Estudo Dirigido #04: SOCKETS – Files 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).

Versões iniciais dos programas estão no Moodle. 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:

- 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;

- Programar o envio do tamanho do arquivo pelo servidor imediatamente antes de enviar os próprios dados. O cliente deve receber esse tamanho e parar de esperar por dados do servidor quando já recebeu dados suficientes.

ATENÇÃO: A sua resposta a essa lista deve ser um arquivo compactado contendo três pastas: **Q1**, **Q2** e **Q3**, cada uma com arquivos que respondem às questões a seguir. O conteúdo de cada pasta varia com a questão, conforme explicado em cada uma delas.

1. Implemente o uso de timeout no programa cliente em anexo e teste a solução. Discuta as vantagens e desvantagens desse método. Apresente três arquivos na pasta de resposta a essa questão: **udp-file-server-v3.py, udp-file-client-v3.py** e **discussao.txt**.

2. Implemente o uso do envio do tamanho do arquivo do servidor para o cliente e teste a solução. Discuta as vantagens e desvantagens desse método. Apresente três arquivos na pasta de resposta a essa questão: **udp-file-server-v4.py**, **udp-file-client-v4.py** e **discussao.txt**.

3. Explique por que tanto os programas de resposta às questões 1 e 2, mesmos sem apresentar erros aparentes não transferem arquivos grandes como deveriam. Gere um pequeno arquivo **discussao.txt** na pasta de resposta a essa questão.