Definição de Protocolo - Trabaho Redes 2023.2

Um protocolo de chat em um contexto de rede de computadores é um conjunto de regras e padrões que governam a comunicação entre os clientes de chat, permitindo que eles troquem mensagens em tempo real. Esses protocolos são essenciais para garantir que as mensagens sejam enviadas, recebidas e exibidas corretamente pelos participantes de um chat.

Formato da mensagem: string 64 bits utf-8
PORTA: 18000
MENSAGEM PARA DESCONECTAR: “:D”
Os grupos se conectam por meio de um cliente utilizando o ip fixo do servidor e a porta estabelecida.

O código é um exemplo de um servidor de chat simples que utiliza sockets e threads em Python. Ele demonstra um protocolo básico de comunicação entre o servidor e os clientes. Vou explicar os principais elementos do protocolo:

1. **Porta de Comunicação**: O servidor está configurado para escutar na porta 18 (`PORTA = 18000`). Essa é a porta pela qual os clientes se conectam ao servidor para iniciar a comunicação.

2. **Endereço do Servidor**: O servidor é vinculado a um endereço IP específico, que neste caso é `''192.168.37.40'` (`SERVER = '192.168.37.40''`). Os clientes usam esse endereço para se conectarem ao servidor.

3. **Formato da Mensagem**: O protocolo define o formato das mensagens trocadas entre os clientes e o servidor, que é codificado em UTF-8 (`FORMAT = 'utf-8''`).

4. **Tamanho do Cabeçalho (HEADER)**: O cabeçalho da mensagem é uma parte importante do protocolo. Ele tem um tamanho fixo de 64 bytes (`HEADER = 64`). O cabeçalho é usado para informar o servidor sobre o tamanho da mensagem real que será enviada, permitindo que o servidor saiba quantos bytes esperar.

5. **Desconexão**: O valor “:D” é definido como a mensagem de desconexão (`DISCONNECT = ':D''`). Quando um cliente envia essa mensagem, o servidor entende que o cliente deseja se desconectar.

6. **Conexão e Comunicação**: O servidor usa um socket para aceitar conexões de clientes e iniciar threads para lidar com cada conexão. O loop no `handler` processa a comunicação com o cliente, recebendo mensagens, verificando o cabeçalho para determinar o tamanho da mensagem e enviando mensagens de volta para o cliente.

7. **Conexões Ativas**: O servidor acompanha o número de conexões ativas com a ajuda da função `threading.active_count()` e exibe esse número sempre que uma nova conexão é aceita.

8. **Início do Servidor**: O servidor é iniciado chamando a função `start()`, que fica ouvindo por novas conexões. Quando uma nova conexão é estabelecida, um novo segmento de execução (thread) é criado para lidar com essa conexão.