1. Simple probabilistic time synchronization algorithm

Alberto Miguel Diez

a. Stop your NTP client altogether. Explain what you do to stop ntp. Consult the practice that we did in the course practices where we provided you details about how to stop NTP.

With the following command I can check if NTP sync is active on my Linux computer.

```
alberto@alberto-Lenovo:~$ systemctl status systemd-timesyncd
● systemd-timesyncd.service - Network Time Synchronization
     Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled; ve>
     Active: active (running) since Fri 2021-11-26 13:40:54 CET; 9min ago
       Docs: man:systemd-timesyncd.service(8)
   Main PID: 812 (systemd-timesyn)
     Status: "Initial synchronization to time server 91.189.89.199:123 (ntp.ubu>
      Tasks: 2 (limit: 9188)
     Memory: 2.2M
     CGroup: /system.slice/systemd-timesyncd.service
             └─812 /lib/systemd/systemd-timesyncd
```

In the image above it can be check that the synchronization is active, therefore I have to stop it with the following command:

```
alberto@alberto-Lenovo:~$ sudo  systemctl stop systemd-timesyncd
alberto@alberto-Lenovo:~$ sudo systemctl status systemd-timesyncd
● systemd-timesyncd.service - Network Time Synchronization
     Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; enabled; v>
     Active: inactive (dead) since Fri 2021-11-26 13:53:16 CET; 6s ago
       Docs: man:systemd-timesyncd.service(8)
    Process: 812 ExecStart=/lib/systemd/systemd-timesyncd (code=exited, status>
   Main PID: 812 (code=exited, status=0/SUCCESS)
     Status: "Shutting down..."
```

For the rest of this exercise, my computer's automatic NTP synchronization will be disabled. It is important to stop it because throughout the exercise, I am going to change the time of my computer to test the operation of the code that I am going to develop. If the computer synchronizes its time automatically, the correct operation of the code could not be verified.

b. Highlight the Linux commands involved in managing the local clock that you used to perform the tests.

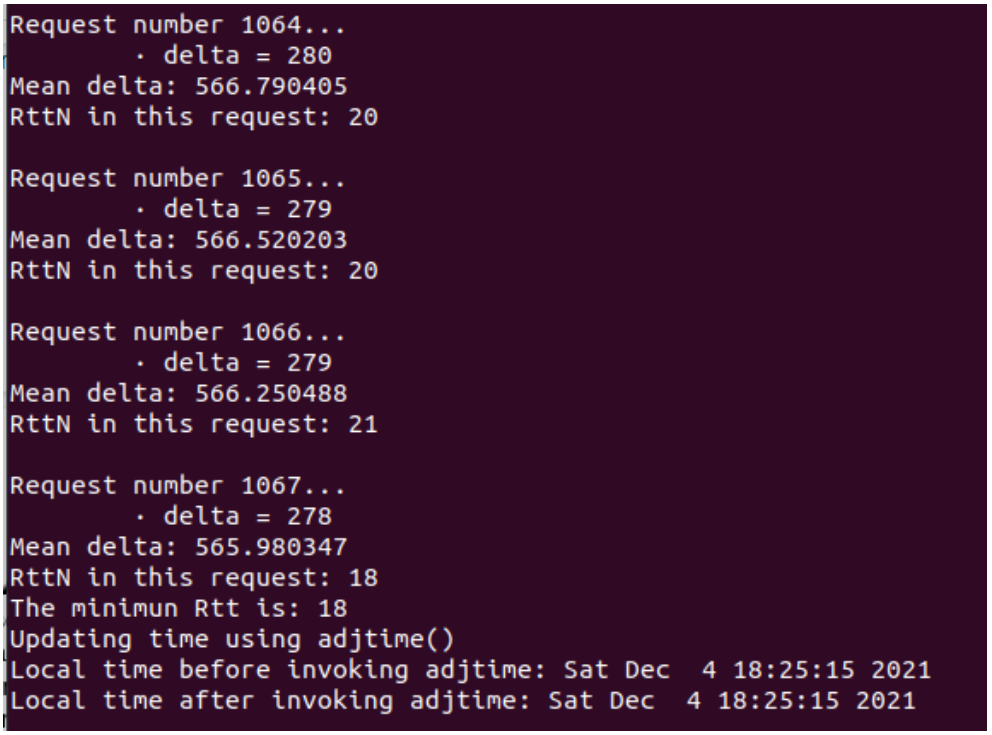
With the command date --rfc-3339 = ns, it can be check the system time with precision of nanoseconds. To do this, it uses the standard described in the NFC 3339 (https://datatracker.ietf.org/doc/html/rfc3339).

```
alberto@alberto-Lenovo:~$  date --rfc-3339=ns
2021-11-26 13:56:19.479513649+01:00
```

With the command *date -s '2021-10-22 09:20:00'* the clock time is changed to a specific day and time. The -s parameter allows setting time described by a string.

c. How long does adjtime() take for reaching a target time that is 5 min forward? Devise an experiment to demonstrate that your results are reasonable.
Instead of doing this experiment with 5 minutes, I'm going to do it with 853 ms so it takes less time to sync the clock. After running the program with *time sudo ./icmptimestamp* it gave me a result of 27 minutes 45 seconds.

Below I show a screenshot of the execution of the program, in which it can be seen the mean delta and a moment in which adjtime () was called.



Execution was stopped when the delta had a value of 0.

d. Explain what tests you will perform to demonstrate that the program functions correctly.
The previous test served to demonstrate the correct operation of the program since the computer's time (which was 853 ms late) was synchronized with the time of the paloalto server. The final delta became 0 in a few moments. In addition, consulting websites such as time.is and comparing with the time on my computer, the difference was negligible.

e. Use paloalto.unileon.es for synchronizing your clock.
To carry out the above tests, the paloalto server was used to synchronize the time found at IP 193.146.101.46. It can be checked in line 292 of the code where the connection with this server is established.

```
285 int main() {
286 
287     unsigned char *packet;
288     int packetLength;
289 
290     pid = getpid();
291 
292     rawSocket = initSocket("193.146.101.46");
293 
294     packet = createPacket(&packetLength);
295 
296     while(1){
297         printf("Request number %d...\n", numberRequest);
298         sendRequest();
299         receiveResponse(packetLength, packet);
300         sleep(1);
301         numberRequest = numberRequest + 1;
302         printf("\n");
303     }
304 
305 
306 } //end of main()
```