada física adequada
    switch(tipoCodificacao) {
        case 1:
            fisicaTransmissora = new CFTManchester();
            fisicaReceptora = new CFRManchester();
            break;
        case 2:
            fisicaTransmissora = new CFTManchesterDiferencial();
            fisicaReceptora = new CFRManchesterDiferencial();
            break;
        default:
               cout << "Tipo de codificação inválido (Utilizamos o tipo de
codificação binária)" << endl;
        case 0:
            fisicaTransmissora = new CFTBinaria();
            fisicaReceptora = new CFRBinaria();
            break;
    }

    // Inicializa a camada de enlace adequada
    switch (tipoEnlace) {
        case 1:
            enlaceTransmissora = new CETInsercaoBytes();
            enlaceReceptora = new CERInsercaoBytes();
            break;
        case 2:
            enlaceTransmissora = new CETInsercaoBits();
            enlaceReceptora = new CERInsercaoBits();
            break;
        default:
               cout << "Tipo de codificação inválido (Utilizamos o tipo de
codificação binária)" << endl;
        case 0:
            enlaceTransmissora = new CETContagemCaracteres();
            enlaceReceptora = new CERContagemCaracteres();
            break;
    }
}
```

Camada de Enlace

Contagem de caracteres

Nesse algoritmo o tamanho do quadro é o tamanho da mensagem que é recebida, com objetivo do método

. Nele é colocado um cabeçalho com o tamanho da mensagem que no máximo pode ser de 255, representando o tamanho da mensagem em bytes.

Inserção por Bytes

Na inserção por byte definimos o quadro com o tamanho fixo de **1 byte** (8 bits). A **flag** foi escolhida sendo **01000000** (**@**) e o **esc** foi escolhida sendo **00100011** (**#**). A implementação do algoritmo consiste na inserção de um byte de flag antes e após o campo de carga útil, caso essa flag esteja no campo de carga útil (seja um dado), deve-se colocar um byte de esc (escape) para indicar que aquela flag é um dado da mensagem, o mesmo ocorre caso o esc seja um dado no campo de carga útil, outro esc deve ser precedido para indicar isso.

Inserção por Bits

Na inserção de bits foi definido o tamanho do quadro como fixo em 1 byte. Quando temos um bit ‘0’ e cinco bits ‘1’ consecutivos no campo de carga útil (011111), acrescenta-se um bit ‘0’ (0111110) para sabermos que trata-se do campo de carga útil. Antes e após cada quadro (que consiste em 1 byte) temos um byte (8 bits) de flag (01111110), sendo assim, fica possível diferenciar a flag do campo de carga útil e fazer a recepção detectando as possíveis falhas de transmissão.

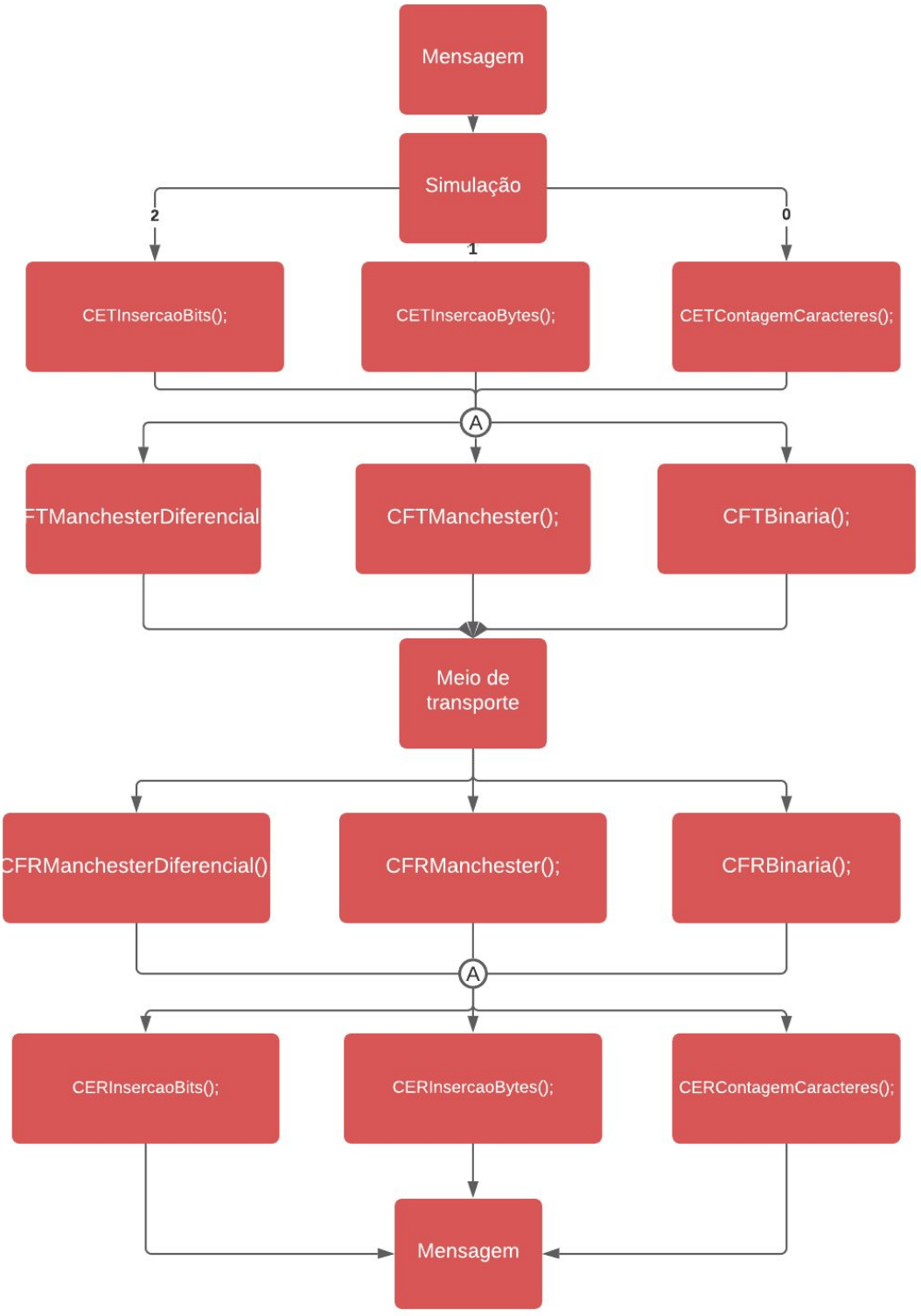
As classes feitas neste trabalho foram feitas pelo mesmo modelo do anterior e vamos exemplificar novamente utilizando a **CamadaEnlaceTransmissora** que é uma classe virtual pura, ou seja, não pode ser instanciada diretamente e serve apenas como uma interface para classes filhas. Por sua vez, as classes filhas herdam o método **execute()** e o sobrescrevem, fazendo com que a implementação desse método seja dependente da classe que o invoca.

Dessa maneira, após instanciar a classe **Simulação**, sempre chamaremos **transmissora->execute()** e o compilador irá decidir, em tempo de execução, qual método executar. Essa prática é interessante, pois abstrai o método de codificação utilizado. E é aplicada essa mesma hierarquia para as receptora e transmissora, das camadas de enlace e física.

A classe BitArray que armazena a informação é basicamente a mesma do trabalho passado, porém foram adicionados novos métodos para facilitar o tratamento da informação na camada de enlace, como os método getByte, getCabecalho.

Utilizando todas as classes supracitadas, o fluxo do código é dado por:

Fluxograma



Membros

Ao início do projeto foi utilizada uma técnica do Extreme Programming (XP) conhecida como pair programming, a diferença é que ao invés de desenvolver com dois, o desenvolvimento ocorreu com três desenvolvedores de forma simultânea utilizando a plataforma discord para conversa e compartilhamento de tela.

O momento inicial citado foi utilizado para definição da arquitetura, técnicas e tecnologias que seriam utilizadas para o desenvolvimento do projeto. Foi feita então as sessões de pair programming necessárias para a implementação de toda a arquitetura do projeto. A linguagem utilizada foi C++ com paradigma orientado à objetos e durante o desenvolvimento de todas as classes e a conclusão da arquitetura principal (a interface que consistiria em todo o enlace da camada física, o esqueleto do projeto) foi feito de forma conjunta.

Após a sessão inicial, na qual havia sido trabalhado os conceitos da disciplina para implementação da arquitetura da camada de enlace físico, o Pedro Augusto fez os métodos de acesso bit a bit, codificação binária e a refatoração/documentação de todo o código realizado, o João Pedro ficou responsável por terminar as codificações restantes (Manchester e Manchester Diferencial) e o Waliff pela confecção do artefato relatório.

Apesar de termos algumas tarefas atribuídas a desenvolvedores específicos, o trabalho foi feito de forma bem conjunta. Inicialmente iríamos utilizar a metodologia Scrum para organizar e distribuir melhor as tarefas de cada iteração, mas ao fazer as sessões iniciais de pair programming e com o auxílio das facilidades de gerência e versionamento que o GitHub proporciona, achamos que não seria tão necessário uma definição tão engessada da gerência para a demanda desse projeto.

Na implementação referente à camada de enlace, foi dividido o trabalho previamente, sendo que o Pedro Augusto ficou responsável pela contagem de caracteres, o João Pedro pela inserção por bits e o Waliff Cordeiro pela inserção por bytes. Sempre que necessário, fazíamos sessões de pair programming para que todos soubessem o que estava sendo tratado em todo o código, tendo um entendimento completo do desenvolvimento e do conteúdo tratado, além de ajudar no desenvolvimento de todas as partes.

Conclusão

O desenvolvimento nos proporcionou exercitar conceitos relevantes sobre redes e o funcionamento do enlace físico. Através da implementação pôde-se perceber na prática alguns conceitos e dificuldades de transmissão e implementação de um enlace de camada.

Tivemos a oportunidade de exercitar os conceitos de transmissão banda base que também é conhecida como transmissão sem modulação. Verificamos na prática a importância de que a codificação de bits em símbolos aconteça com a melhor precisão possível pelo receptor, pois é de nosso interesse saber o momento que um símbolo termina e outro começa para que os bits sejam decodificados com maior assertividade.

Tentando garantir uma maior precisão, várias codificações e protocolos foram desenvolvidos ao longo dos anos. O exercício para trabalhar esses conceitos foi desenvolvido através da implementação das codificações Binária, Manchester e Manchester Diferencial.

A camada de enlace é responsável pela transmissão e recepção de quadros, assim como delimitá-los, e pelo controle de fluxo, ela detecta e pode corrigir eventuais erros que possam aparecer na camada física. Os conceitos de camada de enlace foram trabalhados com base em três conceitos: contagem de caracteres, inserção por bytes e inserção por bits.

O desenvolvimento do projeto utilizando a linguagem de programação C++ em um paradigma de POO, utilizando o design pattern Facade e conceitos de Teleinformática e Redes I proporcionou também a revisão de conceitos importantes de programação. O projeto foi desenvolvido com base na classe Simulação, que guarda ponteiros para quatro interfaces, a CamadaFisicaTransmissora, CamadaFisicaReceptora, CamadaEnlaceTransmissora e a CamadaEnlaceReceptora utilizando conceitos de herança e override para uma implementação menos verbosa e mais eficiente, permitindo no futuro que sejam adicionadas outros tipos de camadas mudando poucas partes do código.

Referências

Materiais disponibilizados ao decorrer da disciplina: livro texto e vídeo aulas ministradas pelo professor Marcelo Antonio Marotta.

Wikipedia. Banda base. Disponível em: <https://pt.wikipedia.org/wiki/Banda_base>. Acesso em: 30 de outubro de 2020.

Até o momento. Design Pattern Facade. Disponível em: <https://www.ateomomento.com.br/facade-padrao-de-projeto/>. Acesso em 30 de outubro de 2020.

Wikipedia. Camada de enlace de dados. Disponível em: <https://pt.wikipedia.org/wiki/Camada_de_enlace_de_dados>. Acesso em 29 de novembro de 2020.