# Implementação de sistemas distribuídos usando Sockets

**Aécio de Oliveira**[1], **Aurelício Pereira**[2], **Gleidson Ramos3, Guilherme França**[4], **Luiz Sacramento**[5]

[1]Instituto de Informática – Universidade Estadual da Bahia (UNEB)
CEP: 41.150-000 – Salvador – Ba – Brazil

## 1. Fundamentação teórica

Criar um sistema distribuído o qual implementa um servidor para armazenamento e manutenção de listas. Além disso, o sistema deve possibilitar a inserção de outros módulos clientes. Esse projeto, trabalha com protocolos pertencentes a camada de aplicação, ou seja, é responsável pela transferência de dados e troca de mensagens. A escolha foi a utilização do socket TCP, por ser mais confiável. Por ultimo, utilizou-se conceitos abordados em aula, sendo eles: **tratamento de falhas e erros** com o **try e catch**, outros conceitos usados foram **concorrência e coordenação** , **ausencia de relógio global e transparência.**

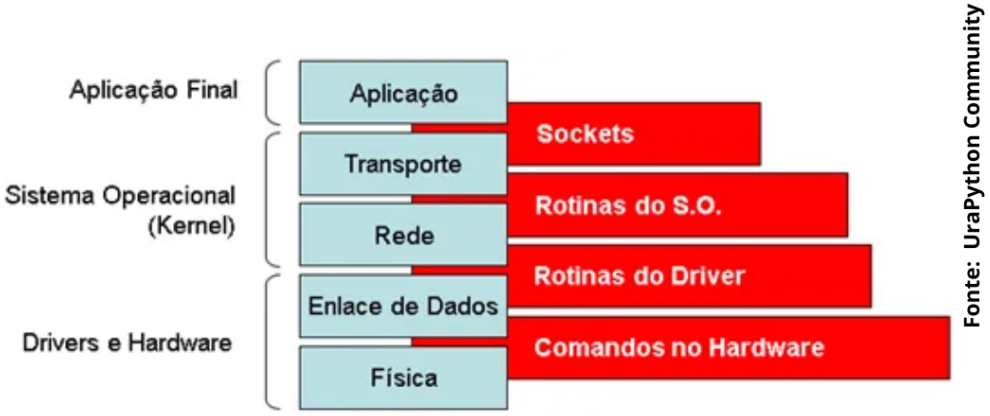


**Figure 1. Camadas pertencentes a um socket**

## 2. Materiais e métodos

A figura acima descreve a comunicação realizada em um Socket. De maneira geral o cliente interage com a camada de aplicação solicitando a criação de uma estrutura que no caso em específico é uma lista. Essa lista permite um leque de possibilidades que são: criar, excluir, inserir e recuperar. Em seguida o serviço será aplicado em suas respectivas camadas mais abaixo que podem ser: transporte, rede, enlace e física.

1. Cliente envia uma requisição para criar uma lista
2. Servidor cria uma lista
3. Cliente envia uma requisição para inserir criar uma lista
4. Servidor responde à mensagem com a inserção dos dados
5. Cliente envia uma requisição para recuperar lista
6. Servidor recupera a lista
7. Cliente envia uma requisição para apagar lista
8. Servidor responde à mensagem notificando que a lista foi apagada

**Coordenação e concorrência:** É feito o uso de conceitos de concorrência e coordenação para a comunicação entre os processos cliente e servidor.

Por exemplo, no lado do servidor, são criadas múltiplas threads para lidar com várias conexões simultâneas permitindo que o servidor atenda a vários clientes ao mesmo tempo. Essas threads podem compartilhar recursos, como dados da lista, e é importante coordenar o acesso a esses recursos para evitar problemas de concorrência, como conflitos de escrita ou leitura simultâneas.

No lado do cliente, é importante coordenar a comunicação com o servidor para que as mensagens sejam enviadas e recebidas na ordem correta e que não haja conflitos de acesso a recursos compartilhados.

Ou seja, a concorrência e coordenação são fundamentais para a comunicação eficiente e segura entre processos cliente e servidor usando sockets.

**Transparência:** A transparência é a habilidade de esconder detalhes de implementação aos usuários, fazendo com que eles possam interagir com a rede de forma simples e intuitiva. No caso dos sockets TCP, alguns aspectos de transparência são alcançados através dos protocolos como os presentes na camada de transporte do modelo OSI.

Por exemplo, o protocolo TCP oferece uma transparência de conexão, permitindo que as aplicações estabeleçam uma conexão entre si sem precisar se preocupar com detalhes de rede, como roteamento e fragmentação de pacotes. O protocolo também oferece mecanismos de confiabilidade, garantindo que os dados transmitidos sejam entregues corretamente e em ordem.

Além disso, a transparência de endereçamento é outro aspecto importante em sockets TCP, permitindo que as aplicações se comuniquem de forma transparente usando endereços IP e números de porta. Esses endereços são normalmente abstraídos pelo sistema operacional e pela API do socket, tornando a comunicação mais fácil para as aplicações.

**Falhas e erros:** o intuito é retornar qualquer inconsistência, como alertas, erros, falhas existentes no código ou elementos que afetem a operabilidade do código.

## 3. Protocolo de comunicação entre servidor e cliente

O protocolo utilizado será o *TCP(Transfer Control Protocol)*, por conta da sua confiabilidade em conjunto com os *HTTP (Hypertext Transfer Protocol)* e *HTTPS(Hypertext Transfer Protocol Secure)* para a transferência dos dados. Já que, o critério utilizado para estabelecer a conexão cliente/servidor não é velocidade, mas sim a confiabilidade dos dados inseridos na lista, optou-se por esses protocolos ditos mais seguro.

No código a seguir a classe ServerSocket é usada para criar um servidor *TCP* e essa classe cria um socket de cliente *TCP*. Além disso, o método *accept()* é usado para aguardar conexões de clientes, e o método *getInputStream()* é usado para receber dados do cliente e o *getOutputStream()* para enviar dados ao cliente.

```
public class Server {

    public static void main(String[] args) throws IOException{
        ServerSocket serverSocket = new ServerSocket(1234);
        System.out.println("Socket servidor iniciado na porta 1234.");

        while (true) {
            Socket socket = null;

            try {
                socket = serverSocket.accept();
                System.out.println("Conex o estabelecida.");

                DataInputStream in = new DataInputStream(socket.getInputStream
                DataOutputStream out = new DataOutputStream(socket.getOutputSt

                System.out.println("Thread parao novo cliente");
                Thread t = new ClientHandler(socket, in, out);
                t.start();
            }
            catch (Exception e){
                socket.close();
                e.printStackTrace();
            }
        }
    }
}
```

## 4. Referências


Beej. **Beej's Guide to Network Programming Using Internet Sockets**. Disponível em: *https://beej.us/guide/bgnet/* acesso 25 de abril de 2023

GeeksforGeeks. **Introducing Threads in Socket Programming in Java**. Disponível em: *https://www.geeksforgeeks.org/introducing-threads-socket-programming-java/* acesso 30 de maio de 2023

SALUTES, Bruno. **O que é HTTP**. Disponível em: *https://canaltech.com.br/internet/o-que-e-http* acesso 18 de abril de 2023

S.K. Rutledge; R.M. Olle; J.M. Cooper . **Atomic oxygen effects on SiO/sub x/ coated Kapton for photovoltaic arrays in low Earth orbit**. Disponível em: *https://beej.us/guide/bgnet/* acesso 28 de abril de 2023

UraPython Community. **Introdução a Sockets em Python** .Disponível em: *https://medium.com/@urapython.community/introdução-sockets-em-python-44d3d55c60d0* acesso 17 de abril de 2023