컴퓨터네트워크 HW#3(10점)

1. (실습과제) 멀티쓰레드 웹서버를 구현하고자 한다. 멀티쓰레드 웹 서버의 단계별 구현을 위해 다음 사항을 수행하시오(5점)

a) 파이썬에서 여러개의 일을 동시에 수행하는 방법은 쓰레드와 다중프로세스를 이용하는 방법이 있다 각 방법을 구현하는 방법을 간단히 설명하시오.

먼저 다중 쓰레드의 경우 하나의 프로세스 안에 필요에 따라 여러 개의 쓰레드를 생성하여 해당 기능을 수행한 후, 쓰레드가 종료되는 방식이다. 이는 파이썬의 threading 라이브러리를 활용하여 쓰레드 기능을 수행할 수 있다. Thread 함수를 호출하여 실행시킬 코드 타겟과 소켓을 매개변수로 입력한 후, 해당 쓰레드를 실행하는 방식으로 작동한다. 이와 달리 다중 프로세스 방식은 하나의 프로세스가 하나의 요청만 수행이 가능하다. 해당 요청이 완료된 후, 다음 요청을 수행하는 방식이다.

b) 멀티쓰레드 웹 서버를 구현하고자 한다. 기존 과제2에서 수행한 WebServer.py파일을 분리하여 쓰레드를 생성하는 파일을 ThreadGen.py라고 하고 웹 서버기능을 수행하는 파일을 HttpRequest.py로 구분하고 각 파일을 작성하시요.

ThreadGen.py

import sys

from socket import \*

from threading import Thread

from HttpRequest import httpReq

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

portNum = 12000

threadNum = 1

serverSocket.bind(("", portNum))

serverSocket.listen(1)

while True:

    print('The server is ready to receive')

    connectionSocket, addr = serverSocket.accept()

    request = Thread(target=httpReq, args=(connectionSocket, threadNum))

    print(f"\nThread {threadNum} started\n")

    request.start()

    threadNum += 1

serverSocket.close()

sys.exit()

HttpRequest.py

from socket import \*

from time import sleep

def httpReq(inputSocket, threadNum):

    try:

        message = inputSocket.recv(2048).decode()

        print(message)

        fileName = message.split()[1]

        print(fileName)

        myfile = open(fileName[1:], "rb")

        response = myfile.read()

        myfile.close()

        header = 'HTTP/1.1 200 OK\n'

        if fileName.endswith(".jpg"):

            fileFormat = "image/jpg"

        elif fileName.endswith(".gif"):

            fileFormat = "image/gif"

        elif fileName.endswith(".mp4"):

            fileFormat = "video/mp4"

        elif fileName.endswith(".wmv"):

            fileFormat = "video/wmv"

        elif fileName.endswith(".html"):

            fileFormat = "text/html"

        else:

            raise IOError

        header += f"Content-Type: {str(fileFormat)}\n\n"

        print(header)

        inputSocket.send(header.encode())

        inputSocket.send(response)

        inputSocket.close()

        for i in range(1, 5):

            sleep(1)

            print(f"thread {threadNum} is working {i}")

    except IOError:

        header = "HTTP/1.1 404 Not Found\n\n"

        response = "<html><head></head><body><h1>Error 404: File not found</h1><p>Python HTTP Server</p></body></html>".encode()

        print(header)

        inputSocket.send(header.encode())

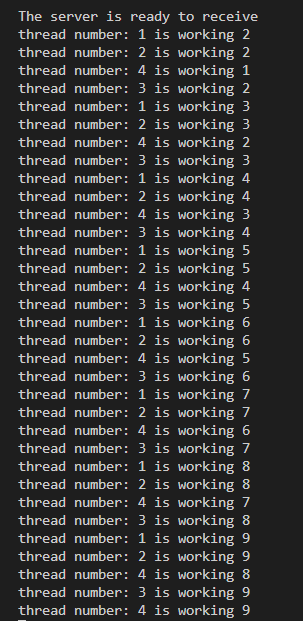
        inputSocket.send(response)

        inputSocket.close()

c) 웹 브라우져에 url값을 입력하여 동영상과 이미지를 동시에 동작시키고 확인하시오(기존 과제2에서 수행한 결과와 멀티쓰레드를 사용한 경우 비교분석 할 것).

과제 2에서 해당 기능을 수행하면 동시에 여러 개의 요청을 할 경우 한 서버당 처리 가능한 클라이언트의 수는 하나다. 그러나 멀티 쓰레드를 이용한 방식은 동시에 병렬적으로 여러 개의 작업을 수행할 수 있게 한다. 따라서 동시에 여러 개의 요청을 수행할 수 있으며, 해당 요청이 종료된 후, 해당 요청을 수행한 쓰레드는 종료된다.

d) 멀티쓰레드 웹서버에 메시지 출력을 통해 멀티쓰레드가 동작함을 보이시오.



2. (실습과제) 현재 우리대학에서 사용되는 출결시스템은 오프라인 수업시 손쉽게 출결을 할 수 있는 장점이 있다. 그러나 실제 출석을 하지 않는 학생도 교수님이 불러준 번호를 카톡으로 받아서 입력함으로써 출결이 가능하게 되는 문제점이 있다. 수업시간에 배운 서브네트워크 개념을 이용하여 내부에서 접속한 학생과 외부에서 접속한 학생을 구분하여 내부에서 접속한 학생만 출석을 인정하도록 코드를 작성하고자 한다.(5점)

a) WebServer.py 코드를 수정하여 인정되는 서브네트워크 리스트를 외부파일로부터 입력받아 출력하도록 작성하시오

accessAddress = []

with open("subnet\_list.txt") as file:

    lines = file.readlines()

    for line in lines:

        accessAddress.append(line[:-1])

file.close()

print(f"List of subnet IP: {accessAddress}")

subnet\_list.txt

127.0.0.1

127.0.0.2

127.0.0.3

127.0.0.4

b) 발신자의 IP주소를 서브네트워크와 호스트로 구분하여 출력하고 서브 네트워크주소가 인정된 서브네트워크이면 HttpReq를 진행하고 다른 서브네트워크이면 위반메시지를 출력하도록 코드를 수정하시오

import sys

from socket import \*

from threading import Thread

from HttpRequest import httpReq

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

portNum = 12000

threadNum = 1

accessAddress = []

serverSocket.bind(("", portNum))

serverSocket.listen(1)

with open("subnet\_list.txt") as file:

    lines = file.readlines()

    for line in lines:

        accessAddress.append(line[:-1])

file.close()

print(f"accessible Sub-network: {accessAddress}")

while True:

    print('The server is ready to receive')

    connectionSocket, addr = serverSocket.accept()

    print(f"source IP subnet address = {addr[0][:-2]}, host: {addr[0][-1]}")

    if not addr[0] in accessAddress:

        print(f"Access Denied, IP=({addr[0]})")

        continue

    request = Thread(target=httpReq, args=(connectionSocket, threadNum))

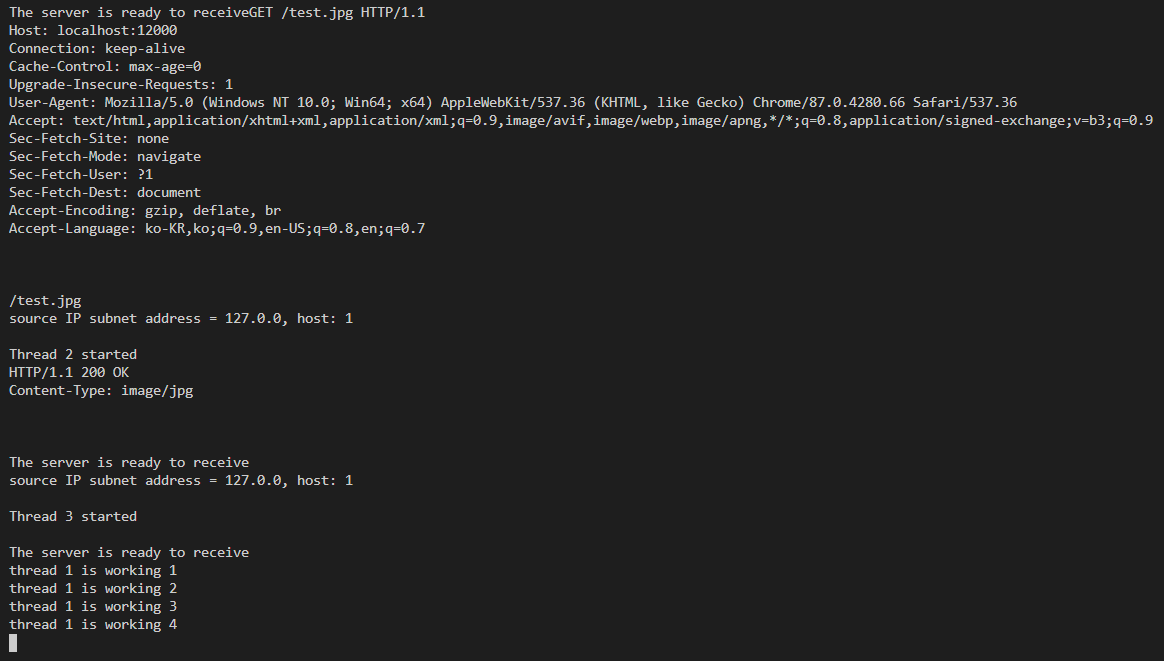
    print(f"\nThread {threadNum} started\n")

    request.start()

    threadNum += 1

serverSocket.close()

sys.exit()



c) 위 방안의 문제점 혹은 좀 더 나은 해결방안이 있으면 제시해 보시오

위 방식의 문제점은 해당 서브네트워크의 모든 리스트를 가져야 한다는 점이다. 이는 수가 점점 많아질수록 리스트 탐색에 소요되는 시간이 늘어난다. 그에 따라 서브네트워크의 리스리스 구성하는 것이 아니라 서브네트워크의 범위를 설정하여 해당 범위 내의 ip를 가진 요청만 승인하고, 나머지 요청들은 거절하는 방식을 취하면 조금 더 효율적인 방식을 사용할 수 있다.

해당 방식으로 작성한 코드

import sys

from socket import \*

from threading import Thread

from HttpRequest import httpReq

from struct import unpack

def atol(a):

    return unpack(">L", inet\_aton(a))[0]

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

portNum = 12000

threadNum = 1

portNum.bind(("", portNum))

portNum.listen(1)

while True:

    print('The server is ready to receive')

    connSocket, addr = portNum.accept()

    print(f"sender subnet address = {addr[0][:-2]}, host: {addr[0][-1]}")

    addrs = atol(addr[0])

    mask = atol("255.255.255.0")

    lo = atol("127.0.0.0")

    hi = atol("127.0.0.255")

    prefix = addrs & mask

    if (lo <= prefix <= hi):

        request = Thread(target=httpReq, args=(connSocket, threadNum))

        print(f"\nThread {threadNum} started\n")

        request.start()

        threadNum = threadNum + 1

    else:

        print(f"Access Failed, IP=({addr[0]})")

        continue

portNum.close()

sys.exit()

위 코드는 로컬 호스트 전체의 요청을 수락하는 서버이며, 로컬 호스트가 아닌 타 ip를 이용하여해당 포트에 접속을 시도 할 시 오류를 반환하는 코드이다. 필요에 따라 범위, 서브넷마스크 값을 조정하여 범위를 설정할 수 있으며, 목적을 달성 할 수 있을 것이다.

실행 결과:

