ASSIGNMENT NO. 06

TITLE: ELECTION

AIM: To implement Bully and Ring algorithm for leader election.

OBJECTIVE:

Students should be able to understand:-

- Election algorithms
- The Bully Algorithm
- The Ring Algorithm

PROBLEM STATEMENT:-

Write a program to implement Bully and Ring algorithm for leader election.

TOOLS / ENVIRONMENT:

- S/W:
  - Can be developed on any platform Windows /Linux.
  - C/C++/Java Language
- H/W:
  - Any basic configuration loaded machine (e.g. P IV)

THEORY:

Distributed Algorithm is a algorithm that runs on a distributed system. Distributed system is a collection of independent computers that do not share their memory. Each processor has its own memory and they communicate via communication networks. Communication in networks is implemented in a process on one machine communicating with a process on other machine. Many algorithms used in distributed system require a coordinator that performs functions needed by other processes in the system. **Election algorithms** are designed to choose a coordinator.

Election Algorithms:
Election algorithms choose a process from group of processors to act as a coordinator. If the coordinator process crashes due to some reasons, then a new coordinator is elected on other processor. Election algorithm basically determines where a new copy of 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.

We have two election algorithms for two different configurations of distributed system.

1. The Bully Algorithm –
This algorithm applies to system where every process can send a message to every other process in the system.

Algorithm – Suppose process P sends a message to the coordinator.

1. If coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.
2. Now process P sends election message to every process with high priority number.
3. It waits for responses, if no one responds for time interval T then process P elects itself as a coordinator.
4. Then it sends a message to all lower priority number processes that it is elected as their new coordinator.
5. However, if an answer is received within time T from any other process Q,
    a. Process P again waits for time interval T' to receive another message from Q that it has been elected as coordinator.
    b. If Q doesn't responds within time interval T' then it is assumed to have failed and algorithm is restarted.

2. The Ring Algorithm –
This 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 right only. Data structure that this algorithm uses is active list, a list that has priority number of all active processes in the system.

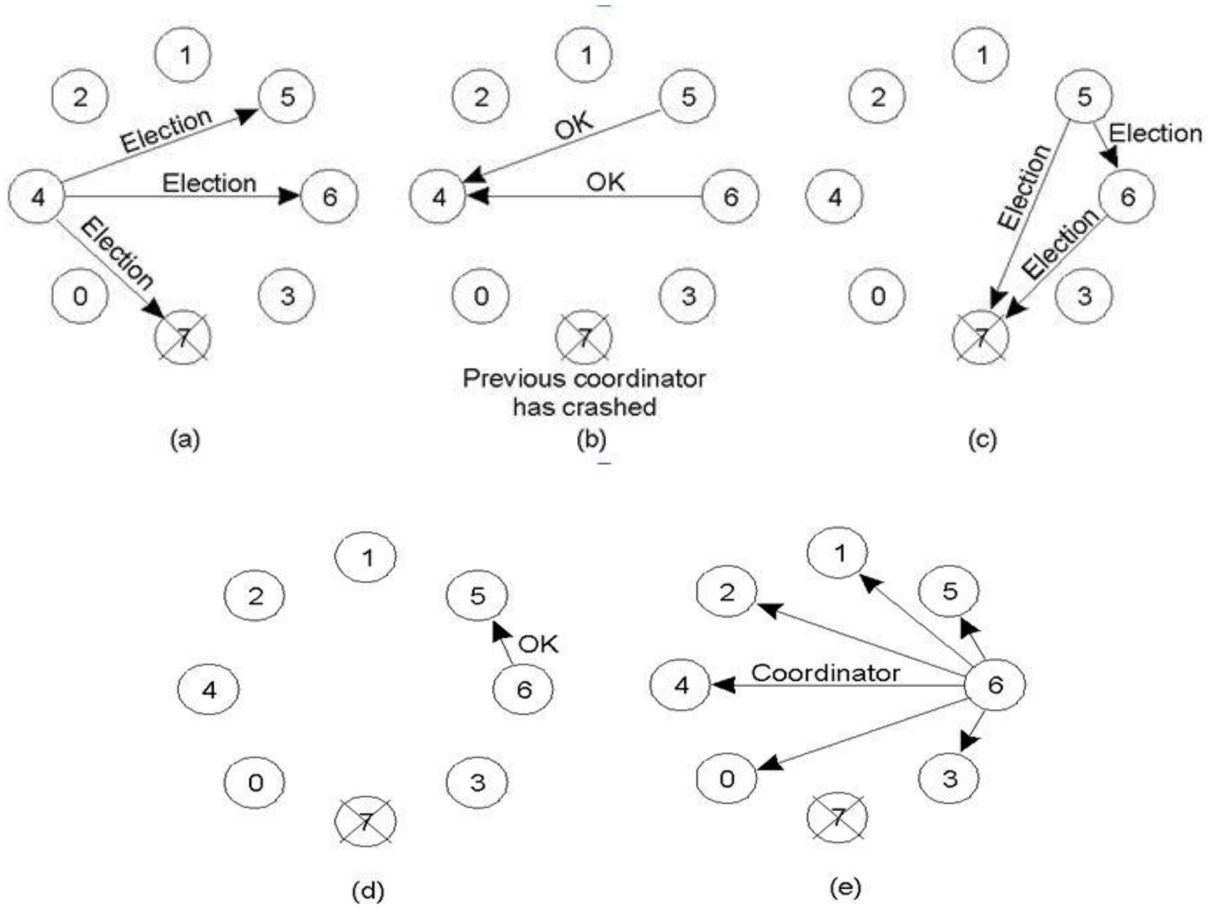
Algorithm –
1. If process P1 detects a coordinator failure, it creates new active list which is empty initially. It sends election message to its neighbor on right and adds number 1 to its active list.
2. If process P2 receives message elect from processes on left, it responds in 3 ways:
    a. If message received does not contain 1 in active list then P1 adds 2 to its active list and forwards the message.
    b. 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.
    c. 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.

Designing the solution:
Bully Algorithm:-

1. Each node has access to some permanent storage that survives node failures.
2. There are no transmission errors.
3. The communication subsystem does not fail

Algorithm :

1. The bully election algorithm
2. Process 4 holds an election
3. Process 5 and 6 respond, telling 4 to stop
4. Now 5 and 6 each hold an election
5. Process 6 tells 5 to stop
6. Process 6 wins and tells everyone

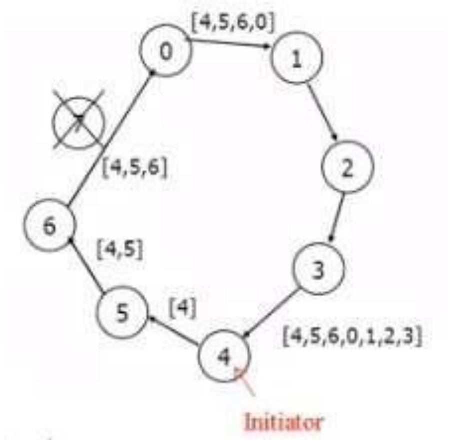


Ring Algorithm:-
Initiation:

1. Consider the Process 4 understands that Process 7 is not responding.
2. Process 4 initiates the Election by sending "ELECTION" message to it's successor (or next alive process) with it's ID.

Leader Election:

3. Messages comes back to initiator. Here the initiator is 4.
4. 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 is Coordinator.



Implementing the solution:

Assumptions and Requirements
Any process can call for an election.
A process can call for at most one election at a time.
Multiple processes can call an election simultaneously.
The result of an election should not depend on which process calls for it.
Each process has
Variable called elected
An attribute value called attr, e.g., id, MAC address, CPU
The non-faulty process with the best (highest) election attribute value (e.g., highest id or address, or fastest cpu, etc.) is elected.
Requirement: A run (execution) of the election algorithm must always guarantee at the end:

- Safety: ∀ P (P's elected = (q: non-failed process with the best attribute value) or ⊥)
- Liveness: ∀ election((election terminates)
    - & ∀ P: non-faulty process, P's elected is not ⊥)

Bully Algorithm

When a process notices that the coordinator is no longer responding to requests, it initiates an election. A 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 he 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 take over the coordinator's job. Thus the biggest guy in town always wins, hence the name "bully algorithm."

For Ring Algorithm:

- *N* Processes are organized in a logical ring.
  - p_i has a communication channel to $p_{(i+1) \bmod N}$.
  - All messages are sent clockwise around the ring.
- Any process p_i that discovers a coordinator has failed initiates an "election" message *<i, p_i.attr>*
- When a process p_j receives an *election* message *<i, p_i.attr>*, it compares the *attr* in the message with its own.
  - If the arrived $p_i.attr > p_j.attr$, then receiver p_j forwards the message *<i, p_i.attr>*.
  - If the arrived $p_i.attr < p_j.attr$ and the receiver p_j has not forwarded an election message earlier, it substitutes its own *<j, p_j.attr>* in the message and forwards it.
  - If the arrived $p_i.attr = p_j.attr$, then this process's $p_j .attr$ must be the greatest, and it becomes the new coordinator. This process then sends an "elected" message to its neighbor announcing the election result.
- When a process p_i receives an elected message, it
  - sets its variable $elected_i \leftarrow$ id of the message.
  - forwards the message if it is not the new coordinator**.**

- (attr:=id)
- The worst-case scenario occurs when the counter-clockwise neighbor has the highest *attr*
- A total of *N-1* messages is required to reach the new coordinator-to-be.
- Another *N* messages are required until the new coordinator-to-be ensures it is the new coordinator.
- Another *N* messages are required to circulate the elected messages.

IMPLEMENTATION:

(Students should write here the implementation for their program. Students should attach printout of their programs with commands used to run the programs. Also attach the proper outputs of programs.)

CONLCUSION:

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 right only.

REFERENCES:

- "**Distributed Systems: Concepts and Design**" by George Coulouris, J Dollimore and Tim Kindberg, Pearson Education, ISBN: 9789332575226, 5th Edition, 2017
- **"Distributed Systems"**, Maarten van Steen, Andrew S. T, Third edition Version

FAQ:

1. Why to conduct election in distributed system?
2. Explain bully algorithm with its advantages and disadvantages.
3. Explain ring algorithm with its advantages and disadvantages.
4. Compare bully and ring algorithm.
5. Suppose that two processes detect the demise of the coordinator simultaneously and both decide to hold an election using the bully algorithm. What happens?