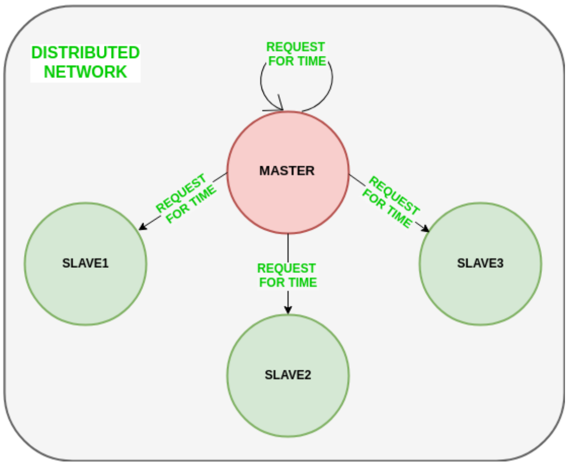
PROGRAM - 2

Aim: To implement Berkeley clock synchronization algorithm.

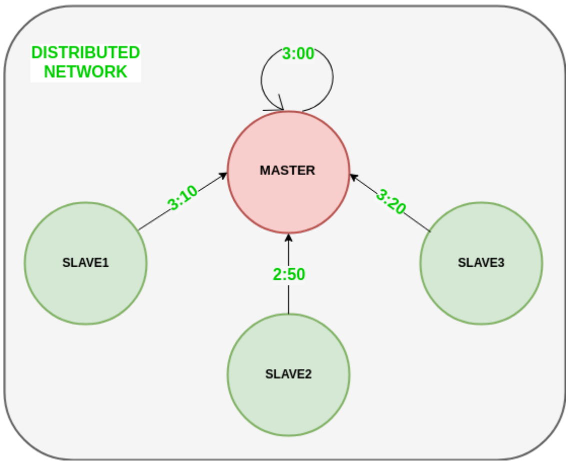
Theory: Berkeley's Algorithm is a clock synchronization technique used in distributed systems. The algorithm assumes that each machine node in the network either doesn't have an accurate time source or doesn't possess an UTC server.

Steps involved in Berkeley's Algorithm:

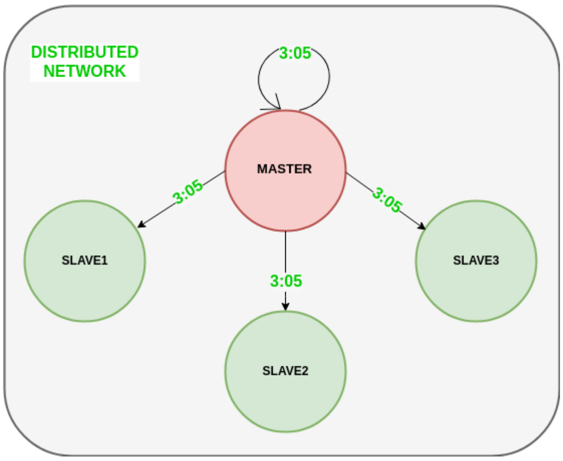
1.



2.



3.



Code:

Server:

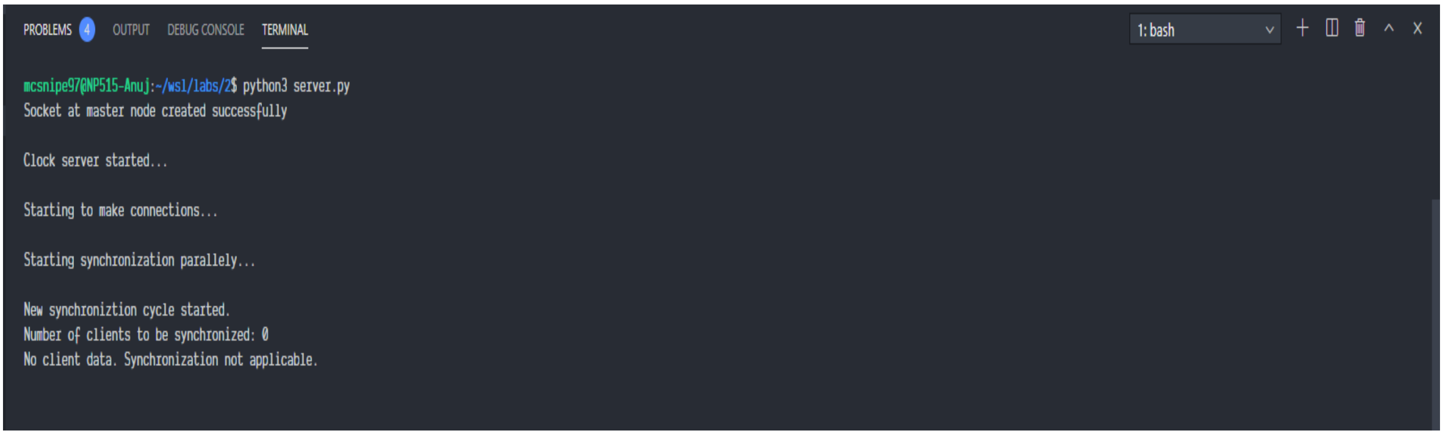
```
from functools import reduce
from dateutil import parser
import threading
import datetime
import socket
import time


# 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


''' master sync thread function used to generate
    cycles of clock synchronization in the network '''


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:

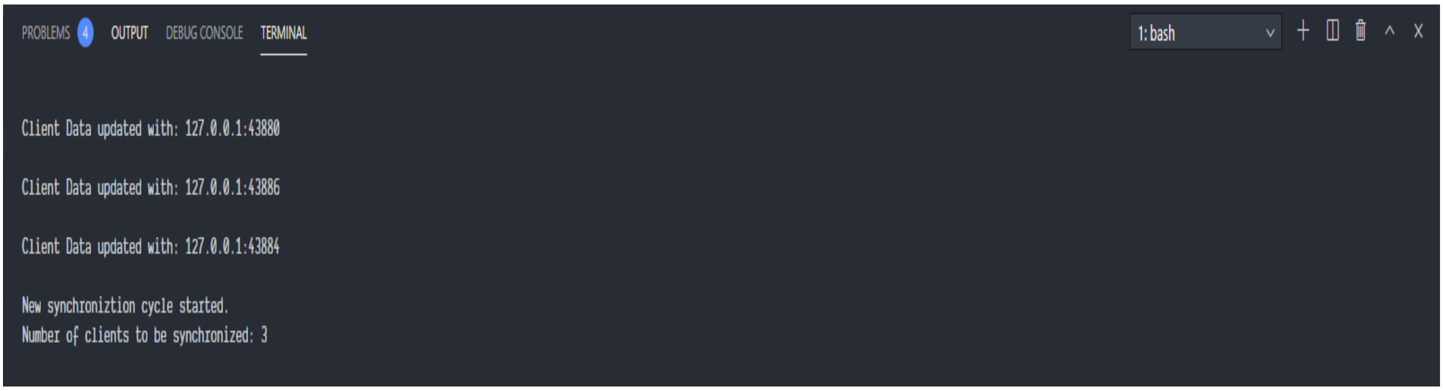
```
from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time


# 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:



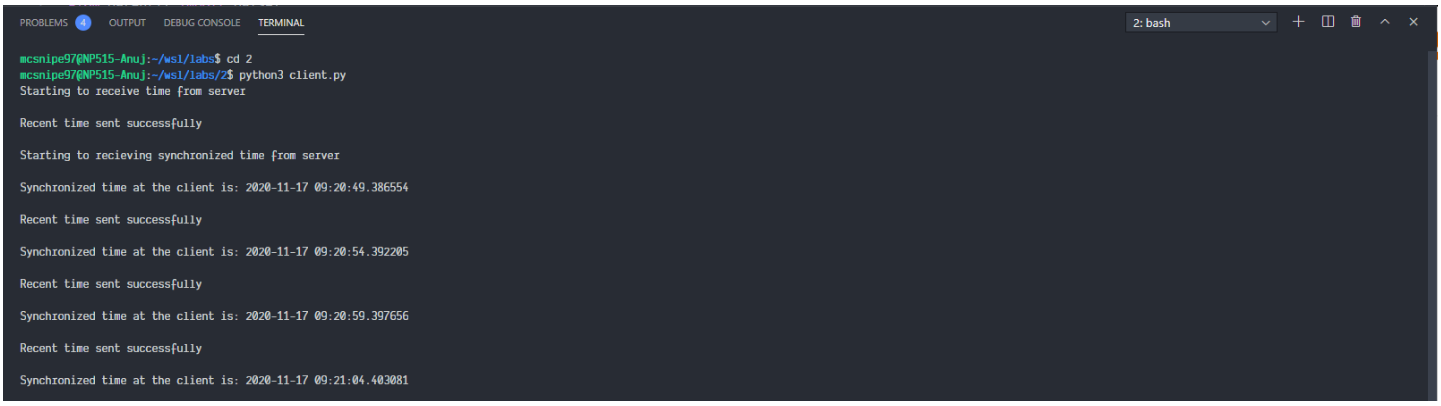
```
mcsnipe97@NP515-Anuj:~/wsl/labs/2$ python3 server.py
Socket at master node created successfully

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.
```



```
Client Data updated with: 127.0.0.1:43880

Client Data updated with: 127.0.0.1:43886

Client Data updated with: 127.0.0.1:43884

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

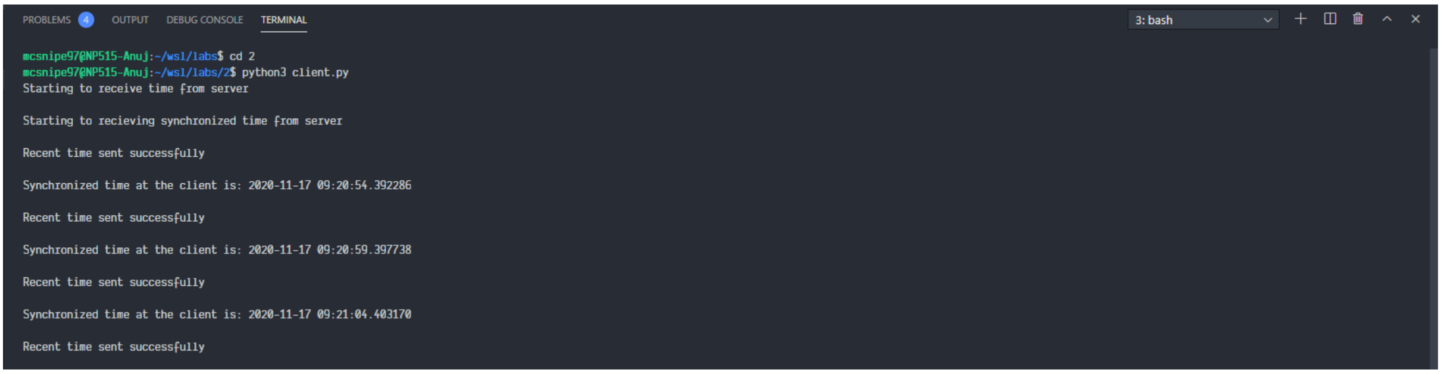
Client:

1:



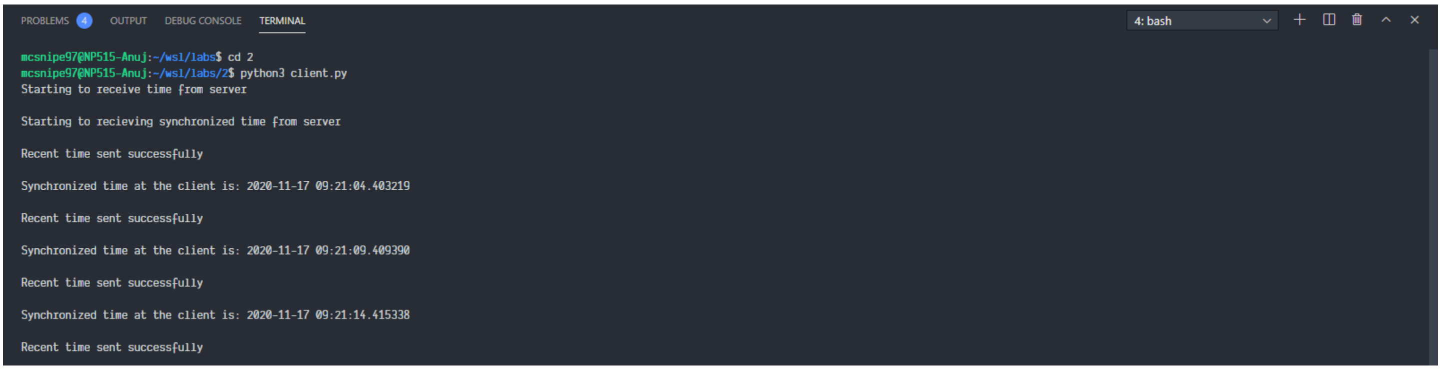
```
mcsnipe97@NP515-Anuj:~/wsl/labs$ cd 2
mcsnipe97@NP515-Anuj:~/wsl/labs/2$ python3 client.py
Starting to receive time from server

Recent time sent successfully

Starting to recieving synchronized time from server

Synchronized time at the client is: 2020-11-17 09:20:49.386554

Recent time sent successfully

Synchronized time at the client is: 2020-11-17 09:20:54.392205

Recent time sent successfully

Synchronized time at the client is: 2020-11-17 09:20:59.397656

Recent time sent successfully

Synchronized time at the client is: 2020-11-17 09:21:04.403081
```

2:



```
mcsnipe97@NP515-Anuj:~/wsl/labs$ cd 2
mcsnipe97@NP515-Anuj:~/wsl/labs/2$ python3 client.py
Starting to receive time from server

Starting to recieving synchronized time from server

Recent time sent successfully

Synchronized time at the client is: 2020-11-17 09:20:54.392286

Recent time sent successfully

Synchronized time at the client is: 2020-11-17 09:20:59.397738

Recent time sent successfully

Synchronized time at the client is: 2020-11-17 09:21:04.403170

Recent time sent successfully
```

3:



```
mcsnipe97@NP515-Anuj:~/wsl/labs$ cd 2
mcsnipe97@NP515-Anuj:~/wsl/labs/2$ python3 client.py
Starting to receive time from server

Starting to recieving synchronized time from server

Recent time sent successfully

Synchronized time at the client is: 2020-11-17 09:21:04.403219

Recent time sent successfully

Synchronized time at the client is: 2020-11-17 09:21:09.409390

Recent time sent successfully

Synchronized time at the client is: 2020-11-17 09:21:14.415338

Recent time sent successfully
```

Conclusion:

Berkeley clock synchronization algorithm is implemented.