EXPERIMENT No. 4

Date of Performance:

Date of Submission:

Aim : To implement Election algorithm

Objectives/ Requirements:

1) Create a dynamic group
2) Create number of processes in the group
3) Calculate number of processes in the group.
4) Decide on the election algorithm (Optional)
5) Elect coordinator.

Theory:

Principle: Many distributed algorithms require that some process acts as a coordinator. The question is how to select this special process dynamically. Note: In many systems the coordinator is chosen by hand (e.g., file servers, DNS servers). This leads to centralized solutions => single point of failure.

– Doesn't matter which process does the job, just needto pick one

– Example: pick a master in Berkeley clocksynchronization algorithm

• Election algorithms: technique to pick a uniquecoordinator

– Assumption: each process has a unique ID

– Goal: find the non-crashed process with the highest ID

Types of election algorithm:

1) Bully's algorithm

Assumptions

– Each process knows the ID and address of every otherprocess

– Communication is reliable

– Need consistent result

• A process initiates an election if it just recovered from failure or it notices that the coordinator has failed

• Three types of messages: Election, OK, Coordinator.

• Several processes can initiate an election simultaneously.

Algorithm:

- Any process P can initiate an election
- P sends Election messages to all process with higher ID sand awaits OK messages

– If no OK messages, P becomes coordinator and sends Coordinator messages to all processes with lower IDs

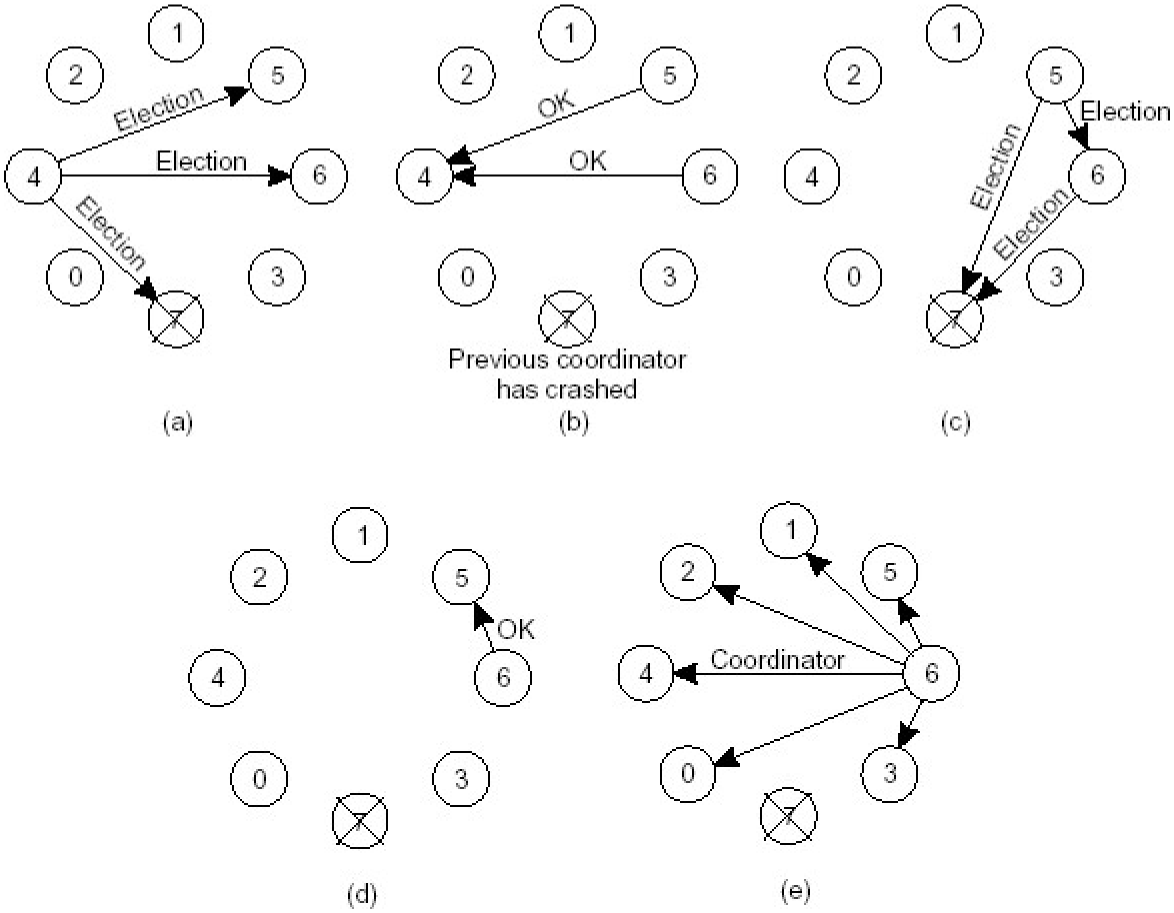
– If it receives an OK, it drops out and waits for an Coordinator message

- If a process receives an Election message

– Immediately sends Coordinator message if it is the process with highest ID

– Otherwise, returns an OK and starts an election

- If a process receives a Coordinator message, it treats sender as the coordinator.



(a) (b) (c)

(d) (e)

2) Ring Algorithm:

Processes are arranged in a logical ring, each process knows the structure of the ring

• A process initiates an election if it just recovered from failure or it notices that the coordinator has failed

• Initiator sends Election message to closest downstream node that is alive

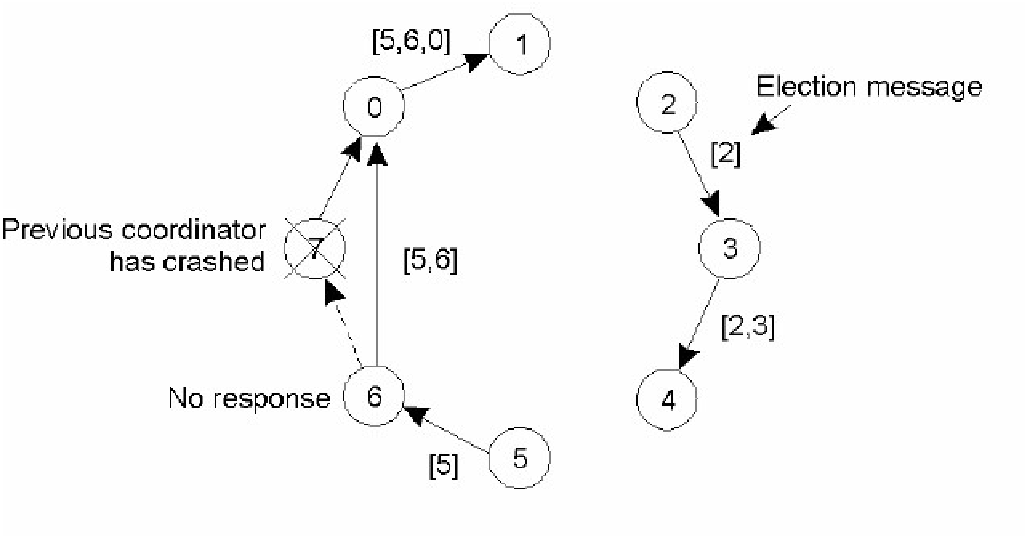
– Election message is forwarded around the ring

– Each process adds its own ID to the Election message

– When Election message comes back, initiator picks node with highest ID and sends a Coordinator message specifying the winner of the election

– Coordinator message is removed when it has circulated once.

• Multiple elections can be in progress



Assume n processes and one election in progress

• Bully algorithm

– Worst case: initiator is node with lowest ID

• Triggers n-2 elections at higher ranked nodes: O(n2) messages

– Best case: initiator is node with highest ID

• Immediate election: n-1 messages

• Ring algorithm

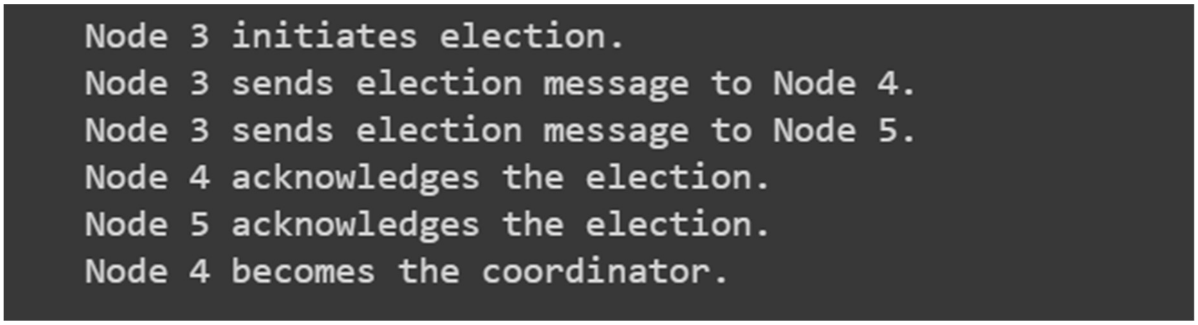
– 2n messages always

CODE:

Bully's algorithm

```
class BullyElectionAlgorithm:
    def __init__(self, nodes):
        self.nodes = nodes
        self.coordinator = None

    def start_election(self, initiating_node):
        print(f"Node {initiating_node} initiates election.")
        higher_nodes = [node for node in self.nodes if node > initiating_node]
        if not higher_nodes:
            self.coordinator = initiating_node
            print(f"Node {self.coordinator} becomes the coordinator.")
            return

        higher_nodes.sort()
        for node in higher_nodes:
            print(f"Node {initiating_node} sends election message to Node {node}.")

        for node in higher_nodes:
            print(f"Node {node} acknowledges the election.")

        for node in higher_nodes:
            if self.check_alive(node):
                self.coordinator = node
                print(f"Node {self.coordinator} becomes the coordinator.")
                return

    def check_alive(self, node):
        # Placeholder function to check if node is alive
        return True

# Example usage:
nodes = [1, 2, 3, 4, 5]
election = BullyElectionAlgorithm(nodes)
election.start_election(3)
```

Output:



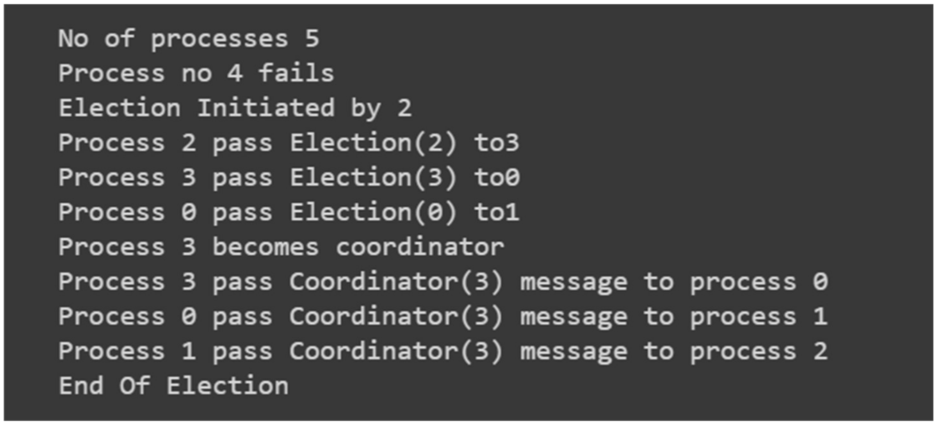
```
Node 3 initiates election.
Node 3 sends election message to Node 4.
Node 3 sends election message to Node 5.
Node 4 acknowledges the election.
Node 5 acknowledges the election.
Node 4 becomes the coordinator.
```

Code :

Ring Algorithm

```
class Pro:
  def __init__(self, id):
    self.id = id
    self.act = True

class GFG:
  def __init__(self):
    self.TotalProcess = 0
    self.process = []

  def initialiseGFG(self):
    print("No of processes 5")
    self.TotalProcess = 5
    self.process = [Pro(i) for i in range(self.TotalProcess)]

  def Election(self):
    print("Process no " + str(self.process[self.FetchMaximum()].id) + " fails")
    self.process[self.FetchMaximum()].act = False
    print("Election Initiated by 2")
    initializedProcess = 2

    old = initializedProcess
    newer = old + 1

    while (True):
      if (self.process[newer].act):
                    print("Process " + str(self.process[old].id) + " pass Election(" +
str(self.process[old].id) + ") to" + str(self.process[newer].id))
        old = newer
      newer = (newer + 1) % self.TotalProcess
      if (newer == initializedProcess):
        break

    print("Process " + str(self.process[self.FetchMaximum()].id) + " becomes coordinator")
    coord = self.process[self.FetchMaximum()].id

    old = coord
    newer = (old + 1) % self.TotalProcess
    while (True):
      if (self.process[newer].act):
          print("Process " + str(self.process[old].id) + " pass Coordinator(" + str(coord) + ")
message to process " + str(self.process[newer].id))
        old = newer
      newer = (newer + 1) % self.TotalProcess
      if (newer == coord):
        print("End Of Election ")
        break
  def FetchMaximum(self):
    maxId = -9999
    ind = 0
    for i in range(self.TotalProcess):
      if (self.process[i].act and self.process[i].id > maxId):
        maxId = self.process[i].id
        ind = i
    return ind

def main():
  object = GFG()
  object.initialiseGFG()
  object.Election()

if __name__ == "__main__":
  main()
```

Output:



```
No of processes 5
Process no 4 fails
Election Initiated by 2
Process 2 pass Election(2) to3
Process 3 pass Election(3) to0
Process 0 pass Election(0) to1
Process 3 becomes coordinator
Process 3 pass Coordinator(3) message to process 0
Process 0 pass Coordinator(3) message to process 1
Process 1 pass Coordinator(3) message to process 2
End Of Election
```

: Conclusion:

Successfully implemented Election algorithm.

SIGN AND REMARK

DATE

R1 **(4 Marks)**	**R2** **(4 Marks)**	**R3** **(4 Marks)**	**R4** **(3 Mark)**	**Total** **(15 Marks)**	**Signature**