CN ASSIGNMENT - 4(Q1-Q3)

Name- Asmit Kumar Panika
Id- 2020UCP1165

1. Write an Echo_server using TCP to estimate the round trip time from client to the server. The server should be such that it can accept multiple connections at any given time , with multiplexed I/O operations.

Server:

```
from _thread import *
import socket
import sys

def clientthread(conn):

    while True:
        data = conn.recv(8192)
        buffer=str(data)
        print('from client : ',buffer[2:-1])
        ans = input('-> : ')
        conn.send(ans.encode())
    conn.close()

def main():
    try:
        host = socket.gethostname()
        port = 6667
        tot_socket = 10 # for creating 10 sockets
        list_sock = []
        for i in range(tot_socket):
            # creating tcp sockets...................
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
            s.bind((host, port+i))
            s.listen(10)
            list_sock.append(s)
            print ("[*] Server listening on %s %d" %(host, (port+i)))

        while 1:
            for j in range(len(list_sock)):
                conn, addr = list_sock[j].accept()
```

```
                print ('[*] Connected with ' + addr[0] + ':' +
str(addr[1]))

                start_new_thread(clientthread ,(conn,))

        s.close()

    except KeyboardInterrupt as msg:
        sys.exit(0)


if __name__ == "__main__":
    main()
```

We can create multiple clients but here I'm attaching codes of only 2 clients.
Client 1:

```
import socket
import time

def client_program():
    host = socket.gethostname()
    port = 6667  # socket server port number

    client_socket = socket.socket()
    t1 = time.time()
    client_socket.connect((host, port))  # connect to the server
    t2 = time.time()
    print('RTT time(in ms) = ',(t2-t1)*1000)
    message = input(" -> ")  # take input

    while message.lower().strip() != 'bye':
        client_socket.send(message.encode())
        data = client_socket.recv(1024).decode()

        print('Received from server: ' + data)

        message = input(" -> ")

    client_socket.close()  # close the connection


if __name__ == '__main__':
    client_program()
```

Client 2:

```
import socket
import time

def client_program():
    host = socket.gethostname()
    port = 6668  # socket server port number

    client_socket = socket.socket()
    t1 = time.time()
    client_socket.connect((host, port))  # connect to the server
    t2 = time.time()
    print('RTT time(in ms) = ',(t2-t1)*1000) # to calculate time
    message = input(" -> ")  # take input

    while message.lower().strip() != 'bye':
        client_socket.send(message.encode())
        data = client_socket.recv(1024).decode()

        print('Received from server: ' + data)

        message = input(" -> ")

    client_socket.close()  # close the connection


if __name__ == '__main__':
    client_program()
```

The only difference between the above two clients is the port number. One client will be connected to port number 6667 and other to port number 6668.

2.Write an Echo_Client using UDP to estimate the round trip time from client to the server. The server should be such that it can accept multiple connections at any given time, with multiplexed I/O operations.

Server:

```
import socket
import sys
import argparse

host = 'localhost'
data_payload = 2048
```

```
def echo_server(port):
    """ A simple echo server """
    # Create a UDP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Bind the socket to the port
    server_address = (host, port)
    print ("Starting up echo server on %s port %s" % server_address)

    sock.bind(server_address)

    while True:
        print ("Waiting to receive message from client")
        data, address = sock.recvfrom(data_payload)

        print ("received %s bytes from %s" % (len(data), address))
        print ("Data: %s" %data)

        if data:
            sent = sock.sendto(data, address)
            print ("sent %s bytes back to %s" % (sent, address))


if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server
Example')
    parser.add_argument('--port', action="store", dest="port",
type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_server(port)
```

Client :

```
import socket
import sys
import argparse
import time

host = 'localhost'
data_payload = 2048

def echo_client(port):
```

```
    """ A simple echo client """
    # Create a UDP socket
    sock = socket.socket(socket.AF_INET,
                         socket.SOCK_DGRAM)

    server_address = (host, port)
    print ("Connecting to %s port %s" % server_address)
    message = ('This is the message.  It will be repeated' )

    try:

        # Send data
        t1 = time.time()
        message = ("Test message. This will be echoed")
        print ("Sending %s" % message)
        sent = sock.sendto(message.encode
               ('utf-8'), server_address)

        # Receive response
        data, server = sock.recvfrom(data_payload)
        print ("received %s" % data)
        t2 = time.time()

        print("RTT time in ms = ",(t2-t1)*1000)

    finally:
        print ("Closing connection to the server")
        sock.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server
Example')
    parser.add_argument('--port', action="store", dest="port",
type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_client(port)
```

3.Program using fork( ) system call.

Server:

```
import os, socket
```

```
host="127.0.0.1"
port=7000
s=socket.socket()
s.bind((host, port))
s.listen(10)

def handle_client(s, addr, i):
    while True:
        data=s.recv(1024)
        decoded_data=data.decode("utf-8")
        if not decoded_data:
                print("\nconnection with client " + str(i) + "
broken\n")
                break
        print("  CLIENT " + str(i) + " -> " + decoded_data)

def server():
    i=1
    while i<=10:
        c, addr=s.accept()
        child_pid=os.fork()
        if child_pid==0:
                print("\nconnection successful with client " + str(i) +
str(addr) + "\n")
                handle_client(c, addr, i)
                break
        else:
                i+=1

server()
```

Client 1:

```
import socket

def client():
    host="127.0.0.1"
    port=7000
    s=socket.socket()
    s.connect((host, port))
    msg=str(input("\n -> "))
    encoded_msg=bytes(msg, "utf-8")
    while msg!='q':
```

```
            s.send(encoded_msg)
            msg=str(input("\n -> "))
            encoded_msg=bytes(msg, "utf-8")

client()
```

Client 2:

```
import socket

def client():
    host="127.0.0.1"
    port=7000
    s=socket.socket()
    s.connect((host, port))
    msg=str(input("\n -> "))
    encoded_msg=bytes(msg, "utf-8")
    while msg!='q':
            s.send(encoded_msg)
            msg=str(input("\n -> "))
            encoded_msg=bytes(msg, "utf-8")

client()
```