ENSF 462

Laboratory 2

Title: Web Server and UDP Pinger

Name: Hongwoo Yoon

UCID: 30113779

Section: B02

Date: Oct 9, 2024

Part 1 Web Server

Server Directory

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Output

멀티미디어 소프트웨어, 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Code

#import socket module

from socket import \*

import threading

def handle\_client(connectionSocket, addr):

    try:

        print(f"Accepted connection from {addr[0]}:{addr[1]}")

        message =  connectionSocket.recv(1024)

        filename = message.split()[1]

        f = open(filename[1:])

        outputdata = f.read()

        #Send one HTTP header line into socket

        connectionSocket.send('HTTP/1.x 200 OK\r\n'.encode())

        connectionSocket.send('Content-Type: text/html\r\n\r\n'.encode())

        #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

        connectionSocket.send('404 Page Not Found\r\n'.encode())

        connectionSocket.send('Content-Type: text/html\r\n\r\n'.encode())

        connectionSocket.send('<html><body><h1>404 Page Not Found</h1></body></html>'.encode())

        #Close client socket

        connectionSocket.close()

serverSocket = socket(AF\_INET, SOCK\_STREAM)

#Prepare a sever socket

serverSocket.bind(('127.0.0.1', 12345))

serverSocket.listen(1)

print("Server is listening...")

while True:

    #Establish the connection

    print('Ready to serve...')

    connectionSocket, addr =  serverSocket.accept()

    thread = threading.Thread(target=handle\_client, args=(connectionSocket, addr))

    thread.start()

serverSocket.close()

Part 2

텍스트, 스크린샷, 폰트, 메뉴이(가) 표시된 사진

자동 생성된 설명

from socket import \*

import time

import numpy as np

UDP\_IP = "127.0.0.1"

UDP\_PORT = 12000

times = []

success = 0

fail = 0

#message = Ping sequence\_number time

#sequence num starts from 1 and ends at 10.

#time = when packet is sent to the server

socket = socket(AF\_INET, SOCK\_DGRAM)

socket.settimeout(1)

for sequence\_number in range(1, 11):

    start = time.time()

    socket.sendto((f'Ping {sequence\_number} {time.time()}').encode(), (UDP\_IP, UDP\_PORT))

    try:

        response, address = socket.recvfrom(1024)

        time\_taken = time.time() - start

        times.append(time\_taken)

        print(f'Sequence Number: {sequence\_number}')

        print(f'Response: {response.decode()}')

        print(f'Round-Trip Time: {time\_taken}')

    except timeout:

        print(f'Sequence Number: {sequence\_number}')

        print("\033[91mRequest time out\033[0m")

        fail += 1

loss\_rate = ((fail / 10) \* 100)

print(f'Minimum RTT: {np.min(times)}')

print(f'Maximum RTT: {np.max(times)}')

print(f'Average RTT: {np.mean(times)}')

print(f'Packet loss rate: {loss\_rate}%')

socket.close()