Assignment No. 4

Aim: Implement Berkeley algorithm for clock synchronization.

Objective: To understand Berkeley algorithm for clock synchronization and its implementation.

Infrastructure: Python environment.

Software Requirements: Python 3.0.

Theory:

Berkeley Algorithm

Berkeley's Algorithm is a distributed algorithm for computing the correct time in a network of computers. The algorithm is designed to work in a network where clocks may be running at slightly different rates, and some computers may experience intermittent communication failures.

The basic idea behind Berkeley's Algorithm is that each computer in the network periodically sends its local time to a designated "master" computer, which then computes the correct time for the network based on the received timestamps. The master computer then sends the correct time back to all the computers in the network, and each computer sets its clock to the received time.

There are several variations of Berkeley's Algorithm that have been proposed, but a common version of the algorithm is as follows –

- Each computer starts with its own local time, and periodically sends its time to the master computer.
- The master computer receives timestamps from all the computers in the network.
- The master computer computes the average time of all the timestamps it has received and sends the average time back to all the computers in the network.
- Each computer sets its clock to the time it receives from the master computer.
- The process is repeated periodically, so that over time, the clocks of all the computers in the network will converge to the correct time.

Benefit: It is relatively simple to implement and understand.
Limitation: The time computed by the algorithm is based on the network conditions and time of sending and receiving timestamps which can't be very accurate and also it has a requirement of a master computer which if failed can cause the algorithm to stop working.

More advance algorithm such as the Network Time Protocol (NTP) which use a more complex algorithm and also consider the network delay and clock drift to get a more accurate time.

Scope of Improvement

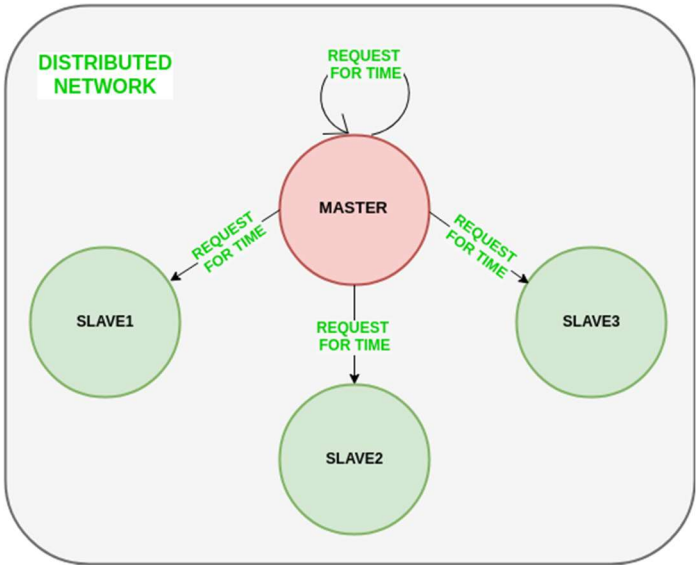
There are several areas where Berkeley's Algorithm can be improved –

- **Accuracy** – The algorithm calculates the average time based on the timestamps received from all the computers in the network, which can lead to inaccuracies, especially if the network has a high degree of jitter or delay.
- **Robustness** – The algorithm requires a master computer, which if it fails, can cause the algorithm to stop working. If the master computer is a single point of failure, it can make the algorithm less robust.
- **Synchronization Quality** – The algorithm assumes that all the clocks in the network are running at the same rate, but in practice, clocks may drift due to temperature, aging, or other factors. The algorithm doesn't consider this drift and may fail to achieve a high degree of synchronization between the clocks in the network.
- **Security** – There is no security measures in the algorithm to prevent malicious computers from tampering with the timestamps they send to the master computer, which can skew the results of the algorithm.
- **Adaptability** – The algorithm doesn't adapt well to changes in the network, for example, if a new computer is added to the network or if the network topology changes.

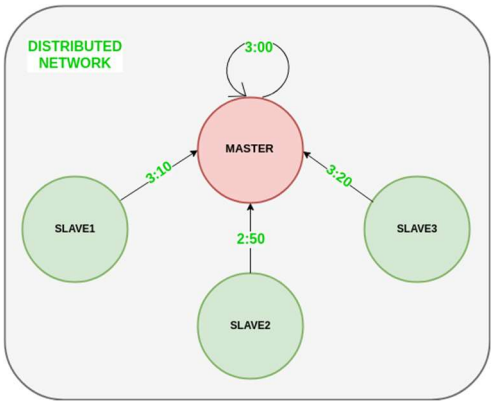
How to use Berkeley's Algorithm

To use Berkeley's Algorithm, you would need to implement the algorithm on each computer in a network of computers. Here is a general overview of the steps you would need to take to implement the algorithm –

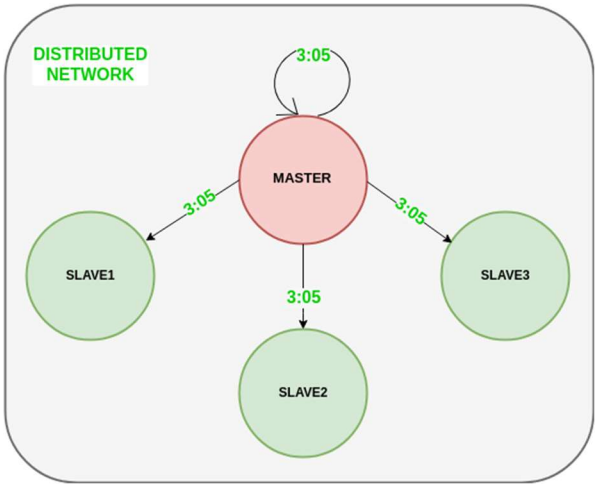
1. Designate one computer in the network as the master computer. This computer will be responsible for receiving timestamps from all the other computers in the network and computing the correct time. The master node is chosen using an election process/leader election algorithm.
2. On each computer in the network, set up a timer to periodically send the computer's local time to the master computer.



3. On the master computer, set up a mechanism for receiving timestamps from all the computers in the network.



4. On the master computer, implement the logic for calculating the average time based on the received timestamps.
5. On the master computer, set up a mechanism for sending the calculated average time back to all the computers in the network.
6. On each computer in the network, set up a mechanism for receiving the time from the master computer and setting the computer's clock to that time.



7. Repeat the process periodically, for example, every 30 seconds or 1 minute, to ensure that the clocks in the network stay synchronized.

Conclusion:
We learnt about Berkeley's algorithm for clock synchronization and its implementation.