Lista De Exercícios #04: SOCKETS – File Server baseado em UDP

Conforme discutido em sala, é possível fazer dois pequenos programas que funcionam como cliente e servidor de arquivos UDP (por mais estranho que isso pareça).

Uma versão inicial dos programas está no GitHub Classroom.

O cliente, no entanto, apresenta um problema: fica travado esperando dados do servidor quando este já terminou de enviar todo o arquivo solicitado, mas o cliente não sabe. Existem pelo menos duas soluções básicas para esse problema:

1. Programar um timeout no lado do cliente (usando o método `settimeout` da API Socket do Python). Assim, se `recvfrom` demorar mais do que o tempo de espera estabelecido, o cliente entende que não mais receberá dados e pode prosseguir;

2. Fazer o servidor enviar uma mensagem especial no final. Por exemplo, depois de enviar todos os dados do arquivo, o servidor poderia enviar um marcador de fim de transmissão e o cliente verificaria se esse marcador foi recebido para encerrar o loop, ao invés de depender apenas do tamanho do arquivo.

Implemente versões para cada uma das soluções apresentadas (crie dentro do assignment do GitHub Classroom pastas separadas para cada uma das implementações). Escreva no README do assignment as vantagens e desvantagens de cada um dos métodos.

Explique (escreva no README do assignment) por que, mesmos sem apresentar erros aparentes, as soluções implementadas não transferem arquivos grandes como deveriam.

Crie uma pasta chamada TCP Model e implemente essa aplicação utilizando o protocolo TCP ao invés do UDP e comente no README as diferenças (vantagens e desvantagens) em relação ao código implementado na pasta UDP Model