PROPOSTA DE PROJETO FINAL:

SIMULADOR INTERATIVO DE PROTOCOLO TCP VIA WEBSOCKETS

1. Proposta Geral

O projeto consiste no desenvolvimento de uma ferramenta educacional interativa. O objetivo principal é esclarecer o funcionamento do protocolo de transporte TCP (Transmission Control Protocol), permitindo que estudantes de redes visualizem e interajam com conceitos abstratos que ocorrem na camada de transporte.

Cenário de Uso:

A aplicação funcionará como um laboratório virtual. Dois usuários por sessão acessarão o sistema via navegador web.

- **Conexão:** O sistema simulará visualmente o 3-Way Handshake para estabelecer uma conexão virtual entre os dois clientes.
- **Troca de Dados:** Um usuário poderá digitar uma mensagem, que será fragmentada em "pacotes virtuais".
- **Simulação de Rede:** O sistema permitirá injetar falhas controladas na comunicação, como perda de pacotes, latência e corrupção.
- **Reação do Protocolo:** A interface gráfica demonstrará como o TCP reage a essas falhas, conceitos que geralmente são invisíveis em aplicações reais.

2. Tecnologias Adotadas

A arquitetura do sistema será baseada no modelo Cliente-Servidor, utilizando comunicação Full-Duplex para garantir interatividade em tempo real.

- **Linguagem Backend:** Python 3.10+.
- **Protocolo de Transporte:** WebSocket (RFC 6455).
- **Bibliotecas Principais:** asyncio, websockets e json.
- **Frontend:** HTML5, CSS3 e JavaScript.
- **Infraestrutura e Deploy:** AWS EC2, Nginx e Systemd.

3. Justificativa

Este projeto foi concebido para aplicar na prática os conceitos fundamentais abordados ao longo do curso, conectando a teoria de sistemas distribuídos com uma implementação moderna e interativa:

Comunicação entre Processos e Sockets:
Embora utilize WebSockets (uma evolução para a web), a aplicação demonstra visualmente o ciclo de vida de uma conexão orientada a fluxo (TCP), ilustrando os conceitos de bind,

listen, accept, connect e send/recv através da simulação do handshake e troca de mensagens, permitindo ao aluno visualizar o que ocorre "por baixo dos panos".

Concorrência:
Para suportar múltiplos alunos interagindo simultaneamente (pares de conexão), o servidor exige um modelo de execução concorrente. O projeto aborda o desafio de gerenciar múltiplos fluxos de execução, optando pelo modelo de I/O assíncrono (asyncio) como uma alternativa viável, garantindo que o bloqueio de um cliente não interfira nos demais.

Sincronização e Ordenação:
A simulação do protocolo TCP impõe a necessidade de lidar com a ordenação de eventos. Em um sistema distribuído real, a ordem de envio não garante a ordem de chegada. O projeto força o aluno a entender os mecanismos de sincronização necessários para reconstruir a mensagem original a partir de pacotes que podem chegar desordenados ou atrasados, ilustrando na prática a complexidade de manter a consistência de dados em ambientes não confiáveis.

Transparência e Tratamento de Falhas:
O projeto aplicará o conceito de Transparência de Falhas, demonstrando a capacidade do sistema de mascarar erros e continuar operando apesar da falha de componentes. De forma didática, a aplicação tornará visível o mecanismo que o TCP utiliza para alcançar essa transparência: o aluno poderá observar graficamente a retransmissão automática de segmentos perdidos e a reordenação de pacotes fora de sequência. Isso ilustra na prática como o protocolo oculta a instabilidade da rede, fazendo com que o sistema pareça, para a camada de aplicação, uma unidade única e confiável que entrega a mensagem íntegra, independentemente das falhas ocorridas no transporte.

4. Equipe e Responsabilidades Técnicas

O projeto será desenvolvido individualmente.

Thalles Kenedy Oliveira Maia:

Arquitetura e Backend: Desenvolvimento do servidor de aplicação em Python, implementação da lógica de simulação de rede e gerenciamento de conexões assíncronas via asyncio.

Frontend e Interface: Criação da interface gráfica e desenvolvimento da lógica cliente em JavaScript para renderização das animações e controle de estado da conexão TCP virtual.

Infraestrutura: Configuração do ambiente em nuvem (AWS EC2), configuração do servidor web Nginx como Proxy Reverso, implementação de segurança (SSL/TLS) e automação de serviços via Systemd.

5. Repositório do Projeto

https://github.com/TkMaia7/tcp-simulator