COMPUTER NETWORKS

PROBLEM SHEET-2

NAME : Aswajith.S **ROLL NO :** 22PT05

1.Develop an banking application based on UDP/TCP Sockets that allows the server to record deposits and withdrawals of accounts of clients connecting to the server. MyClient will have this functionality deposit method will deposit credit to the account residing in MyServer and withdraw method will withdraw credit. You need to check this in MyServer that if enough credit exists for the operation. Any other message sent to MyServer will be ignored. MyClient needs to print out everything the server sends back. . MyServer must check exactly what the client message is and resolve which request is being asked. If a proper request is found, MyServer must take the necessary action. MyServer should start with an initial credit amount of 1000 for every new client request. After each deposit, it needs to increase the credit amount and send back to MyClient the total credit amount it has. After each withdrawal, MyServer first needs to check whether enough credit exists in the account. After a successful withdrawal, it again needs to send back to MyClient the remaining total credit amount. If withdrawal is unsuccessful; then it needs to send back the message “Not enough credit!” First, make sure that server and client work correctly on your localPC. Then use the server and client on different machines.

Solution:

Client:

```
import socket
import sys

def client(HostName,Port):

        client_socket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        while True :
                message = input('Enter a message to the server [ Operation_<amount> ] : ')
                client_socket.sendto(message.encode(),(HostName,Port))
                if message[:4].lower() == 'exit':
                        break

                reply = client_socket.recv(1024).decode()
                print('The response from the server : ',reply)
```

```
if __name__ == '__main__':
	if len(sys.argv)!=1:
		client(sys.argv[1],int(sys.argv[2]))
	else:
		HostName = input('Enter the host IP : ')
		Port = int(input('Enter the Port Number : '))
	client(HostName,Port)
```

Server:

```
import socket
import threading

def handle_client(server_socket,dictionary):

	while True:
		msg,address = server_socket.recvfrom(1024)
		message = msg.decode()

		if address not in dictionary.keys():
			dictionary[address]=1000
		print(address,dictionary[address])
		amount = dictionary[address]

		if message[:4].lower() == 'exit':
			print('Client 1 Exited')
			break
		elif message[:7].lower() == 'deposit':
			amount += int(message[8:])
			server_socket.sendto(str(amount).encode(),address)
		elif message[:8].lower() == 'withdraw':
			if amount < int(message[9:]) :
				server_socket.sendto(b'Not enough credit !',address)
			else:
				amount-=int(message[9:])
				server_socket.sendto(str(amount).encode(),address)
				dictionary[address]=amount

def start(HostName,Port):

	server_socket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
	server_socket.bind((HostName,Port))
	print('Server is Active!')

	client_dict={}
```

```
        while True:
                Thread = threading.Thread(target=handle_client(server_socket,client_dict))
                Thread.start()

if __name__=='__main__':
        HostName = 'localHost'
        Port = 8000
        start(HostName,Port)
```

2. A chat room is an interface that allows two or more people to chat and send messages to everyone in the room. Set up a simple Chat Room server and allow multiple clients to connect to it using Socket Programming .The server and clients processes should run on different machines.

Client:

```
import socket
import threading
import sys

def recieve(client_socket):
        while True:
                try:
                        msg , _ =client_socket.recvfrom(1024)
                        print(msg.decode())
                        if 'left the room' in msg.decode():
                                break
                except Exception as e:
                        print(f'Error in Recieving message from the server : {e}')

if __name__ == '__main__':
        HostName = ''
        Port = 0

        if len(sys.argv)!=1:
                HostName = sys.argv[1]
                Port = int(sys.argv[2])
        else:
                HostName = input('Enter Host IP : ')
                Port = int(input("Enter Port Number : "))
        print(HostName,Port)
        client = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        name = input('Enter your user name : ')
```

```
t = threading.Thread(target=recieve,args=(client,))
t.start()

while True:
    message = input('\n\n')
    if message.lower() == 'exit' or message.lower() == 'quit':
        client.sendto(f'{name}+Exit'.encode(),(HostName,Port))
        break
    else:
        client.sendto((name +'+'+ message).encode(),(HostName,Port))
```

Server:

```
import socket
import threading
import queue

messages = {}
names = {}
send_queue = queue.Queue()

def recieve_msg(server_socket):
    while True:
        try:
            data , address = server_socket.recvfrom(1024)
            name , msg = data.decode().split('+')
            print(msg,name,msg=='Exit',msg.lower())
            if msg == 'Exit':
                print(f'{name} left the room')
                server_socket.sendto(f'{name} left the room'.encode(),address)
                del messages[address]
                del names[address]
                break
            send_queue.put(address)
            if address not in messages.keys():
                names[address]=name
                print(f'{name} is connected ')
                print(f'{name} : {msg}')
            else:
                print(f'{names[address]} : {msg}')
                messages[address] = msg
        except Exception as e:
            print(f'Error in Recieving Message : {e}')
```

```
def send_msg(server_socket):

        try:
                while True:
                        while not send_queue.empty():
                                current = send_queue.get()
                                print(current)
                                for i in messages.keys():
                                        print(f'{names[current]} : {messages[current]}')
                                        if i != current:
                                                server_socket.sendto(f'{names[current]} : {messages[current]}'.encode(),i)
                                        else:
                                                server_socket.sendto(f'You : {messages[current]}'.encode(),current)
        except Exception as e:
                print(f'Error in sending Message : {e}')


def start(HostName:str,Port:int):
        server = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        server.bind((HostName,Port))
        print('Server is Active')

        recieveThread = threading.Thread(target=recieve_msg,args=(server,))
        sendThread = threading.Thread(target=send_msg,args=(server,))

        recieveThread.start()
        sendThread.start()

if __name__ == '__main__':
        HostName = 'localhost'
        Port = 8001
        start(HostName=HostName,Port=Port)
```

3. Imagine a Client-Server architecture (As shown in Figure), where user stores the file on a server. The main server splits that file into two or more fragments and store each fragment on separate storage server. When client retrieve the file from the main server, the main server again retrieves the file in fragments from storage servers and present it as a one file to user.

Client:

```
import os
import sys
import socket

def client(HostName:str,Port:int):

        client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        client_socket.connect((HostName,Port))

        while True:
                print('\nFile Storage Server ')
                print('\n\t1.Store a file\n\t2.Retrieve a file from storage\n\t3.Exit\n')

                choice = input('Enter your choice : ')
                if choice == '1':
                        file_name = input('Enter the file name to be stored [with extension]: ')
                        try:
                                file_name.split('.')
                        except:
                                print('Failed to recogonize file')
                                continue
                        file_size = os.path.getsize(file_name)

                        try:
                                with open(file_name,'rb') as file:
                                        data = file.read()
                                        client_socket.send(('1 __ '+file_name+' __ '+str(file_size)+' __ ').encode()+data)
                                        # client_socket.send('<TMNTR>'.encode())

                                        print(f'\nResult :  \n\t Name : {file_name}\n\t File size : {file_size}\n\t Operation :
                        Transfer\n\t Status : Success')
                        except Exception as e:
                                print(f'\nResult :  \n\n Name : {file_name}\n\t File size : {file_size}\n\t Operation : Transfer\n\t
                Status : Failed\n\t Error : {e}')

                elif choice == '2':

                        file_name = input('Enter the file name to be retrived [with extension]: ')
                        try:
                                file_name.split('.')
                        except Exception as e:
                                print(f'Failed to recogonize file {e}')
                                continue
```

```
                                file_size = os.path.getsize(file_name)
                                client_socket.send(('2 __ '+file_name+' __ '+str(file_size)+' __ NULL').encode())

                                data = client_socket.recv(1024)
                                print(data)
                                try:
                                        with open('Retrived'+file_name,'wb') as file:
                                                file.write(data)
                                                print(f'\nResult :  \n\t Name : {file_name}\n\t File size : {file_size}\n\t Operation :
                                        Retrival\n\t Status : Success')
                                except Exception as e:
                                        print(f'\nResult :  \n\n Name : {file_name}\n\t File size : {file_size}\n\t Operation : Retrival\n\t
                                Status : Failed\n\t Error : {e}')

                        elif choice =='3':
                                client_socket.send(('3 __ '+'NONE.txt'+' __ '+'123'+' __ NULL').encode())
                                break
                        else:
                                print('~~~Invalid choice Please Enter a valid choice !~~~')
                client_socket.close()

if __name__ == '__main__':

                HostName = ''
                Port = 0
                if len(sys.argv)!=1:
                        HostName = sys.argv[1]
                        Port = int(sys.argv[2])
                else:
                        HostName = input('Enter Host IP : ')
                        Port = int(input('Enter Port Number : '))

                client(HostName=HostName,Port=Port)
```

Server:

```
import socket
import threading
import os

def Broadcast_Storage(name,extension,message,is_first=True):
                try:
                        storage = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                        if is_first:
                                storage.connect((Storage_Host1,Storage_Port1))
```

```
                else:
                        storage.connect((Storage_Host2,Storage_Port2))
                        storage.send(b'1 __ '+name.encode()+b' __ '+extension.encode()+b' __ '+message.encode())
                        storage.close()
        except Exception as e:
                print(e)

def Retrive(name,extension,is_first=True):
        try:
                message = 'NULL'
                server = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                if is_first:
                        server.connect((Storage_Host1,Storage_Port1))
                else:
                        server.connect((Storage_Host2,Storage_Port2))
                server.send(b'2 __ '+name.encode()+b' __ '+extension.encode()+b' __ '+message.encode())
                data = server.recv(1024)
                server.close()
                return data
        except Exception as e:
                print(e)

def accept_connections(server_socket):
        while True:
                try :
                        client,address = server_socket.accept()
                        while True:
                                try:
                                        choice,file,size,data = client.recv(1024).decode().split(' __ ')
                                        file_name , file_extension = file.split('.')
                                        file_size = int(size)

                                        print(choice,file,size)

                                        if choice == '1':
                                                file1_bytes = b''
                                                file2_bytes = b''

                                                file1_bytes = data[:int(file_size/2)]
                                                file2_bytes = data[int(file_size/2):]

                                                #print('\n\n',file1_bytes,'\n\n',file2_bytes)

                                                Broadcast_Storage(file_name,file_extension,file1_bytes,True)
```

```
                                        Broadcast_Storage(file_name,file_extension,file2_bytes,False)

                                elif choice == '2':
                                        data1 = Retrive(file_name,file_extension,True)
                                        data2 = Retrive(file_name,file_extension,False)

                                        client.send(data1+data2)
                                else:
                                        client.close()
                                        break
                        except ValueError as v:
                                print(v)

                except Exception as e:
                        print(f'Error in Recieving file : {e}')

if __name__=='__main__':
        HostName = 'localhost'
        Port = 8000

        server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        server_socket.bind((HostName,Port))
        server_socket.listen(5)

        Storage_Host1 = 'localhost'
        Storage_Host2 = 'localhost'

        Storage_Port2 = 7000
        Storage_Port1 = 9000

        print('Server is Active Ready to accept connections!')

        accept_connections(server_socket=server_socket)
        # thread = threading.Thread(target=accept_connections,args=(server_socket,))
        # thread.start()
        server_socket.close()
```

Storage Server:

```
import socket
import threading

def connections(server_socket):
        while True:
```

```
        client , address = server_socket.accept()
        choice,name,extension,data = client.recv(1024).decode().split(' __ ')
        if choice =='1':
                try:
                        with open(name+'A.'+extension,'wb') as file:
                                file.write(data.encode())
                        print(f'File Name : {name}\nStore Operation : Success')

                except Exception as e:
                        print(f'Error in receiving content [Storage 1] : {e}')
        elif choice == '2':
        try:
                data = b''
                with open(name+'A.'+extension,'rb') as file:
                data += file.read()

                client.send(data)
                print(f'File Name : {name}\nRetrival Operation : Success')
        except Exception as e:
                print(f'Error in sending content [Storage 1] : {e}')
        client.close()

if __name__=='__main__':

        HOSTNAME = 'localhost'
        PORT = 9000

        server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        server_socket.bind((HOSTNAME,PORT))
        server_socket.listen(10)
        print('Storage Server 1 is Active, ready to accept connections')

        connections(server_socket=server_socket)
```

Storage Server:

```
import socket
import threading

def connections(server_socket):
        while True:
        client , address = server_socket.accept()
        choice,name,extension,data = client.recv(1024).decode().split(' __ ')
        if choice =='1':
```

```
                try:
                        with open(name+'B.'+extension,'wb') as file:
                                file.write(data.encode())
                        print(f'File Name : {name}\nStore Operation : Success')

                except Exception as e:
                        print(f'Error in receiving content [Storage 1] : {e}')
        elif choice == '2':
        try:
                data = b''
                with open(name+'B.'+extension,'rb') as file:
                data += file.read()

                client.send(data)
                print(f'File Name : {name}\nRetrival Operation : Success')
        except Exception as e:
                print(f'Error in sending content [Storage 1] : {e}')
        client.close()

if __name__=='__main__':

        HOSTNAME = 'localhost'
        PORT = 9000

        server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        server_socket.bind((HOSTNAME,PORT))
        server_socket.listen(10)
        print('Storage Server 1 is Active, ready to accept connections')

        connections(server_socket=server_socket)
```

4. Develop a distributed tic-tac-toe game using multi threads and networking with socket streams. A distributed tic-tac-toe game enables users to play on different machines from anywhere .You need to develop a server for multiple clients. The server creates a server socket and accepts connections from every two players to form a session. Each session is a thread that communicates with the two players and determines the status of the game. The server can establish any number of sessions. For each session, the first client connecting to the server is identified as player 1 with token X, and the second client connecting is identified as player 2 with token O. The server notifies the players of their respective tokens. Once two clients are connected to it, the server starts a thread to facilitate the game between the two players by performing the steps repeatedly,

Solution:

Client:

```
import socket
import threading

client=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
client.connect(('127.0.0.1',9999))

while True:
  message=client.recv(1024).decode()
  print(message)
  if "Your turn" in message:
    position=input()
    client.send(position.encode())
  elif 'Congratulations!' in message or 'lose' in message:
    break
```

Server:

```
import socket
import threading

def handle_session(player1, player2):
  global board
  print("correct")
  player1['socket'].send("You are player1 with token 'X'".encode())
  player2['socket'].send("You are player2 with token 'O'".encode())
  try:
    while True:
      send_board(player1, player2)
      handle_player_turn(player1)
      if check_winner(player1['token']):
        send_board(player1, player2)
        notify_winner(player1,player2)
        break

      send_board(player1, player2)
      handle_player_turn(player2)
      if check_winner(player2['token']):
        send_board(player1, player2)
        notify_winner(player2,player1)
        break
  except Exception as e:
    print(f"Error in handle_session: {e}")
```

```
def handle_player_turn(player):
    player['socket'].send("Your turn. Enter position (1-9): ".encode())
    position = int(player['socket'].recv(1024).decode()) - 1

    if is_valid_move(position):
        board[position] = player['token']
    else:
        player['socket'].send("Invalid move. Try again.\n".encode())
        handle_player_turn(player)

def send_board(player1, player2):
    board_str = format_board()
    player1['socket'].send(board_str.encode())
    player2['socket'].send(board_str.encode())

def format_board():
    return f" {board[0]} | {board[1]} | {board[2]}\n" \
           f"-----------\n" \
           f" {board[3]} | {board[4]} | {board[5]}\n" \
           f"-----------\n" \
           f" {board[6]} | {board[7]} | {board[8]}\n"

def is_valid_move(position):
    return 0 <= position <= 8 and board[position] == ''

def check_winner(token):
    winning_combinations = [(0,1,2),(3,4,5),(6,7,8),
                            (0,3,6),(1,4,7),(2,5,8),
                            (0,4,8),(2,4,6)]
    for combo in winning_combinations:
        if all(board[i] == token for i in combo):
            return True
    return False

def notify_winner(winner, loser):
    winner['socket'].send("Congratulations! You win.".encode())
    loser['socket'].send("You lose. Better luck next time.".encode())
    reset_board()

def reset_board():
    global board
    board = [''] * 9

board = [''] * 9
```

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('127.0.0.1', 9999))
server_socket.listen(8)

players = []
while len(players) < 2:

  client, addr = server_socket.accept()
  print(client)
  print(f"Connection established with {addr}")
  players.append({'socket': client, 'token': 'X' if len(players) == 1 else 'O'})

  if len(players) == 2:

    print(players[0])
    print(players[1])
    threading.Thread(target=handle_session, args=(players[0], players[1])).start()
```