Centro de Ciências e Tecnologia

Ciência da Computação

UNIVERSIDADE FEDERAL DO CARIRI

Projeto: Servidor Web

Disciplina: Lab Redes (CC0044) Professora: Camila Oliveira

Objetivos

Neste projeto, você trabalhará com os conceitos básicos de programação de socket para conexões TCP em Python: como criar um socket, vinculá-lo a um endereço e porta específicos, bem como enviar e receber um pacote HTTP. Você também vai aprender alguns conceitos básicos do formato de cabeçalho HTTP.

Você desenvolverá um servidor web que lida com uma solicitação HTTP por vez. Seu servidor web deve aceitar e analisar a solicitação HTTP, obter o arquivo solicitado do sistema de arquivos do servidor, criar uma resposta HTTP que consiste no arquivo solicitado precedido por linhas de cabeçalho e, em seguida, enviar a resposta diretamente para o cliente. Se o arquivo solicitado não estiver presente no servidor, o servidor deve enviar um HTTP “404 Not Found” de volta para o cliente.

Código

Abaixo você encontrará o código esqueleto para o servidor Web. Você deve completar o código esqueleto. Os locais onde você precisa preencher o código estão marcados com #início e #fim. Cada lugar delimitado pode exigir uma ou mais linhas de código.

```
#import socket module
import socket
# In order to terminate the program
import sys

serverSocket = socket(AF_INET, SOCK_STREAM)
```

```
#Prepare um sevidor socket
#início
#fim
while True:
        #Estabeleça a conexão
        print('Pronto para servir...')
        connectionSocket, addr = #início                        #fim
        try:
                message = #início                               #fim
                filename = message.split()[1]
                f = open(filename[1:])
                outputdata = #início                            #fim
                #Send one HTTP header line into socket
                #início
                #fim
                #Send the content of the requested file to the client
                for i in range(0, len(outputdata)):
                        connectionSocket.send(outputdata[i].encode())
                connectionSocket.send("\r\n".encode())
                connectionSocket.close()
        except IOError:
                #Send response message for file not found
                #início
                 #fim
                #Close client socket
                #início
                #fim
serverSocket.close()
sys.exit()#Terminate the program after sending the corresponding data
```

Executando o servidor

Coloque um arquivo HTML (por exemplo, HelloWorld.html) no mesmo diretório em que o servidor está. Execute o programa servidor. Determine o endereço IP do host que está

executando o servidor (por exemplo, 128.238.251.26). A partir de outro host, abra um navegador e forneça a URL correspondente. Por exemplo: http://128.238.251.26:6789/HelloWorld.html. ‘HelloWorld.html’ é o nome do arquivo que você colocou no diretório do servidor. Observe também o uso da porta, número após os dois pontos. Você precisa substituir este número de porta por qualquer porta que você tenha usado no código do servidor. No exemplo acima, usamos o número da porta 6789. O navegador deve então exibir o conteúdo de HelloWorld.html. Se você omitir ":6789", o navegador assumirá a porta 80 e você obterá a página da web do servidor apenas se o seu servidor estiver escutando na porta 80.
Em seguida, tente obter um arquivo que não esteja presente no servidor. Você deve receber uma mensagem “404 Not Found”.

Servidor com threads

Atualmente, o servidor web trata apenas uma solicitação HTTP por vez. Agora, você deve implementar um servidor multithread que é capaz de atender a várias solicitações simultaneamente. Usando o encadeamento, primeiro crie um encadeamento principal em que seu servidor modificado escuta clientes em uma porta fixa. Quando recebe uma conexão TCP, solicitação de um cliente, ele estabelecerá a conexão TCP através de outra porta e atenderá o pedido do cliente em uma thread separada. Haverá uma conexão TCP separada em um thread separado para cada par solicitação/resposta.

Teste usando 3 clientes acessando o servidor ao mesmo tempo.

Cliente HTTP

Em vez de usar um navegador, escreva seu próprio cliente HTTP para testar seu servidor. Seu cliente se conectará com o servidor usando uma conexão TCP, enviará uma solicitação HTTP para o servidor e exibirá a resposta do servidor como saída. Você pode assumir que a solicitação HTTP enviada é um método GET. **Você não pode usar as bibliotecas HTTP do python.**
O cliente deve receber argumentos de linha de comando especificando o endereço IP do servidor ou o nome do host, a porta na qual o servidor está escutando e o caminho no qual o objeto solicitado está armazenado no servidor.
A seguir está um formato de comando de entrada para executar o cliente:
client.py server_host server_port filename

Entregar

Você entregará o código completo do servidor junto com as capturas de tela do navegador do cliente e do console servidor dos teste executados em cada etapa.
Também entregará o códgio do seu cliente HTTP juntamente com as capturas de tela do teste de seu funcionamento.