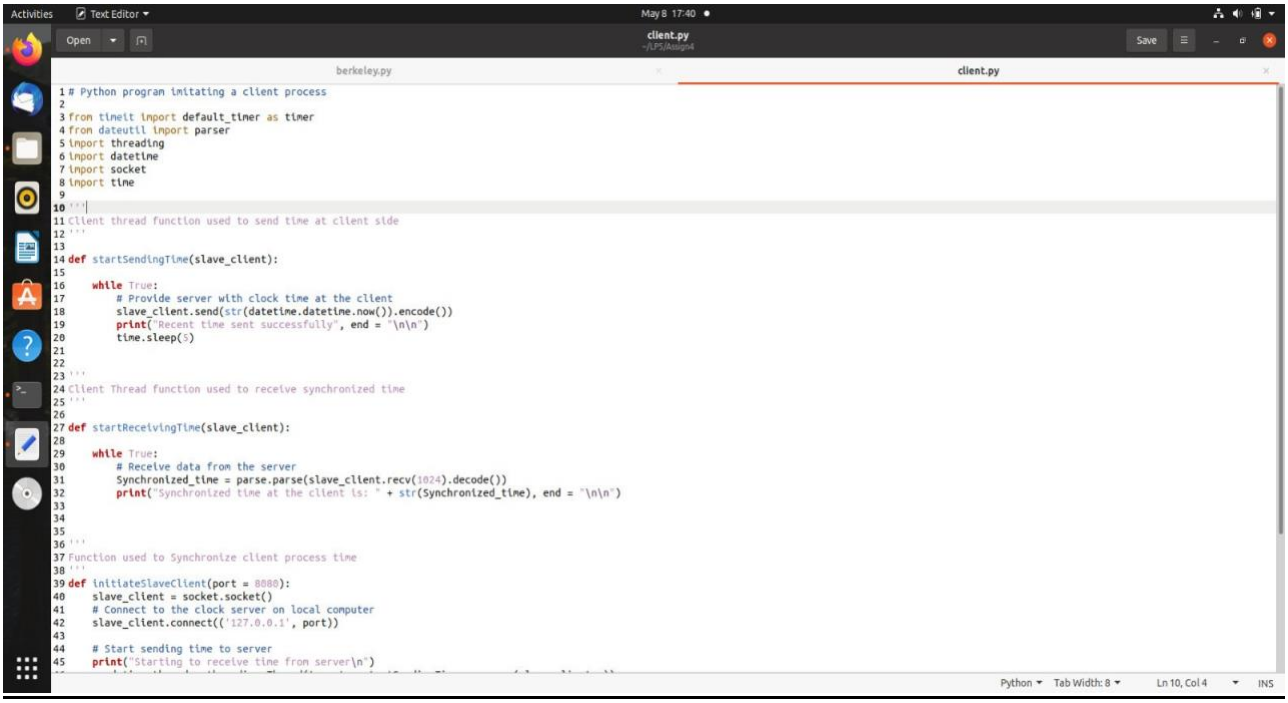
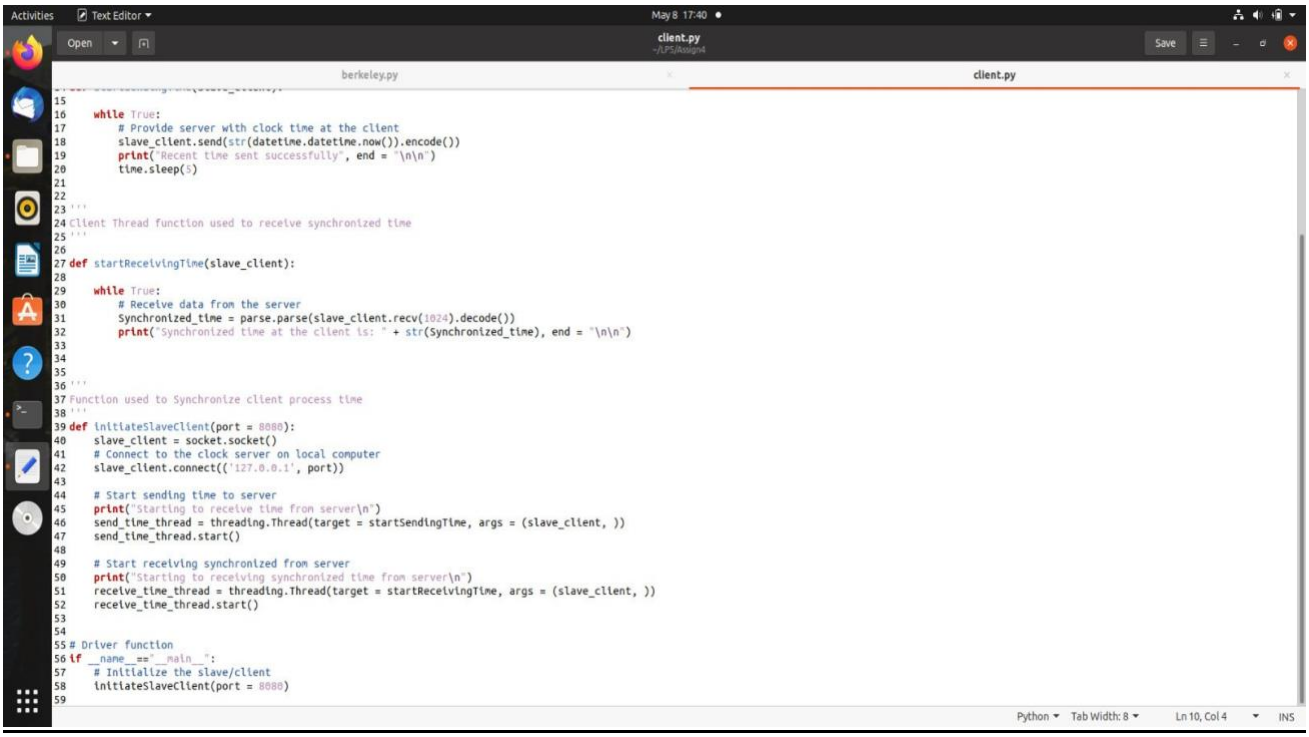
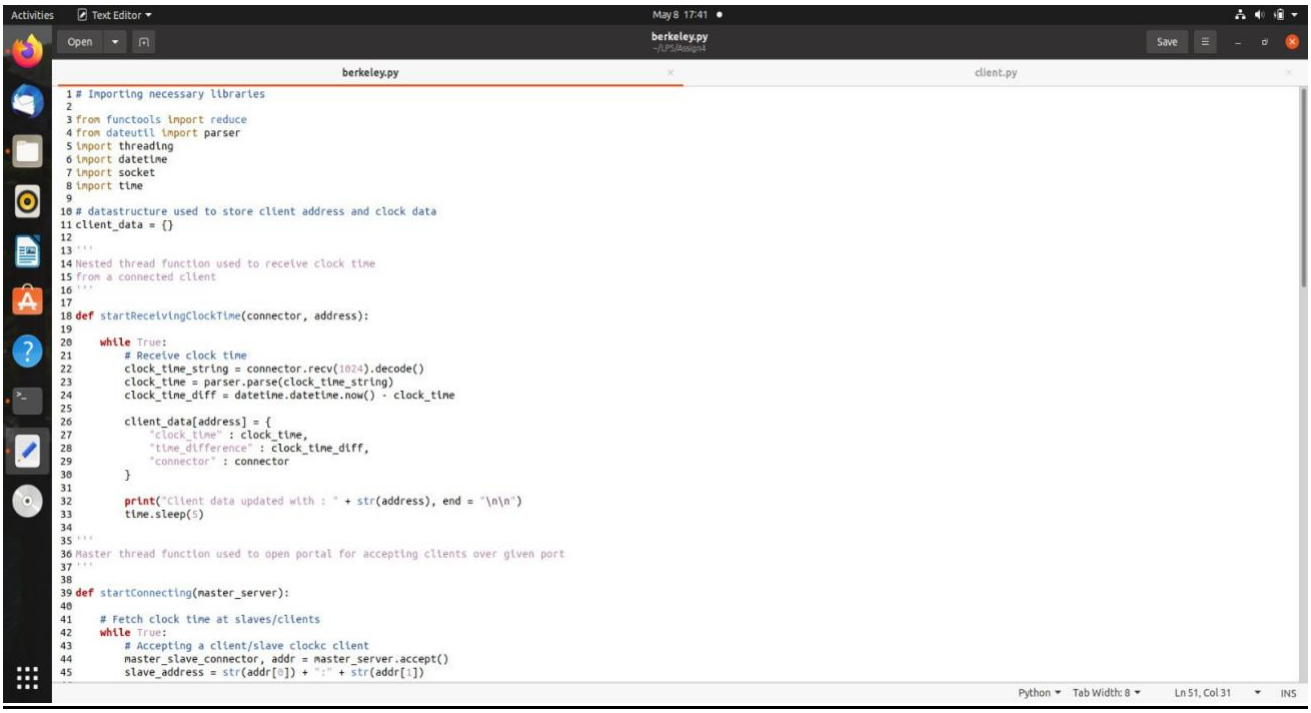
CODE IMPLEMENTATION :

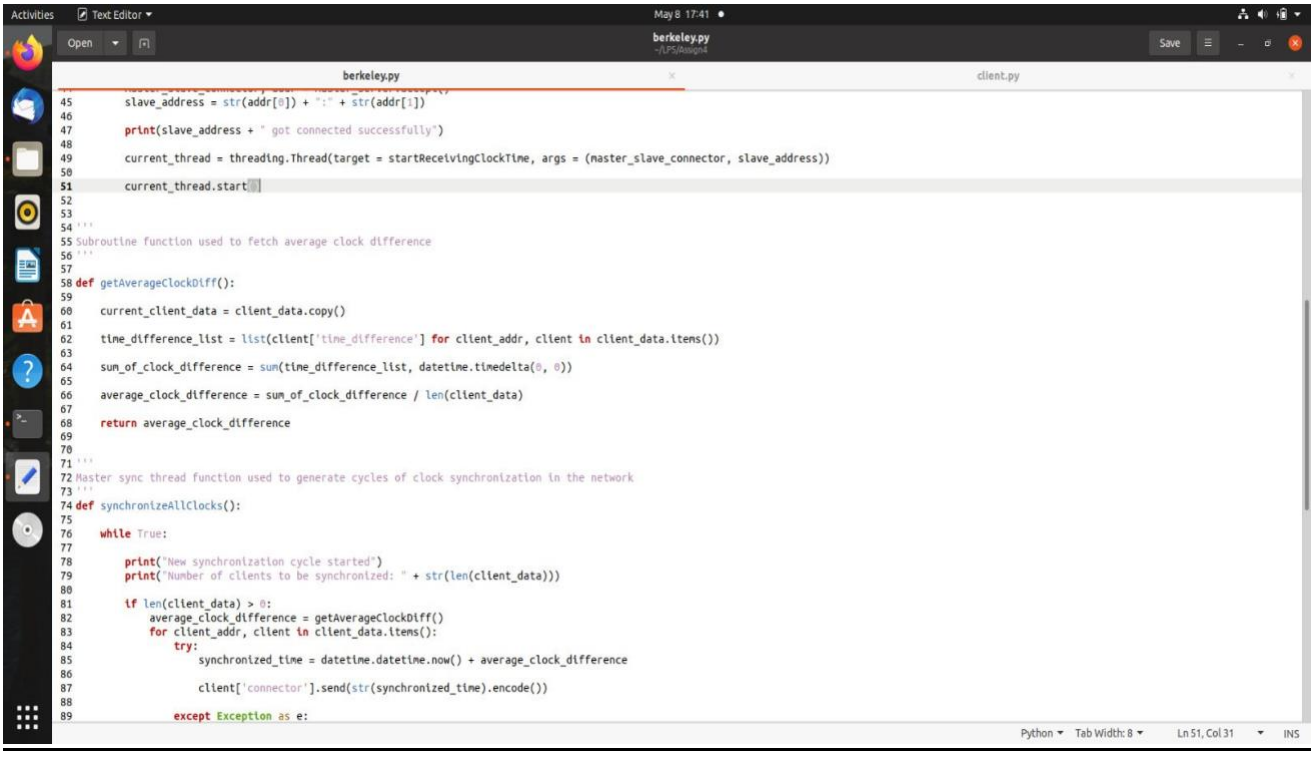


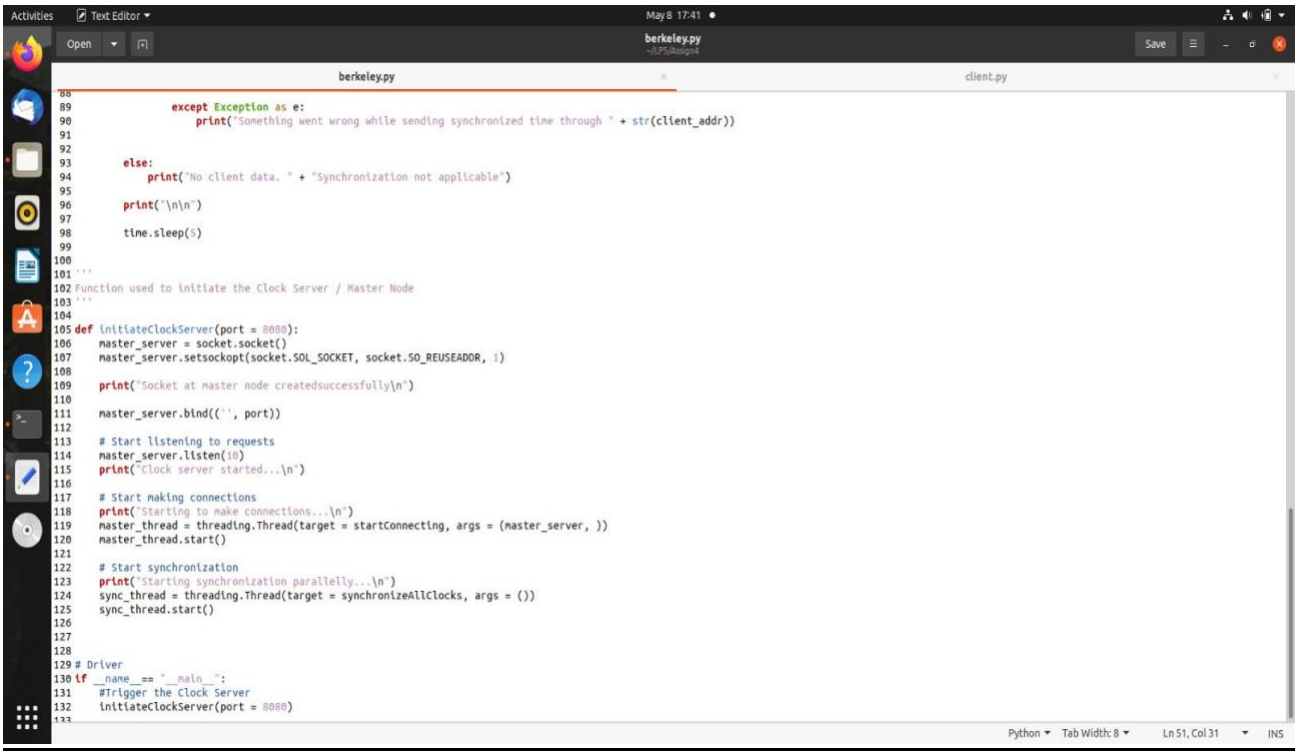
```python
# Python program imitating a client process

from tinelt 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 = parse.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")
```

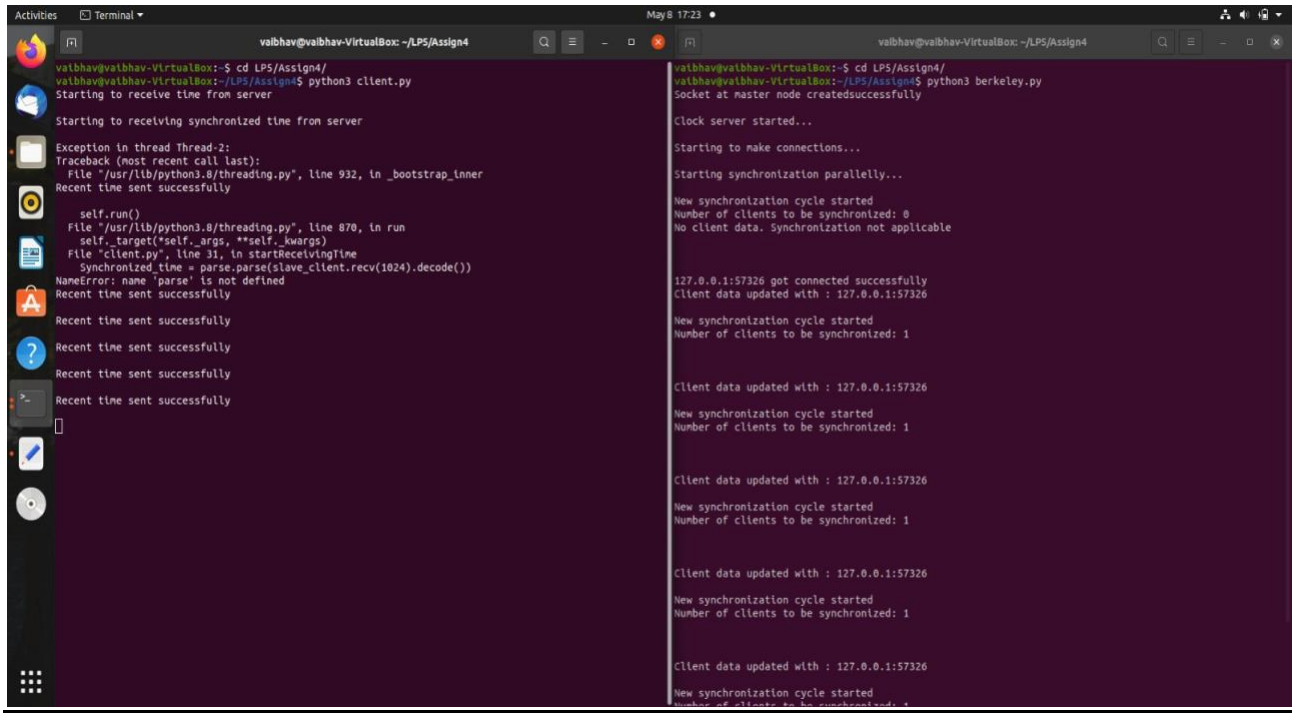


```python

    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 = parse.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 receiving synchronized from server
    print("Starting to receiving 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)
```


Activities
Text Editor
May 8 17:41
Open
berkeley.py
~/LPS/Assign4
Save
berkeley.py
client.py
# Importing necessary libraries
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 startReceivingClockTime(connector, address):
while True:
# Receive 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 clockc client
master_slave_connector, addr = master_server.accept()
slave_address = str(addr[0]) + ":" + str(addr[1])
Python
Tab Width: 8
Ln 51, Col 31
INS


Activities
Text Editor
May 8 17:41
Open
berkeley.py
~/LPS/Assign4
Save
berkeley.py
client.py
slave_address = str(addr[0]) + ":" + str(addr[1])
print(slave_address + " got connected successfully")
current_thread = threading.Thread(target = startReceivingClockTime, 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
'''
Master sync thread function used to generate cycles of clock synchronization in the network
'''
def synchronizeAllClocks():
while True:
print("New synchronization 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:
Python
Tab Width: 8
Ln 51, Col 31
INS


Activities
Text Editor
May 8 17:41
Open
berkeley.py
~/LP5/Assign4
Save
berkeley.py
client.py
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 createdsuccessfully\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 synchronization
print("Starting synchronization parallelly...\n")
sync_thread = threading.Thread(target = synchronizeAllClocks, args = ())
sync_thread.start()
# Driver
if __name__== "__main__":
#Trigger the Clock Server
initiateClockServer(port = 8080)
Python
Tab Width: 8
Ln 51, Col 31
INS


Activities
Terminal
May 8 17:23
vaibhav@vaibhav-VirtualBox: ~/LP5/Assign4
vaibhav@vaibhav-VirtualBox:~$ cd LP5/Assign4/
vaibhav@vaibhav-VirtualBox:~/LP5/Assign4$ python3 client.py
Starting to receive time from server
Starting to receiving synchronized tine from server
Exception in thread Thread-2:
Traceback (most recent call last):
File "/usr/lib/python3.8/threading.py", line 932, in _bootstrap_inner
Recent time sent successfully
self.run()
File "/usr/lib/python3.8/threading.py", line 870, in run
self._target(*self._args, **self._kwargs)
File "client.py", line 31, in startReceivingTime
synchronized_time = parse.parse(slave_client.recv(1024).decode())
NameError: name 'parse' is not defined
Recent time sent successfully
Recent time sent successfully
Recent time sent successfully
Recent time sent successfully
Recent time sent successfully
vaibhav@vaibhav-VirtualBox: ~/LP5/Assign4
vaibhav@vaibhav-VirtualBox:~$ cd LP5/Assign4/
vaibhav@vaibhav-VirtualBox:~/LP5/Assign4$ python3 berkeley.py
Socket at master node createdsuccessfully
Clock server started...
Starting to make connections...
Starting synchronization parallelly...
New synchronization cycle started
Number of clients to be synchronized: 0
No client data. Synchronization not applicable
127.0.0.1:57326 got connected successfully
Client data updated with : 127.0.0.1:57326
New synchronization cycle started
Number of clients to be synchronized: 1
Client data updated with : 127.0.0.1:57326
New synchronization cycle started
Number of clients to be synchronized: 1
Client data updated with : 127.0.0.1:57326
New synchronization cycle started
Number of clients to be synchronized: 1
Client data updated with : 127.0.0.1:57326
New synchronization cycle started
Number of clients to be synchronized: 1
Client data updated with : 127.0.0.1:57326
New synchronization cycle started