****

**Name of the Student: choudhari Arti Rajendra Roll no: 08**

**CLASS:- B.E.[I.T] Division: A Course:- 2015**

**Assignment No.4**

**COMPUTER LABORATORY-IX**

**Marks: /10**

**Date of Performance: Sign with Date:**

## ASSIGNMENTNO.4

**ProblemStatement:**

To develop Token Ring distributed algorithm for leader election.

**Objective:**

1) The course aims to provide an understanding of the principles on which the distributed systems are based; their architecture, algorithms and how they meet the demands of Distributed applications.

2) The course covers the building blocks for a study related to the design and the implementation of distributed systems and applications.

**Outcomes:**

1) Demonstrate knowledge of the core concepts and techniques in distributed systems.

2) Learn how to apply principles of state-of-the-Art Distributed systems in practical application.

3) Design, build and test application programs on distributed systems.

**PEOs:2; POs: a,b,c,d,f,g,i, l, m ; PSOs: 1,2,3 and COs satisfied: 1, 2, 3.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assignment No.** | **Assignment Title** | **Assignment Statement** | **Scenarios** | **Software Required** |
| 4 | Election Algorithm | To develop any distributed algorithm for leader election. | 1. The program can be used in situations when the block chain related transactions are involved | 1) JDK version 8 mandatory  2) eclipse java(EE) version (June 2017) (Oxygen)  / Netbeans 8.2 (optional). |

## Tools /Environment:

Java Programming Environment, JDK 1.8, Eclipse Neon(EE).

## RelatedTheory:

## ElectionAlgorithm:

1. Many distributed algorithms require a process to act as acoordinator.
2. The coordinator can be any process that organizes actions of otherprocesses.
3. *A* coordinator mayfail.
4. How is a new coordinator chosen orelected?

## Assumptions:

Each process has a unique number to distinguishthem. Processes know each other's process number.

There are two types of Distributed Algorithms:

1. BullyAlgorithm
2. RingAlgorithm

## Bully Algorithm:

1. **When a process, P, notices that the coordinator is no longer responding to requests,it initiates anelection.**
2. P sends an ELECTION message to all processes with highernumbers.
3. If no one responds, P wins the election and becomes acoordinator.
4. If one of the higher-ups answers, it takes over. P’s job isdone.

## When a process gets an ELECTION message from one of itslower-numbered colleagues:

* 1. Receiver sends an OK message back to the sender to indicate that he is alive and will take over.
  2. Eventually, all processes give up apart of one, and that one is the newcoordinator.
  3. The new coordinator announces *its* victory by sending all processes a**CO-ORDINATOR**

message telling them that it is the new coordinator.

## If a process that *was* previously down comesback:

* 1. It holds an election.
  2. If it happens to be the highest process currently running, it will win the election andtake over the coordinatorsjob.

## “Biggest guy" always wins and hence the name bully algorithm.

**Ring Algorithm:**

**Initiation:**

1. When a process notices that coordinator is notfunctioning:
2. Another process (initiator) initiates the election by sending "ELECTION" message (containing its own processnumber)

## Leader Election:

1. Initiator sends the message to it's successor (if successor is down, sender skips over it and goes to the next member along the ring, or the one after that, until a running process islocated).
2. At each step, sender adds its own process number to the list in themessage.
3. When the message gets back to the process that started it all: Message comes back toinitiator. In the queue the **process with maximum ID Numberwins**.

Initiator announces the winner by sending another message around the ring.

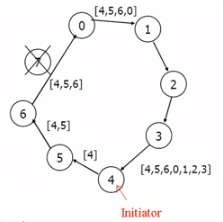
## Designing thesolution:

* 1. **For Ring Algorithm Initiation:**

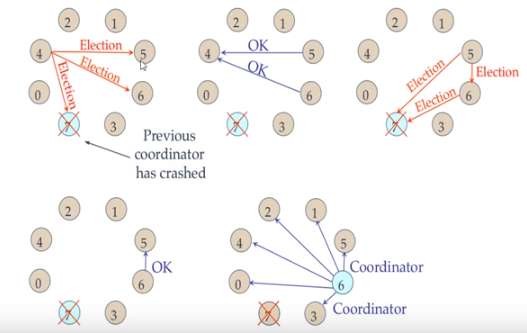
1. Consider the Process 4 understands that Process 7 is notresponding.
2. Process 4 initiates the Election by sending "ELECTION" message to it's successor (or next alive process) with it's ID.

## Leader Election:

1. Messages comes back to initiator. Here the initiator is4.
2. Initiator announces the winner by sending another message around the ring. Here the process with highest process ID is 6. The initiator will announce that Process 6 isCoordinator.



## For BullyAlgorithm:



**Implementingthesolution: For Ring Algorithm:**

* + 1. Creating Class for Process whichincludes
       1. State: Active / Inactive
       2. Index: Stores index ofprocess.
       3. ID: ProcessID
    2. Import Scanner Class for getting input fromConsole
    3. Getting input from User for number of Processes and store them into object ofclasses.
    4. Sort these objects on the basis of processid.
    5. Make the last process id as"inactive".
    6. Ask for menu 1.Election2.Exit
    7. Ask for initializing electionprocess.
    8. These inputs will be used by RingAlgorithm.

**Compiling and Executing thesolution:**

1. Create Java Project in Eclipse
2. CreatePackage
3. Add class in package Ring.java.
4. Compile and Execute in Eclipse.

The output is associated in the above section.

**Conclusion:** Election algorithms **are designed to choose a coordinator.** We have two election algorithms for two different configurations of distributed system. **The Bully** algorithm applies to system where every process can send a message to every other process in the system and **The Ring** algorithm applies to systems organized as a ring (logically or physically). In this algorithm we assume that the link between the process are unidirectional and every process can message to the process on its rightonly.

**Code:-**

//Ring

import java.util.Scanner;

public class Ring {

public static void main(String[] args) {

// TODO Auto-generated method stub

int temp, i, j;

char str[] = new char[10];

Rr proc[] = new Rr[10];

// object initialisation

for (i = 0; i<proc.length; i++)

proc[i] = new Rr(); //implicit object declaration

// scanner used for getting input from console

Scanner in = new Scanner(System.in);

System.out.println("Enter the number of process : ");

int num = in.nextInt();

// getting input from users

for (i = 0; i<num; i++) {

proc[i].index = i;

System.out.println("Enter the id of process : ");

proc[i].id = in.nextInt();

proc[i].state = "active";

proc[i].f = 0;

}

// sorting the processes from on the basis of id

for (i = 0; i<num - 1; i++) {

for (j = 0; j <num - 1; j++) {

if (proc[j].id >proc[j + 1].id) //bubble sort condition

{

temp = proc[j].id;

proc[j].id = proc[j + 1].id;

proc[j + 1].id = temp;

}

}

}

for (i = 0; i<num; i++) {

System.out.print(" [" + i + "]" + " " + proc[i].id);

}

int init;

int ch;

int temp1;

int temp2;

int ch1;

int arr[] = new int[10];

proc[num - 1].state = "inactive";

System.out.println("\n process " + proc[num - 1].id + "select as co-ordinator");

while (true) {

System.out.println("\n 1.election 2.quit ");

ch = in.nextInt();

for (i = 0; i<num; i++) {

proc[i].f = 0;

}

switch (ch) {

case 1:

System.out.println("\n Enter the Process number who initialsiedelection : ");

init = in.nextInt();

temp2 = init;

temp1 = init + 1;

i = 0;

while (temp2 != temp1) {

if ("active".equals(proc[temp1].state) && proc[temp1].f == 0) {

System.out.println("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);

proc[temp1].f = 1; //f stands for flag

init = temp1;

arr[i] = proc[temp1].id;

i++;

}

if (temp1 == num) {

temp1 = 0;

} else {

temp1++;

}

}

System.out.println("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);

arr[i] = proc[temp1].id;

i++;

int max = -1;

// finding maximum for co-ordinator selection

for (j = 0; j <i; j++) {

if (max <arr[j]) {

max = arr[j];

}

}

// co-ordinator is found then printing on console

System.out.println("\n process " + max + "select as co-ordinator");

for (i = 0; i<num; i++) {

if (proc[i].id == max) {

proc[i].state = "inactive";

}

}

break;

case 2:

System.out.println("Program terminated ...");

return ;

default:

System.out.println("\n invalid response \n");

break;

}

}

}

}

class Rr {

public int index; // to store the index of process

public int id; // to store id/name of process

public int f;

String state; // indiactes whether active or inactive state of node

}

/\*OUTPUT

Enter the number of process :

5

Enter the id of process :

1

Enter the id of process :

2

Enter the id of process :

3

Enter the id of process :

4

Enter the id of process :

5

[0] 1 [1] 2 [2] 3 [3] 4 [4] 5

process 5select as co-ordinator

1.election 2.quit

1

Enter the Process number who initialsiedelection :

4

Process 5 send message to 1

Process 1 send message to 2

Process 2 send message to 3

Process 3 send message to 4

Process 4 send message to 5

process 5select as co-ordinator

1.election 2.quit

2

Program terminated ...\*/

//BULLY

import java.io.InputStream;//importing class inside the package

import java.io.PrintStream;

import java.util.Scanner;

public class Bully {

static boolean[] state = new boolean[5];

int coordinator;

public static void up(int up)//non-static to static is not allowed;hence we declared it

{

if (state[up - 1]) {

System.out.println("process" + up + "is already up");

} else {

int i;

Bully.state[up - 1] = true;

System.out.println("process " + up + "held election");

for (i = up; i< 5; ++i) {

System.out.println("election message sent from process" + up + "to process" + (i + 1));

}

for (i = up + 1; i<= 5; ++i) {

if (!state[i - 1]) continue;

System.out.println("alive message send from process" + i + "to process" + up);

break;

}

}

}

public static void down(int down) {

if (!state[down - 1]) {

System.out.println("process " + down + "is already dowm.");

} else {

Bully.state[down - 1] = false;

}

}

public static void mess(int mess) {

if (state[mess - 1]) {

if (state[4]) {

System.out.println("0K");

} else if (!state[4]) {

int i;

System.out.println("process" + mess + "election");

for (i = mess; i< 5; ++i) {

System.out.println("election send from process" + mess + "to process " + (i + 1));

}

for (i = 5; i>= mess; --i) {

if (!state[i - 1]) continue;

System.out.println("Coordinator message send from process" + i + "to all");

break;

}

}

} else {

System.out.println("Prccess" + mess + "is down");

}

}

public static void main(String[] args) {

int choice;

Scanner sc = new Scanner(System.in);//for the input accepting

for (int i = 0; i< 5; ++i) {

Bully.state[i] = true;

}

System.out.println("5 active process are:");

System.out.println("Process up = p1 p2 p3 p4 p5");

System.out.println("Process 5 is coordinator");

do {

System.out.println(".........");

System.out.println("1 up a process.");

System.out.println("2.down a process");

System.out.println("3 send a message");

System.out.println("4.Exit");

choice = sc.nextInt();

switch (choice) {

case 1: {

System.out.println("bring process up");

int up = sc.nextInt();

if (up == 5) {

System.out.println("process 5 is co-ordinator");

Bully.state[4] = true;

break;

}

Bully.up(up);

break;

}

case 2: {

System.out.println("bring down any process.");

int down = sc.nextInt();

Bully.down(down);

break;

}

case 3: {

System.out.println("which process will send message");

int mess = sc.nextInt();

Bully.mess(mess);

}

}

} while (choice != 4);

}

}

/\*OUTPUT

5 active process are:

Process up = p1 p2 p3 p4 p5

Process 5 is coordinator

.........

1 up a process.

2.down a process

3 send a message

4.Exit

1

bring process up

2

process2is already up

.........

1 up a process.

2.down a process

3 send a message

4.Exit

2

bring down any process.

4

.........

1 up a process.

2.down a process

3 send a message

4.Exit

3

which process will send message

4

Prccess4is down

.........

1 up a process.

2.down a process

3 send a message

4.Exit

3

which process will send message

2

0K

.........

1 up a process.

2.down a process

3 send a message

4.Exit

4 \*/