Trabalho prático – Programação com Sockets

Alunos: Angelo Henrique Peres Cestari Junior e Francisco Ferreira Lima Neto

Professora: Hana Karina Salles Rubinsztejn

Descrição da implementação do servidor:

O servidor foi implementado em Python, com o uso de threads para lidar com as solicitações dos clientes e com a mensagem recorrente de status do servidor.

O servidor contém um loop principal para leitura dos dados recebidos dos clientes. A cada requisição de um cliente, é criada uma thread para tratá-la e o servidor já volta a escutar por outras requisições.

O tratamento das requisições realizado pelas threads criadas realiza a interpretação destas, as manipulações necessárias no registro de identificadores e o envio das respostas, em acordo com a especificação do trabalho.

É mantido um dicionário com os identificadores dos clientes registrados como chave e que mantém como dado o endereço do cliente.

Cada vez que um cliente da um OI ou TCHAU, é salvo em uma lista de logs uma string contendo o acontecimento e a mesma também é apresentada no console que o servidor foi executado.

As mensagens temporizadas com o status do servidor são tratadas por uma thread separada que fica em um loop de enviar as mensagens para os clientes e esperar 60 segundos para enviá-las novamente.

Foi implementado um quinto tipo de mensagem, representado pelo número 4, que representa a solicitação de uma lista com os atuais clientes registrados no servidor. Dessa forma, se o servidor recebe uma mensagem de tipo 4, mesmo se a origem ainda não estiver registrada, ele retorna uma mensagem de tipo 4 com o campo de texto da mensagem listando os identificadores registrados separados por vírgulas.

A fim de evitar problemas de concorrência, quando uma requisição que só lê da lista de clientes é executada, esta consulta uma cópia da lista gerada no começo do tratamento da requisição.

Assim, o servidor contém dois loops executando concorrentemente via separação por thread, um para recepção de mensagens e outro para o envio do status a cada 60 segundos. Cada vez que uma mensagem é recebida pelo servidor, uma thread é criada para lidar com essa mensagem.

Descrição da implementação do cliente:

O cliente foi desenvolvido em Python utilizando a biblioteca Streamlit para criar uma interface gráfica simples e interativa. A comunicação com o servidor é feita via protocolo UDP usando o módulo socket. Para garantir que o cliente possa receber mensagens do servidor sem interromper a interação do usuário com a interface, foi utilizada threading, permitindo que a recepção de mensagens aconteça em segundo plano.

A interface é organizada com um menu lateral que permite ao usuário navegar entre as diferentes funcionalidades: Criar Conexão, Enviar Mensagens, Encerrar Conexão, Receber Mensagens e Solicitar Lista de Clientes. Cada funcionalidade é apresentada em uma página separada para facilitar o uso.

- **Criação de Conexão**: Permite ao usuário registrar-se no servidor inserindo um ID, nome de usuário, endereço IP e porta.
- **Envio de Mensagens**: O cliente pode enviar mensagens para outro cliente ou para todos os clientes conectados.
- **Encerramento de Conexão**: Envia uma solicitação ao servidor para desconectar o cliente.
- **Recepção de Mensagens**: As mensagens recebidas do servidor são exibidas automaticamente em uma página dedicada, graças à thread de escuta em segundo plano.
- **Solicitação de Lista de Clientes**: Permite que o cliente solicite ao servidor a lista de todos os clientes conectados.

A utilização de threads para escutar mensagens do servidor garante que a interface permaneça responsiva enquanto o cliente aguarda ou recebe mensagens. Além disso, são feitas validações de entrada para garantir que o ID do usuário e a porta do servidor sejam corretos, e o tratamento de exceções assegura que qualquer erro de comunicação seja informado de forma clara ao usuário.

Essa implementação garante uma experiência de uso fluida e eficiente, permitindo que múltiplas operações ocorram de forma simultânea, sem travar a interface.