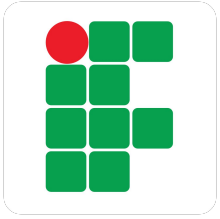
API Berkeley Sockets

Docente: Adriano Antunes Prates
Discente: Victor Gonçalves Alencar



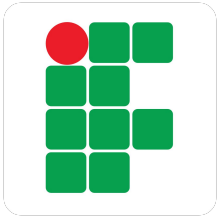
Importância Histórica

O que é a API Berkeley Sockets?

- Interface de programação para comunicação em rede.
- Criada no início dos anos 1980 pela Universidade de Berkeley.
- Parte integrante do sistema BSD Unix.

Necessidade na década de 1980:

- A expansão das redes de computadores exigia uma interface padrão.
- Sistemas operacionais precisavam de formas mais simples para permitir a troca de dados em redes.
- A ARPANET (precursora da internet) destacou a necessidade de interoperabilidade.
- A Solução de Berkeley : Desenvolvimento do BSD Unix com suporte ao protocolo TCP/IP.
    - API Berkeley Sockets lançada em 1983.



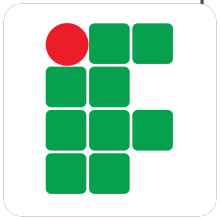
Importância Histórica

Por que a API foi criada?

1. Facilitar a comunicação em rede: Interface padronizada para programas usarem TCP/IP.
2. Proporcionar interoperabilidade:Compatibilidade entre diferentes sistemas operacionais.
3. Abstrair a complexidade das redes: Simplificar a programação com detalhes técnicos invisíveis ao programador.

Impacto da API Berkeley Sockets

1. Padrão para comunicação em redes: Adoção em Linux, Windows (Winsock) e Mac OS.
2. Fundação da Internet moderna: Protocolo HTTP (web), SMTP (e-mails).
3. Base para novas tecnologias: Suporte a UDP, IPv6, sistemas distribuídos e peer-to-peer (P2P).
4. Simplificação no desenvolvimento de software: Permitiu criação de aplicações de rede de forma portátil e eficiente.



Principais Primitivas da API

socket(): Cria um novo socket, que é usado para a comunicação entre processos.

bind(): Associa o socket a um endereço e porta específicos.

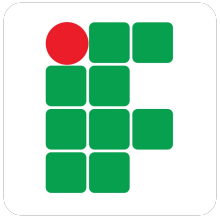
Listen(): Colocar o socket em modo de escuta, aguardando conexões de clientes.

accept(): Aceita uma conexão de um cliente.

c**onnect()**: Estabelece uma conexão com o servidor.

send() e **recv()**: Envia e recebe dados entre o cliente eo servidor.

close(): Fecha a conexão.



Biblioteca socket do Python

socket.socket(): Cria um novo socket.

socket.bind(): Associa o socket a um endereço e porta.

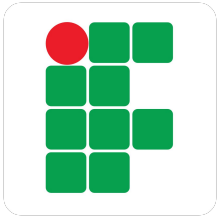
socket.listen(): Colocar o socket em modo de escuta.

socket.accept(): Aceita uma conexão de um cliente.

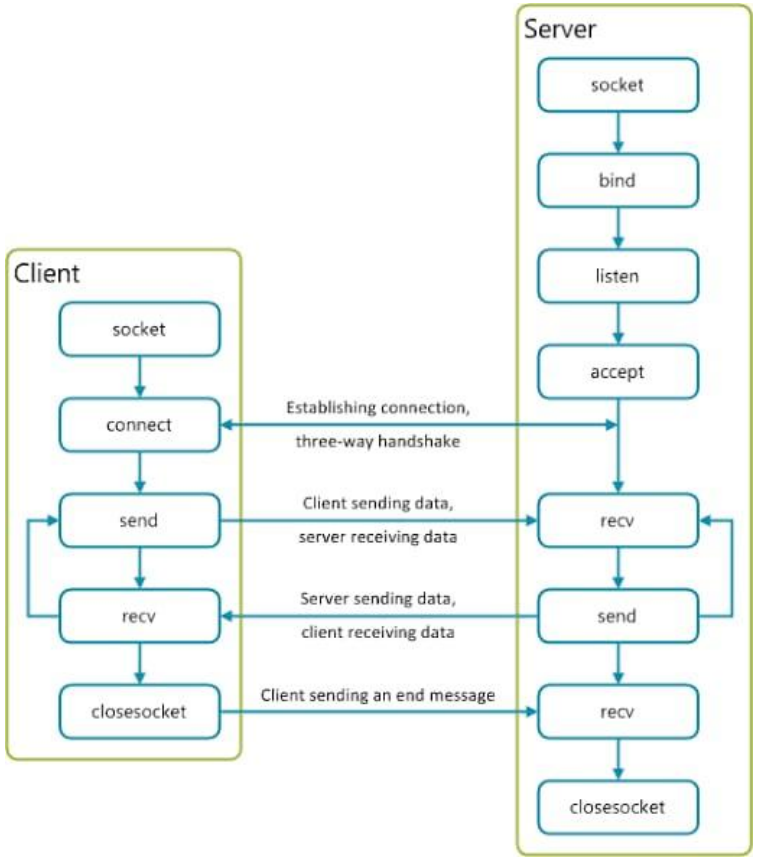
socket.connect(): Conecta-se a um servidor.

socket.send() e **socket.recv()**: Envia e recebe dados através do socket.

socket.close(): Fecha o socket, encerrando a comunicação.



Fluxo de operações Aplicação cliente-servidor



Problema

Problema a ser resolvido:

- Um servidor abre um socket em modo listen.
- Um cliente se conecta ao servidor.
- O servidor exibe o IP e a porta do cliente, e envia uma mensagem de "bem-vindo".
- O cliente exibe a mensagem recebida e envia "obrigado, servidor".
- O servidor exibe a mensagem do cliente e encerra a conexão.