LAB – 6

CLOCK SYNCHRONIZATION

Solved examples

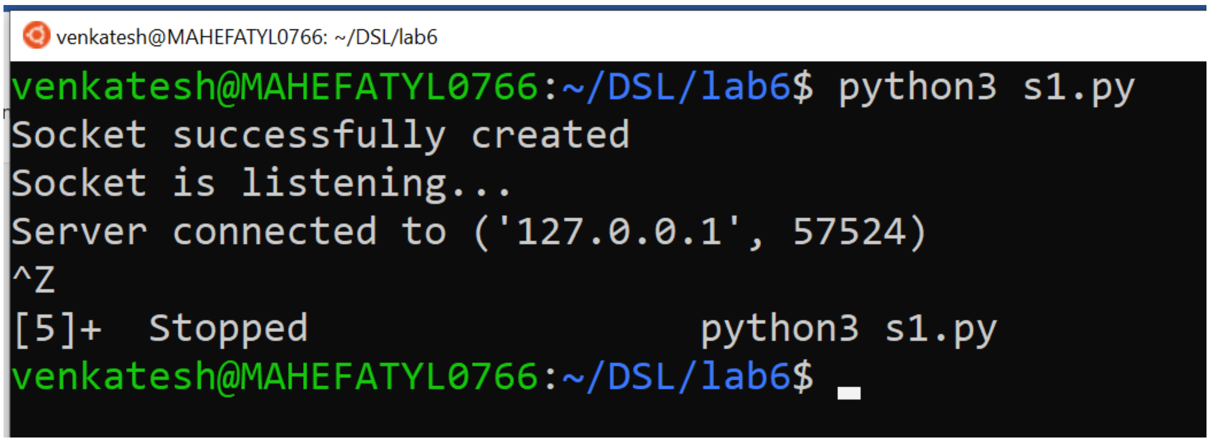
Cristian’s algorithm

1) **To initiate a prototype of a clock server on local machine:**

server.py

```
# Cristian’s algorithm
# To initiate a prototype of a clock server on local machine: Server
# Python3 program imitating a clock server

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 = 9014
    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()
```

OUTPUT AT THE SERVER TERMINAL

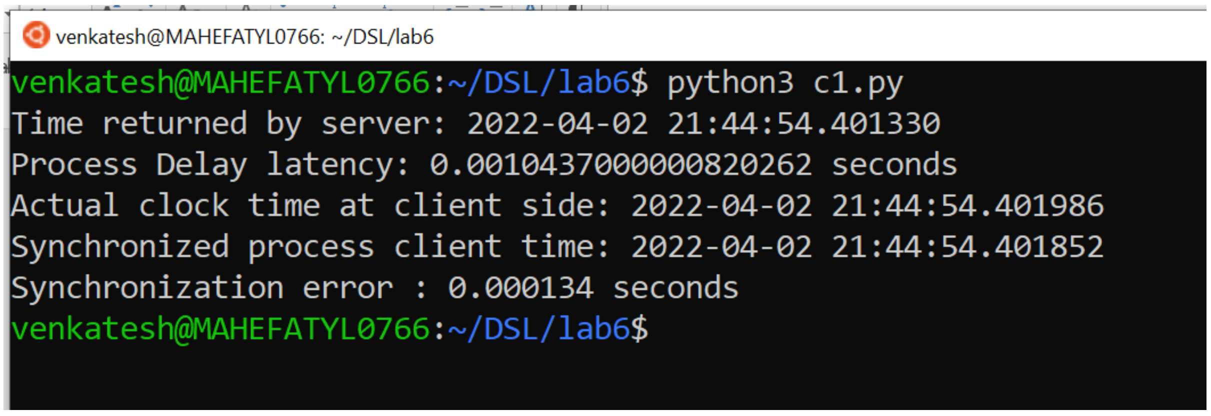


```
venkatesh@MAHEFATYL0766:~/DSL/lab6$ python3 s1.py
Socket successfully created
Socket is listening...
Server connected to ('127.0.0.1', 57524)
^Z
[5]+  Stopped                 python3 s1.py
venkatesh@MAHEFATYL0766:~/DSL/lab6$ 
```

client.py

```
# Code below is used to initiate a prototype of a client process on local
machine:
# Python3 program imitating a client process

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 = 9014
    # 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()
```

<u>OUTPUT AT THE CLIENT TERMINAL</u>



```
venkatesh@MAHEFATYL0766:~/DSL/lab6$ python3 c1.py
Time returned by server: 2022-04-02 21:44:54.401330
Process Delay latency: 0.0010437000000820262 seconds
Actual clock time at client side: 2022-04-02 21:44:54.401986
Synchronized process client time: 2022-04-02 21:44:54.401852
Synchronization error : 0.000134 seconds
venkatesh@MAHEFATYL0766:~/DSL/lab6$
```

https://github.com/tezansahu/Berkeley_Clock_Synchronization/blob/master/master.py

Second solved program

# Python3 program imitating a clock server

#Server.py

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

client_data = {}
port = 8001

''' nested thread function used to receive
    clock time from a connected client '''
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 = port):

    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))

    master_server.listen(10)
    print("Clock server started...\n")
    print("Starting to make connections...\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 = port)
```

client.py

```
# Python3 program imitating a client process

from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time
port = 8001
def startSendingTime(slave_client):
    while True:
        slave_client.send(str(
                        datetime.datetime.now()).encode())
        print("Recent 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 = port):
    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 = port)
```

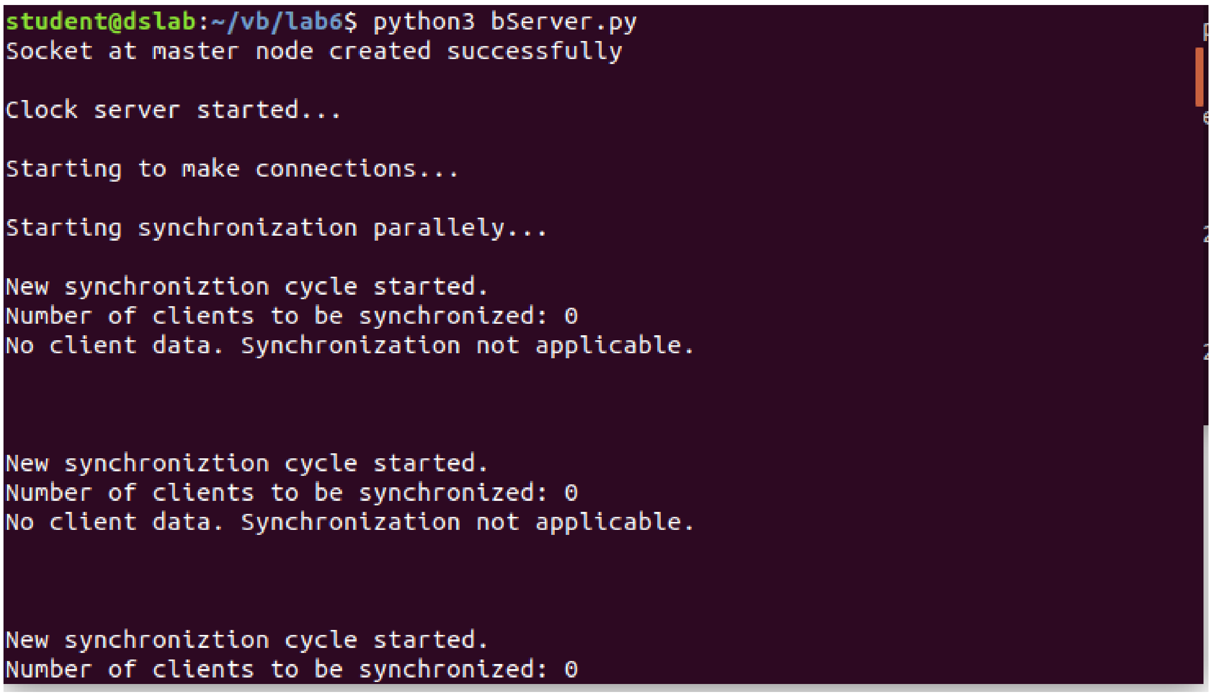
Steps of execution for 2nd program

1) Execute the server program in one terminal

2) Open other terminal and execute the client program

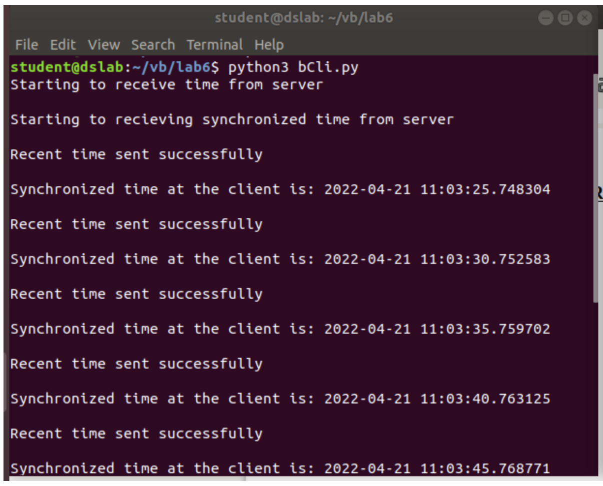
(Can also open multiple terminals and run the client program parallelly)

<u>OUTPUT AT THE SERVER TERMINAL</u>

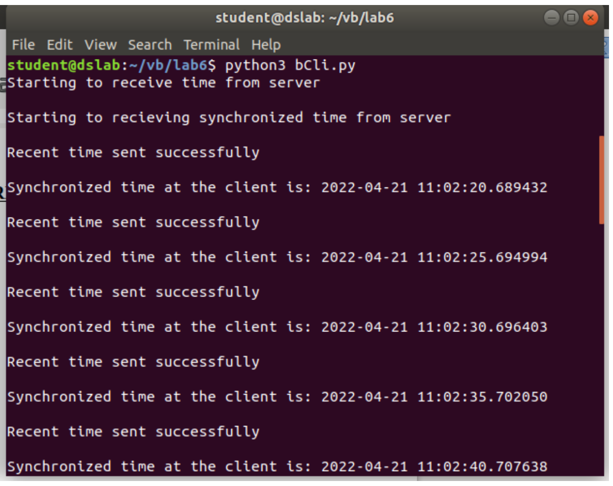


```
student@dslab:~/vb/lab6$ python3 bServer.py
Socket at master node created successfully

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.



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
```

<u>OUTPUT AT THE CLIENT TERMINAL</u>



```
student@dslab:~/vb/lab6$ python3 bCli.py
Starting to receive time from server

Starting to recieving synchronized time from server

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:03:25.748304

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:03:30.752583

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:03:35.759702

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:03:40.763125

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:03:45.768771
```



```
student@dslab:~/vb/lab6$ python3 bCli.py
Starting to receive time from server

Starting to recieving synchronized time from server

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:02:20.689432

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:02:25.694994

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:02:30.696403

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:02:35.702050

Recent time sent successfully

Synchronized time at the client is: 2022-04-21 11:02:40.707638
```

Client-1 **Client-2**