Student's name: Nguyễn Thanh Tùng

Class: ICT04-K62

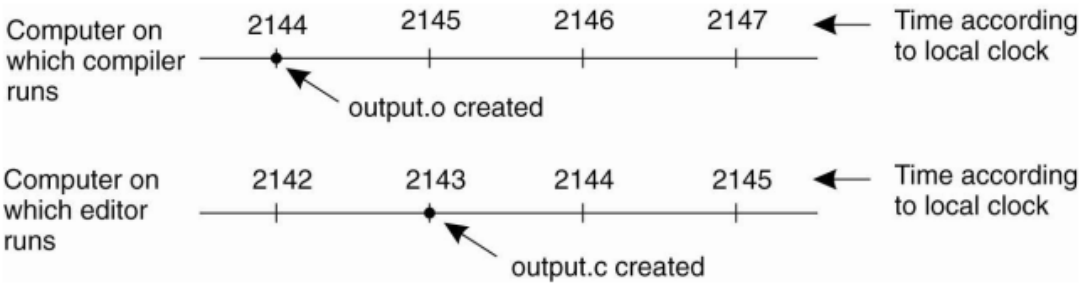
Class Exercises

Module: Distributed Systems

Chapter 4: Synchronization

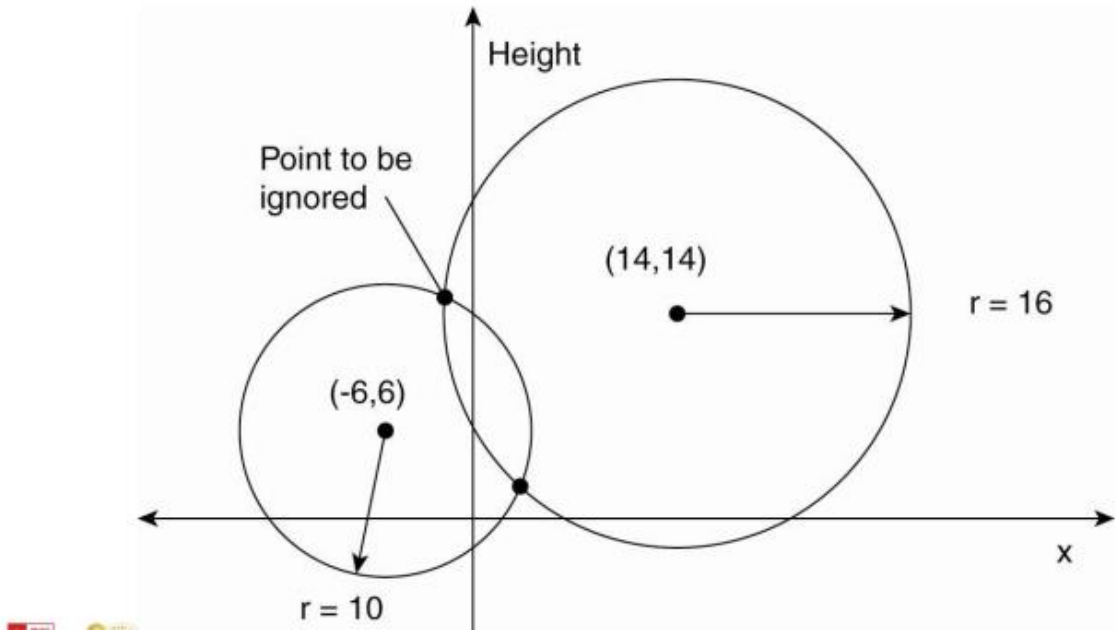
Question 1.

- **Example 1: Programming in DS**



When each machine has its own clock, an event that occurred after another event may nevertheless be assigned an earlier time

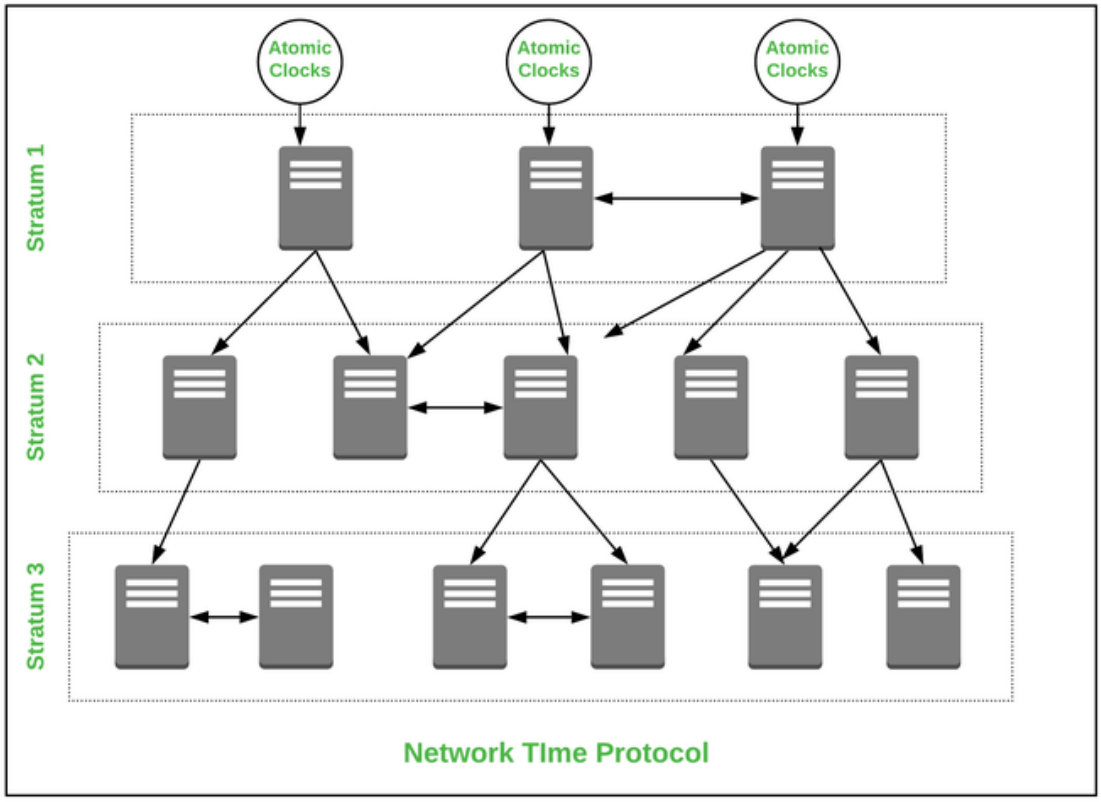
- **Example 2: Global Positioning System**



It takes a while before data on a satellite's position reaches the receiver. The receiver's clock is generally not in synch with that of a satellite.

Question 2.

- **Network time protocol**



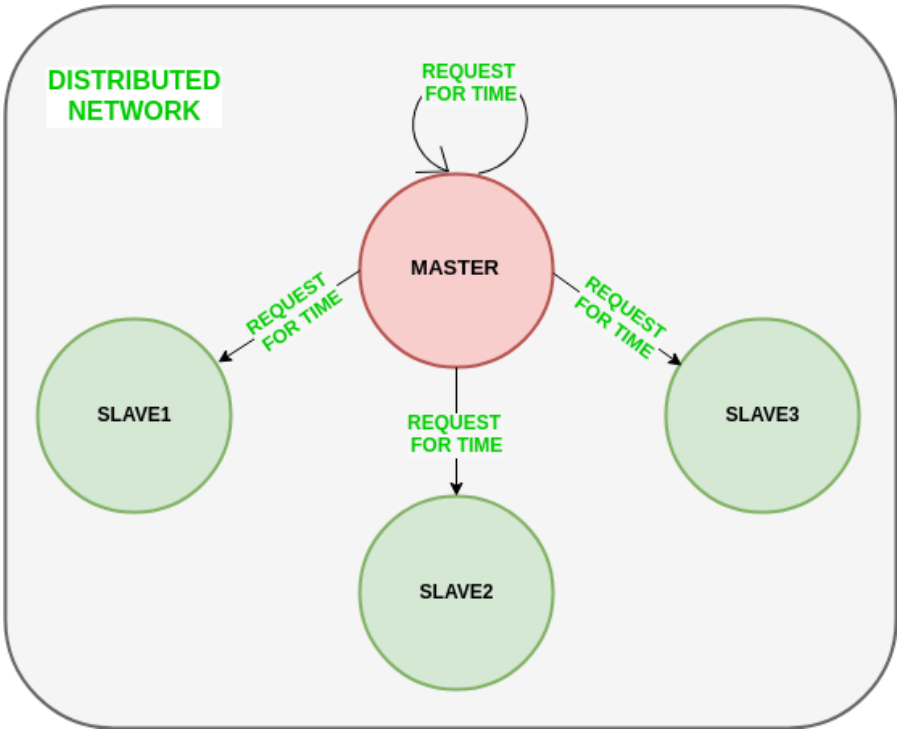
NTP is a protocol that works over the application layer, it uses a hierarchical system of time resources and provides synchronization within the stratum servers. First, at the topmost level, there is highly accurate time resources' ex. atomic or GPS clocks. These clock resources are called stratum 0 servers, and they are linked to the below NTP server called Stratum 2 or 3 and so on. These servers then provide the accurate date and time so that communicating hosts are synced to each other.

- **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. Master node is chosen using a election process/leader election algorithm.

2) Master node periodically pings slaves nodes and fetches clock time at them using Cristian's algorithm.

Diagram below illustrates how the master sends request to slave nodes.



Question 3.

The primary concern of all wireless applications is energy conservation. A clock synchronization algorithm (CSA for short) must carefully regiment its frequency of resynchronization, and avoid flooding. In addition, the algorithm cannot typically rely on a power-hungry source of real time such as GPS. Another characteristic of wireless networks is unexpected and possibly frequent changes in network topology. Thus, a CSA in a wireless medium must continue to function in the face of node failures and recoveries. Lastly, many applications in wireless settings are local in nature. That is, only nearby nodes in the network need to participate in some activity. Thus, a desirable property for a CSA is that it closely synchronizes nodes which are nearby, while possibly allowing faraway nodes to be more loosely synchronized.

Question 4.

- **Physical clock**

  It is a physical process and also a method of measuring that process to record the passage of time. For example, the rotation of the Earth measured in solar days. Most of the physical clocks are based on cyclic processes such as a celestial rotation.

- **Logical clock**

It is a mechanism for capturing causal and chronological relationships in a distributed system. A physically synchronous global clock may not be present in a distributed system. In such systems a logical clock allows global ordering on events from different processes.

Question 5.

Updating counter Ci for process Pi

1. Before executing an event Pi executes
    a. Ci ← Ci + 1.
2. When process Pi sends a message m to Pj, it sets m's timestamp ts (m) equal to Ci after having executed the previous step.
3. Upon the receipt of a message m, process Pj adjusts its own local counter as Cj ← max{Cj , ts (m)}, after which it then executes the first step and delivers the message to the application.

Question 6.

a) RTT refers to the combined time taken by the network and the server to transfer request to the server, process the request and returning the response back to the client process. The time at client side differs from actual time by at most RTT/2 seconds. Using above statement we can draw a conclusion that the error in synchronization can be at most RTT/2 seconds.

Hence $error\ \epsilon \left[\frac{-RTT}{2}, \frac{RTT}{2}\right]$

b) By defining a minimum transfer time, with a high confidence, we can say that the server time will always be generated after $T_0 + T_{min}$ and the T_{server} will always be generated before $T_1 - T_{min}$, Here synchronization error can be formulated as follows:

$$error\ \epsilon \left[\frac{-RTT}{2} - T_{min}, \frac{RTT}{2} - T_{min}\right]$$

The synchronized clock time in this case can be calculated as:

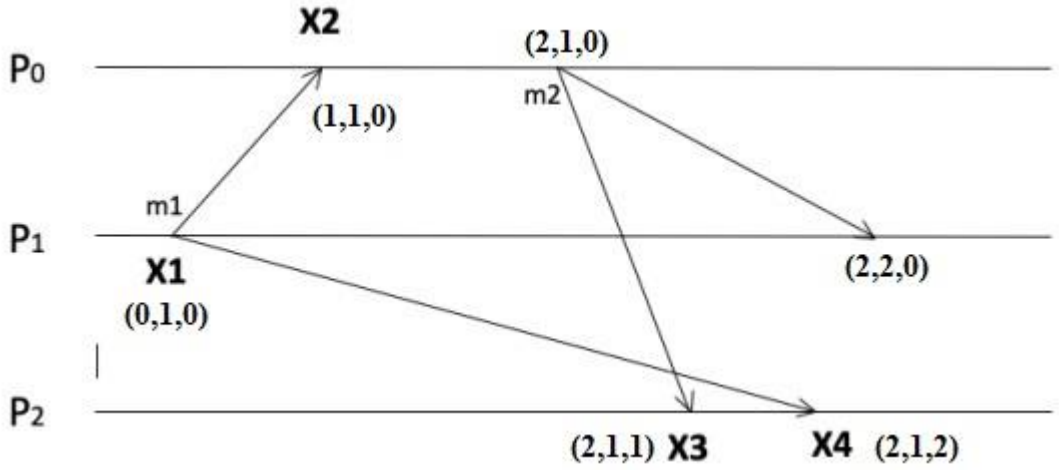
$$T_{client} = T_{server} + \frac{RTT}{2} + \frac{T_{min}}{2}$$

Question 7.

a)

1. $ts(m)[i] = VC_j[i]+1$
2. $ts(m)[k] \leq VC_j[k]$ for all $k \neq i$

b)



Question 8.

In Distributed systems, we neither have shared memory nor a common physical clock and there for we can not solve mutual exclusion problem using shared variables. To eliminate the mutual exclusion problem in distributed system approach based on message passing is used. Below are the three approaches based on message passing to implement mutual exclusion in distributed systems:

- Token Based Algorithm:
- Non-token based approach:
- Quorum based approach:

Question 9.

- The coordinator is a single point of failure, the entire system may go down if it crashes.
- If processes normally block after making a request, they cannot distinguish a dead coordinator from ‘‘permission denied’’ since no message comes back.
- In a large system a single coordinator can become a performance bottleneck.

Question 10.

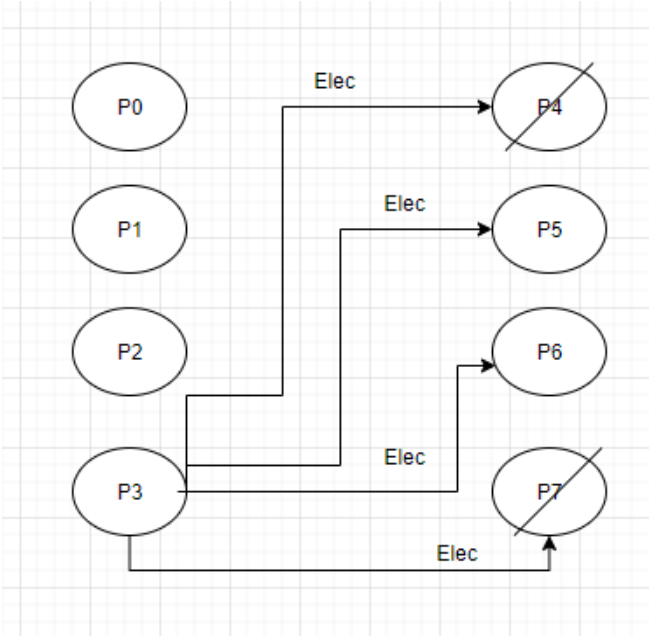
- Complicated
- n points of failure

- Broadcasting mechanism scalability
- More expensive, but less robust than the centralized algorithm

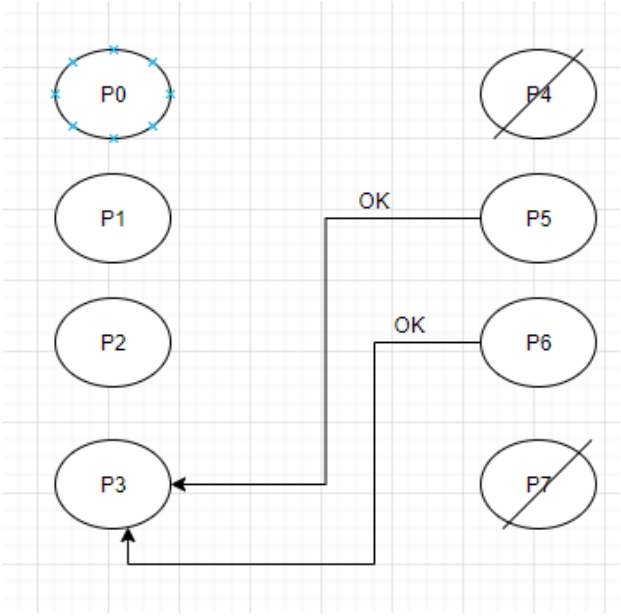
Question 11.

There should be a defensive mechanism again the loss of token. We can allow the process to generate new Token. Process will wait for Max Token Return Time and if the token is not received then it will generate a new Token. However, this raise a new problem of concurrent token creation → need collision resolution algorithms. For the collision resolution, we can use a locking mechanism such as mutex to protect the token value.

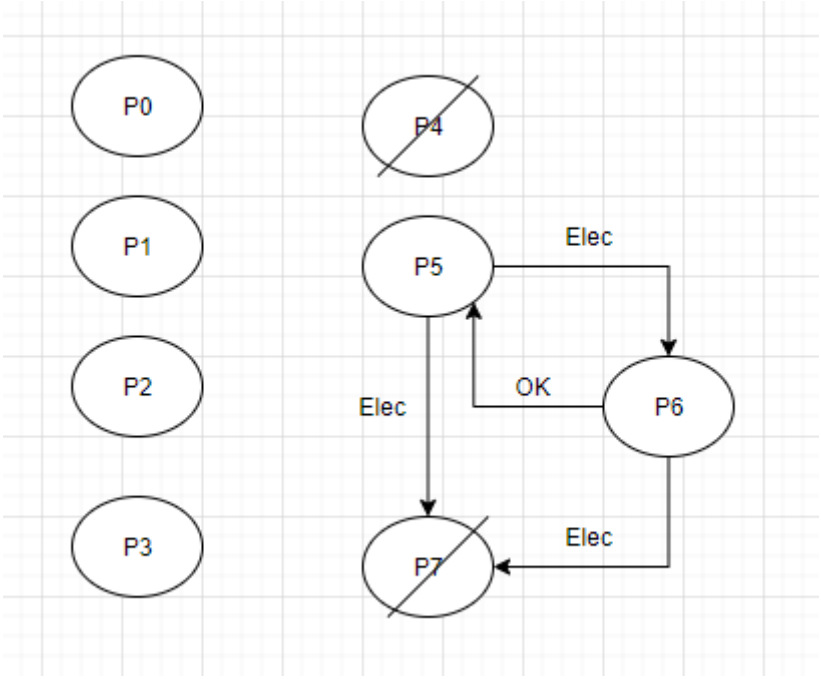
Question 12.



First the P3 start its election, the elec message is sent to all the process with higher number → 4 message is created.



Then, since P4 and P7 broken, only P5 and P6 reply to P3 telling P3 to stop.



Then, P5 and P6 host the election by sending 3 elec messages P5 → P6, P5 → P7 and P6 → P7. Afterward, P6 send response to P5 election and become the coordinator.

So, there were 4 + 3 = 7 messages needed to be sent before the coordinator is selected.

Question 13:

a) The number of messages needed to vote for a new coordinator is N + 1 (consider the broadcast message will not be sent to broken nodes). If the broadcast message will be sent to every node include broken nodes. We consider 3 scenarios. Where the node with id just below the broken coordinator is running, the total messages needed is N + 1. Where the new coordinator is PN , we need 2N + i + 1messages. And where the new coordinator is Pi itself, the number of messages is 2N + i.

b) Consider proposing new algorithm for recovering node, we send a message RECOVER from recovered node to the coordinator node, then the coordinator node send back OK and broadcast to every node to update their status table, total messages needed is N + 2.

Question 14:

a) Total messages needed is: 2N - 2 messages include N - 1 message REQUEST from Pi and N – 1 REPLY messages to Pi to let Pi use SR (just use, did not includes release).

b) This improvement work in this specific scenario since Pi do not need REPLY from Pj to confirm that its request have the lowest timestamp to use SR, so, the total message is 2N – 2 – i where I is the number of processed sent the same time with Pi but have timestamp higher than Pi .(just messages to use SR, not include release it)