LAB – 7
CLOCK SYNCHRONIZATION

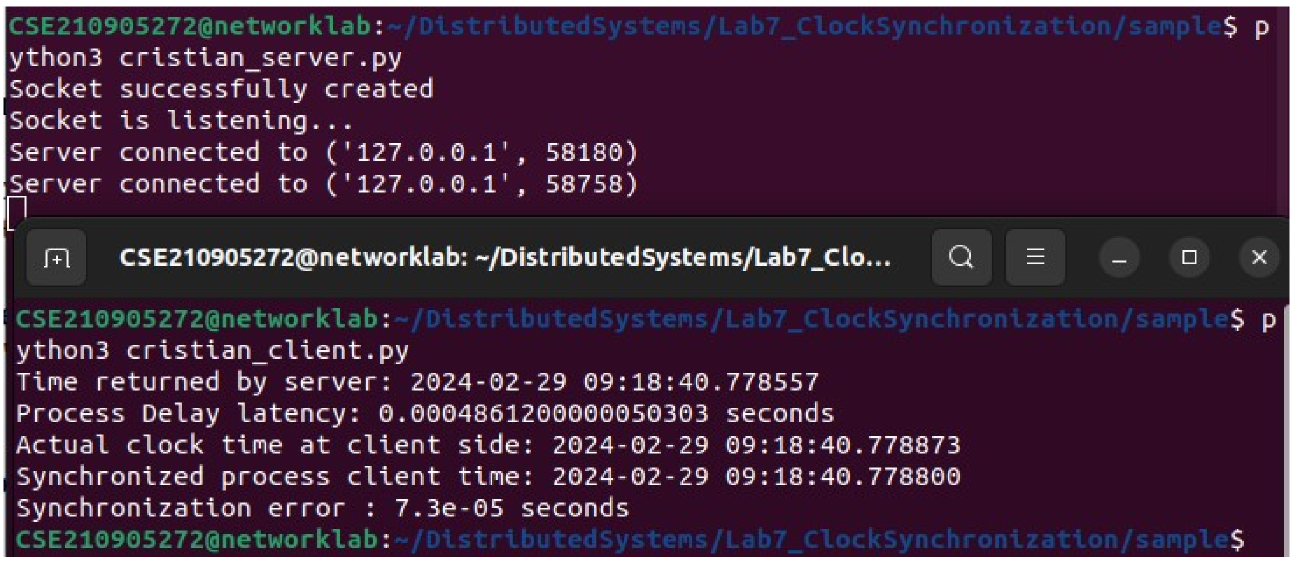
Cristian Algorithm

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
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__':
        synchronizeTime()
```



```
CSE210905272@networklab:~/DistributedSystems/Lab7_ClockSynchronization/sample$ p
ython3 cristian_server.py
Socket successfully created
Socket is listening...
Server connected to ('127.0.0.1', 58180)
Server connected to ('127.0.0.1', 58758)
```

```
CSE210905272@networklab:~/DistributedSystems/Lab7_ClockSynchronization/sample$ p
ython3 cristian_client.py
Time returned by server: 2024-02-29 09:18:40.778557
Process Delay latency: 0.00048612000000050303 seconds
Actual clock time at client side: 2024-02-29 09:18:40.778873
Synchronized process client time: 2024-02-29 09:18:40.778800
Synchronization error : 7.3e-05 seconds
CSE210905272@networklab:~/DistributedSystems/Lab7_ClockSynchronization/sample$
```

.

Berkeley's algorithm

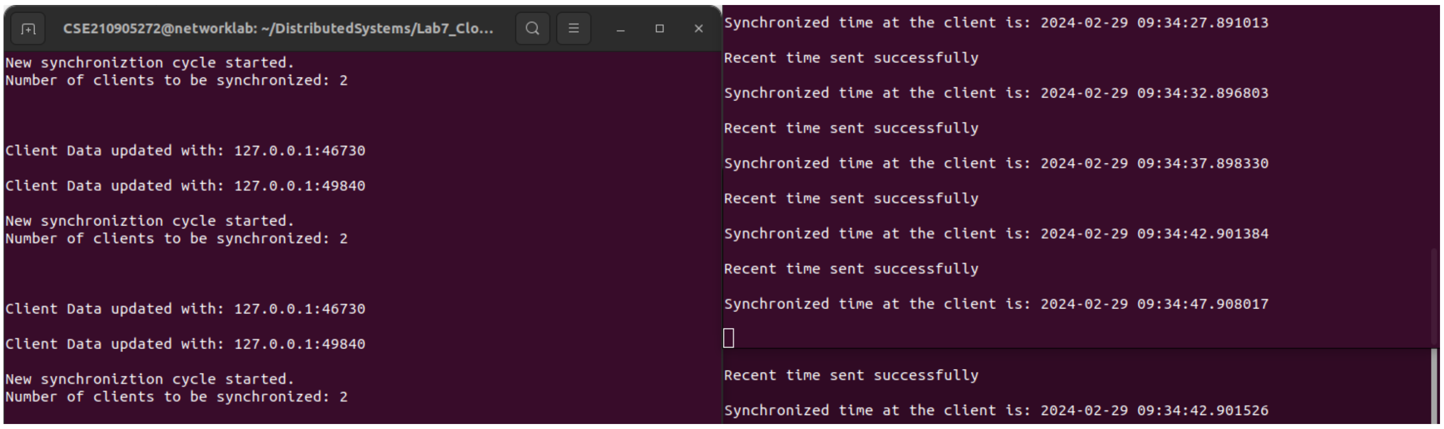
server.py

```
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)
```

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)
```



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


Client Data updated with: 127.0.0.1:46730

Client Data updated with: 127.0.0.1:49840

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


Client Data updated with: 127.0.0.1:46730

Client Data updated with: 127.0.0.1:49840

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

```
Synchronized time at the client is: 2024-02-29 09:34:27.891013

Recent time sent successfully

Synchronized time at the client is: 2024-02-29 09:34:32.896803

Recent time sent successfully

Synchronized time at the client is: 2024-02-29 09:34:37.898330

Recent time sent successfully

Synchronized time at the client is: 2024-02-29 09:34:42.901384

Recent time sent successfully

Synchronized time at the client is: 2024-02-29 09:34:47.908017

Recent time sent successfully

Synchronized time at the client is: 2024-02-29 09:34:42.901526
```

.

LAB EXERCISES

Q1) The Manipal Foodie is a renowned automated food processing outlet known for its tiffin service to students. The various processes involved are food production, filling and packing. Every day more than 3000 orders are received on an average from the students in manipal. There are total of 4 production lines for orders received from KMC, MIT, TAPMI and SOLS students, each of them has a digital clock which needs to be in synchronization with the master clock. The master clock mounted in the testing lab controls the entire clock system. Design an appropriate solution using Berkeley's algorithm for the above scenario. Assume that the clocks at the institutes are slave/clients.

->
testinglab.py

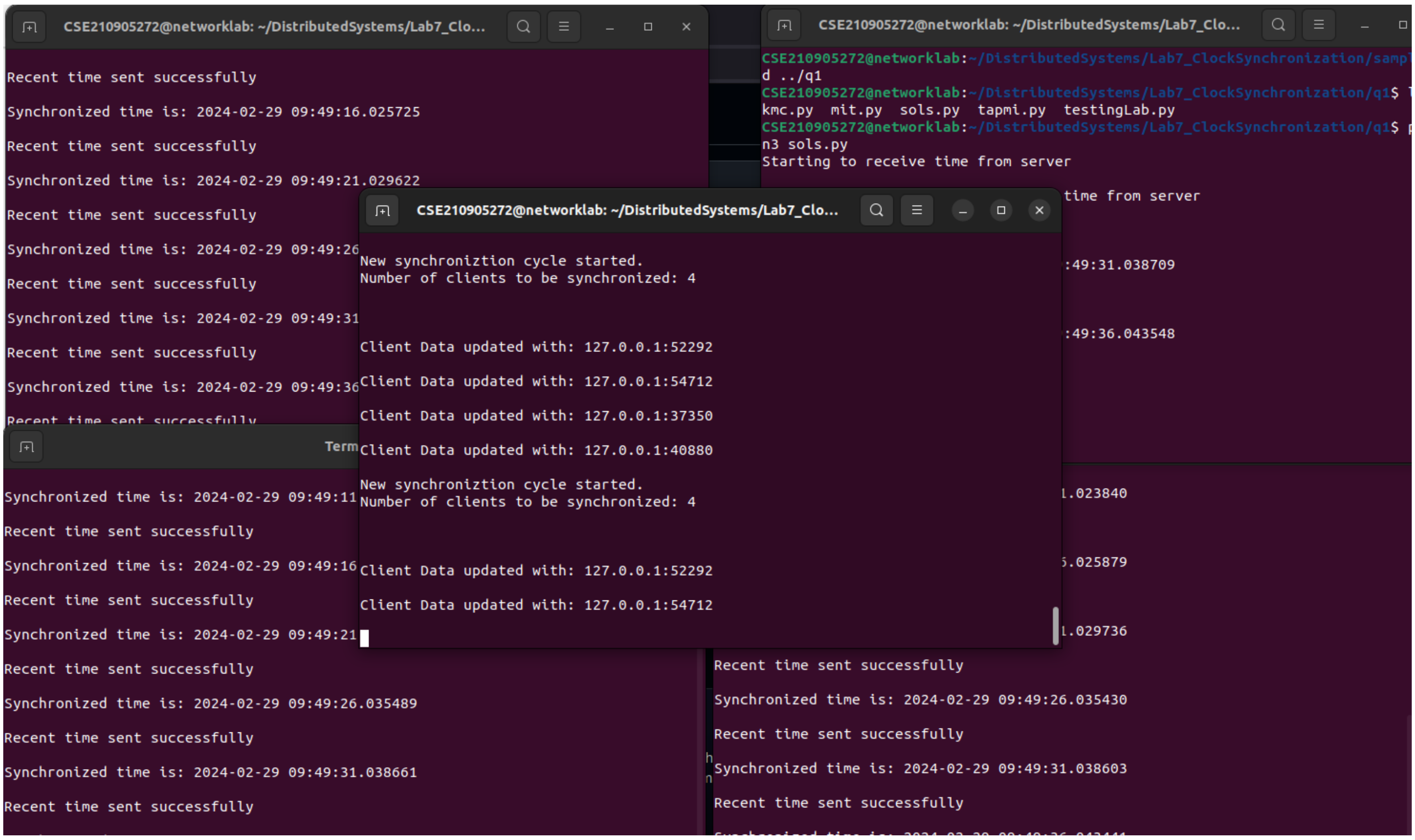
```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time


# 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("This is the master clock at the testing lab.\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)
```

<u>commonCLient.py</u>

```
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 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("This is KMC time client.\nStarting 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)
```



```
CSE210905272@networklab: ~/DistributedSystems/Lab7_Clo...

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:16.025725

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:21.029622

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:26

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:31

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:36
```

```
CSE210905272@networklab: ~/DistributedSystems/Lab7_Clo...
CSE210905272@networklab:~/DistributedSystems/Lab7_ClockSynchronization/samp
d ../q1
CSE210905272@networklab:~/DistributedSystems/Lab7_ClockSynchronization/q1$
kmc.py  mit.py  sols.py  tapmi.py  testingLab.py
CSE210905272@networklab:~/DistributedSystems/Lab7_ClockSynchronization/q1$
n3 sols.py
Starting to receive time from server

                                        time from server


                                        :49:31.038709


                                        :49:36.043548
```

```
CSE210905272@networklab: ~/DistributedSystems/Lab7_Clo...

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


Client Data updated with: 127.0.0.1:52292

Client Data updated with: 127.0.0.1:54712

Client Data updated with: 127.0.0.1:37350

Client Data updated with: 127.0.0.1:40880

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


Client Data updated with: 127.0.0.1:52292

Client Data updated with: 127.0.0.1:54712
```

```
Term

Synchronized time is: 2024-02-29 09:49:11

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:16

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:21

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:26.035489

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:31.038661

Recent time sent successfully
```

```
1.023840


5.025879


1.029736
Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:26.035430

Recent time sent successfully

Synchronized time is: 2024-02-29 09:49:31.038603

Recent time sent successfully
```

.

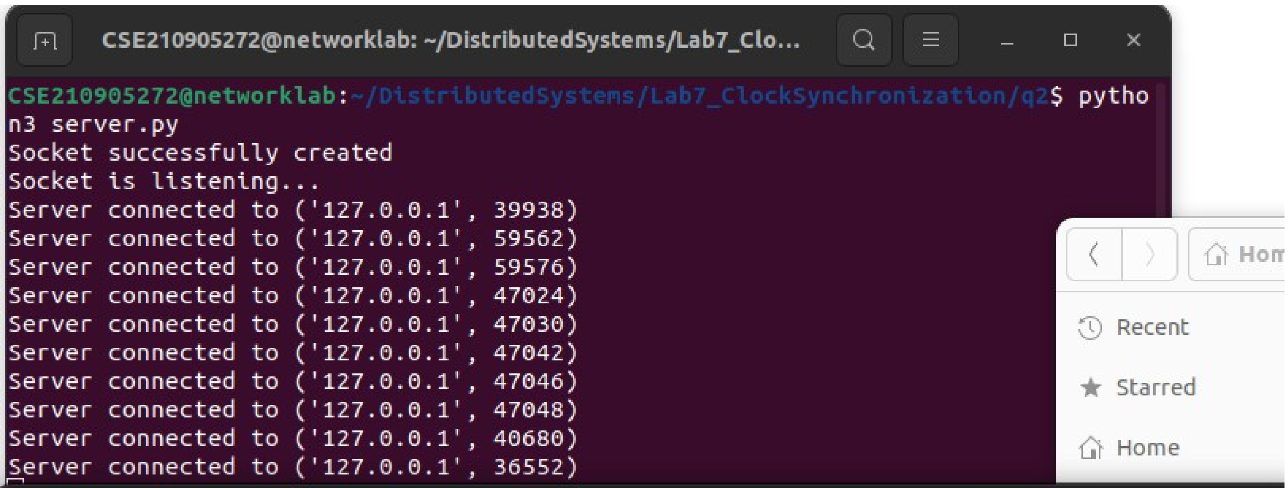
Q2) Manipal Buddy is a banking and education application for the students and staff of MIT, Manipal. Mr Vinay, a sixth semester student wants to pay the end semester exams fees for a re-registered course. He simultaneously wishes to register for a course on NPTEL through the app. To register for exam he uses the mobile app whereas to register for NPTEL course he uses his laptop to log in. As he needs to finish both the registrations on the same day, he tries to do both the tasks simultaneously. Analyse and demonstrate using a program how Cristian's algorithm can be used in the above case to synchronize the clocks. Assume the relevant parameters.

->server.py

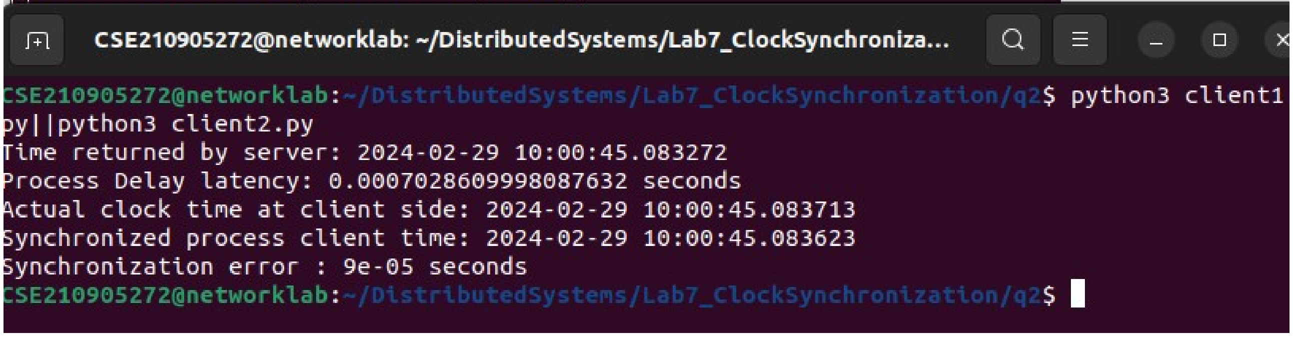
```
import socket
import datetime
import time
# function used to initiate the Clock Server
def initiateClockServer():
        s = socket.socket()
        print("Socket successfully created")
        # Server port
        port = 8011
        s.bind(('', port))
        # Start listening to requests
        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
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__':
        synchronizeTime()
```



```
CSE210905272@networklab:~/DistributedSystems/Lab7_ClockSynchronization/q2$ pytho
n3 server.py
Socket successfully created
Socket is listening...
Server connected to ('127.0.0.1', 39938)
Server connected to ('127.0.0.1', 59562)
Server connected to ('127.0.0.1', 59576)
Server connected to ('127.0.0.1', 47024)
Server connected to ('127.0.0.1', 47030)
Server connected to ('127.0.0.1', 47042)
Server connected to ('127.0.0.1', 47046)
Server connected to ('127.0.0.1', 47048)
Server connected to ('127.0.0.1', 40680)
Server connected to ('127.0.0.1', 36552)
```



CSE210905272@networklab: ~/DistributedSystems/Lab7_ClockSynchroniza...

```
CSE210905272@networklab:~/DistributedSystems/Lab7_ClockSynchronization/q2$ python3 client1
py||python3 client2.py
Time returned by server: 2024-02-29 10:00:45.083272
Process Delay latency: 0.0007028609998087632 seconds
Actual clock time at client side: 2024-02-29 10:00:45.083713
Synchronized process client time: 2024-02-29 10:00:45.083623
Synchronization error : 9e-05 seconds
CSE210905272@networklab:~/DistributedSystems/Lab7_ClockSynchronization/q2$
```