Exercise 1 – Simple probabilistic time synchronization algorithm

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.

First we need to see the status of the *systemd-timesyncd* service.

```
{▸ 0} [~] $ sudo service systemd-timesyncd status
● systemd-timesyncd.service - Network Time Synchronization
     Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; disabled; vendor preset: enabled)
     Active: active (running) since Sat 2021-12-04 12:15:54 CET; 3min 26s ago
       Docs: man:systemd-timesyncd.service(8)
   Main PID: 15395 (systemd-timesyn)
     Status: "Initial synchronization to time server 91.189.89.199:123 (ntp.ubuntu.com)."
      Tasks: 2 (limit: 18935)
     Memory: 1.3M
     CGroup: /system.slice/systemd-timesyncd.service
             └─15395 /lib/systemd/systemd-timesyncd

dic 04 12:15:54 samuel systemd[1]: Starting Network Time Synchronization...
dic 04 12:15:54 samuel systemd-timesyncd[15395]: The system is configured to read the RTC time in the loca
dic 04 12:15:54 samuel systemd[1]: Started Network Time Synchronization.
dic 04 12:18:25 samuel systemd-timesyncd[15395]: Initial synchronization to time server 91.189.89.199:123
```

Figure 1: State of systemd-timesyncd service before stopping it

As we can see in the image, it is currently active (if it is inactive, ignore this step). In order to stop it and disable the computer to adjust the time automatically, we must run the following command as **super user**. Later we check that the service is inactive (executing the previous command).

- $ **sudo service systemd-timesyncd stop**

```
{▸ 0} [~] $ sudo service systemd-timesyncd status
● systemd-timesyncd.service - Network Time Synchronization
     Loaded: loaded (/lib/systemd/system/systemd-timesyncd.service; disabled; vendor preset: enabled)
     Active: inactive (dead) since Sat 2021-12-04 12:21:40 CET; 45s ago
       Docs: man:systemd-timesyncd.service(8)
    Process: 15395 ExecStart=/lib/systemd/systemd-timesyncd (code=exited, status=0/SUCCESS)
   Main PID: 15395 (code=exited, status=0/SUCCESS)
     Status: "Shutting down..."

dic 04 12:15:54 samuel systemd[1]: Starting Network Time Synchronization...
dic 04 12:15:54 samuel systemd-timesyncd[15395]: The system is configured to read the RTC time in the loca
dic 04 12:15:54 samuel systemd[1]: Started Network Time Synchronization.
dic 04 12:18:25 samuel systemd-timesyncd[15395]: Initial synchronization to time server 91.189.89.199:123
dic 04 12:21:40 samuel systemd[1]: Stopping Network Time Synchronization...
dic 04 12:21:40 samuel systemd[1]: systemd-timesyncd.service: Succeeded.
dic 04 12:21:40 samuel systemd[1]: Stopped Network Time Synchronization.
```

Figure 2: State of systemd-timesyncd service after stopping it

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

- $ **date -s 'yyyy-mm-dd hh:mm:ss'** → Set a custom date. This is used for running the clock backwards or forwards.
- $ **date --rfc-3339=ns** → View the current date with nanoseconds precision.

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.

In this exercise, I run a test with only a few milliseconds because 5 minutes takes a lot of time to synchronize.

```
{▸ 130} [build] $ time make
------------------------------------------------------------
        -> Delta = 80
        -> Mean delta = 80.000
        -> Current minimum rtt = 2147483647

Adjusting time (new min rtt = 29)
Current time before adjtime Sat Dec  4 13:04:41 2021
Current time after adjtime Sat Dec  4 13:04:41 2021
------------------------------------------------------------
```

Figure 3: Start trace

```
------------------------------------------------------------
        -> Delta = 0
        -> Mean delta = 39.929
        -> Current minimum rtt = 27

make  0,10s user 0,04s system 0% cpu 2:39,56 total
```

Figure 4: Last trace

This example is explained in the following exercise.

Explain what tests you will perform to demonstrate that the program functions correctly.

The time that the program is executing for forwarding the local clock 80ms until synchronization is 2 minutes and 39 seconds (last trace of the previous exercise). The current date is not showed in the second trace because it is only displayed every time a new minimum rtt is obtained.

```
void adjust(const struct timeval delta) {
    print_time("Current time before adjtime");
    adjtime(&delta, (struct timeval *) 0);
    print_time("Current time after adjtime");
}
```

Figure 5: Adjust time

This function is called when a new minimum RTT is obtanied. It receives a time delta to accelerate or decelerate the system's clock. Other functionality I have implemented is to update the RTT after 15 times in which the RTT is not updated (see source code for more precise documentation).