Practical 3

Distributed Operating System

Name - Ritesh Parkhi

Roll No. – 46

Batch – 3

AIM:

Build a program to implement concept of distributed mutual exclusion using Non-token based algorithm.

THEORY:

Ricart-Agrawala algorithm is an algorithm to for mutual exclusion in a distributed system proposed by Glenn Ricart and Ashok Agrawala. This algorithm is an extension and optimization of Lamport's Distributed Mutual Exclusion Algorithm. Like Lamport's Algorithm, it also follows permission based approach to ensure mutual exclusion.

In this algorithm:

- Two type of messages (REQUEST and REPLY) are used and communication channels are assumed to follow FIFO order.
- A site send a REQUEST message to all other site to get their permission to enter critical section.
- A site send a REPLY message to other site to give its permission to enter the critical section.
- A timestamp is given to each critical section request using Lamport's logical clock.
- Timestamp is used to determine priority of critical section requests. Smaller timestamp gets high priority over larger timestamp. The execution of critical section request is always in the order of their timestamp.

Algorithm:

Requesting Site

- Sends a message to all sites. This message includes the site's name, and the current timestamp of the system according to its logical clock (which is assumed to be synchronized with the other sites)

Receiving Site

- Upon reception of a request message, immediately sending a timestamped reply message if and only if:
    - the receiving process is not currently interested in the critical section OR
    - the receiving process has a lower priority (usually this means having a later timestamp)
- Otherwise, the receiving process will defer the reply message. This means that a reply will be sent only after the receiving process has finished using the critical section itself.

Critical Section:

- Requesting site enters its critical section only after receiving all reply messages.
- Upon exiting the critical section, the site sends all deferred reply messages.

Drawbacks of Ricart-Agrawala algorithm:

- Unreliable approach: failure of any one of node in the system can halt the progress of the system. In this situation, the process will starve forever.
- The problem of failure of node can be solved by detecting failure after some timeout.

PROGRAM:

```
#include<iostream>
#include<vector>
#include<map>
using namespace std;

int main()
{
    int ns, ncs, timestamp, site;
    cout << "Enter number of sites :";
    cin >> ns;
    cout << "Enter number of sites which want to enter critical section:";
    cin >> ncs;
    vector<int> ts(ns, 0);
    vector<int> request;
    map<int, int> mp;
    for (int i = 0; i < ncs; i++)
    {
        cout << "\nEnter timestamp:";
        cin >> timestamp;
        cout << "Enter site number:";
```

```
        cin >> site;
        ts[site - 1] = timestamp;
        request.push_back(site);
        mp[timestamp] = site;
    }

    cout << "\nSites and Timestamp:\n";
    for (int i = 0; i < ts.size(); i++)
    {
        cout << i + 1 << " \t\t" << ts[i] << endl;
    }

    for (int i = 0; i < request.size(); i++)
    {
        cout << "\nRequest from site:" << request[i] << endl;
        for (int j = 0; j < ts.size(); j++)
        {
            if (request[i] != (j + 1))
            {
                if (ts[j] > ts[request[i] - 1] || ts[j] == 0)
                    cout << j + 1 << ": Replied\n";
                else
                    cout << j + 1 << ": Deferred\n";
            }
        }
    }

    cout << endl;
    map<int, int>::iterator it;
    it = mp.begin();
    int c = 0;
    for (it = mp.begin(); it != mp.end(); it++)
    {
        cout << "Site " << it->second << " entered Critical Section \n";
        if (c == 0)
            cout << "\nSimilarly,\n\n";
        c++;
    }
    return 0;
}
```

OUTPUT:

CONCLUSION:

Hence we have successfully built a program to implement concept of distributed mutual exclusion using Ricart-Agrawala algorithm.