Aim - WAP to demonstrate Berkeley clock synchronization algorithm

Theory -

Clock synchronization : deal with understanding the temporal ordering of events produced by concurrent process

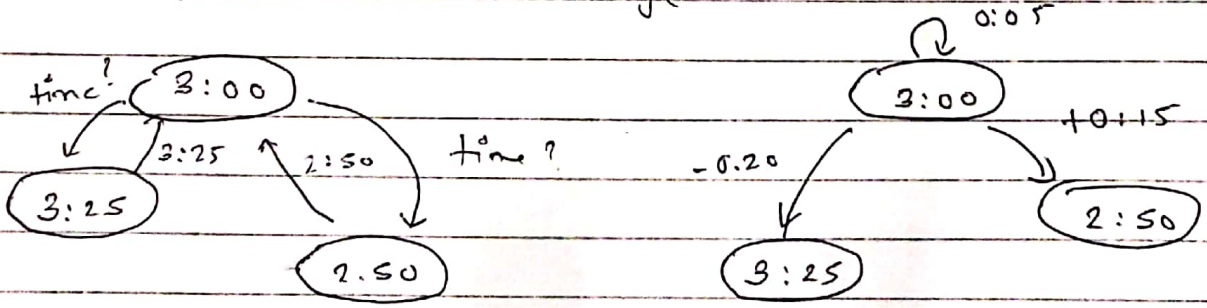
It is usefull for synchronizing senders & receivers of message controlling joint activities of the serializing concurrent access to shared objects.

In a centralized system - the solution is trivial, The centralized server will & dictate the system time. Cristian's algo & Berkeley Algorithm are some solution to clock synchronization problem.

Berkeley Algorithm:-

The Berkeley Algorithm was developed by Gusella & Zatti in 1989, does not assume that any machine has an accurate time source with which to synchronize

Instead, it opts for obtaining an average time from participating computers & synchronizing all machine to that average



Teacher's Sign.: _____

The machine with time 3:00 is the server. It sends out synchronization query to other machine in the group.

Each machine sends a timestamp as a response to the query

Server averages the three timestamps
- the two it received
- and its own

Now it sends an offset to each machine so that the machine's time will be synchronized to average.

Machine with a time of 3:25 gets offset of -0.20
Machine with a time of 2:50 gets offset of +0:15
server adjust its own time by +0:05

Conclusion:-

Algorithm also has provisions to ignore reading from clocks whose skew is too great.

If master machine fails any other slave could be elected to take over

Understand the concepts of clock synchronization. Berkeley algorithm is successfully implemented

Teacher's Sign.: _____

Berkley's Clock Synchronization

Code:

Berkley_Server.py

```
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 = {}


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:
```

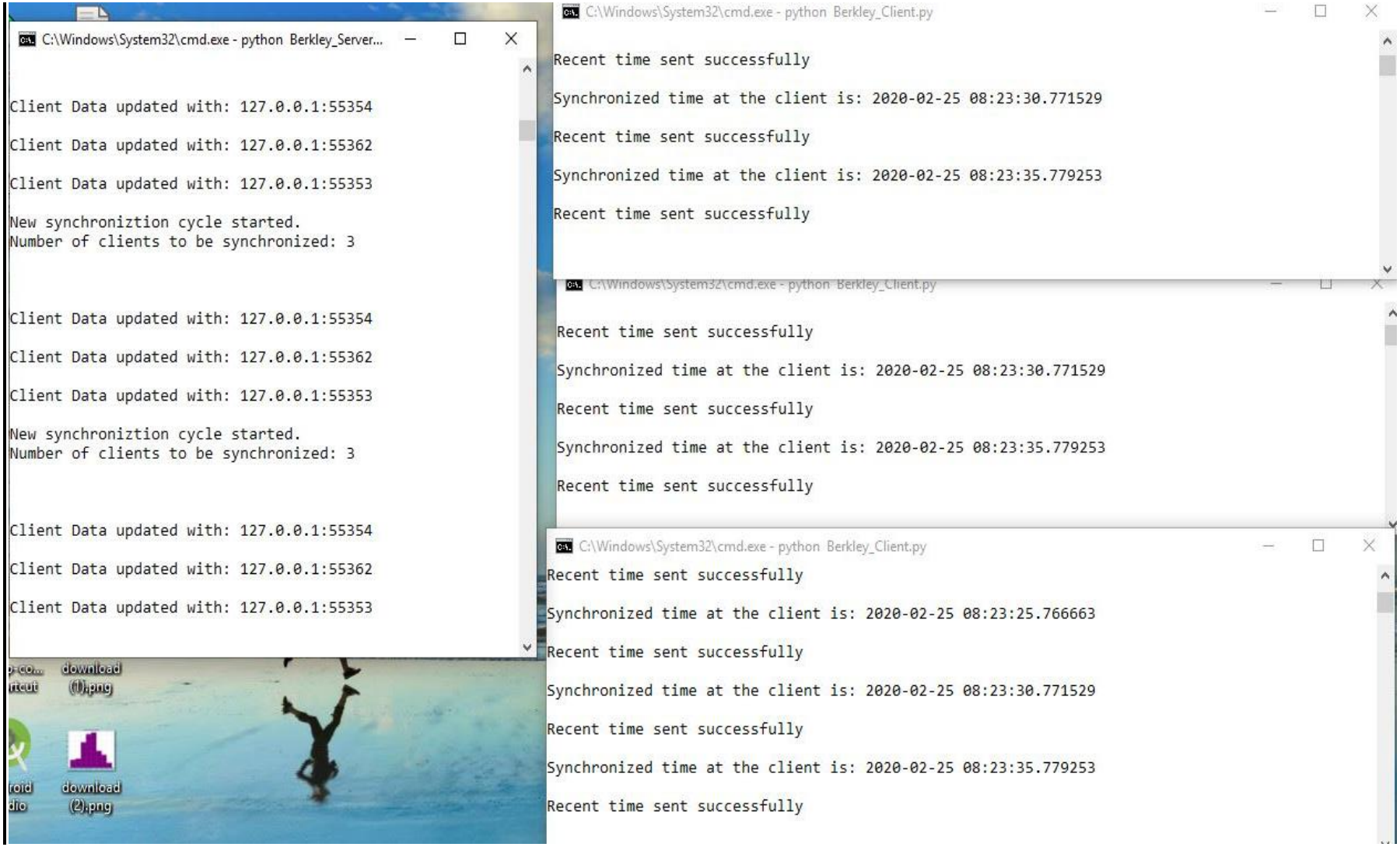
```
                print("Something went wrong while " +"sending synchronized time " + "through " +
str(client_addr))

        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)
```

Berkley_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)
```


C:\Windows\System32\cmd.exe - python Berkley_Server...
Client Data updated with: 127.0.0.1:55354
Client Data updated with: 127.0.0.1:55362
Client Data updated with: 127.0.0.1:55353
New synchroniztion cycle started.
Number of clients to be synchronized: 3
Client Data updated with: 127.0.0.1:55354
Client Data updated with: 127.0.0.1:55362
Client Data updated with: 127.0.0.1:55353
New synchroniztion cycle started.
Number of clients to be synchronized: 3
Client Data updated with: 127.0.0.1:55354
Client Data updated with: 127.0.0.1:55362
Client Data updated with: 127.0.0.1:55353
C:\Windows\System32\cmd.exe - python Berkley_Client.py
Recent time sent successfully
Synchronized time at the client is: 2020-02-25 08:23:30.771529
Recent time sent successfully
Synchronized time at the client is: 2020-02-25 08:23:35.779253
Recent time sent successfully
C:\Windows\System32\cmd.exe - python Berkley_Client.py
Recent time sent successfully
Synchronized time at the client is: 2020-02-25 08:23:30.771529
Recent time sent successfully
Synchronized time at the client is: 2020-02-25 08:23:35.779253
Recent time sent successfully
C:\Windows\System32\cmd.exe - python Berkley_Client.py
Recent time sent successfully
Synchronized time at the client is: 2020-02-25 08:23:25.766663
Recent time sent successfully
Synchronized time at the client is: 2020-02-25 08:23:30.771529
Recent time sent successfully
Synchronized time at the client is: 2020-02-25 08:23:35.779253
Recent time sent successfully
download (1).png
download (2).png