Distributed Systems Lab
Car Hire Booking Distributed System

Team Members:
Tania Rajabally (2017130047)
Shruti Rampure(2017130048)
Rahul Ramteke(2017130049)

Batch: C

Problem Statement:
The purpose of this system is to monitor and control the bookings of cars in a distributed environment. All the features of a typical Car Hire Booking System are discussed here by considering a distributed system.

<u>ELECTION ALGORITHM</u>

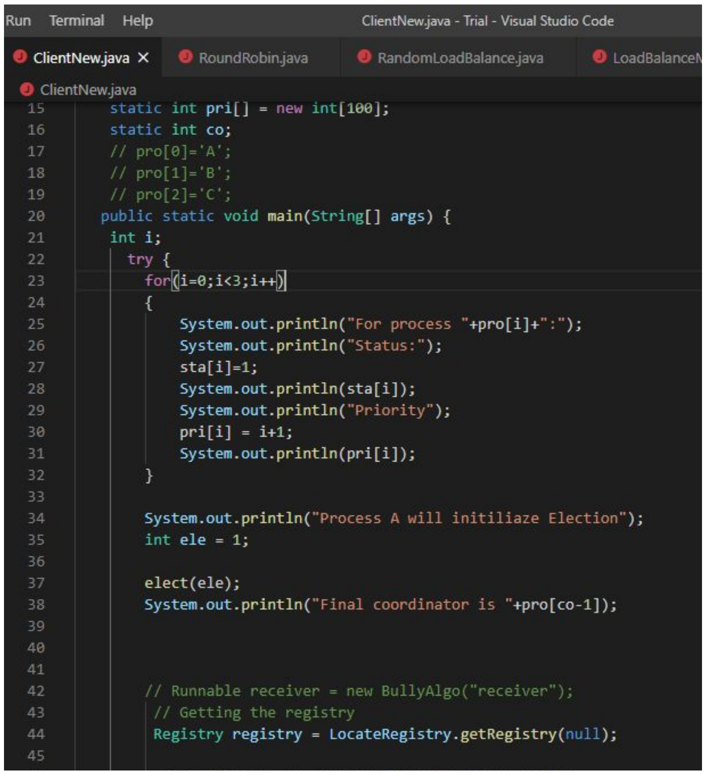
Election algorithms choose a process from a group of processors to act as a coordinator. If the coordinator process crashes due to some reasons, then a new coordinator is elected on another processor. Election algorithm basically determines where a new copy of the coordinator should be restarted.
Election algorithm assumes that every active process in the system has a unique priority number. The process with highest priority will be chosen as a new coordinator. Hence, when a coordinator fails, this algorithm elects that active process which has highest priority number.Then this number is send to every active process in the distributed system.

In distributed computing, the bully algorithm is a method for dynamically electing a coordinator or leader from a group of distributed computer processes. The process with the highest process ID number from amongst the non-failed processes is selected as the coordinator.

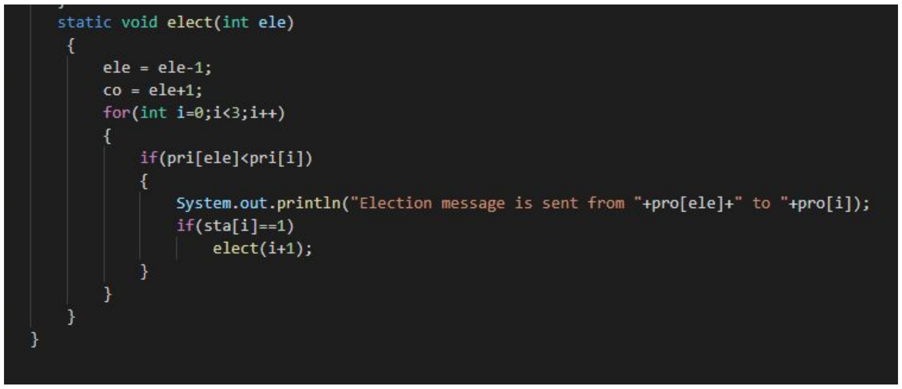
The algorithm assumes that:
- the system is synchronous.
- processes may fail at any time, including during execution of the algorithm.
- a process fails by stopping and returns from failure by restarting.
- there is a failure detector which detects failed processes.
- message delivery between processes is reliable.
- each process knows its own process id and address, and that of every other process.

Code:

ClientNew.java



```java
    static int pri[] = new int[100];
    static int co;
    // pro[0]='A';
    // pro[1]='B';
    // pro[2]='C';
   public static void main(String[] args) {
    int i;
      try {
        for(i=0;i<3;i++)
        {
            System.out.println("For process "+pro[i]+":");
            System.out.println("Status:");
            sta[i]=1;
            System.out.println(sta[i]);
            System.out.println("Priority");
            pri[i] = i+1;
            System.out.println(pri[i]);
        }

        System.out.println("Process A will initiliaze Election");
        int ele = 1;

        elect(ele);
        System.out.println("Final coordinator is "+pro[co-1]);



        // Runnable receiver = new BullyAlgo("receiver");
         // Getting the registry
         Registry registry = LocateRegistry.getRegistry(null);

```



```java
   static void elect(int ele)
    {
        ele = ele-1;
        co = ele+1;
        for(int i=0;i<3;i++)
        {
            if(pri[ele]<pri[i])
            {
                System.out.println("Election message is sent from "+pro[ele]+" to "+pro[i]);
                if(sta[i]==1)
                    elect(i+1);
            }
        }
    }
}
```

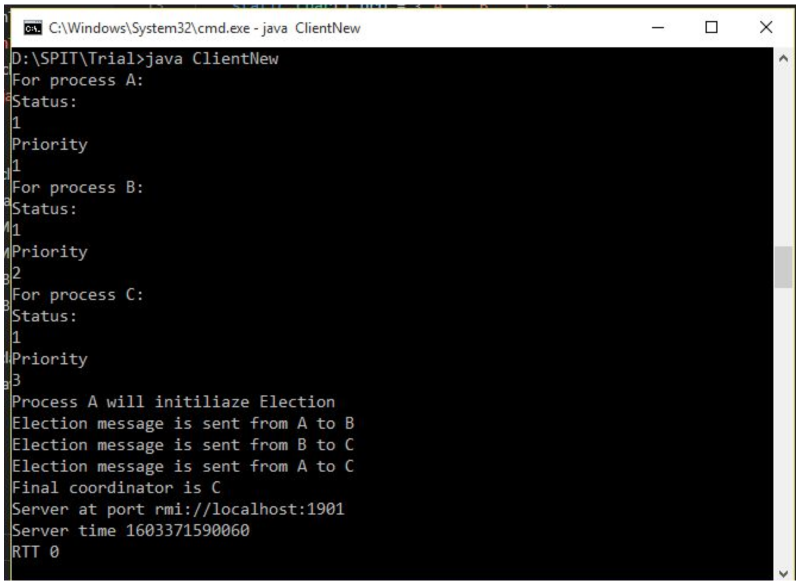
Explanation of Implementation:
There are 3 processes in our system- A,B and C. First the process with highest priority is selected as the coordinator. In our system process C has the highest priority. Each process can send a message to the coordinator. If the coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed. Now process P sends an election message to every process with a high priority number. It waits for responses, if no one responds for time interval T then process P elects itself as a coordinator. Then it sends a message to all lower priority number processes that it is elected as their new coordinator. However, if an answer is received within time T from any other process Q, Process P again waits for time interval T’ to receive another message from Q that it has been elected as coordinator or if Q

doesn’t respond within time interval T’ then it is assumed to have failed and the algorithm is restarted. This is how the bully algorithm is implemented in our system.

Steps to run:

1. Start rmiregistry
2. Compile the files : javac *.java
3. Run the command:
   java ServerA so that the serverA will be ready.
   In a new window run :
   Java ServerB so that serverB is ready.
   In a new window run :
   Java ServerC so that serverC is ready.
4. In a new window, run the command: java ClientNew to run the client side program. The process with highest priority is selected using the bully algorithm.

Screenshots:



```
D:\SPIT\Trial>java ClientNew
For process A:
Status:
1
Priority
1
For process B:
Status:
1
Priority
2
For process C:
Status:
1
Priority
3
Process A will initiliaze Election
Election message is sent from A to B
Election message is sent from B to C
Election message is sent from A to C
Final coordinator is C
Server at port rmi://localhost:1901
Server time 1603371590060
RTT 0
```

Process C has the highest priority in our system. Hence it is selected as the coordinator. Election messages are sent from the processes to the coordinator.

Conclusion:

Bully algorithm was used for implementing election algorithm. Every process can send a message to every other process in our car booking system. This algorithm has been integrated in our system.