Client.py-

```
import threading

import datetime

import socket

import time


def send_time(slave_client):

   while True:

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

      print("Time sent successfully")

      time.sleep(5)


def receive_time(slave_client):

   while True:

      synchronized_time = datetime.datetime.strptime(slave_client.recv(1024).decode(), "%Y-%m-%d
%H:%M:%S.%f")

      print("Synchronized time at the client is:", synchronized_time)


def initiate_slave_client(port=8080):

   slave_client = socket.socket()

   slave_client.connect(('127.0.0.1', port))

   print("Starting to receive time from server")

   threading.Thread(target=send_time, args=(slave_client,)).start()

   print("Starting to receive synchronized time from server")

   threading.Thread(target=receive_time, args=(slave_client,)).start()


if __name__ == '__main__':

   initiate_slave_client(port=8080)
```

server.py-

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

client_data = {}

def start_receiving_clock_time(connector, address):
    while True:
        clock_time = parser.parse(connector.recv(1024).decode())
        clock_time_diff = datetime.datetime.now() - clock_time
        client_data[address] = {"clock_time": clock_time, "time_difference": clock_time_diff,
"connector": connector}
        time.sleep(5)

def start_connecting(master_server):
    while True:
        master_slave_connector, addr = master_server.accept()
        client_address = f"{addr[0]}:{addr[1]}"
        threading.Thread(target=start_receiving_clock_time, args=(master_slave_connector,
client_address)).start()
        print(f"Client connected from address {client_address}")
def synchronize_all_clocks():
    while True:
        if len(client_data) > 0:
            avg_clock_diff = sum((client['time_difference'] for client in client_data.values()),
datetime.timedelta()) / len(client_data)
            for client in client_data.values():
                synchronized_time = datetime.datetime.now() + avg_clock_diff
                try:
                    client['connector'].send(str(synchronized_time).encode())
            except Exception as e:
                print(f"Error sending synchronized time to {client['address']}: {e}")
        time.sleep(5)

def initiate_clock_server(port=8080):
    master_server = socket.socket()
    master_server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    master_server.bind(('', port))
    master_server.listen(10)
    print("Clock server started...")
    threading.Thread(target=start_connecting, args=(master_server,)).start()
    threading.Thread(target=synchronize_all_clocks).start()

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

Info =

Probleam Statement: Implement Berkeley algorithm for clock synchronization.

Commands To Execute Assignment-4:

On Terminal-1:

```
python server.py
```

On Terminal-2:

```
python client.py
```