Lab 8: Clock Synchronization

Clock synchronization is a topic in computer science and engineering that aims to coordinate independent clocks. Even when initially set accurately, real clocks will differ after some amount of time due to clock drift, caused by clocks counting time at slightly different rates.

In a distributed system the problem takes on more complexity because a global time is not easily known. The most used Physical clock synchronization solution on the Internet is the Network Time Protocol (NTP) which is a layered client-server architecture based on UDP message passing.

Exercise 1. Physical Clocks. NTP Implementation

NTP or Network Time Protocol is a protocol that is used to synchronize all system clocks in a network to use the same time. When we use the term NTP, we are referring to the protocol itself and also the client and server programs running on the networked computers.

Exercise 1.1. Launch an Amazon EC2 Instance

Launch Amazon EC2 instance with Ubuntu Server 18.04 LTS and connect to it from your host machine using ssh.

Exercise 1.2. Install and configure NTP Server on the EC2 Instance

Until recently, most network time synchronization was handled by the *Network Time Protocol daemon* or *ntpd*. This service connects to a pool of other NTP servers that provide it with constant and accurate time updates.

Ubuntu's default install now uses *timesyncd* instead of *ntpd*. *timesyncd* connects to the same time servers and works in roughly the same way, but is more lightweight and more integrated with *systemd* and the low level workings of Ubuntu.

1. Query the status of *timesyncd* by running `timedatectl` with no arguments. You don't need to use sudo in this case:

   ```
   $ timedatectl
                         Local time: Mon 2019-10-07 19:03:48 UTC
                     Universal time: Mon 2019-10-07 19:03:48 UTC
                           RTC time: Mon 2019-10-07 19:03:49
                          Time zone: Etc/UTC (UTC, +0000)
          System clock synchronized: yes
   systemd-timesyncd.service active: yes
                    RTC in local TZ: no
   ```

   This prints out the *local time*, *universal time* (which may be the same as local time, if you didn't switch from the UTC time zone), and some network time status information. `System clock synchronized: yes` indicates that the time has been successfully synced, and `systemd-timesyncd.service active: yes` means that *timesyncd* is enabled and running.

   Though timesyncd is fine for most purposes, some applications that are very sensitive to even the slightest perturbations in time may be better served by *ntpd*, as it uses more sophisticated techniques to constantly and gradually keep the system time on track.

2. Before installing *ntpd*, we should turn off *timesyncd*:

   ```
   $ sudo timedatectl set-ntp no
   ```

3. Verify that timesyncd is off:

   ```
   $ timedatectl
   ```

4. Look for `systemd-timesyncd.service active: no` in the output. This means *timesyncd* has been stopped.
   We can now install the ntp package with apt.
5. Update repository index
   In order to install the latest available version of software from the Internet repositories, repository index needs to be in line with them. Run the following command as sudo in order to update your local repository index:

   ```
   $ sudo apt update
   ```

6. Install NTP Server
   Please run the following command as sudo in order to install NTP server daemon from the APT repositories:

   ```
   $ sudo apt install ntp
   ```

7. The system might provide you with a *Y/n* option to continue the installation. Enter *Y* and then hit enter; NTP server will then be installed on your system. The process may, however, take some time depending on your Internet speed.
8. Verify your NTP installation and also check the version number by running the following command in your Terminal:

   ```
   $ sntp --version
   ```

9. When you install the NTP server, it is mostly configured to fetch proper time. However, you can switch any server pool. This includes making some changes in the `/etc/ntp.conf` file.
   Open the file in the nano editor as sudo by running this following command:

   ```
   $ sudo nano /etc/ntp.conf
   ```

10. In this file, you will be able to see a pool list. Replace this pool list by a pool of time servers located in Russian Federation. To choose a pool list, visit support.ntp.org.
11. Quit the file by hitting `Ctrl+X` and then by entering `y` for saving changes.
12. Restart the NTP server
    In order for the above changes to take effect, you need to restart the NTP server. Run the following command as sudo in order to do so:

    ```
    $ sudo service ntp restart
    ```

13. Verify that the NTP Server is running

    ```
    $ sudo service ntp status
    ```

    The `Active` status verifies that your NTP server is up and running.

14. Configure Firewall so that client(s) can access NTP server
    It is time to configure your system's UFW firewall so that incoming connections can access the NTP server at UDP Port number 123. Run the following command as sudo to open port 123 for incoming traffic:

    ```
    $ sudo ufw allow from any to any port 123 proto udp
    ```

15. Configure Network Security Group of your EC2 Instance to allow inbound traffic for UDP Port number 123.

Your EC2 Instance machine is now configured to be used as an NTP server.

Q1: What is a stratum in terms of NTP?

Exercise 1.3. Configure NTP Client to be Time Synced with the NTP Server
Let us now configure our client host machine to be time synchronized with the NTP server.

1. Install ntpdate
   The ntpdate command will let you manually check your connection configuration with the NTP-server. Open the Terminal application on your host machine and enter the following command as sudo:

   ```
   $ sudo apt install ntpdate
   ```

2. Check if the host machine's time is synchronized with NTP server
   The following ntpdate command will let you manually check if time is synchronized between the client and server systems:

   ```
   $ sudo ntpdate <aws-public-dns-ntp-server>
   ```

   The output should ideally show a time offset between the two systems.

3. Disable the systemd timesyncd service on the host
   Because we want our client to sync time with the NTP server, let us disable the *timesyncd* service on the host machine. Enter the following command to do so:

   ```
   $ sudo timedatectl set-ntp no
   ```

4. Install NTP on your host machine
   Run the following command as sudo in order to install NTP on your host machine:

   ```
   $ sudo apt install ntp
   ```

5. Configure the `/etc/ntp.conf` file to add your NTP server as the new time server
   Now we want our host machine to use our own NTP host server to be used as the default time server. For this, we need to edit the `/etc/ntp.conf` file on the host machine.
   Run the following command as sudo in order to open the file in the Nano editor:

   ```
   $ sudo nano /etc/ntp.conf
   ```

6. Comment everything and add the following line in the file:

   ```
   server <aws-public-dns-ntp-server> prefer iburst
   ```

   where `<aws-public-dns-ntp-server>` is the hostname you specified for your Amazon NTP server
7. Hit `Ctrl+x` in order to quit the file and then enter y to save the changes.
8. Restart the NTP server
   In order for the above changes to take effect, you need to restart the NTP service. Run the following command as sudo in order to do so:

   ```
   $ sudo service ntp restart
   ```

9. View the Time Synchronization Queue
   Now your client and server machines are configured to be time-synced. You can view the time synchronization queue by running the following command:

   ```
   $ ntpq -p
   ```

10. You should be able to see NTP server as the time synchronization host/source in the queue.

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

So this was all you needed to know about installing and configuring NTP to synchronize time on your networked machines. The process may seem a little cumbersome but if you follow all of the above steps carefully, one-by-one, your machines will be synced in no time.

Exercise 2. Logical Clock Implementation

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 first implementation of logical clocks was proposed by Leslie Lamport in 1978 and is known as the Lamport timestamps. Vector clocks are also concepts of the logical clock in distributed systems.

To understand how to implement logical clock algorithms using Python's multiprocessing library for the situation shown in Figure 1, study this article.

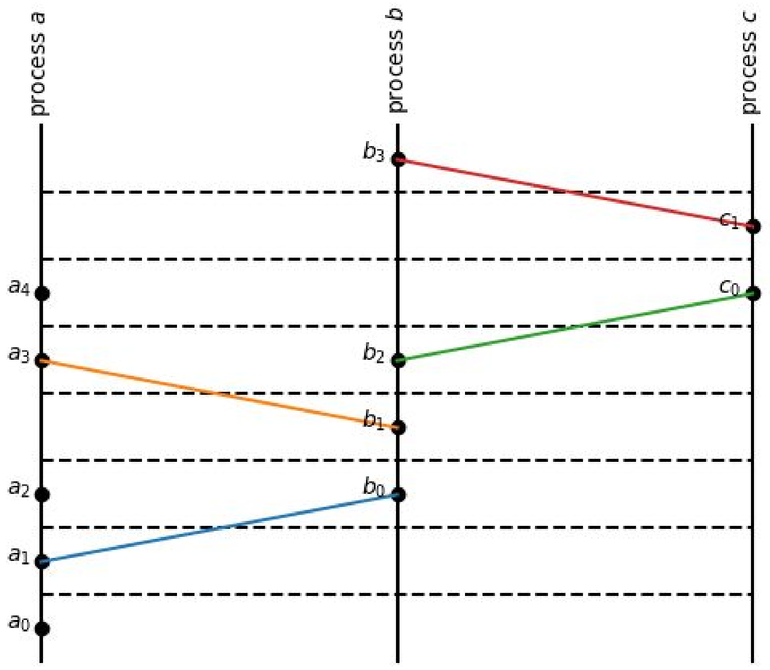

Figure 1. Three processes, each with its own logical clock.

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

Assignment

Implement logical vector clock algorithm using Process and Pipe approaches on any programming language for the situation shown in Figure 2.

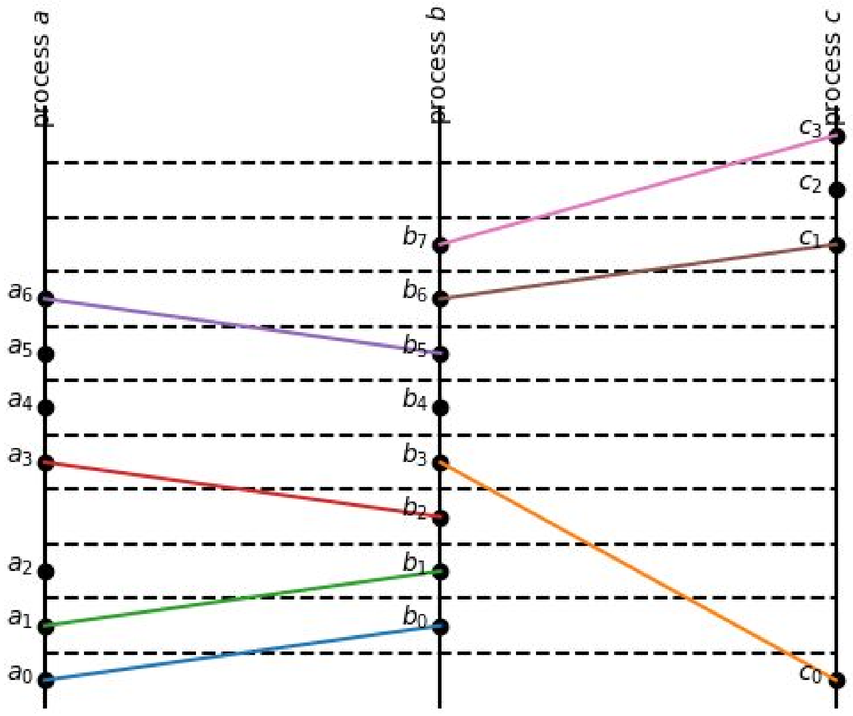

Figure 2. Three processes, each with its own vector logical clock

In Moodle, submit the report in **pdf** format and include the following sections in it:

1. Answers on questions
2. The final vector state of each process.
   Example format:
   - Process a [5, 2, 0]
   - Process b [2, 4, 2]
   - Process c [2, 3, 2]
3. Link to the GitHub repo or Gist with source code of your vector clock implementation

Do not forget to clean up AWS test resources when you're done using them. Delete ntpd and turn on timesyncd on your host machine.