DS Lab 8- Clock Synchronization

Student name: Okonicha Ozioma
Course: SE-01

Some screenshots from the process

Exercise 1. Configure NTP Client to be Time Synced with the NTP Server

```
ubuntu@ip-172-31-29-6:~$ timedatectl
               Local time: Thu 2020-09-24 08:11:05 UTC
           Universal time: Thu 2020-09-24 08:11:05 UTC
                 RTC time: Thu 2020-09-24 08:11:06
                Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
ubuntu@ip-172-31-29-6:~$
```

```
ubuntu@ip-172-31-29-6:~$ sudo timedatectl set-ntp no
```

```
ubuntu@ip-172-31-29-6:~$ timedatectl
               Local time: Thu 2020-09-24 08:21:03 UTC
           Universal time: Thu 2020-09-24 08:21:03 UTC
                 RTC time: Thu 2020-09-24 08:21:04
                Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: inactive
          RTC in local TZ: no
ubuntu@ip-172-31-29-6:~$
```

As seen above after running `sudo timedatectl set-ntp no` NTP service changes from **active** to **inactive**

```
ubuntu@ip-172-31-29-6:~$ sntp --version
sntp 4.2.8p12@1.3728-o (1)
ubuntu@ip-172-31-29-6:~$
```

```
ubuntu@ip-172-31-29-6:~$ sudo service ntp restart
ubuntu@ip-172-31-29-6:~$ sudo service ntp status
● ntp.service - Network Time Service
     Loaded: loaded (/lib/systemd/system/ntp.service; enabled; vendor preset:>
     Active: active (running) since Thu 2020-09-24 09:27:37 UTC; 6s ago
       Docs: man:ntpd(8)
    Process: 28635 ExecStart=/usr/lib/ntp/ntp-systemd-wrapper (code=exited, s>
   Main PID: 28650 (ntpd)
      Tasks: 2 (limit: 1164)
     Memory: 1.2M
     CGroup: /system.slice/ntp.service
             └─28650 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 113:119
```

Q1: *What is a stratum in terms of NTP?*
A1: The Network Time Protocol (NTP) is a hierarchical protocol divided into “stratum levels” that define the distance from a reference clock timing source. So basically a strata indicates the distance between the server and a reference clock

```
ozzie_kins@ozzie-kins-X542UN:~$ sudo ntpdate ec2-54-173-157-255.compute-1.amazonaws.com
24 Sep 12:51:28 ntpdate[951859]: adjust time server 54.173.157.255 offset 0.087381 sec
ozzie_kins@ozzie-kins-X542UN:~$
```

Q2: *Provide the output of ntpq -p command from the client and describe the meaning of the following fields: remote, refid, st, t, when, poll, reach, delay, offset, and jitter.*

A2:

```
ozzie_kins@ozzie-kins-X542UN:~$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 ec2-54-173-157- 163.237.218.19   2 u    2   64    1  279.022   -9.655  20.159
ozzie_kins@ozzie-kins-X542UN:~$
```

- remote: this shows my **instance** itself
- refid: this shows where or what the remote peer or server itself is **synchronized** to.
- st: this column refers to a server's **stratum**, which refers to how close the server is from us (the lower the number, the closer, and typically, better).
- t: is the type, when specified to be u, it specifies a **unicast** or manycast client
- when: here the number of seconds since **last response** is shown
- poll: this is the poll interval, which shows the **interval** at which the ntp client sends ntp packets
- reach: The reach peer variable is an 8-bit shift register displayed in octal format. When a valid packet is received, the rightmost bit is lit. When a packet is sent, the register is shifted left one bit with 0 replacing the rightmost bit. If the reach value is nonzero, the server is reachable; otherwise, it is unreachable. So here our server is **reachable**
- delay: this delay is the **round-trip** delay to peer is reported in milliseconds.
- offset: as we already now, this is the time **difference** between the peers or between the server and client.
- jitter: this column refers to the network **latency** between your server and theirs.

Exercise 2. Logical Clock Implementation

Q3: *What are the lacks of using the Lamport's algorithm?*

A3: People use physical time to order events. For example, we say that an event at 8:15 AM occurs before an event at 8:16 AM. In distributed systems, physical clocks are not always precise, so we can't rely on physical time to order events. Instead, we can use logical clocks to create a partial or total ordering of events.

The Lamport timestamp algorithm is a simple logical clock algorithm used to determine the order of events in a distributed computer system. Within a distributed system in particular we want to determine if an event at one node affects or causes an event at another node, but when a system is distributed, there is no global clock.

However, some disadvantages are:

- clocks are entirely under the control of the logical-clock protocol and may as a result make huge jumps when a message is received.
- the system may encounter anomalous behaviors as logical clocks do not consider external events to part of their event sets.
- if one process fails, the entire system's systems progress stops.
- there is now a high message complexity
- the algorithm can't tell you when two events are concurrent

The final vector state of each process.

```
ozzie_kins@ozzie-kins-X542UN:~/Documents/3rd year 1st semester block_one/Distributed Systems$ python3 vector.py
Process a [7, 6, 1]
Process b [2, 8, 1]
Process c [2, 8, 4]
ozzie_kins@ozzie-kins-X542UN:~/Documents/3rd year 1st semester block_one/Distributed Systems$
```

Link to GitHub repo or gist with source code.

lab8