**18CS621: Distributed Systems**

**Lab Assignment: Coordination**

**Date:- 28/11/2020**

**Individual Contributions to Assignment**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Name** | **Contributions** |
| 20010 | C H Vamshi | Mutual Exclusion (Token Ring) |
| 20017 | K Koushik | Leader Election (Ring Election) |
| 20031 | 1. Aathith Bhalaje | Vector Clocks Synchronization |

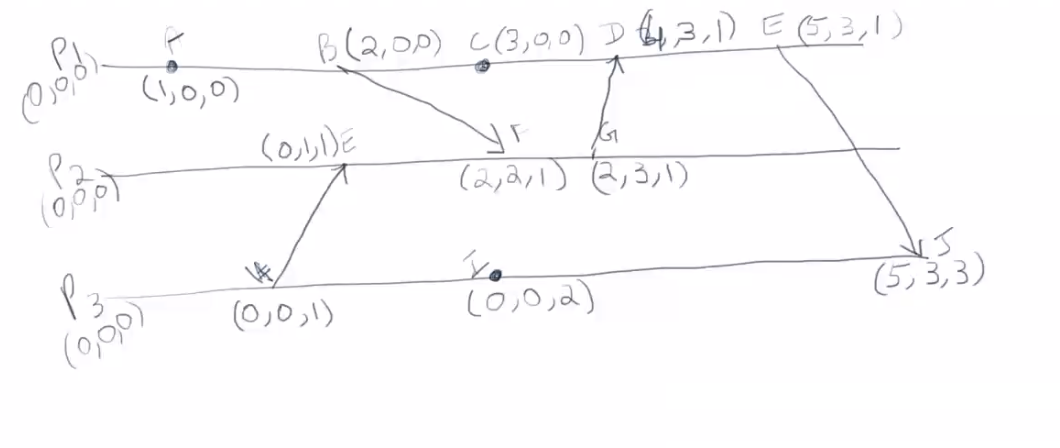
**1). Lamport Logical Clock Synchronisation (Vector Clock)**

**How does the vector clock algorithm work :-**

1. Initially, all the clocks are **set to zero**.
2. Every time, an Internal event occurs in a process, the value of the processes logical clock in the vector is **incremented by 1**
3. Also, every time a process sends a message, the value of the processes logical clock in the vector is **incremented by 1**.

Every time, a process receives a message, the value of the processes logical clock in the vector is **incremented by 1**, and moreover, each element is updated by taking the maximum of the value in its own vector clock and the value in the vector in the received message (*for every element*).

**Diagram :-**



**How we tried to Implement** :-

We tried to implement the above Lamport Logical Clock Synchronization (Vector Clock) diagram and wrapped it using a real time Bank Application, in which there will be 3 accounts namely A,B,C :-

They can;

1). Transfer money between their accounts.

2). Check the Balance.

3). Show the Synchronized timers in each process.

**NOTE** :-

As per vector clock occurring of an event increases **Time Stamp + 1**, at a particular process.

Here, the Events can be considered as Transferring the money or Checking the balance. While Transferring the money the message is transferred from one account to other account which demonstrates the message passing between different processes.

Checking the balance can be the event which occurs at single process.

**Instructions to Run Code: -**

* **Files Structure: -**

-Co-Ordination Assignment

|-Problem1(Vector Clock)

|- vectorClock.go

|- VectorClock.docx (Documentation)

|-Problem3(Mutual Exclusion)

|-tokenRingMuTex.go

|-Token Ring Mutual Exclusion.docx (Documentation)

|-Problem4(Leader Election)

|-ringLeaderElection.go

|-Election Leader.docx (Documentation)

* **How to Give inputs :-**

According to the Diagram :-

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **CHECKING BALANCE** | **TRANSFERING MONEY** | **SYNCHRONISING TIMERS** |
| 1). |  | In Process P3 & P2, there is a transaction between the events H & E is happening. | So, the ***INITIAL*** Time Stamp is H(0,0,1) & E(0,1,1). |
| 2). | In Process P3, a random event I checks it’s balance. |  | So, the Time stamp will be I(0,0,2). |
| 3). | In Process P1, a random event A checks it’s balance. |  | So, the Time stamp will be A(1,0,0). |
| 4). |  | In Process P1 & P2, there is another transaction is happening between the events B&F. | So, the Time Stamp is B(2,0,0) & F(2,2,1). |
| 5). | In Process P1, a random event C checks it’s balance. |  | So, the Time stamp will be C(3,0,0). |
| 6). |  | In Process P2 & P1, there is a transaction is happening between the events G & D. | So, the Time Stamp is G(2,3,1) & F(4,3,1). |
| 7). |  | In Process P1 & P3, there is a transaction is happening between the events E & J. | So, the Time Stamp is E(5,3,1) & J(5,3,3). |

**Output Screenshots** :-

