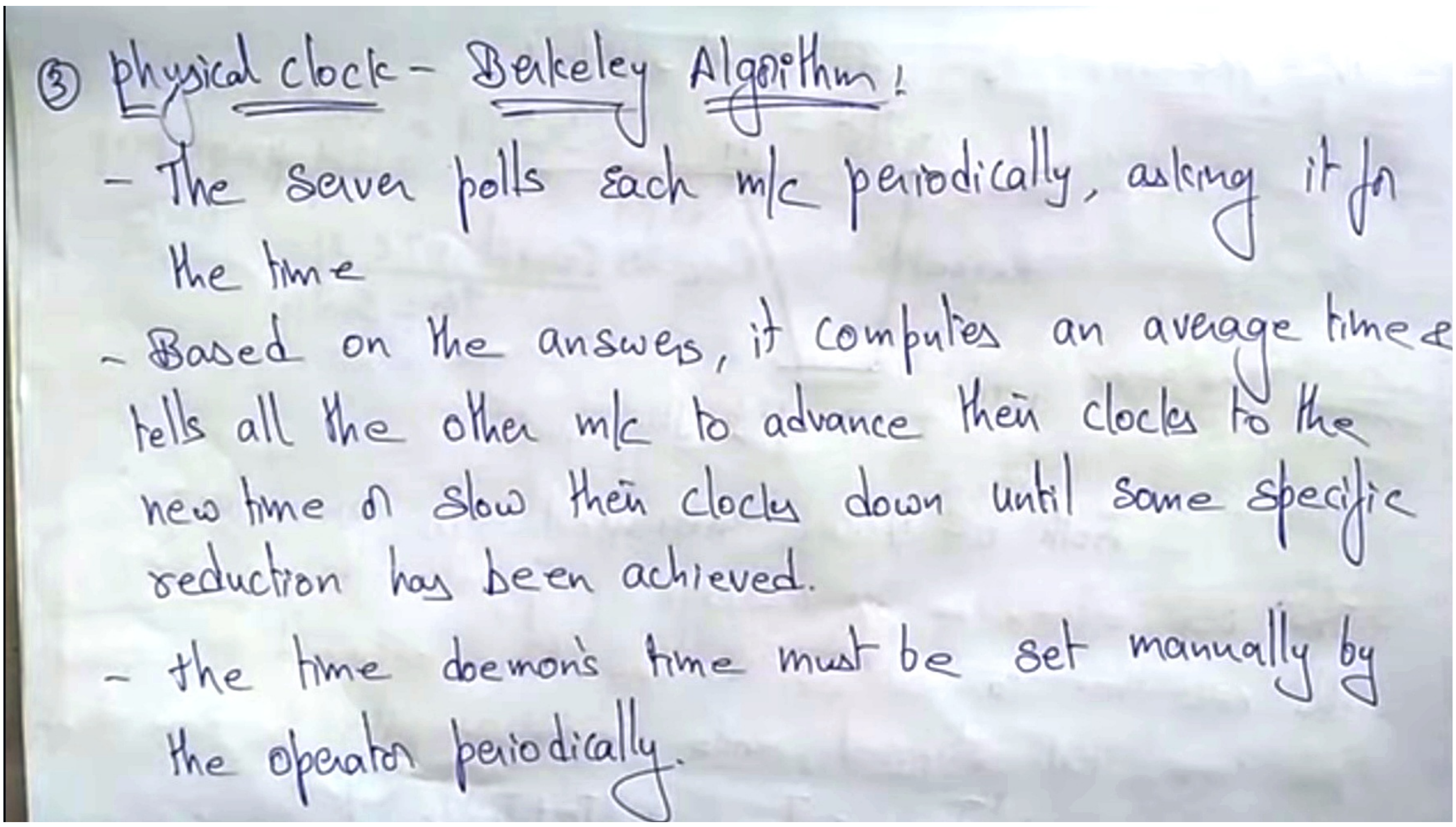
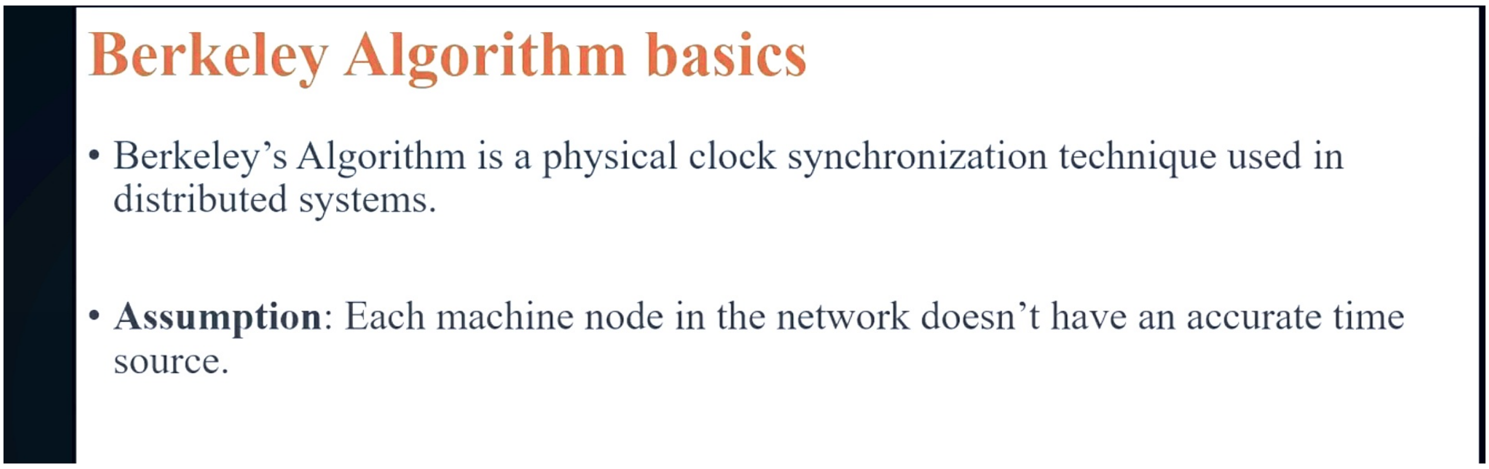


③ <u>physical clock</u> - <u>Berkeley Algorithm</u>:

- The server polls each m/c periodically, asking it for the time
- Based on the answers, it computes an average time & tells all the other m/c to advance their clocks to the new time or slow their clocks down until some specific reduction has been achieved.
- the time doemon's time must be set manually by the operator periodically.

Berkeley's Algorithm
(Physical clock synchronization algorithm)

Berkeley Algorithm illustration

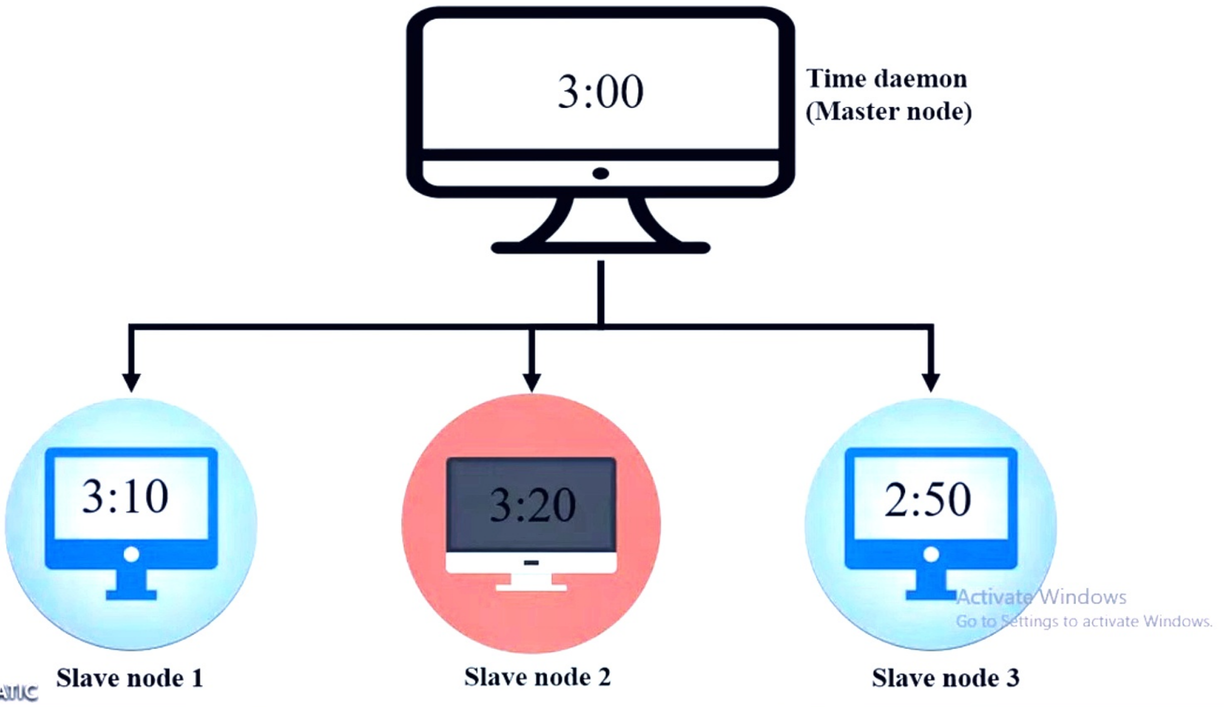


Berkeley Algorithm basics

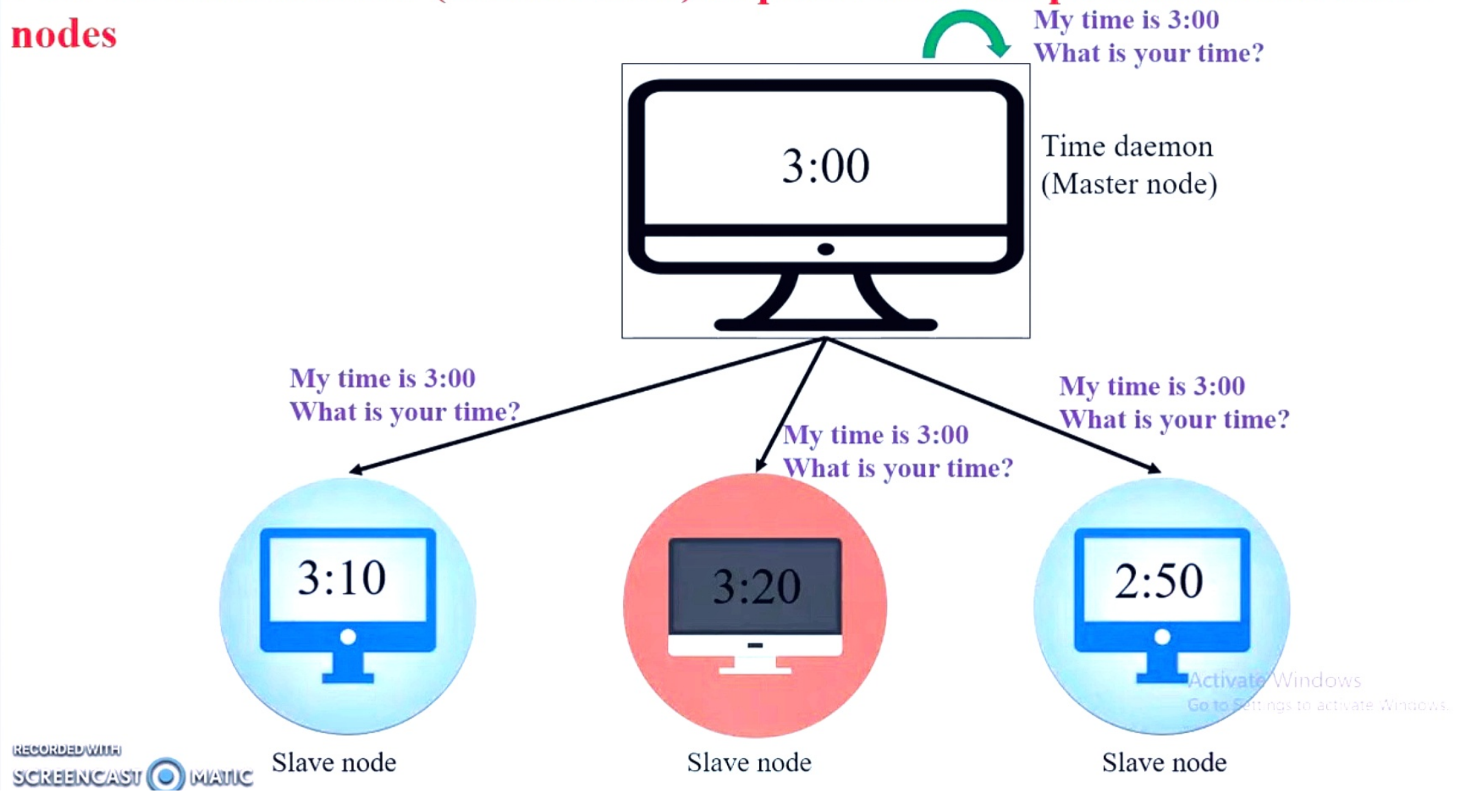
- Berkeley's Algorithm is a physical clock synchronization technique used in distributed systems.

- **Assumption**: Each machine node in the network doesn't have an accurate time source.

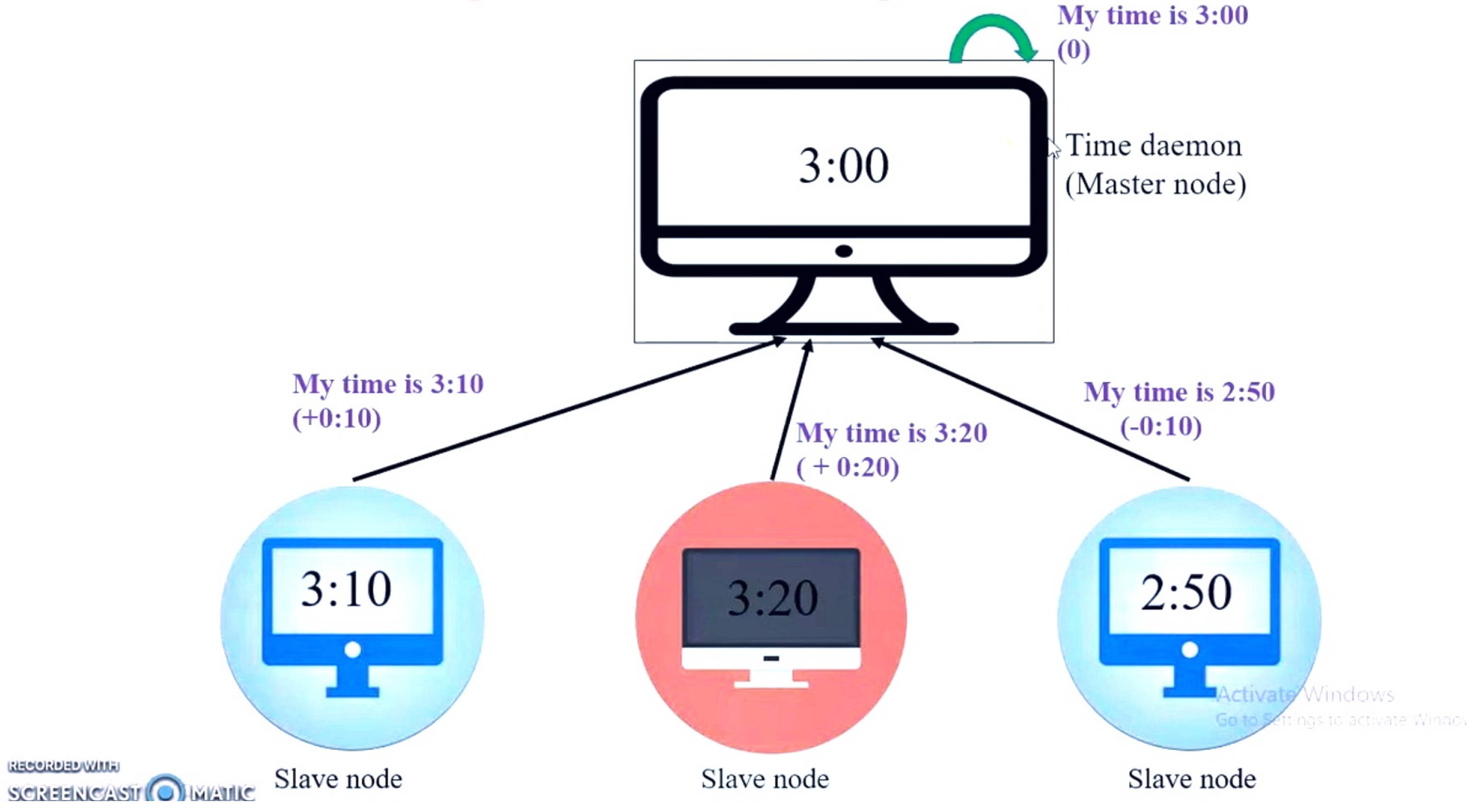
Berkeley Algorithm illustration



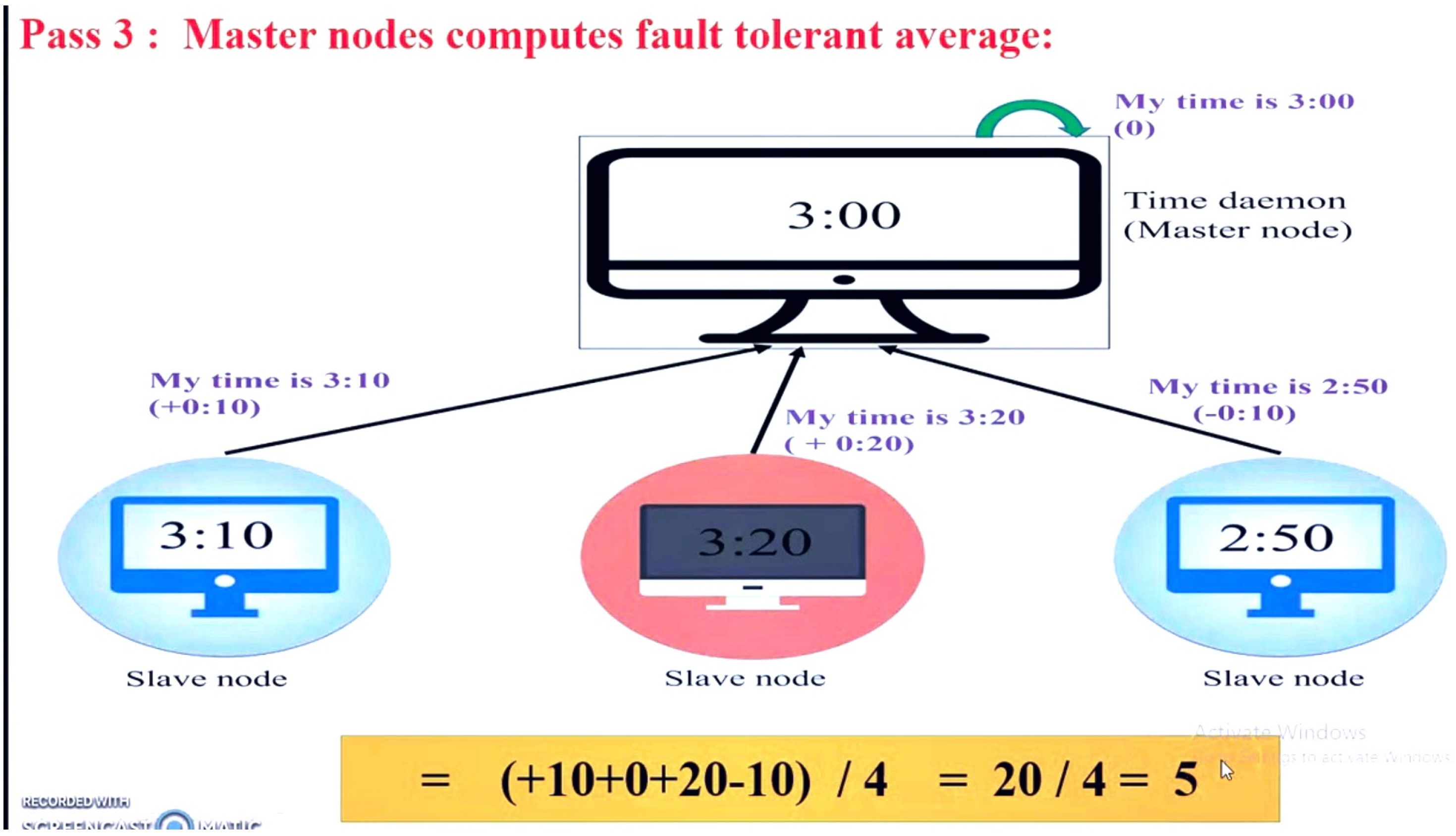
Pass 1 : Time daemon (Master node) requests timestamps from all the slave nodes



Pass 2 : Slave nodes responds their timestamps to master node

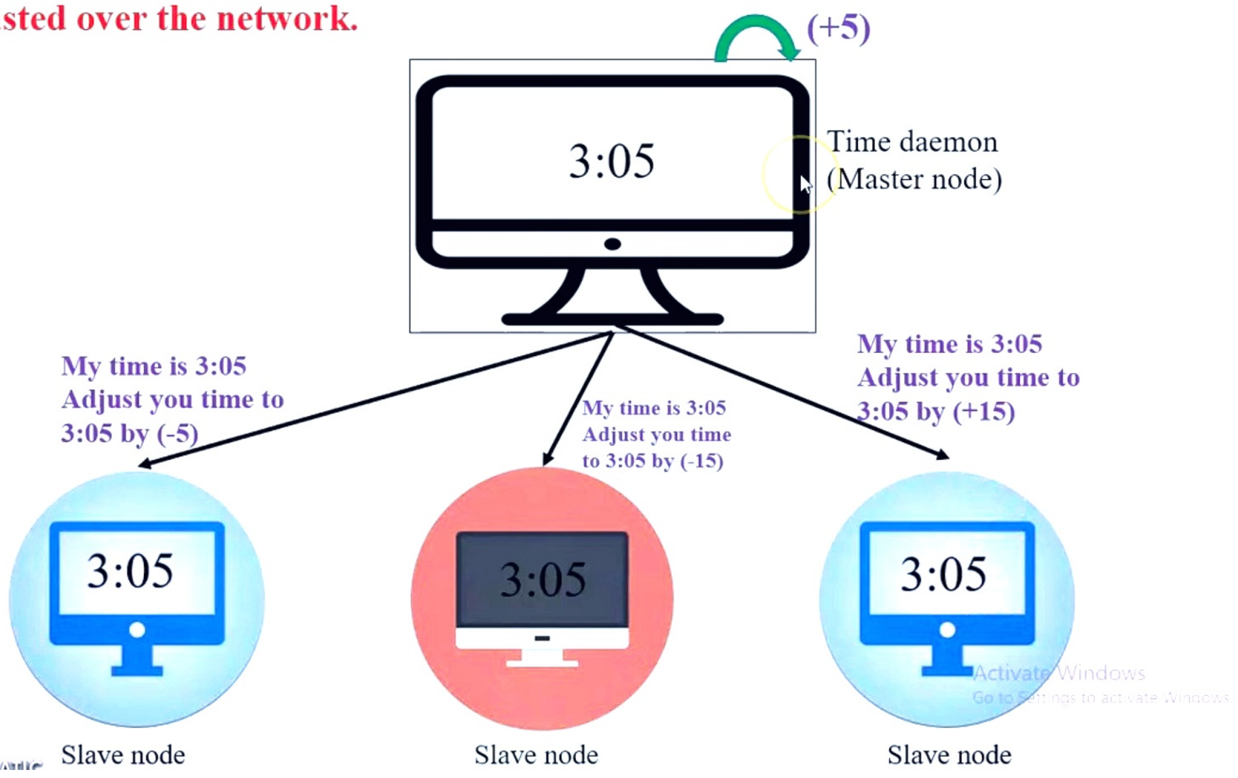


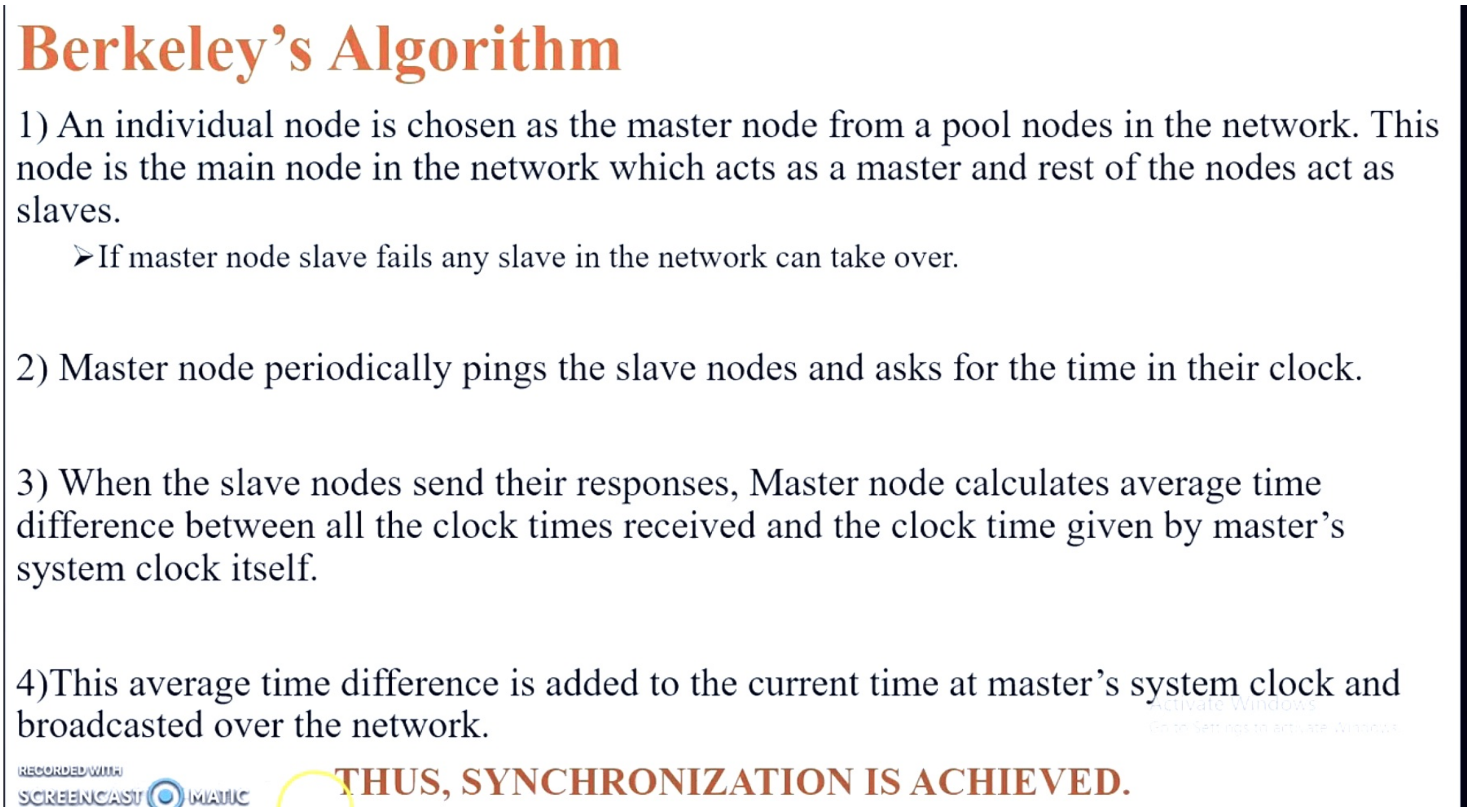
Pass 3 : Master nodes computes fault tolerant average:



= (+10+0+20-10) / 4 = 20 / 4 = 5

Pass 4 : This average time difference is added to the current time at master's system clock and broadcasted over the network.





Berkeley's Algorithm

1) An individual node is chosen as the master node from a pool nodes in the network. This node is the main node in the network which acts as a master and rest of the nodes act as slaves.

- If master node slave fails any slave in the network can take over.

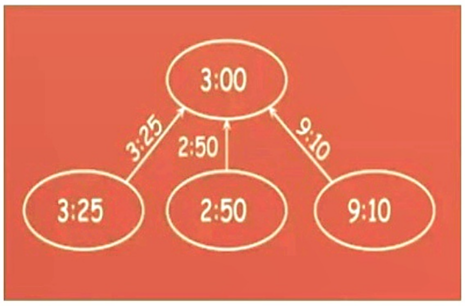
2) Master node periodically pings the slave nodes and asks for the time in their clock.

3) When the slave nodes send their responses, Master node calculates average time difference between all the clock times received and the clock time given by master's system clock itself.

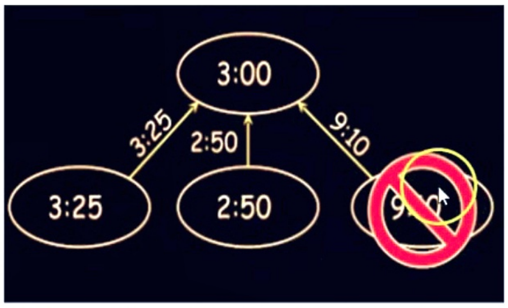
4)This average time difference is added to the current time at master's system clock and broadcasted over the network.

THUS, SYNCHRONIZATION IS ACHIEVED.

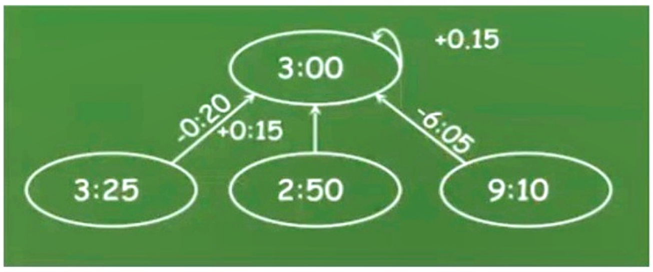
➢ Algorithm has provisions for ignoring readings from the clocks whose skew is too large



1. Request timestamps from all slaves



2. Ignore the reading from the clock whose skew is large and compute fault tolerant average



3. Send offset to each client