Name :- Rushikesh Kazbhazi Palve
Roll No. 31258

: Date :	Page No :
/ /20	

(L)

Assignment No. 8

DOP :- 15-11-2021 DOJ :- 17-11-2021

Problem Deffnition :-

Simulation of election algorithms (Ring and Bully). (Unix C progeamming / Java)

Objectives :-

(i) To study the use of coordinator.
(ii) To understand the concept of deriding coordinator for accessing shared resource.

Learning Outcomes :-

~~After~~ completion of the assignment, students will be able to -
(i) Understand process synchronization.
(ii) Simulate the Bully and Ring algorithms.

Theory :-

Many distributed algorithms require one process to act as coordinator, initiator or to perform some other spérial role. In the centralized mutual exclusion algorithm, one process is elected as the coordinator.

For instance, the process running on the machine with the highest network address might be selected. Whenever a process wants to enter a critical region, it sends a request message to the coordinator stating which critical region it wants to enter and asking for permission. If no other process is currently in that region, the coordinator sends back a reply granting permission.

It does not matter which process takes on the special responsibility of coordinator, but one of them has to do it. In general, election algorithms attempt to locate the process with the highest process number and designate it as coordinator. It is assumed that every process knows the process number of every other process. What the processes do not know is which ones are currently up and which ones are currently down. The goal of election algorithm is to ensure that when an election starts, it concludes with all process agreeing on who the new coordinator is to be.

Bully Election Algorithm :-

The Bully Algorithm was devised by Garcia-Molina in 1982. When a process notices that the coordinator is no longer responding to requests, it initiates an election. Process P, holds an election

as follows :

1) P sends an ELECTION message to all processes with higher numbers.

2) If no one responds, P wins the election and becomes coordinator.

3) If one of the higher-ups answers, it takes over. P's job is done.

At any moment, a process can get an ELECTION message from one of its lower-numbered colleagues. When such a message arrives, the receiver sends an OK message back to the sender to indicate that it is alive and will take over. The receiver then holds an election, unless it is already holding one. Eventually, all processes give up but one, and that one is the new coordinator. It announces its victory by sending all processes a message telling them that starting immediately it is the new coordinator.

If a process that was previously down comes back up, it holds an election. If it happens to be the highest-numbered process currently running, it will win the election and will take over the coordinator's job. Thus the biggest guy in town always wins, hence the name "Bully Algorithm".

Ring Election Algorithm :-

This election algorithm is based on the use of a ring. We assume that the processes are physically or logically ordered, so that each process knows who its successor is. When any process notices that the coordinator is not functioning, it builds an ELECTION message containing its own process number and sends the message to its successor. If the successor is down, the sender skips over the successor and goes to the next number along the ring, or the one after that, until a running process is located. At each step, the sender adds its own process number to the list in the message. Eventually, the message gets back to the process that started it all. That process recognizes this event when it receives an incoming message containing its own process number. At that point, the message type is changed to COORDINATOR and circulated once again, this time to inform everyone else who the coordinator is (designated by the list member with the highest number) and who the members of the new ring are. When this message has circulated once, it is removed and everyone goes back to work.

ROLL NO. 31258

: Date :	Page No :
/ /20	

ALGORITHM :-

1] Ring Election Algorithm :-

Step 1: Start

Step 2: If process P1 detects a coordinator failure, it creates new active list which is empty initially. It sends election message to its neighbour on right and adds number 1 to its active list.

Step 3: If process P2 receives message elect from processes on left, it responds in 3 ways:

- (I) If message received does not contain 1 in active list then P1 adds 2 to its active list and forwards the message.
- (II) If this is the first election message it has received or sent, P1 creates new active list with numbers 1 and 2. It then sends election message 1 followed by 2.
- (III) If process P1 receives its own election message 1 then active list for P1 now contains numbers of all the active processes in the system. Now process P1 detects highest priority number from list and elects it as the new coordinator.

Step 4: Stop.

2] Bully Election Algorithm :-

Suppose process P sends a message to the coordinator -

Step 1 : If coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.

Step 2 : Now process P sends election message to every process with high priority number.

Step 3 : It waits for responses, if no one responds for time interval T then process P elects itself as a coordinator.

Step 4 : Then it sends a message to all lower priority number processes that it is elected as their new coordinator.

Step 5 : However, if an answer is received within time T from any other process Q,

- (I) process P again waits for time interval T' to receive another message from Q that it has been elected as coordinator.
- (II) If Q doesn't responds within time interval T' then it is assumed to have failed and algorithm is restarted.

CONCLUSION :

We have successfully understood the concept and simulated Ring and Bully election algorithms.

CODE :-

1.] Ring Election Algorithm

```
/*
 * Problem Statement :-
    Simulation of election algorithms(Ring and Bully).
*/


package assignmentNo_8;


import java.util.*;
public class Ring
{
    static int token[] = new int[100];
    static int l = 0;
    public static void main(String args[])
    {
        int winner, max = -1;
        Scanner sc = new Scanner(System.in);
        System.out.print("\n\t Enter the Total Number of processes = ");
        int n = sc.nextInt();
        winner = n;
        int processes[] = new int[n+1];
        int status[] = new int[n+1];
        System.out.println("\n\t Enter the status of processes (1/0) .....");
        for(int i=0; i<n; i++)
        {
            processes[i] = i;
            System.out.print("\n\t Enter the status of process "+ i +" = ");
            status[i] = sc.nextInt();
        }
        System.out.print("\n\t Enter the process initiator = ");
        int x = sc.nextInt();

        int i = x;
        while(i<n)
        {
            if(status[i]==1)
            {
                int next = i+1;
                while(next<n)
                {
                    if(status[next]==1)
                    {
                        System.out.println("\n\t Election message is sent from
"+ i +" to "
                                + next);
                        token[l] = i;
                        l++;
```

```
                    print();
                    winner = next;
                    break;
                }
                else
                {
                    next++;
                }
            }
        }
        i++;
    }

        System.out.println("\n\t Election message is sent from " + winner + "
to 0");
        token[l] = winner;
        l++;
        print();
        i = 0;
        while(i<x)
        {
            if(status[i]==1)
            {
                int next = i+1;
                while(next<n)
                {
                    if(status[next]==1)
                    {
                        System.out.println("\n\t Election message is sent from
"+ i +" to "
                                + next);
                        token[l] = i;
                        l++;
                        print();
                        break;
                    }
                    else
                    {
                        next++;
                    }
                }
            }
        }
        i++;
    }

    for(int j=0; j<l; j++)
    {
        if(token[j] > max)
            max = token[j];
    }
    System.out.println("\n\t Co-ordinator is " + max);

    sc.close();
```

```
    }

    public static void print()
    {
        System.out.print("\n\t ___ Token received : ");
        for(int i=0; i<l; i++)
            System.out.print(token[i] + ", ");
        System.out.println();
    }
}
```

2.] Bully Election Algorithm

```
/*
 * Problem Statement :-
    Simulation of election algorithms(Ring and Bully).
*/


package assignmentNo_8;

import java.io.*;
import java.util.Scanner;

public class Bully
{
    static int n;
    static int pro[] = new int[100];
    static int sta[] = new int[100];
    static int co;

    public static void main(String args[])throws IOException
    {
        System.out.print("\n\t Enter Total number of processes : ");
        Scanner in = new Scanner(System.in);
        n = in.nextInt();


        for(int i=0;i<n;i++)
        {
            System.out.print("\n\t For process " + (i+1) + ":");
            System.out.print("\n\t\t Status (0/1) : ");
            sta[i] = in.nextInt();
            System.out.print("\n\t\t Priority : ");
            pro[i] = in.nextInt();
        }

        System.out.print("\n\t Which process will initiate election? : ");
        int ele = in.nextInt();

        elect(ele);
        System.out.println("\n\t Final coordinator is " + co);
```

```
        in.close();
    }

    static void elect(int ele)
    {
        ele--;
        co = ele+1;
        for(int i=0; i<n; i++)
        {
            if(pro[ele] < pro[i])
            {
                System.out.println("\n\t Election message is sent from " +
(ele+1)
                        + " to " + (i+1));
                if(sta[i]==1)
                    elect(i+1);
            }
        }
    }
}
```

OUTPUT :-

1.] Ring Election Algorithm

```
Enter the Total Number of processes = 5

Enter the status of processes (1/0) .....

Enter the status of process 0 = 1

Enter the status of process 1 = 0

Enter the status of process 2 = 1

Enter the status of process 3 = 1

Enter the status of process 4 = 1

Enter the process initiator = 2

Election message is sent from 2 to 3

___ Token received : 2,

Election message is sent from 3 to 4

___ Token received : 2, 3,
```

```
   Election message is sent from 4 to 0

   ___ Token received : 2, 3, 4,

   Election message is sent from 0 to 2

   ___ Token received : 2, 3, 4, 0,

   Co-ordinator is 4
```

2.] Bully Election Algorithm

```
Enter Total number of processes : 5

For process 1:
   Status (0/1) : 1

   Priority : 6

For process 2:
   Status (0/1) : 0

   Priority : 7

For process 3:
   Status (0/1) : 1

   Priority : 4

For process 4:
   Status (0/1) : 1

   Priority : 2

For process 5:
   Status (0/1) : 1

   Priority : 3

Which process will initiate election? : 3

Election message is sent from 3 to 1

Election message is sent from 1 to 2

Election message is sent from 3 to 2

Final coordinator is 1
```