Prática – UDP e TCP

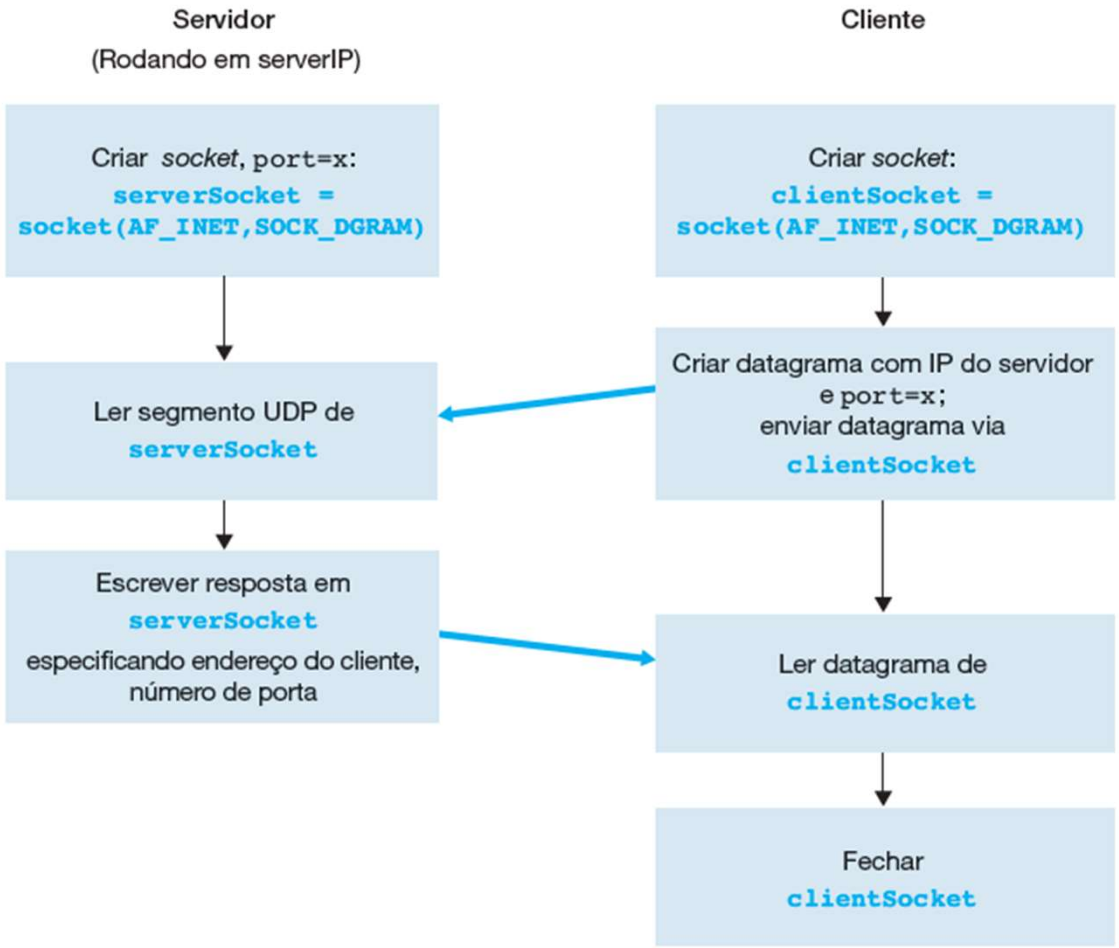
Prof. Jean Lima

Exemplo Prático

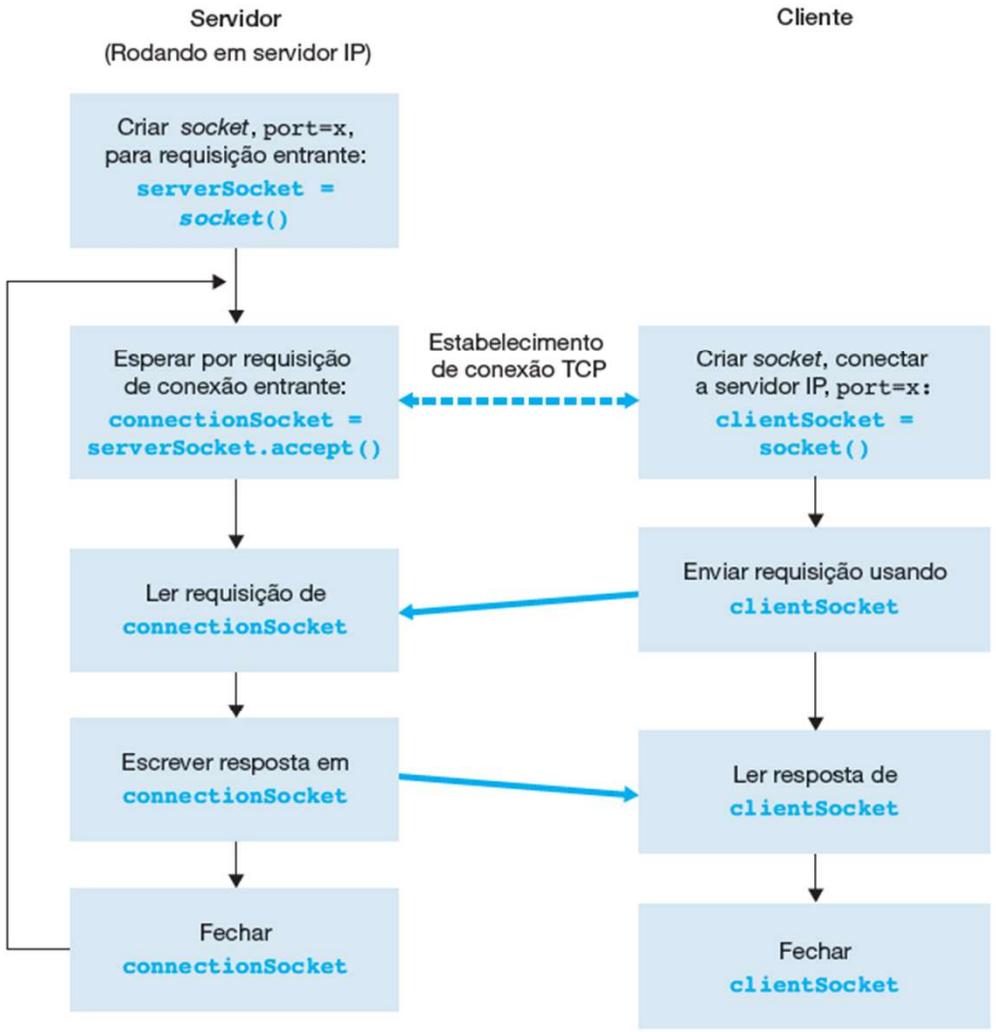
Problema:

1. Um cliente le uma linha de caracteres (dados) do teclado e a envia para o servidor.
2. O servidor recebe os dados e converte os caracteres para maiúsculas.
3. O servidor envia os dados modificados ao cliente.
4. O cliente recebe os dados modificados e apresenta a linha em sua tela.

Exemplo Prático – Solução com UDP



Exemplo Prático – Solução com TCP



Tarefa 1 - Servidor *Web*

Nesta tarefa, você desenvolverá um servidor Web simples em Python, capaz de processar apenas uma requisição. Seu servidor Web (i) criará um *socket* de conexão quando contatado por um cliente (navegador); (ii) receberá a requisição HTTP dessa conexão; (iii) analisará a requisição para determinar o arquivo específico sendo requisitado; (iv) obterá o arquivo requisitado do sistema de arquivo do servidor; (v) criará uma mensagem de resposta HTTP consistindo no arquivo requisitado precedido por linhas de cabeçalho; e (vi) enviará a resposta pela conexão TCP ao navegador requisitante. Se um navegador requisitar um arquivo que não está presente no seu servidor, seu servidor deverá retornar uma mensagem de erro "*404 Not Found*".

Tarefa 2 – UDP *Pinger*

Nesta tarefa de programação, você escreverá um programa *ping* do cliente em Python. Seu cliente enviará uma mensagem *ping* simples a um servidor, receberá uma mensagem *pong* correspondente de volta do servidor e determinará o atraso entre o momento em que o cliente enviou a mensagem *ping* e recebeu a mensagem *pong*. Esse atraso é denominado tempo de viagem de ida e volta (*round-trip time* — RTT). A funcionalidade oferecida pelo cliente e servidor e semelhante a fornecida pelo programa *ping* padrão, disponível nos sistemas operacionais modernos. Porém, os programas *ping* padrão usam o Internet Control Message Protocol (ICMP). Aqui, criaremos um programa *ping* baseado em UDP, fora do padrão (porem simples!).

Seu programa *ping* deverá enviar 10 mensagens *ping* ao servidor de destino por meio de UDP. Para cada mensagem, seu cliente deverá determinar e imprimir o RTT quando a mensagem *pong* correspondente for retornada. Como o UDP é um protocolo não confiável, um pacote enviado pelo cliente ou servidor poderá ser perdido. Por esse motivo, o cliente não poderá esperar indefinidamente por uma resposta a uma mensagem *ping*. Você deverá fazer que o cliente espere até 1s por uma resposta do servidor; se nenhuma resposta for recebida, o cliente deverá considerar que o pacote foi perdido e imprimir uma mensagem de acordo.

Por fim, após a realização dessa rotina, o programa cliente deve imprimir na tela o:

- Total de pacotes enviados;
- Total de recebidos;
- Percentual de pacotes perdidos;
- Tempo mínimo de resposta;
- Tempo Máximo de Resposta;
- Tempo Médio de Resposta.

Tarefa 3 – Cliente de Correio

O objetivo desta tarefa de programação é criar um cliente de correio simples, que envia e-mail a qualquer destinatário. Seu cliente precisará estabelecer uma conexão TCP com um servidor de correio (por exemplo, um servidor de correio do Google), dialogar com esse servidor usando o protocolo SMTP, enviar uma mensagem de correio a um destinatário (por exemplo, seu amigo) pelo servidor de correio e, por fim, fechar a conexão TCP com o servidor de correio.

Tarefa 4 - Servidor *Proxy Web multithreaded*

Nesta tarefa, você desenvolverá um *proxy* da Web. Quando seu *proxy* receber de um navegador uma requisição HTTP para um objeto, ele gerará uma nova requisição HTTP para o mesmo objeto e a enviará para o servidor de origem. Quando o *proxy* receber a resposta HTTP correspondente com o objeto do servidor de origem, ele criará uma nova resposta HTTP, incluindo o objeto, e a enviará ao cliente. Esse *proxy* será *multithreaded*, de modo que poderá lidar com várias requisições ao mesmo tempo.