Ayush Goyal
190905522 CSE D 62

Distributed Systems Week 6: Lab 6: Clock Synchronization

Lab Exercises:

1. The Manipal Foodie is a renowned automated food processing outlet known for its tiffin service to students. The various processes involved are food production, filling and packing. Every day more than 3000 orders are received on an average from the students inmanipal. There are total of 4 production lines for orders received from KMC, MIT, TAPMI and SOLS students, each of them has a digital clock which needs to be in synchroniza tion with the master clock. The master clock mounted in the testing lab controls the entire clock system. Design an appropriate solution using Berkeley'salgorithm for the above scenario. Assume that the clocks at the institutes are slave/clients.

Code:

1server.py:

```
from functools import reduce
from dateutil import parser
import threading
import datetime
import socket
import time
client_data = {}

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 = 8080):
    master_server = socket.socket()
    master_server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    print("The Manipal Foodie\n")
    master_server.bind(('', port))
    master_server.listen(10)
    print("Clock server print\n")
    print("Connecitng to production lines...\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 = 8080)
```

1mit.py:

```
from timeit import default_timer as timer
from dateutil import parser
```

```
import threading
import datetime
import socket
import time

def startSendingTime(slave_client):
    while True:
        slave_client.send(str(datetime.datetime.now()).encode())
        print("MIT 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 = 8080):
    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 = 8080)
```

1kmc.py:

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

def startSendingTime(slave_client):
    while True:
        slave_client.send(str(datetime.datetime.now()).encode())
        print("KMC 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 = 8080):
    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 = 8080)
```

1tapmi.py:

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

def startSendingTime(slave_client):
    while True:
        slave_client.send(str(datetime.datetime.now()).encode())
        print("TAPMI 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 = 8080):
    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 = 8080)
```

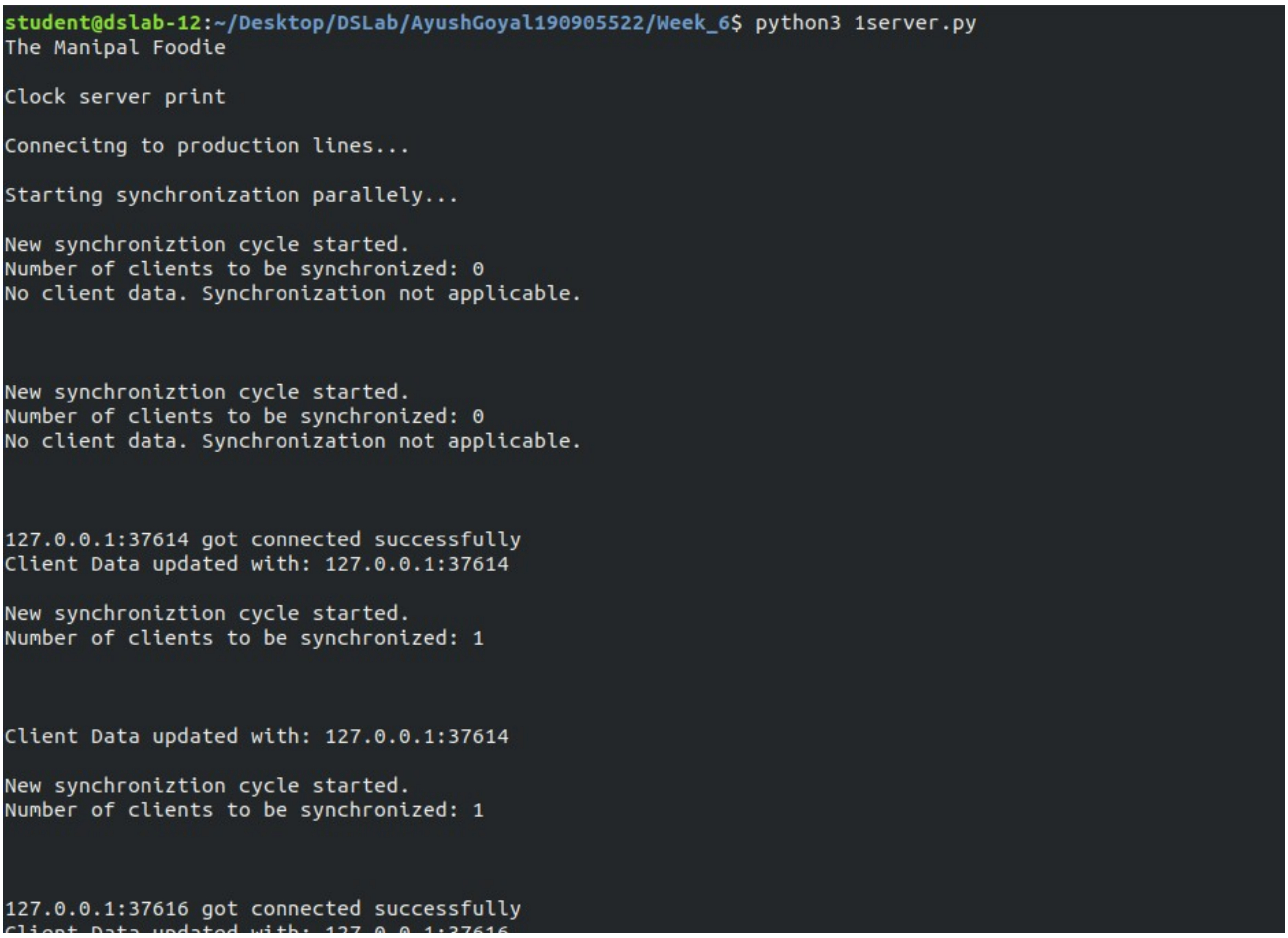
1sols.py:

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

def startSendingTime(slave_client):
    while True:
        slave_client.send(str(datetime.datetime.now()).encode())
        print("SOLS 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 = 8080):
    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 = 8080)
```

OUTPUT:

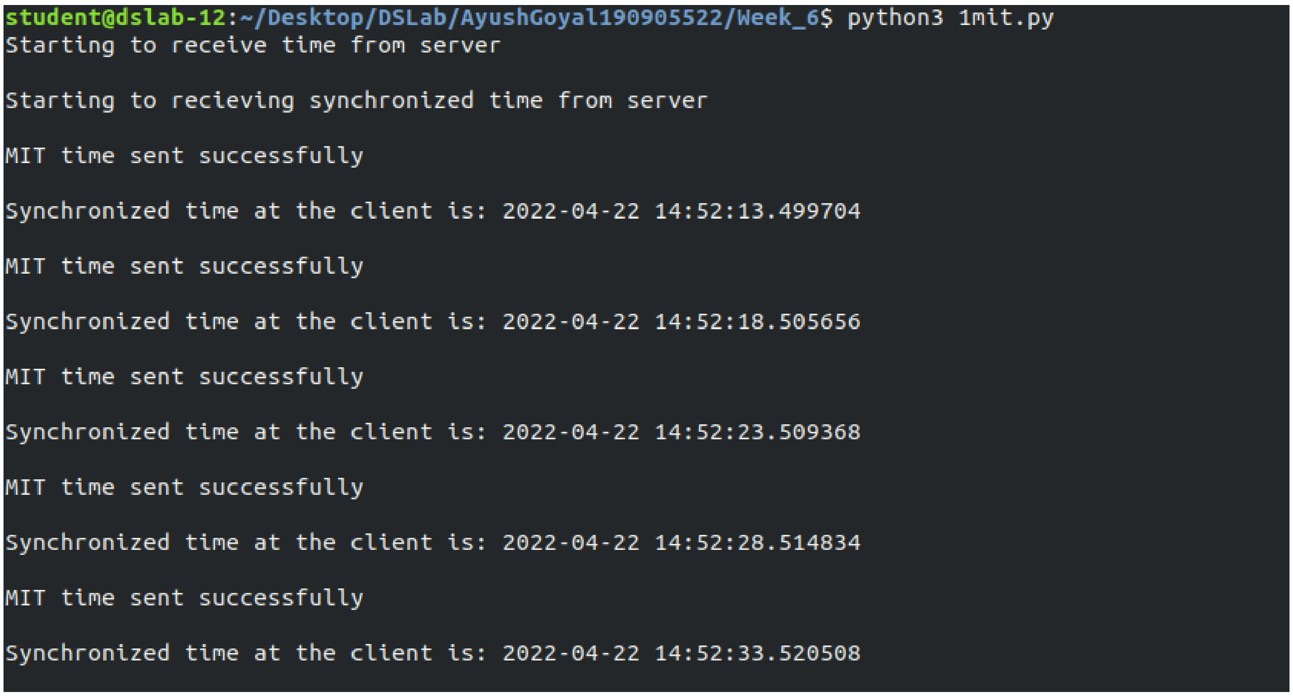
Firstly, we run the server code in one terminal and thus the server will be active. After that we start running each of the four clients one by one.

After running 1server.py:



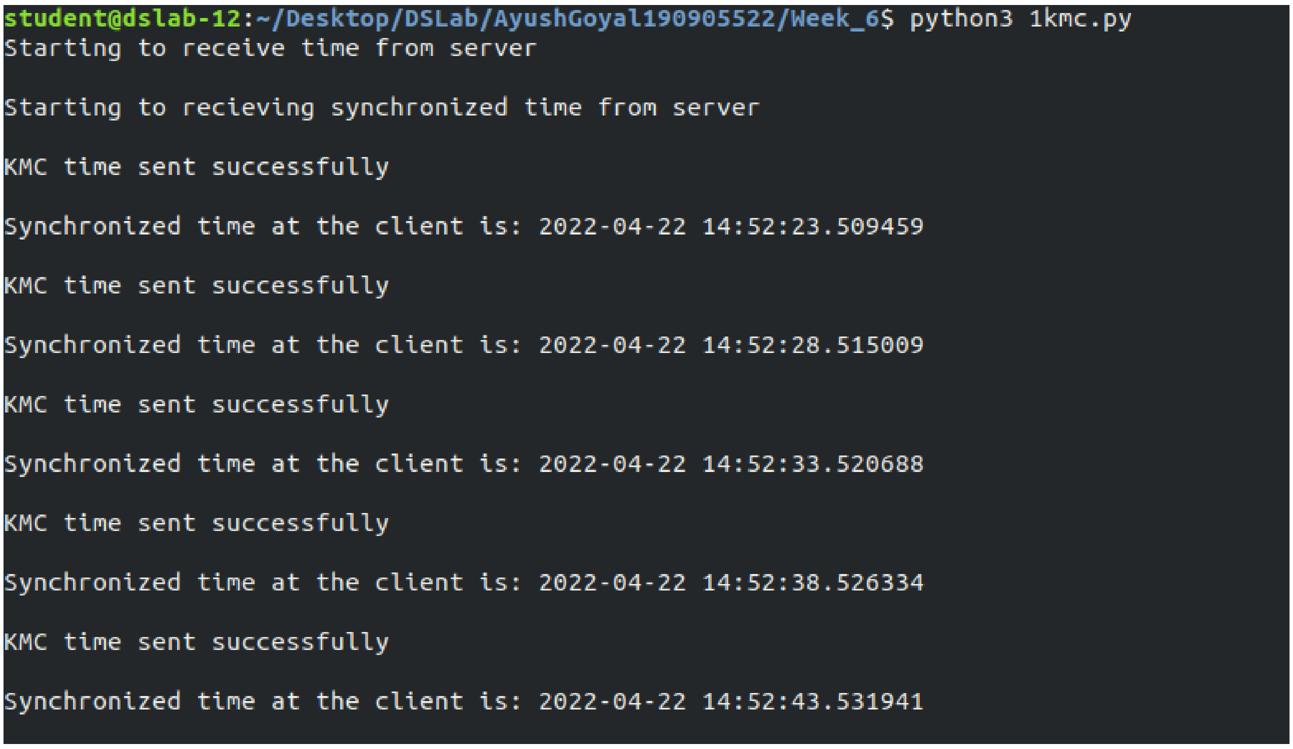
```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$ python3 1server.py
The Manipal Foodie

Clock server print

Connecitng to production lines...

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.



127.0.0.1:37614 got connected successfully
Client Data updated with: 127.0.0.1:37614

New synchroniztion cycle started.
Number of clients to be synchronized: 1



Client Data updated with: 127.0.0.1:37614

New synchroniztion cycle started.
Number of clients to be synchronized: 1



127.0.0.1:37616 got connected successfully
```

After running 1mit.py:



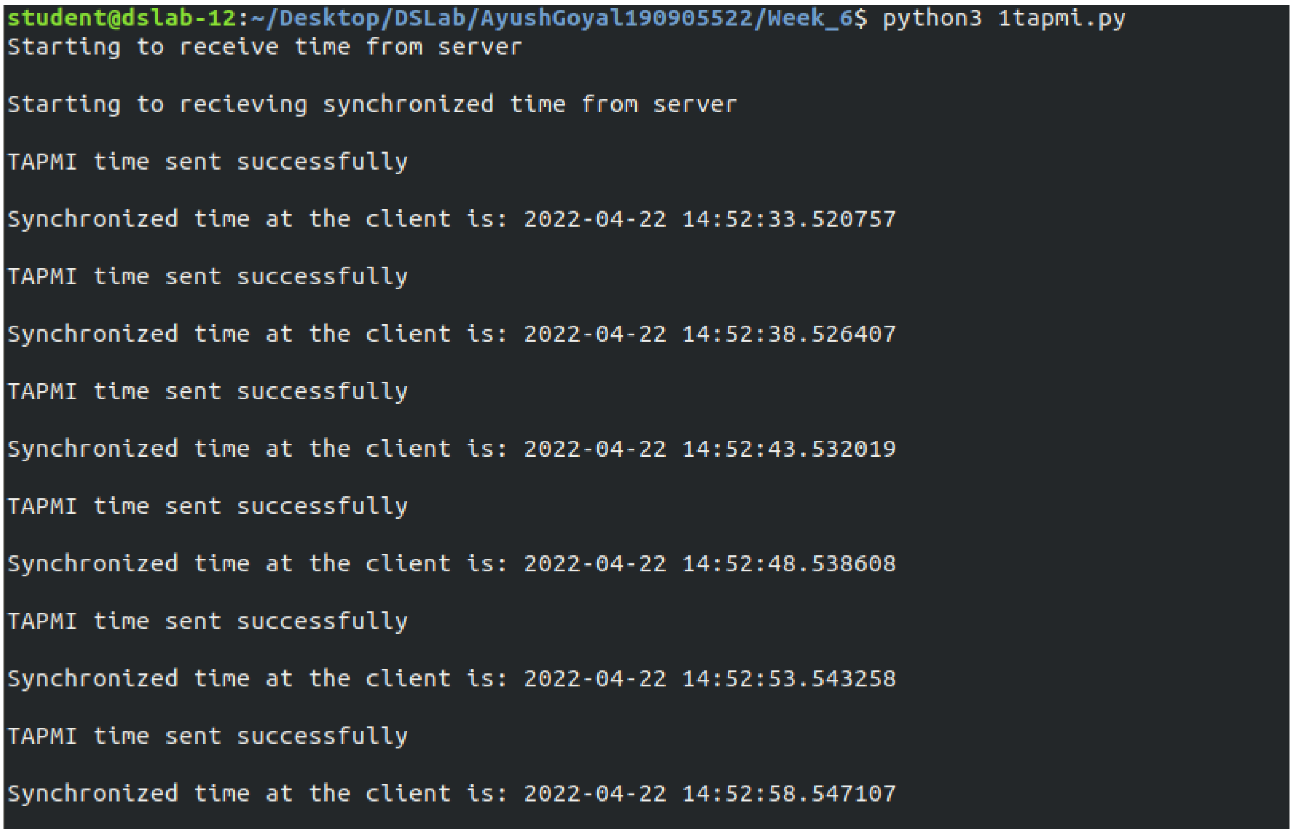
```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$ python3 1mit.py
Starting to receive time from server

Starting to recieving synchronized time from server

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:13.499704

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:18.505656

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:23.509368

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:28.514834

MIT time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:33.520508
```

After running 1kmc.py:



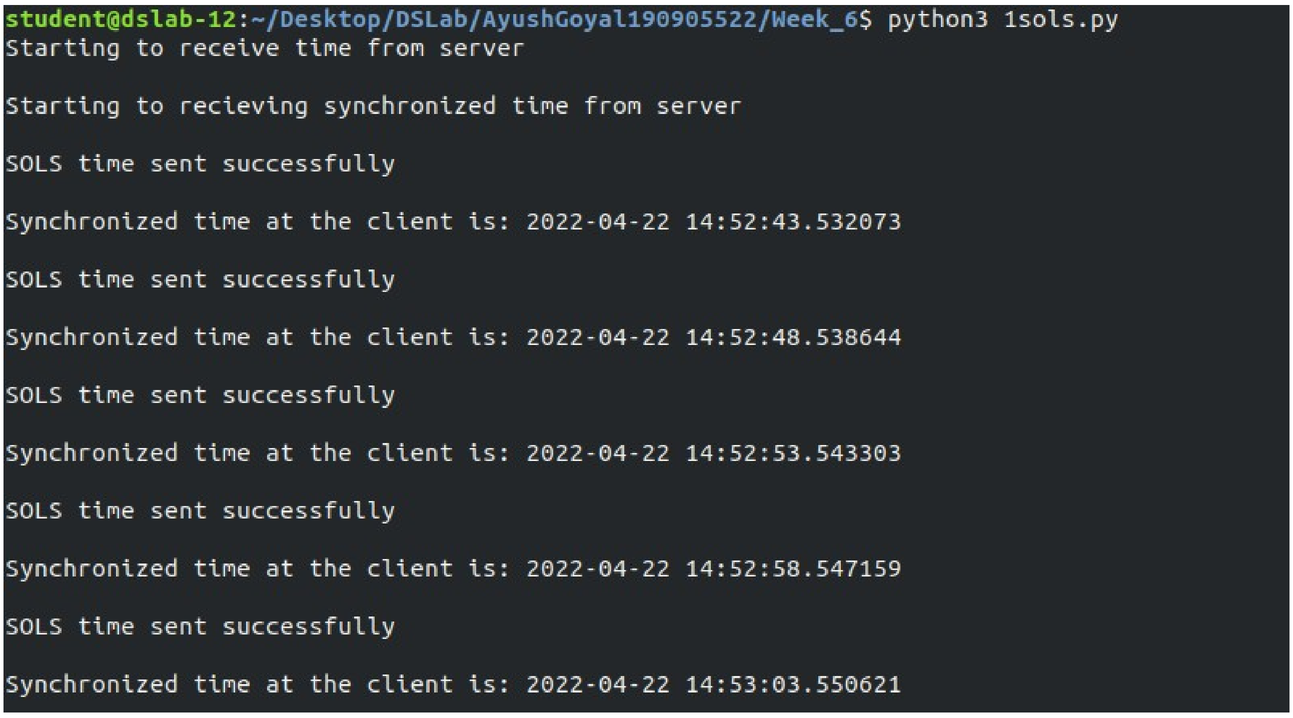
```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$ python3 1kmc.py
Starting to receive time from server

Starting to recieving synchronized time from server

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:23.509459

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:28.515009

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:33.520688

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:38.526334

KMC time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:43.531941
```

After running 1tapmi.py:



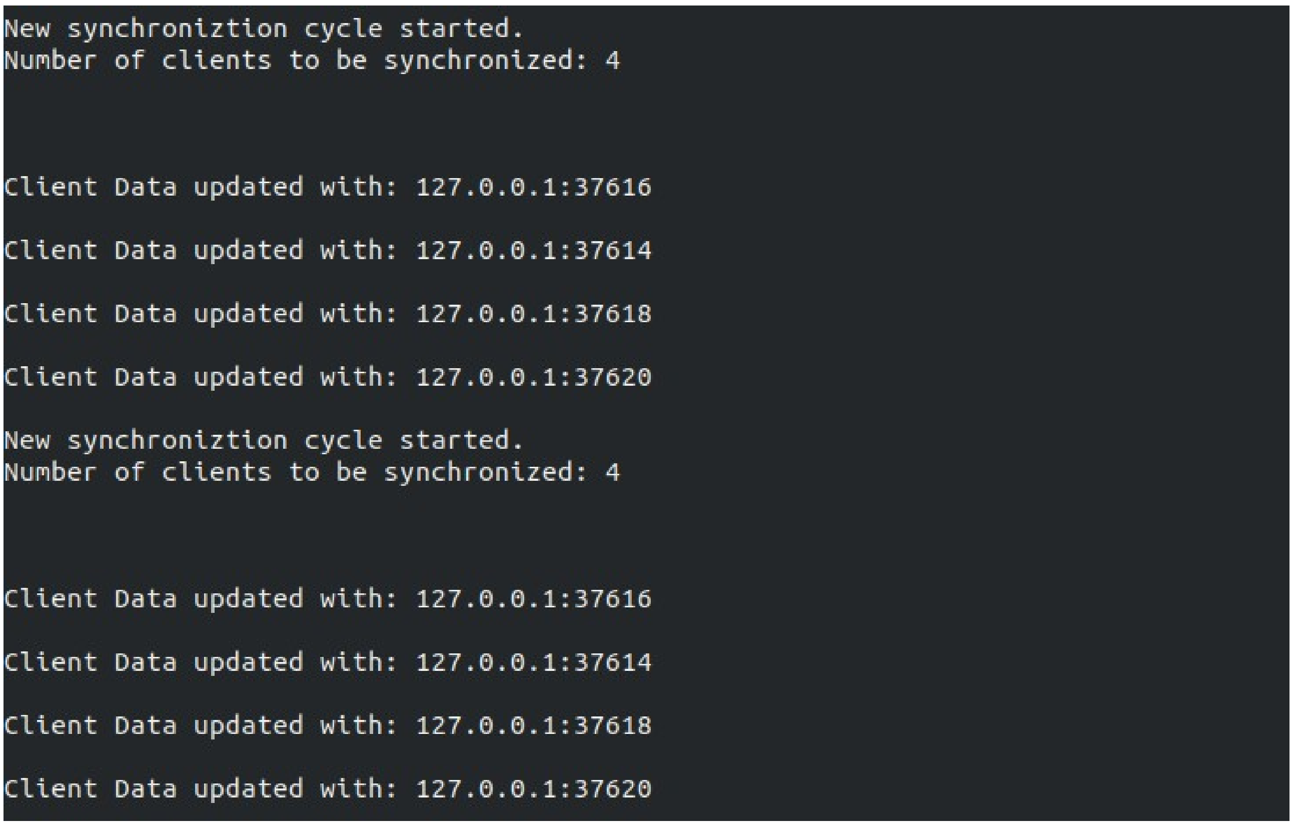
```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$ python3 1tapmi.py
Starting to receive time from server

Starting to recieving synchronized time from server

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:33.520757

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:38.526407

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:43.532019

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:48.538608

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:53.543258

TAPMI time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:58.547107
```

After running 1sols.py:



```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$ python3 1sols.py
Starting to receive time from server

Starting to recieving synchronized time from server

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:43.532073

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:48.538644

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:53.543303

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 14:52:58.547159

SOLS time sent successfully

Synchronized time at the client is: 2022-04-22 14:53:03.550621
```

Finally, we can see the output of 1server.py terminals as:



```
New synchroniztion cycle started.
Number of clients to be synchronized: 4



Client Data updated with: 127.0.0.1:37616

Client Data updated with: 127.0.0.1:37614

Client Data updated with: 127.0.0.1:37618

Client Data updated with: 127.0.0.1:37620

New synchroniztion cycle started.
Number of clients to be synchronized: 4



Client Data updated with: 127.0.0.1:37616

Client Data updated with: 127.0.0.1:37614

Client Data updated with: 127.0.0.1:37618

Client Data updated with: 127.0.0.1:37620
```

2. Manipal Buddy is a banking and education application for the students and staff of MIT, Manipal. Mr Vinay, a sixth semester student wants to pay the end semester exams fees for a re-registered course. He simultaneous ly wishes to register for a course on NPTEL through the app. To register for exam he uses the mobile app whereas to register for NPTEL course he uses his laptop to log in. As he needs to finish both the registrations on the same day, he tries to do both the tasks simultaneous ly. Analyse and demonstrate using a program how Cristian's algorithm can be used in the above case to synchronize the clocks. Assume the relevant parameters.

<u>Code:</u>

2server.py:

```
import socket
import datetime
import time

def initiateClockServer():
    s = socket.socket()
    print("Manipal Buddy Banking")
    port = 8011
    s.bind(('', port))
    s.listen(5)
    print("Waiting for client...")
    while True:
        connection, address = s.accept()
        print('Server connected to', address)
        connection.send(str(datetime.datetime.now()).encode())
        connection.close()

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

2client1-mobileapp-examfees.py:

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

def synchronizeTime():
    print("MOBILE APP\n")
    s = socket.socket()
    port = 8011
    s.connect(('127.0.0.1', port))
    request_time = timer()
    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))
    client_time = server_time + datetime.timedelta(seconds = (process_delay_latency) / 2)
    print("Synchronized process client time: " + str(client_time))
    time.sleep(10)
    s.close()

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

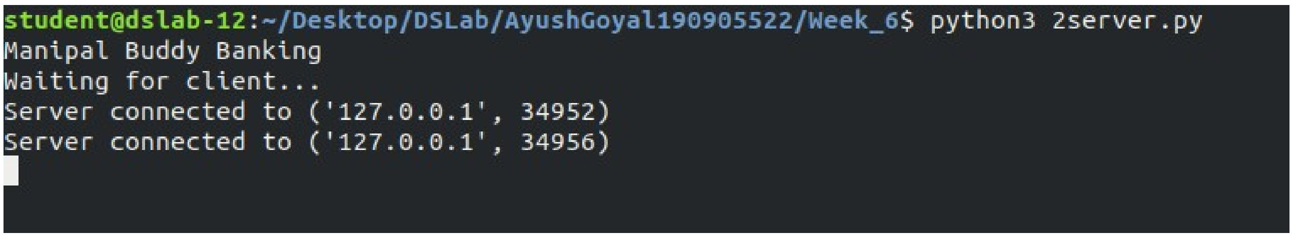
2client2-webbrowser-NPTEL.py:

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

def synchronizeTime():
    print("WEB BROWSER\n")
    s = socket.socket()
    port = 8011
    s.connect(('127.0.0.1', port))
    request_time = timer()
    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))
    client_time = server_time + datetime.timedelta(seconds = (process_delay_latency) / 2)
    print("Synchronized process client time: " + str(client_time))
    time.sleep(10)
    s.close()

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

OUTPUT:

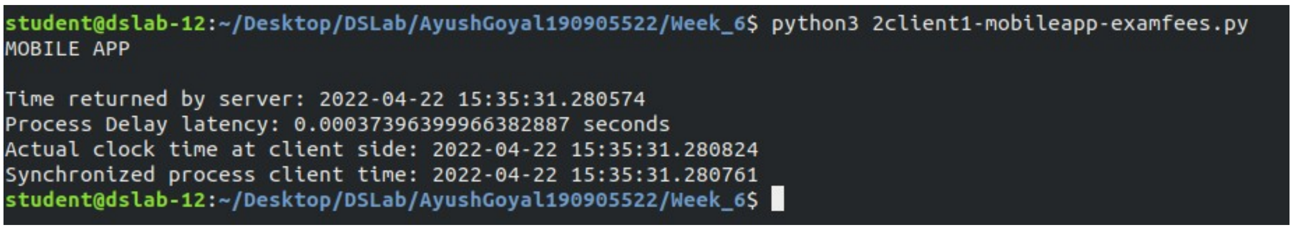
Firstly, we run the server code in one terminal and thus the server will be active. After that we start running the two clients one by one.

After running 2server.py:



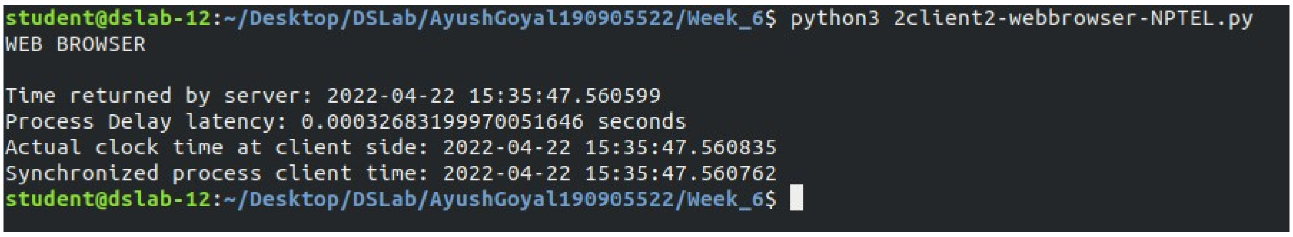
```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$ python3 2server.py
Manipal Buddy Banking
Waiting for client...
Server connected to ('127.0.0.1', 34952)
Server connected to ('127.0.0.1', 34956)
```

After running 2client1-mobileapp-examfees.py:



```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$ python3 2client1-mobileapp-examfees.py
MOBILE APP

Time returned by server: 2022-04-22 15:35:31.280574
Process Delay latency: 0.00037396399966382887 seconds
Actual clock time at client side: 2022-04-22 15:35:31.280824
Synchronized process client time: 2022-04-22 15:35:31.280761
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$
```

After running 2client2-webbrowser-NPTEL.py:



```
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$ python3 2client2-webbrowser-NPTEL.py
WEB BROWSER

Time returned by server: 2022-04-22 15:35:47.560599
Process Delay latency: 0.00032683199970051646 seconds
Actual clock time at client side: 2022-04-22 15:35:47.560835
Synchronized process client time: 2022-04-22 15:35:47.560762
student@dslab-12:~/Desktop/DSLab/AyushGoyal190905522/Week_6$
```

<u>THE END</u>