Student’s name: Nguyễn Đức Dũng

Class: ICT.02-K61

**Class Exercises**

**Module: Distributed Systems**

**Chapter 6: Synchronization**

**Theoretical Exercises:**

**Part I**

**Question 1:** Give two examples to demonstrate the importance and the need of synchronization mechanism between processes in distributed systems.

* Example 1: Power generating station with multiple generators. Each generator is independent but the various AC power phases need to be aligned, or the generators would be "fighting" each other and there would be significant distortion and inefficiency.
* Example 2: AirBnB cottage, if there are two clients trying to make a reservation for the same day, only one can win, and the other must get an error.

**Question 2:** Compare Network Time Protocol and Berkeley algorithm.

* **Type of algorithm:**
* Network Time Protocol: Distributed.
* Berkeley algorithm: Centralized.
* **Approach**:
* Network Time Protocol: External Clock is used as reference time server. Based on Multiple time server arranged in levels.
* Berkeley algorithm: Active Time Server and based on Internal clock synchronization approach.
* **Scalability**:
* Network Time Protocol: Good.
* Berkeley algorithm: Poor.
* **Fault tolerance:**
* Network Time Protocol: Yes.
* Berkeley algorithm: No.
* **Reasons for implementation:**
* Network Time Protocol: To minimize propagation time (in milliseconds) and faster access of correct time value.
* Berkeley algorithm: To minimize the maximum difference between any two clock (in milliseconds).
* **Limitations:**
* Network Time Protocol: Supports in UNIX system only.
* Berkeley algorithm: Server becomes bottleneck.

**Question 3:** What is the typical characteristic of synchronization algorithm for wireless networks?

The machines don’t change their local time system, they store offset values which are the time differences with other machines within system.

**Question 4:** What is the difference between physical synchronization and logical synchronization?

* Physical synchronization is a physical process and also a method of measuring that process to record the passage of time.
* Logical synchronization is a mechanism for capturing causal and chronical relationships in a distributed system.

**Question 5:** What are the update steps of counters to implement Lamport’s  
logical clock?

All the process counters start with value 0. Each process maintains a logical clock that ticks (increments its counter) before sending its value along with an outgoing message, when a process receives a message, it synchronizes its logical clock with the senders if the received counter value is greater than its own, then increments it.

**Question 6:**

a) The value of TP is not absolutely correct in the case (T4-T3) is different from (T2 – T1)

b) We have:

* - min

**Question 7:**

1. Two conditions:
2. Vector clock values:

* X1:
* VC0 = [0,0,0]
* VC1 = [0,1,0]
* VC2 = [0,0,0]
* X2:
* VC0 = [0,1,0]
* VC1 = [0,1,0]
* VC2 = [0,0,0]
* X3:
* VC0 = [1,1,0]
* VC1 = [0,1,0]
* VC2 = [0,0,0]
* X4:
* VC0 = [1,1,0]
* VC1 = [0,1,0]
* VC2 = [0,1,0]

1. m2 is kept in the middleware since when m2 arrives with ts(m2) does not satisfy the two condition.

**PART II**

**Question 1:**

Mutual exclusion algorithm is an algorithm which is created to allow a process to obtain exclusive access to a resource.

**Question 2:**

* Process cannot distinguish being block from a dead coordinator.
* Centralized server can be a bottleneck.

**Question 3:**

Algorithm is suitable only for small group of processes, identity of all processes is needed making it complex.

**Question 4:**

When a process P1 wants to enter into its critical section, it sends request to the coordinator. If the coordination retains the token, it then sends the token to the requesting process (P1). After getting the token, the process will send an acknowledgement to the coordinator, and enters the critical section. During the execution of the critical section, P1 will continually send an exists signal to the coordinator at certain time interval, so that, the coordinator becomes acquainted that the token is alive and it has not lost.

**Question 5:**

P3 starts the election, it sends messages to P4,P5,P6,P7. Since P4 & P7 are broken, there are two processes respond P5 and P6. Process P5 sends election messages to P6 & P7, only process P6 answers and take over the election. Process P6 sends out only one election message to P7, when process 7 does not respond, process 6 declares itself the winner and broadcasts coordinator message to the whole system. So, we need 16 messages.

**Question 6:**

1. Assign *m* is the number of broken nodes, m = {0,N-i}

S1: Pi -Election🡪 PN-k

S2: There are two cases, the first case is PN-k running (N – k = i)

🡺 PN-k -coordinator🡪 Pj with j = {1,n-k-1}

With second case, PN-k broken 🡪 k = k + 1 then return to step 1.

🡺 Total number of messages:

* + With OK message: N
  + Without OK message: N – 1

1. If the broken node with node ID smaller than the current coordinator ID, it will initiate an election from current state, the recovered node will be the new coordinator and update the process status table and sends the coordinator message to all other processes in the system and takes over the coordinator’s job from the currently active coordinator. Number of messages in this case = n – 1.

If the current coordinator ID larger than recovered node’s ID, recovered node will send its priority number and an update messages to all other processes in the system. Number of messages in this case = n – 1.

Since we just need to consider if the recovered node’s ID has higher priority than the current coordinator so when a crashed process recover, it just sends messages to all processes with higher process number than it.

**Question 7:**

Pi broadcasts messages: n - 1 messages

Pi received reply messages: n – 1 messages

Pi broadcasts release messages: n – 1 messages

Hence, let a process use SR, it need 3n – 3 messages

It work. For this improvement, a process only use SR when it not receive any REPLY. Supposed Pi send a REQUEST message and there are m REPLY messages return, it means there are m processes has to use SR before Pi, so Pi has to wait m RELEASE messages to use SR. Hence, to use SR, process Pi need the amount of messages:

Broadcasts messages: n – 1

Release messages: n – 1

REPLY messages received: m

Number of messages: 2n – 2 + m