COMPSCI 578 - PA1 : TSDesign

Team members and Spire IDs:

Ojas Jeetendra Raundale (34029615) Manish Karna (34061932)

We have tackled the problem of time sync using the **NTP algorithm**.
The programs are coded in Python and use the **TCP sockets** to implement this algorithm.

TSServer.py

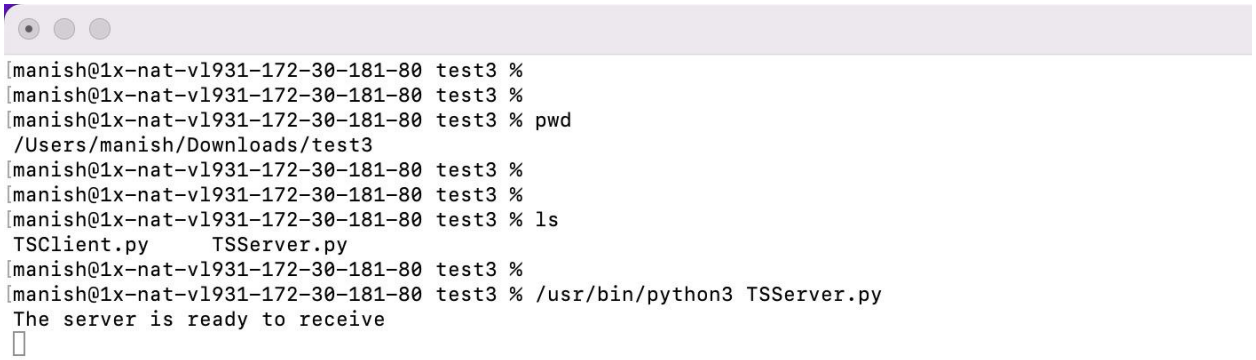
The server program runs on the 11932 port and listens for incoming connections. As soon as a connection is accepted, it launches a thread that is now responsible for dealing with this connection, and the main loop now doesn't have to wait for this connection to end and is open to listening to new incoming connections.
The spawned connection thread captures the "connection-acceptance" Unix timestamp (T2) and "message-send" Unix timestamp (T3) and sends it as a tab-separated message back to the client.
If multiple Clients are trying to connect at the same time, all are assigned a separate thread and dealt with separately.

TSClient.py

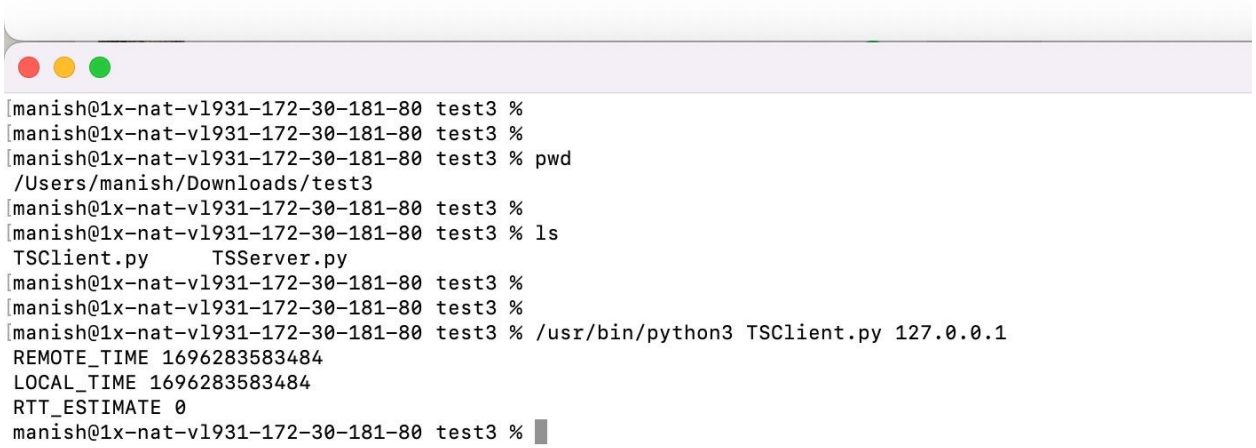
The client program initiates a connection with the server and records the Unix timestamp (T1) just before sending a message to the client. The client does all its shenanigans and sends back the tab-separated T2 and T3. The client records T4, the "response-arrival" Unix timestamp, as soon as the response is received. Then it applies the NTP algorithm and the following formulas to calculate the RTT, and the remote time:

RTT = [(T2 - T1) + (T4 - T3)]/2
Local Time = T4
Remote time = T3 + RTT/2
(Here we are assuming that the Remote time means the Client's estimate of the server's current time. We could have also derived Remote time as T4 - θ but both of them should give the same answer if we assume $d_1 = d_2 = RTT/2$.)

Screenshot of Client and Server on the local machine:

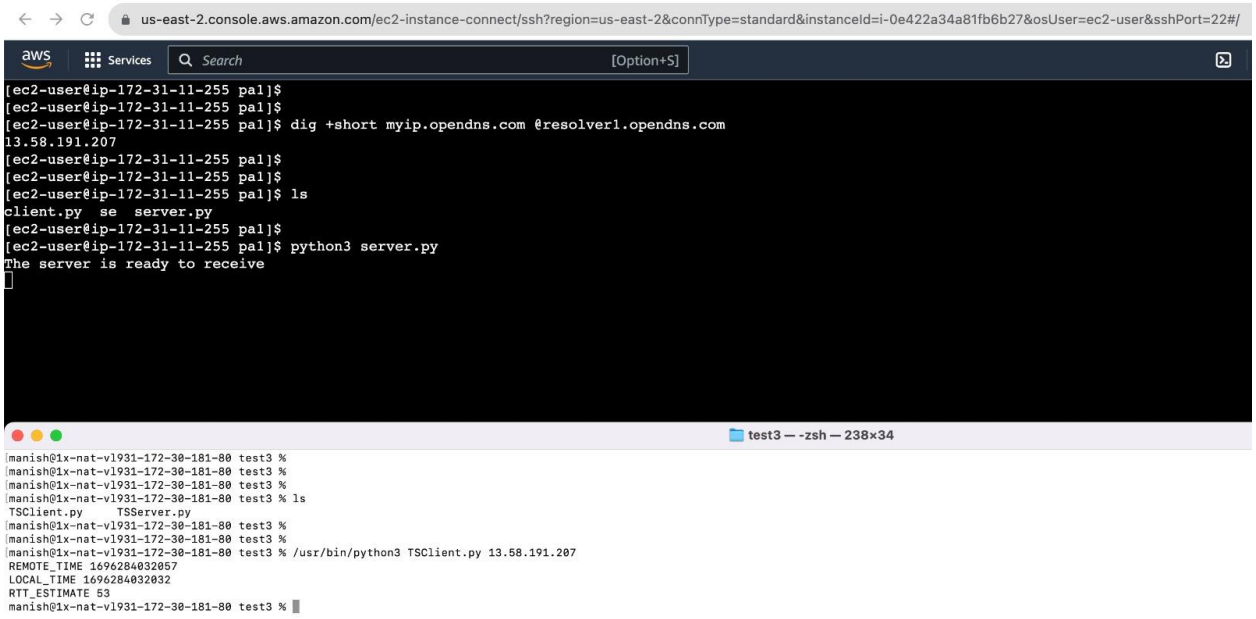


```
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 % pwd
/Users/manish/Downloads/test3
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 % ls
TSClient.py     TSServer.py
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 % /usr/bin/python3 TSServer.py
The server is ready to receive
```



```
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 % pwd
/Users/manish/Downloads/test3
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 % ls
TSClient.py     TSServer.py
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 %
manish@1x-nat-vl931-172-30-181-80 test3 % /usr/bin/python3 TSClient.py 127.0.0.1
REMOTE_TIME 1696283583484
LOCAL_TIME 1696283583484
RTT_ESTIMATE 0
manish@1x-nat-vl931-172-30-181-80 test3 %
```

Screenshot of server running on Amazon EC2 cloud instance and client running on Local Machine:



Screenshot of the server running on Amazon EC2 cloud instance and 2 instances of our clients back to back: