ASSIGNMENT NO. 6

Problem Statement :

Implement Bully and Ring algorithm for leader election.

Tools/Environment :

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

Theory :

Election Algorithm:

1. Many distributed algorithms require a process to act as a coordinator.

2. The coordinator can be any process that organizes actions of other processes.
3. A coordinator may fail.

4. How is a new coordinator chosen or elected?

Assumptions:

Each process has a unique number to distinguish them. Processes know each other's process numbers.

There are two types of Distributed Algorithms:

1. Bully Algorithm

2. Ring Algorithm

Bully Algorithm:

A . When a process, P, notices that the coordinator is no longer responding to requests, it initiates an election.

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

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

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

B. When a process gets an ELECTION message from one of its lower-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 a part of one, and that one is the new coordinator.

3. The new coordinator announces its victory by sending all processes a CO-ORDINATOR message telling them that it is the new coordinator.

C . If a process that previously down came back:

1. It holds an election.

2. If it happens to be the highest process currently running, it will win the election and take over the coordinator’s job.

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

Ring Algorithm:

Initiation:

1. When a process notices that the coordinator is not functioning:

2. Another process (initiator) initiates the election by sending an "ELECTION" message (containing its process number) Leader Election:

3. Initiator sends the message to its 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 is located).

4. At each step, the sender adds its own process number to the list in the message.

5. When the message gets back to the process that started it all: The message comes back to the initiator. In the queue, the process with the maximum ID Number wins. Initiator announces the winner by sending another message around the ring.

Designing the solution:

A . For Ring Algorithm

Initiation:

1. Consider the Process 4 understands that Process 7 is not responding.

2.Process 4 initiates the Election by sending an "ELECTION" message to its

successor (or next alive process) with its ID. Leader Election:

3.Messages come back to the initiator. Here the initiator is 4.

4. Initiator announces the winner by sending another message around the ring. Here the process with the highest process ID is 6. The initiator will announce that Process 6 is Coordinator.

B. For Bully Algorithm:

1. Creating Class for Process which includes

   i) State: Active / Inactive

   ii) Index: Stores index of process.

   iii) ID: Process ID

2. Import Scanner Class for getting input from Console

3. Getting input from the User for several Processes and store them into an

object of classes.

4. Sort these objects based on process id.

5. Make the last process id as "inactive".

6. Ask for menu 1.Election 2.Exit

7. Ask for initializing the election process.

8.These inputs will be used by Ring Algorithm.

Compiling and Executing the solution:

1. Create Java Project in Eclipse

2. Create Package

3. Add a class in the package Ring.java.

4. Compile and Execute in Eclipse.

Conclusion:

Election algorithms are designed to choose a coordinator. We have two election algorithms for two different configurations of the distributed systems. The Bully algorithm applies to the 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 is unidirectional and every process can message to the process on its right only.