Lista de Exercícios - Programação com Sockets

Abstract

Essa é a nossa primeira lista de implementação. Implemente suas soluções, e compartilhe o código num relatório unificado ou, alternativamente, no github. Em todo caso, você precisa entregar um PDF único para ser avaliado. A leitura de textos adicionais ao PDF será opcional.

Estude o capítulo 2 do livro texto e implemente os seguintes programas.

1. Na seção 2.7.1 do livro de referência, na 7ª edição, são implementados os códigos do cliente (UDPClient.py) e do servidor (UDPServer.py) para uma aplicação em que o servidor recebe um único trecho de texto do cliente e o devolve com todas as letras em maiúsculo. No entanto, o cliente só consegue enviar um único trecho de texto antes de finalizar a conexão. Altere o código do cliente para permitir que o cliente envie vários trechos de textos, ou seja, de tal forma que o cliente envie texto múltiplas vezes. (Obs: ambos os códigos precisarão ser ligeiramente adaptados? ou apenas o cliente?).

2. Repita a questão anterior para o caso em que o cliente e servidor usam TCP. Discuta as possibilidades, vantagens e desvantagens, ao implementar no modo persistente e não persistente. Experimente executar vários clientes em paralelo, e veja o que acontece.

3. Repita a questão anterior, mas agora de tal forma que o servidor possa atender vários clientes em paralelo. Faça a questão tanto para UDP quanto TCP. No caso do UDP, cada thread ou processo será responsável por um trecho de texto, de forma completamente independente, como se os clientes emitindo vários textos, na realidade, fossem vários clientes. No caso de TCP, você pode escolher se irá considerar o modo persistente ou não persistente.

4. Faça um programa cliente cuja única função é enviar os N primeiros números naturais para um servidor pré-especificado. Faça também um programa servidor que imprima esses números no terminal à medida que eles vão chegando. Para esta questão, considere experimentar os dois tipos de protocolos de transporte, TCP e UDP. Você observou alguma diferença? Caso sim, qual a causa dessa diferença? Caso não, alguma diferença era esperada?

5. Nesta tarefa, você terá que restringir o acesso a um dado que está no servidor para somente usuários autenticados. Para isso, faça uma aplicação cliente que solicite o usuário e a senha, envie esses dados para o servidor (não se preocupe com criptografia) e então imprima no terminal a mensagem enviada pelo servidor. No lado do servidor, você deve implementar uma verificação do usuário e da senha enviados pelo cliente (pode ser com uma simples comparação com valores pré-estabelecidos) e então enviar de volta ou uma mensagem de erro (caso as informações de login estejam incorretas) ou enviar uma mensagem de sua preferência (caso as informações de login estejam corretas).

6. Bora fazer um chatbot pra ajudar a tirar dúvidas sobre a nossa disciplina? A aplicação é a seguinte: o comportamento do lado do servidor é descrito na Figura 1, onde as elipses correspondem a estados, as caixas vermelhas correspondem às mensagens que o servidor envia para um cliente assim que atinge cada estado e as caixas verdes são as condições que precisam ser satisfeitas para haver a troca de estado. A função do lado do cliente, por sua vez, é simplesmente se manter em um loop com três atividades simples: solicita ao usuário uma entrada de texto, envia esse texto pro servidor e imprime a resposta que receberá do servidor. Você deve implementar a interrupção desse loop para que o cliente finalize sua conexão com o servidor quando este finalizar a conexão com o cliente, ou seja, quando o servidor atingir o estado "Finalizar". Utilize sockets TCP para a comunicação nesta aplicação. Note que essa é uma comunicação *stateful* diferente do HTTP que é *stateless*. Comente as vantagens e desvantagens das duas abordagens de comunicação.

7. faça o problema p29: comente sobre os problemas que podem ocorrer ao se chamar o comando bind

8. faça o problema p30: comente sobre a vantagem de conexões múltiplas

9. faça o problema p31: como ponto adicional, leia o artigo

   `https://www.ibm.com/docs/en/i/`

   `7.1?topic=programming-socket-application-design-recommendations`

   e explique porque e quando é recomendável usar a flag MSG_WAITALL no TCP

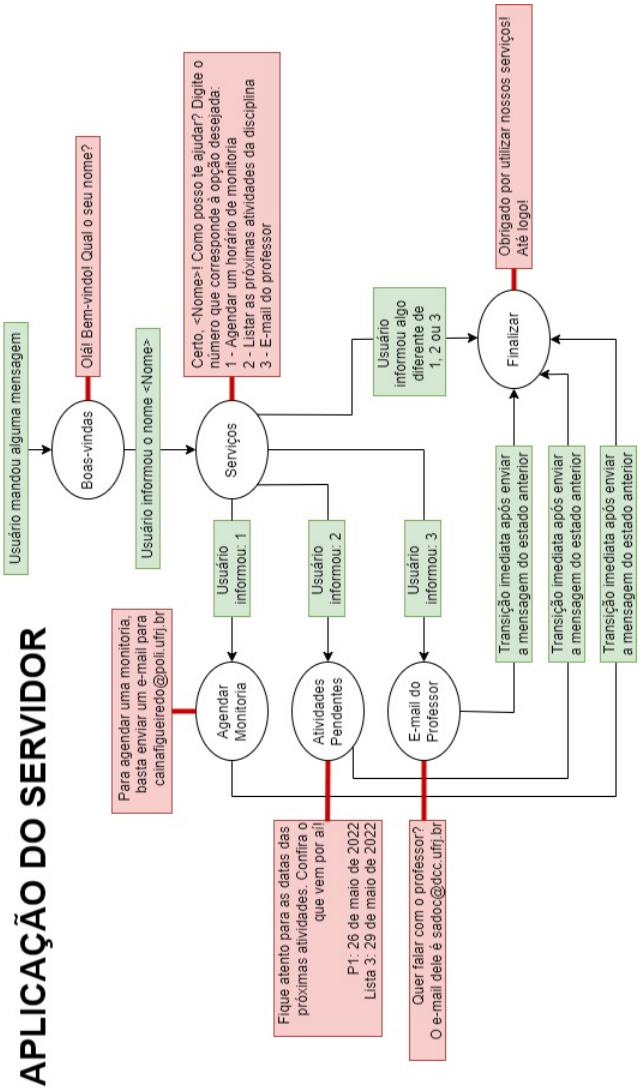

Figure 1: Comportamento do servidor do chatbot da disciplina