ASSIGNMENT No.4

TITLE: Implement Berkeley algorithm for clock synchronization

Tools / Environment:

Java Programming Environment, jdk 1.8, Eclipse IDE.

Related Theory:

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.

One benefit of Berkeley's Algorithm is that it is relatively simple to implement and understand. However, It has a limitation that 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.

There is other 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.

As you see, Berkeley algorithm, despite its simplicity, has some limitations. Network Time Protocol (NTP) is widely used in the industry to overcome some of the limitations of Berkeley's Algorithm. NTP uses a more complex algorithm and also considers the network delay and clock drift to get a more accurate time. Additionally, NTP uses a hierarchical structure, which makes it more robust and adaptable to changes in the network. NTP also includes cryptographic mechanisms to authenticate the time messages and protect against malicious attacks.

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 −

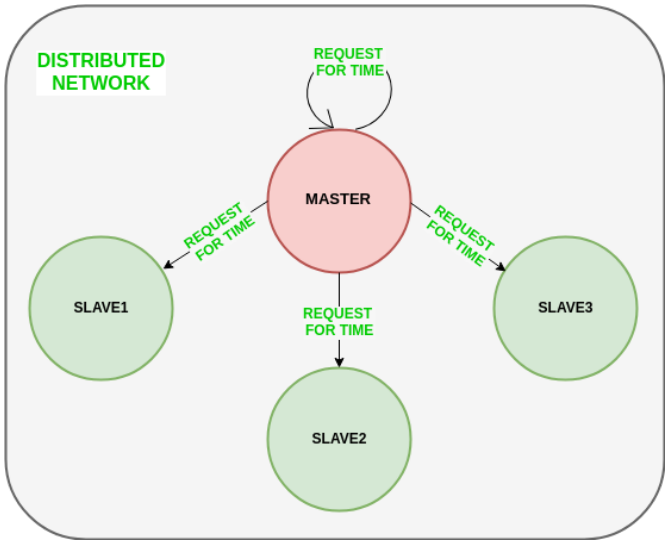
- Designate one computer in the network as the master computer. This computer will be responsible for receiving timestamps from all the other computers in the network and computing the correct time.
- On each computer in the network, set up a timer to periodically send the computer's local time to the master computer.
- On the master computer, set up a mechanism for receiving timestamps from all the computers in the network.
- On the master computer, implement the logic for calculating the average time based on the received timestamps.
- On the master computer, set up a mechanism for sending the calculated average time back to all the computers in the network.
- 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.
- Repeat the process periodically, for example, every 30 seconds or 1 minute, to ensure that the clocks in the network stay synchronized.

It's worth noting that this is a high-level description of the steps to implement the algorithm and in practice, many implementation details will depend on the programming language, operating system, and network infrastructure you are using. Additionally, as explained before, Berkeley Algorithm have some limitations, If you need to have a more accurate and robust solution, you may consider using other time synchronization protocols like NTP which have been designed to overcome some of the limitations of Berkeley's algorithm.

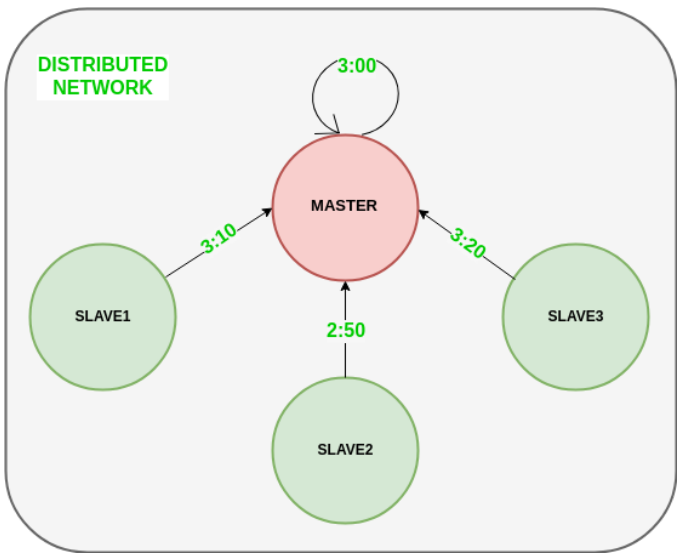
Algorithm

1) An individual node is chosen as the master node from a pool node in the network. This

node is the main node in the network which acts as a master and the rest of the nodes act as slaves. The master node is chosen using an election process/leader election algorithm.
2) Master node periodically pings slaves nodes and fetches clock time at them using Cristian's algorithm.
The diagram below illustrates how the master sends requests to slave nodes.

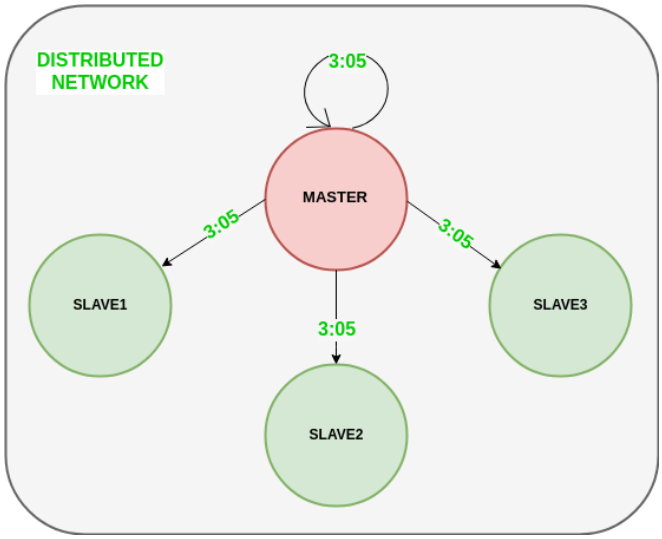


The diagram below illustrates how slave nodes send back time given by their system clock.



3) Master node calculates the average time difference between all the clock times received and the clock time given by the master's system clock itself. This average time difference is added to the current time at the master's system clock and broadcasted over the network.
Pseudocode for above step:

The diagram below illustrates the last step of Berkeley's algorithm.



Scope of Improvement

- Improvision inaccuracy of Cristian's algorithm.
- Ignoring significant outliers in the calculation of average time difference
- In case the master node fails/corrupts, a secondary leader must be ready/pre-chosen to take the place of the master node to reduce downtime caused due to the master's unavailability.
- Instead of sending the synchronized time, master broadcasts relative inverse time difference, which leads to a decrease in latency induced by traversal time in the network while the time of calculation at slave node.

features of Berkeley's Algorithm:

- **Centralized time coordinator:** Berkeley's Algorithm uses a centralized time coordinator, which is responsible for maintaining the global time and distributing it to all the client machines.
- **Clock adjustment:** The algorithm adjusts the clock of each client machine based on the difference between its local time and the time received from the time coordinator.
- **Average calculation:** The algorithm calculates the average time difference between the client machines and the time coordinator to reduce the effect of any clock drift.
- **Fault tolerance:** Berkeley's Algorithm is fault-tolerant, as it can handle failures in the network or the time coordinator by using backup time coordinators.
- **Accuracy:** The algorithm provides accurate time synchronization across all the client machines, reducing the chances of errors due to time discrepancies.
- **Scalability:** The algorithm is scalable, as it can handle a large number of client machines, and the time coordinator can be easily replicated to provide high availability.
- **Security:** Berkeley's Algorithm provides security mechanisms such as authentication and encryption to protect the time information from unauthorized access or tampering.

Conclusion: Hence, we have studied and implemented Berkeley algorithm for clock synchronization.