ANISH ADNANI
D17B/01

EXPERIMENT 8

Aim: Implementation of Bully Election Algorithm.

Theory:

The coordinator election problem is to choose a process from among a group of processes on different processes.
In a distributed system to act as a central coordinator distributed algorithm requires the need of the coordinator to be elected amongst the ones which are currently active.
In Bully election algorithm, it is assumed that every process knows the priority no. of every other process in the system.

Working of Bully Election Algorithm.

i) Process 'p' calls an election when it notices that the coordinator is no longer responding. High numbered process 'bully' low numbered process out of the election, until only one process remains.

ii) When a crash process reboots, it holds an election. If it is now the highest numbered live process, it will win.

iii) Process 'p' sends an election message to all higher numbered processes in the system.

iv) If no process responds, then 'p' becomes the coordinator.

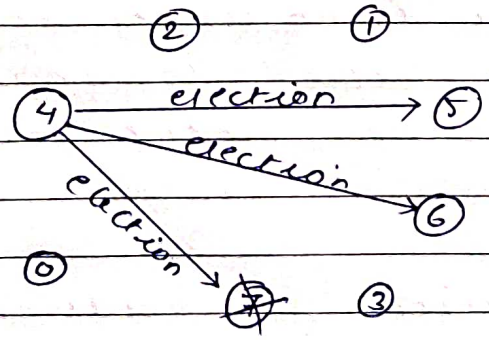
v) If a higher level process '@' responds it sends 'p' a message that terminates p's role in the algorithm

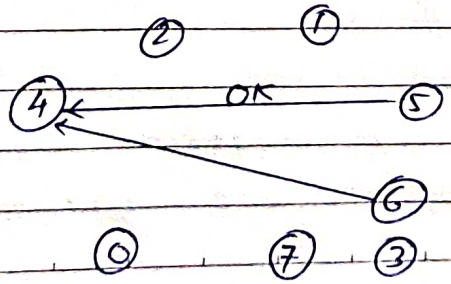
vi) The process '@' now calls an election

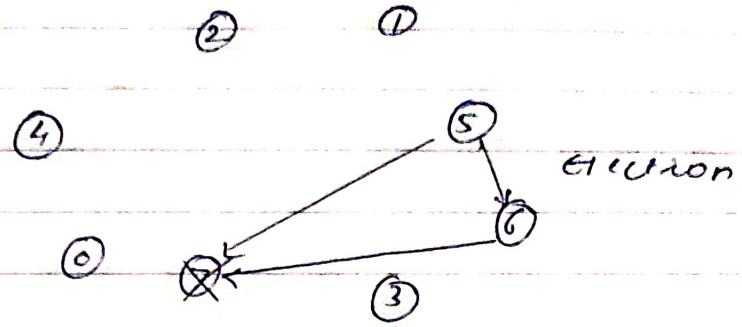
vii) Repeat until no higher level process responds. The last process to call an election 'wins' the election.

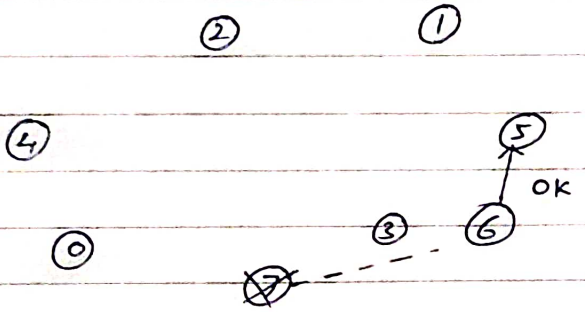
viii) The winner sends a message to other processes announcing itself as the new coordinator.

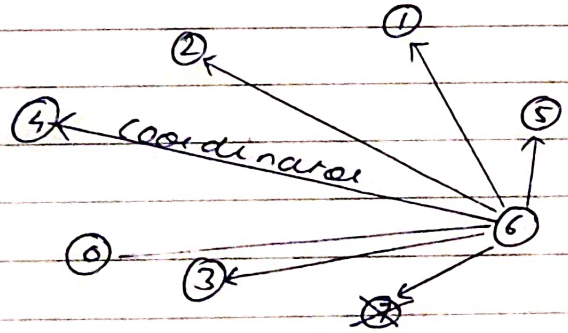
Bully Algorithm example:











Conclusion: In Bully election algorithm, every node in the system has a unique priority number. And every node in the system knows the priority of all nodes.
When an election is held, the node with highest priority number among the current live nodes is elected as the coordinator.

ANISH ADNANI D17B ROLL NO : 01

```
import java.io.*;

class BullyAlgorithm
{
    int code, ch, crash;
    int prc[];
    public void election(int n) throws IOException
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("\nThe Coordinator has Crashed !");
        int flag=1;

        while(flag == 1)
        {
            crash = 0;
            for(int i1 = 0; i1<n; i1++)
            {
                if(prc[i1] == 0)
                    crash++;
            }
            if(crash == n)
            {
                System.out.println("\n All Processes are Crashed ***!!!");
            }
            else
            {
                System.out.println("\n Enter the Initiator : ");
                int init = Integer.parseInt(br.readLine());
                if(init < 1 || init > n || prc[init - 1] == 0)
                {
                    System.out.println("\n Invalid Initiator... Try again");
                    continue;
                }
                for(int i1 = init - 1; i1 < n; i1++)
                {
                    System.out.println("Process "+(i1+1)+" called for Election.\n");
                }
                for(int i1 = init - 1; i1 < n; i1++)
                {
                    if(prc[i1] == 0)
                        System.out.println("Process "+(i1+1)+" is Dead.");
                    else
                        System.out.println("Process "+(i1+1)+" is In.");
                }
```

```
            for(int i1 = n-1; i1 >= 0; i1--)
            {
                if(prc[i1] == 1)
                {
                    code = (i1+1);
                    System.out.println("\n ***New Coordinator is "+code+" ***");
                    flag = 0;
                    break;
                }
            }
        }
    } // end of while
} // end of election() method

public void Bully() throws IOException
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the Number of Processes : ");
    int n = Integer.parseInt(br.readLine());
    prc = new int[n];
    crash = 0;
    for(int i = 0; i < n; i++)
        prc[i] =1;
    code = n;
    do
    {
        System.out.println("\n\t1. Crash A Process");
        System.out.println("\t2. Recover A Process");
        System.out.println("\t3. Display New Coordinator");
        System.out.println("\t4. Exit");
        ch=Integer.parseInt(br.readLine());
        switch(ch)
        {
            case 1: System.out.println("\nEnter A Process To Crash");
                    int cp=Integer.parseInt(br.readLine());
                    if((cp > n) || (cp < 1))
                    {
                      System.out.println("Invalid Process! Enter A Valid Process");
                    }
                    else if((prc[cp - 1] == 1) && (code != cp))
                    {
                      prc[cp - 1] = 0;
                      System.out.println("\nProcess "+cp+ " Has Been Crashed");
                    }
```

```
        else if((prc[cp - 1] == 1) && (code == cp))
        {
          prc[cp - 1] = 0;
          election(n);
        }
        else
          System.out.println("\nProcess "+cp+" Is Already Crashed");
        break;

case 2: System.out.println("\nCrashed Processes Are: \n");
        for(int i = 0; i < n; i++)
        {
          if(prc[i] == 0)
          System.out.println(i+1);
          crash++;
        }
        System.out.println("Enter The Process You Want To Recover");
        int rp=Integer.parseInt(br.readLine());
        if((rp < 1) || (rp > n))
          System.out.println("\nInvalid Process. Enter A Valid ID");
        else if((prc[rp - 1] == 0) && (rp > code))
        {
          prc[rp - 1] = 1;
          System.out.println("\nProcess "+rp+" Has Recovered");
          code = rp;
          System.out.println("\nProcess "+rp+ " Is The New Coordinator");
        }
        else if(crash == n)
        {
          prc[rp - 1] = 1;
          code = rp;
          System.out.println("\nProcess "+rp+ " Is The New Coordinator");
          crash--;
        }
        else if((prc[rp - 1] == 0) && (rp < code))
        {
          prc[rp - 1] = 1;
          System.out.println("\nProcess "+rp+" Has Recovered");
        }
        else
          System.out.println("\nProcess "+rp+" Is Not A Crashed Process");
        break;
case 3:
        System.out.println("\nCurrent Coordinator Is " +code);
```

```
                    break;
                case 4:
                    System.exit(0);
                    break;
                default:
                    System.out.println("\nInvalid Entry!");
                    break;
            }//end switch
        }while(ch!=4);
    }//end of Bully()

    public static void main(String args[]) throws IOException
    {
        BullyAlgorithm ob=new BullyAlgorithm();
        ob.Bully();
    }
}
```

C:\Windows\System32\cmd.exe - java BullyAlgorithm

```
C:\Users\User\Desktop\sem8-exps-anish\DC\exp8>javac BullyAlgorithm.java

C:\Users\User\Desktop\sem8-exps-anish\DC\exp8>java BullyAlgorithm
Enter the Number of Processes :
5

        1. Crash A Process
        2. Recover A Process
        3. Display New Coordinator
        4. Exit
3

Current Coordinator Is 5

        1. Crash A Process
        2. Recover A Process
        3. Display New Coordinator
        4. Exit
1

Enter A Process To Crash
2

Process 2 Has Been Crashed

        1. Crash A Process
        2. Recover A Process
        3. Display New Coordinator
        4. Exit
1

Enter A Process To Crash
5

The Coordinator has Crashed !

 Enter the Initiator :
1
Process 1 called for Election.

Process 2 called for Election.

Process 3 called for Election.

Process 4 called for Election.

Process 5 called for Election.

Process 1 is In.
```

C:\Windows\System32\cmd.exe - java BullyAlgorithm

```
Process 4 called for Election.

Process 5 called for Election.

Process 1 is In.
Process 2 is Dead.
Process 3 is In.
Process 4 is In.
Process 5 is Dead.

 ***New Coordinator is 4 ***

        1. Crash A Process
        2. Recover A Process
        3. Display New Coordinator
        4. Exit
3

Current Coordinator Is 4

        1. Crash A Process
        2. Recover A Process
        3. Display New Coordinator
        4. Exit
2

Crashed Processes Are:

2
5
Enter The Process You Want To Recover
5

Process 5 Has Recovered

Process 5 Is The New Coordinator

        1. Crash A Process
        2. Recover A Process
        3. Display New Coordinator
        4. Exit
3

Current Coordinator Is 5

        1. Crash A Process
        2. Recover A Process
        3. Display New Coordinator
        4. Exit
```