DS Lab Week #6

Angad Sandhu
190905494
1/04/2022

solved questions

1)

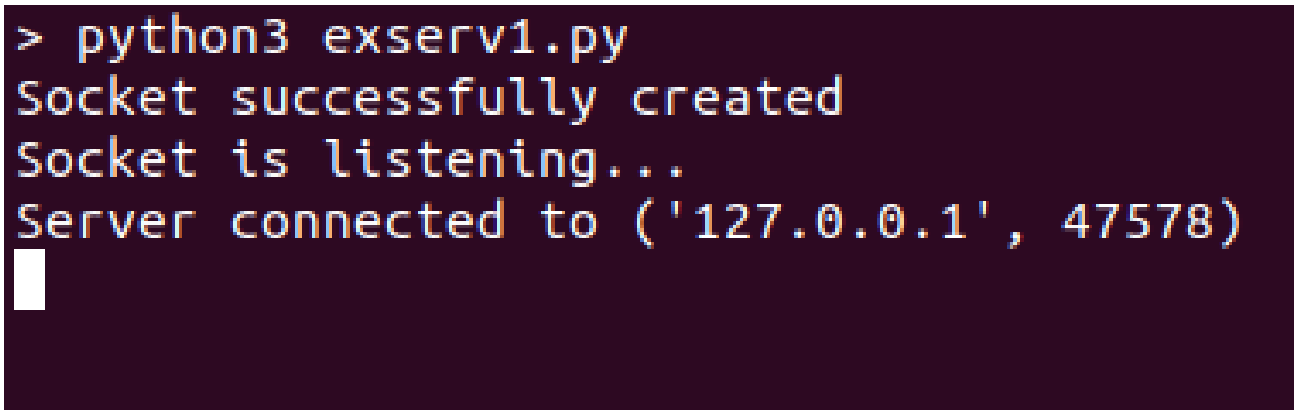
#server.py

```
import socket
import datetime
import time

# function used to initiate the Clock Server
def initiateClockServer():
	s = socket.socket()
	print("Socket successfully created")
	# Server port
	port = 8011
	s.bind(('', port))
	# Start listening to requests
	s.listen(5)
	print("Socket is listening...")
	# Clock Server Running forever
	while True:
	# Establish connection with client
		connection, address = s.accept()
		print('Server connected to', address)
	# Respond the client with server clock time
		connection.send(str(datetime.datetime.now()).encode())
	# Close the connection with the client process
	connection.close()

# Driver function
if __name__ == '__main__':
	# Trigger the Clock Server
	initiateClockServer()
```

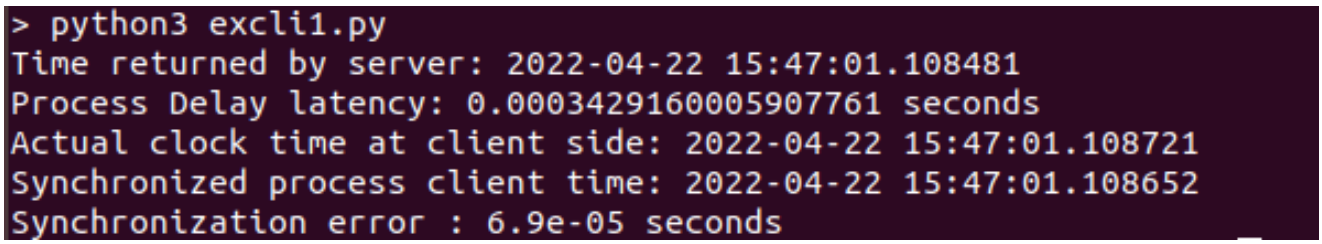
#client.py

```
import socket
import datetime
from dateutil import parser
from timeit import default_timer as timer

# function used to Synchronize client process time
def synchronizeTime():
    s = socket.socket()
    # Server port
    port = 8011
    # connect to the clock server on local computer
    s.connect(('127.0.0.1', port))
    request_time = timer()
    # receive data from the server
    server_time = parser.parse(s.recv(1024).decode())
    response_time = timer()
    actual_time = datetime.datetime.now()
    print("Time returned by server: " + str(server_time))
    process_delay_latency = response_time - request_time
    print("Process Delay latency: " + str(process_delay_latency) + " seconds")
    print("Actual clock time at client side: " + str(actual_time))
    # synchronize process client clock time
    client_time = server_time + datetime.timedelta(seconds =
(process_delay_latency) / 2)
    print("Synchronized process client time: " + str(client_time))
    # calculate synchronization error
    error = actual_time - client_time
    print("Synchronization error : " + str(error.total_seconds()) + " seconds")
    s.close()

# Driver function
if __name__ == '__main__':
    # synchronize time using clock server
    synchronizeTime()
```

Output:
Server



Client



2)

#server.py

```
from functools import reduce
from dateutil import parser
import threading
import datetime
import socket
import time


# datastructure used to store client address and clock data
client_data = {}

''' nested thread function used to receive
    clock time from a connected client '''
def startRecieveingClockTime(connector, address):

    while True:
        # recieve clock time
        clock_time_string = connector.recv(1024).decode()
        clock_time = parser.parse(clock_time_string)
        clock_time_diff = datetime.datetime.now() - \
```

```
                                    clock_time

        client_data[address] = {
                    "clock_time"      : clock_time,
                    "time_difference" : clock_time_diff,
                    "connector"       : connector
                    }

        print("Client Data updated with: "+ str(address),
                                    end = "\n\n")
        time.sleep(5)


''' master thread function used to open portal for
    accepting clients over given port '''
def startConnecting(master_server):

    # fetch clock time at slaves / clients
    while True:
        # accepting a client / slave clock client
        master_slave_connector, addr = master_server.accept()
        slave_address = str(addr[0]) + ":" + str(addr[1])

        print(slave_address + " got connected successfully")

        current_thread = threading.Thread(
                    target = startRecieveingClockTime,
                    args = (master_slave_connector,
                                slave_address, ))
        current_thread.start()


# subroutine function used to fetch average clock difference
def getAverageClockDiff():

    current_client_data = client_data.copy()

    time_difference_list = list(client['time_difference']
                            for client_addr, client
                               in client_data.items())


    sum_of_clock_difference = sum(time_difference_list, \
                               datetime.timedelta(0, 0))
```

```
    average_clock_difference = sum_of_clock_difference \
                                / len(client_data)

    return  average_clock_difference

def synchronizeAllClocks():

    while True:

        print("New synchroniztion cycle started.")
        print("Number of clients to be synchronized: " + \
                                str(len(client_data)))

        if len(client_data) > 0:

            average_clock_difference = getAverageClockDiff()

            for client_addr, client in client_data.items():
                try:
                    synchronized_time = \
                        datetime.datetime.now() + \
                                average_clock_difference

                    client['connector'].send(str(
                            synchronized_time).encode())

                except Exception as e:
                    print("Something went wrong while " + \
                         "sending synchronized time " + \
                         "through " + str(client_addr))

        else :
            print("No client data." + \
                    " Synchronization not applicable.")

        print("\n\n")

        time.sleep(5)


# function used to initiate the Clock Server / Master Node
def initiateClockServer(port = 8080):

    master_server = socket.socket()
    master_server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

```
    print("Socket at master node created successfully\n")

    master_server.bind(('', port))

    # Start listening to requests
    master_server.listen(10)
    print("Clock server started...\n")

    # start making connections
    print("Starting to make connections...\n")
    master_thread = threading.Thread(
                    target = startConnecting,
                    args = (master_server, ))
    master_thread.start()

    # start synchroniztion
    print("Starting synchronization parallely...\n")
    sync_thread = threading.Thread(
                      target = synchronizeAllClocks,
                      args = ())
    sync_thread.start()



# Driver function
if __name__ == '__main__':

    # Trigger the Clock Server
    initiateClockServer(port = 8080)
```

#client.py

```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time

# client thread function used to send time at client side
def startSendingTime(slave_client):

    while True:
        # provide server with clock time at the client
```

```
        slave_client.send(str(
                    datetime.datetime.now()).encode())

        print("Recent time sent successfully",
                                    end = "\n\n")
        time.sleep(5)


# client thread function used to receive synchronized time
def startReceivingTime(slave_client):

    while True:
        # receive data from the server
        Synchronized_time = parser.parse(
                      slave_client.recv(1024).decode())

        print("Synchronized time at the client is: " + \
                              str(Synchronized_time),
                              end = "\n\n")


# function used to Synchronize client process time
def initiateSlaveClient(port = 8080):

    slave_client = socket.socket()

    # connect to the clock server on local computer
    slave_client.connect(('127.0.0.1', port))

    # start sending time to server
    print("Starting to receive time from server\n")
    send_time_thread = threading.Thread(
                    target = startSendingTime,
                    args = (slave_client, ))
    send_time_thread.start()


    # start recieving synchronized from server
    print("Starting to recieving " + \
                      "synchronized time from server\n")
    receive_time_thread = threading.Thread(
                     target = startReceivingTime,
                     args = (slave_client, ))
    receive_time_thread.start()
```

```
# Driver function
if __name__ == '__main__':

    # initialize the Slave / Client
    initiateSlaveClient(port = 8080)
```

Output:

Server

```
> python3 exserv2.py
Socket at master node created successfully

Clock server started...

Starting to make connections...

Starting synchronization parallely...

New synchroniztion cycle started.
Number of clients to be synchronized: 0
No client data. Synchronization not applicable.



127.0.0.1:49240 got connected successfully
Client Data updated with: 127.0.0.1:49240

New synchroniztion cycle started.
Number of clients to be synchronized: 1



Client Data updated with: 127.0.0.1:49240

New synchroniztion cycle started.
Number of clients to be synchronized: 1



Client Data updated with: 127.0.0.1:49240

New synchroniztion cycle started.
Number of clients to be synchronized: 1



Client Data updated with: 127.0.0.1:49240

New synchroniztion cycle started.
Number of clients to be synchronized: 1
```

Client

```
> python3 excli2.py
Starting to receive time from server

Starting to recieving synchronized time from server

Recent time sent successfully

Synchronized time at the client is: 2022-04-22 15:48:31.436069

Recent time sent successfully

Synchronized time at the client is: 2022-04-22 15:48:36.441700

Recent time sent successfully

Synchronized time at the client is: 2022-04-22 15:48:41.447294

Recent time sent successfully

Synchronized time at the client is: 2022-04-22 15:48:46.453031

Recent time sent successfully

Synchronized time at the client is: 2022-04-22 15:48:51.458577

Recent time sent successfully

Synchronized time at the client is: 2022-04-22 15:48:56.463472

Recent time sent successfully

Synchronized time at the client is: 2022-04-22 15:49:01.466466

Recent time sent successfully

Synchronized time at the client is: 2022-04-22 15:49:06.470042

Recent time sent successfully

Synchronized time at the client is: 2022-04-22 15:49:11.473888

Recent time sent successfully
```

exersize questions

1)

#server.py

```
from functools import reduce
from dateutil import parser
import threading
import datetime
```

```
import socket
import time
client_data = {}
def startRecieveingClockTime(connector, address):
    while True:
        clock_time_string = connector.recv(1024).decode()
        clock_time = parser.parse(clock_time_string)
        clock_time_diff = datetime.datetime.now() - clock_time
        client_data[address] = {
            "clock_time": clock_time,
            "time_difference": clock_time_diff,
            "connector": connector
        }
        print("Client Data updated with: " + str(address), end="\n\n")
        time.sleep(5)

def startConnecting(master_server):
    while True:
        master_slave_connector, addr = master_server.accept()
        slave_address = str(addr[0]) + ":" + str(addr[1])
        print(slave_address + " got connected successfully")
        current_thread = threading.Thread(
            target=startRecieveingClockTime,
            args=(master_slave_connector, slave_address, ))
        current_thread.start()

def getAverageClockDiff():
    current_client_data = client_data.copy()
    time_difference_list = list(client['time_difference']
                                for client_addr, client in client_data.items())
    sum_of_clock_difference = sum(
    time_difference_list, datetime.timedelta(0, 0))
    average_clock_difference = sum_of_clock_difference / len(client_data)
    return average_clock_difference

def synchronizeAllClocks():
    while True:
        print("New synchroniztion cycle started.")
        print("Number of clients to be synchronized: " + str(len(client_data)))
        if len(client_data) > 0:
            average_clock_difference = getAverageClockDiff()
            for client_addr, client in client_data.items():
                try:
                    synchronized_time = datetime.datetime.now() +
average_clock_difference
```

```
                client['connector'].send(str(synchronized_time).encode())
            except Exception as e:
                print("Something went wrong while sending synchronized time through"
+ str(client_addr))
        else:
            print("No client data. Synchronization not applicable.")
        print("\n\n")
        time.sleep(5)

def initiateClockServer(port=8059):
    master_server = socket.socket()
    master_server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    print("The Manipal Foodie\n")
    master_server.bind(('', port))
    master_server.listen(10)
    print("Clock server print\n")
    print("Connecitng to production lines...\n")
    master_thread = threading.Thread(
        target=startConnecting,
        args=(master_server, ))
    master_thread.start()
    print("Starting synchronization parallely...\n")
    sync_thread = threading.Thread(
        target=synchronizeAllClocks,
        args=())
    sync_thread.start()

if __name__ == '__main__':
    initiateClockServer(port=8059)
```

#client1.py

```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time
def startSendingTime(slave_client):
    while True:
        slave_client.send(str(datetime.datetime.now()).encode())
        print("KMC time sent successfully", end="\n\n")
        time.sleep(5)
```

```
def startReceivingTime(slave_client):
    while True:
        Synchronized_time = parser.parse(slave_client.recv(1024).decode())
        print("Synchronized time at the client is: " +
        str(Synchronized_time), end="\n\n")

def initiateSlaveClient(port=8059):
    slave_client = socket.socket()
    slave_client.connect(('127.0.0.1', port))
    print("Starting to receive time from server\n")
    send_time_thread = threading.Thread(
        target=startSendingTime,
        args=(slave_client, ))
    send_time_thread.start()
    print("Starting to recieving synchronized time from server\n")
    receive_time_thread = threading.Thread(
        target=startReceivingTime,
        args=(slave_client, ))
    receive_time_thread.start()

if __name__ == '__main__':
    initiateSlaveClient(port=8059)
```

#client2.py

```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time

def startSendingTime(slave_client):
    while True:
        slave_client.send(str(datetime.datetime.now()).encode())
        print("MIT time sent successfully", end="\n\n")
        time.sleep(5)

def startReceivingTime(slave_client):
    while True:
        Synchronized_time = parser.parse(slave_client.recv(1024).decode())
        print("Synchronized time at the client is: " +
        str(Synchronized_time), end="\n\n")
```

```
def initiateSlaveClient(port=8059):
    slave_client = socket.socket()
    slave_client.connect(('127.0.0.1', port))
    print("Starting to receive time from server\n")
    send_time_thread = threading.Thread(
        target=startSendingTime,
        args=(slave_client, ))
    send_time_thread.start()
    print("Starting to recieving synchronized time from server\n")
    receive_time_thread = threading.Thread(
        target=startReceivingTime,
        args=(slave_client, ))
    receive_time_thread.start()

if __name__ == '__main__':
    initiateSlaveClient(port=8059)
```

#client3.py

```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time

def startSendingTime(slave_client):
    while True:
        slave_client.send(str(datetime.datetime.now()).encode())
        print("TAPMI time sent successfully", end="\n\n")
        time.sleep(5)

def startReceivingTime(slave_client):
    while True:
        Synchronized_time = parser.parse(slave_client.recv(1024).decode())
        print("Synchronized time at the client is: " +
        str(Synchronized_time), end="\n\n")

def initiateSlaveClient(port=8059):
    slave_client = socket.socket()
    slave_client.connect(('127.0.0.1', port))
    print("Starting to receive time from server\n")
    send_time_thread = threading.Thread(
        target=startSendingTime,
```

```
        args=(slave_client, ))
    send_time_thread.start()
    print("Starting to recieving synchronized time from server\n")
    receive_time_thread = threading.Thread(
        target=startReceivingTime,
        args=(slave_client, ))
    receive_time_thread.start()

if __name__ == '__main__':
    initiateSlaveClient(port=8059)
```

#client4.py

```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time

def startSendingTime(slave_client):
    while True:
        slave_client.send(str(datetime.datetime.now()).encode())
        print("SOLS time sent successfully", end="\n\n")
        time.sleep(5)

def startReceivingTime(slave_client):
    while True:
        Synchronized_time = parser.parse(slave_client.recv(1024).decode())
        print("Synchronized time at the client is: " + str(Synchronized_time), end="\n\
n")
def initiateSlaveClient(port=8059):
    slave_client = socket.socket()
    slave_client.connect(('127.0.0.1', port))
    print("Starting to receive time from server\n")
    send_time_thread = threading.Thread(
        target=startSendingTime,
        args=(slave_client, ))
    send_time_thread.start()
    print("Starting to recieving synchronized time from server\n")
    receive_time_thread = threading.Thread(
        target=startReceivingTime,
        args=(slave_client, ))
    receive_time_thread.start()
```

```
if __name__ == '__main__':
    initiateSlaveClient(port=8059)
```

Output:

Server:

```
> python3 q1server.py
The Manipal Foodie

Clock server print

Connecitng to production lines...

Starting synchronization parallely...

New synchroniztion cycle started.
Number of clients to be synchronized: 0
No client data. Synchronization not applicable.



New synchroniztion cycle started.
Number of clients to be synchronized: 0
No client data. Synchronization not applicable.



New synchroniztion cycle started.
Number of clients to be synchronized: 0
No client data. Synchronization not applicable.



New synchroniztion cycle started.
Number of clients to be synchronized: 0
No client data. Synchronization not applicable.



New synchroniztion cycle started.
Number of clients to be synchronized: 0
No client data. Synchronization not applicable.



New synchroniztion cycle started.
Number of clients to be synchronized: 0
No client data. Synchronization not applicable.
```

```
New synchroniztion cycle started.
Number of clients to be synchronized: 0
No client data. Synchronization not applicable.



127.0.0.1:55620 got connected successfully
Client Data updated with: 127.0.0.1:55620

127.0.0.1:55622 got connected successfully
Client Data updated with: 127.0.0.1:55622

New synchroniztion cycle started.
Number of clients to be synchronized: 2



Client Data updated with: 127.0.0.1:55620

Client Data updated with: 127.0.0.1:55622

New synchroniztion cycle started.
Number of clients to be synchronized: 2



Client Data updated with: 127.0.0.1:55620

Client Data updated with: 127.0.0.1:55622

127.0.0.1:55624 got connected successfully
Client Data updated with: 127.0.0.1:55624

New synchroniztion cycle started.
Number of clients to be synchronized: 3



127.0.0.1:55626 got connected successfully
Client Data updated with: 127.0.0.1:55626

Client Data updated with: 127.0.0.1:55620

Client Data updated with: 127.0.0.1:55622

Client Data updated with: 127.0.0.1:55624
```

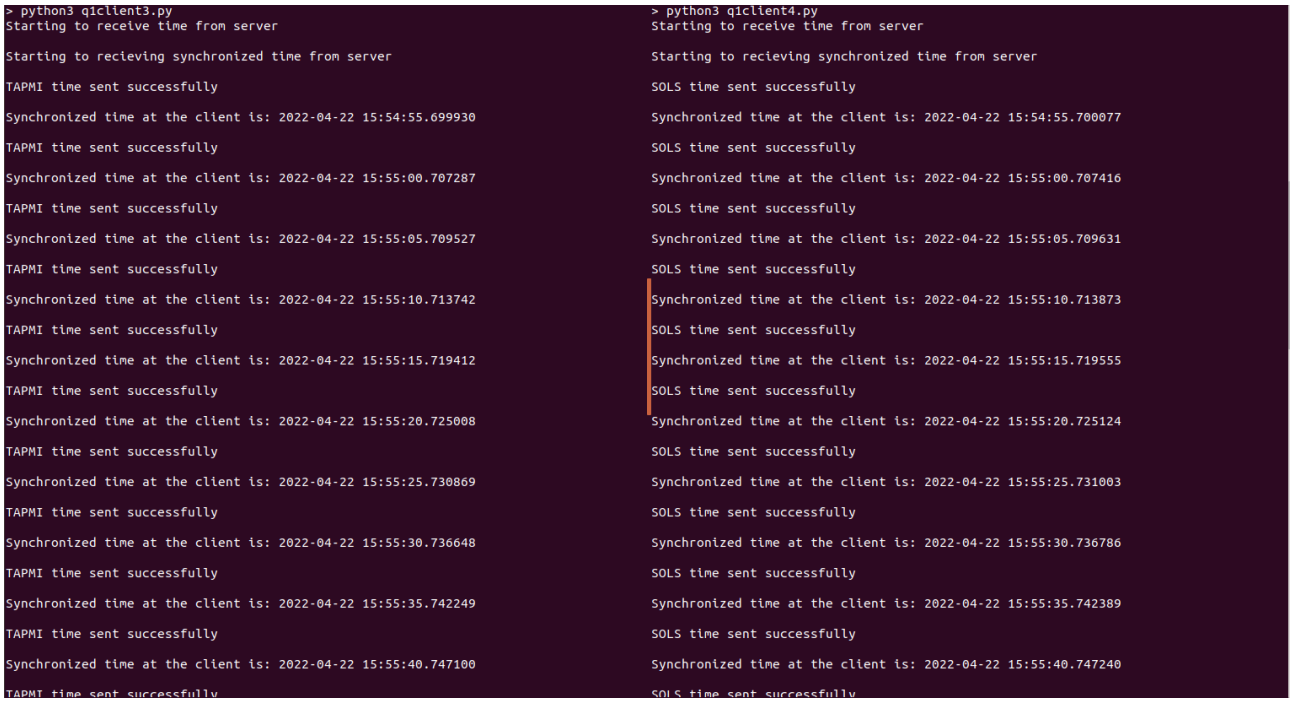
```
Client Data updated with: 127.0.0.1:55620

Client Data updated with: 127.0.0.1:55622

Client Data updated with: 127.0.0.1:55624

New synchroniztion cycle started.
Number of clients to be synchronized: 4



Client Data updated with: 127.0.0.1:55626

Client Data updated with: 127.0.0.1:55620

Client Data updated with: 127.0.0.1:55622

Client Data updated with: 127.0.0.1:55624

New synchroniztion cycle started.
Number of clients to be synchronized: 4



Client Data updated with: 127.0.0.1:55626

Client Data updated with: 127.0.0.1:55620

Client Data updated with: 127.0.0.1:55622

Client Data updated with: 127.0.0.1:55624

New synchroniztion cycle started.
Number of clients to be synchronized: 4
```

Client 1 & Client 2

```
> python3 q1client1.py
Starting to receive time from server

Starting to recieving synchronized time from server

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:05.709684

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:10.713925

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:15.719615

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:20.725178

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:25.731056

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:30.736844

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:35.742448

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:40.747299

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:45.752134

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:50.755897
```

```
> python3 q1client2.py
Starting to receive time from server

MIT time sent successfully

Starting to recieving synchronized time from server

Synchronized time at the client is: 2022-04-22 15:55:10.713974

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:15.719666

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:20.725227

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:25.731105

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:30.736892

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:35.742500

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:40.747353

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:45.752185

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:50.755950

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:55.762244
```

Client 3 & Client 4



```
> python3 q1client3.py
Starting to receive time from server

Starting to recieving synchronized time from server

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:54:55.699930

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:00.707287

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:05.709527

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:10.713742

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:15.719412

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:20.725008

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:25.730869

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:30.736648

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:35.742249

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:40.747100

TAPMI time sent successfully
```

```
> python3 q1client4.py
Starting to receive time from server

Starting to recieving synchronized time from server

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:54:55.700077

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:00.707416

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:05.709631

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:10.713873

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:15.719555

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:20.725124

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:25.731003

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:30.736786

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:35.742389

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 15:55:40.747240

SOLS time sent successfully
```

2)

#server.py

```
import socket
import datetime
import time

def initiateClockServer():
    s = socket.socket()
    print("Manipal Buddy Banking")
    port = 8011
    s.bind(('', port))
    s.listen(5)
    print("Waiting for client...")
    while True:
        connection, address = s.accept()
        print('Server connected to', address)
        connection.send(str(datetime.datetime.now()).encode())
        connection.close()

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

#client1.py

```
import socket
import datetime
import time
from dateutil import parser
from timeit import default_timer as timer

def synchronizeTime():
    print("MOBILE APP\n")
    s = socket.socket()
    port = 8011
    s.connect(('127.0.0.1', port))
    request_time = timer()
    server_time = parser.parse(s.recv(1024).decode())
    response_time = timer()
    actual_time = datetime.datetime.now()
    print("Time returned by server: " + str(server_time))
    process_delay_latency = response_time - request_time
    print("Process Delay latency: " + str(process_delay_latency) + " seconds")
    print("Actual clock time at client side: " + str(actual_time))
    client_time = server_time + \
        datetime.timedelta(seconds=(process_delay_latency) / 2)
    print("Synchronized process client time: " + str(client_time))
    time.sleep(10)
    s.close()

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

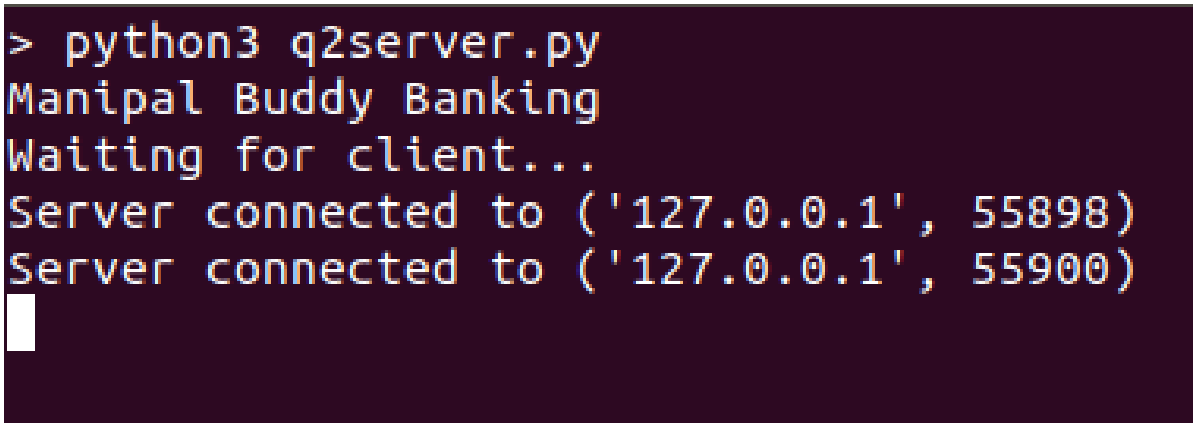
#client2.py

```
import socket
import datetime
import time
from dateutil import parser
from timeit import default_timer as timer

def synchronizeTime():
    print("WEB BROWSER\n")
    s = socket.socket()
    port = 8011
    s.connect(('127.0.0.1', port))
    request_time = timer()
    server_time = parser.parse(s.recv(1024).decode())
```

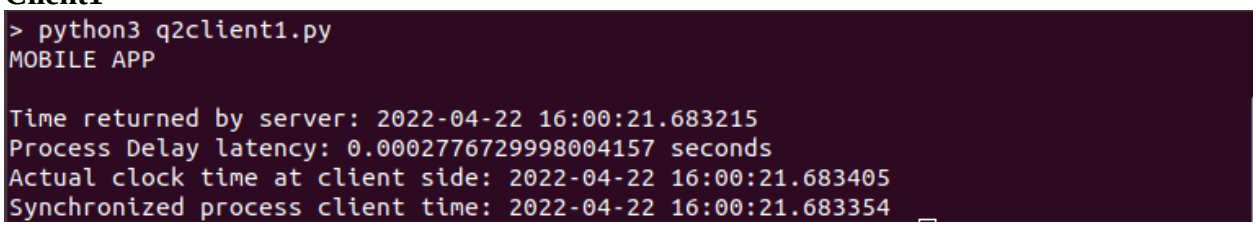
```
    response_time = timer()
    actual_time = datetime.datetime.now()
    print("Time returned by server: " + str(server_time))
    process_delay_latency = response_time - request_time
    print("Process Delay latency: " + str(process_delay_latency) + " seconds")
    print("Actual clock time at client side: " + str(actual_time))
    client_time = server_time + datetime.timedelta(seconds =
(process_delay_latency) /2)
    print("Synchronized process client time: " + str(client_time))
    time.sleep(10)
    s.close()

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

Output:

Server



Client1



Client2