Experiment 4

Name: Rebecca Dias **Roll No.: 18** **Class: BE CMPN A** **PID: 182027**

Aim: Implementation of Election Algorithm (Bully/Ring Algorithm)

Theory:

Election Algorithms

- In Election algorithms, a process from a group of processors is chosen to act as a central coordinator.
- If the coordinator process crashes due to some reasons, then a new coordinator is elected by another 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, a new coordinator gets elected with an active process which has the highest priority number.Then this number is sent to every active process in the distributed system.
- In the election algorithm it is assumed that processes have unique IDs, such that one is highest and it is assumed that the priority of process Pi is i
- We have two election algorithms for two different configurations of distributed systems.

(a) Bully Algorithm:

This algorithm applies to systems where every process can send a message to every other process in the system.

Background: any process Pi sends a message to the current coordinator; if no response in T time units, Pi tries to elect itself as leader.

Details follow:

Algorithm – Suppose process P sends a message to the coordinator as the lack of coordinator is detected.

1. If the coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.
2. Now process P sends an election message to every process with a 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,

   I. Process P again waits for time interval T' to receive another message from Q that it has been elected as coordinator.

   II. If Q doesn't respond within the time interval T' then it is assumed to have failed and the algorithm is restarted.

Below diagram shows the working of Bully Algorithm.

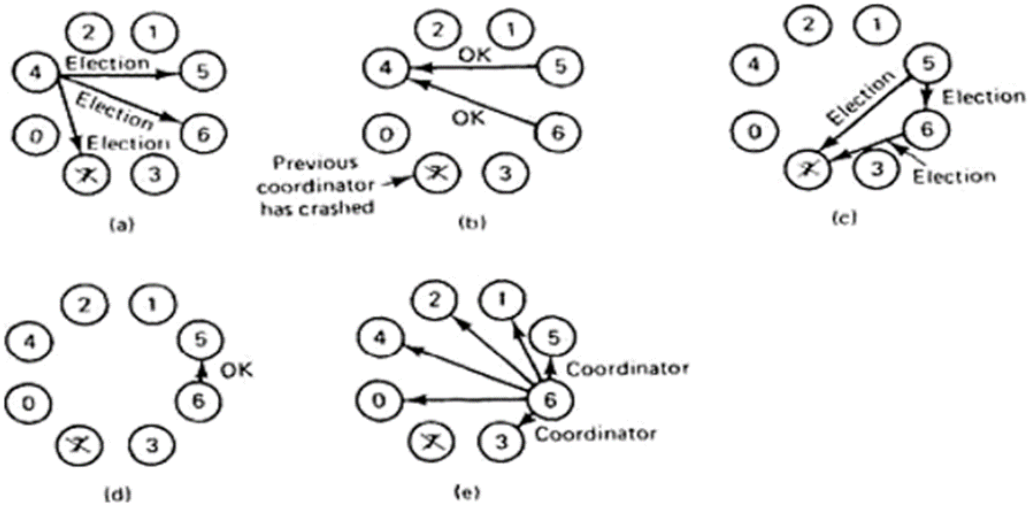

Fig 4.1: Working of Bully Algorithm

(b) 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 processes are unidirectional and every process can message to the process on its right only.
- Data structure that this algorithm uses is an active list, a list that has the priority number of all active processes in the system.

Algorithm

If process P1 detects a coordinator failure, it creates a new active list which is empty initially. It sends election messages to its neighbor on the right and adds number 1 to its active list.

If process P2 receives message elect from processes on left, it responds in 3 ways:

(I) If the message received does not contain 1 in the active list then P1 adds 2 to its active list and forwards the message.

(II) If this is the first election message it has received or sent, P1 creates a new active list with numbers 1 and 2. It then sends election message 1 followed by 2.

(III) If Process P1 receives its own election message 1 then the active list for P1 now contains numbers of all the active processes in the system. Now Process P1 detects the highest priority number from the list and elects it as the new coordinator.

Below diagram shows the working of Ring Algorithm.

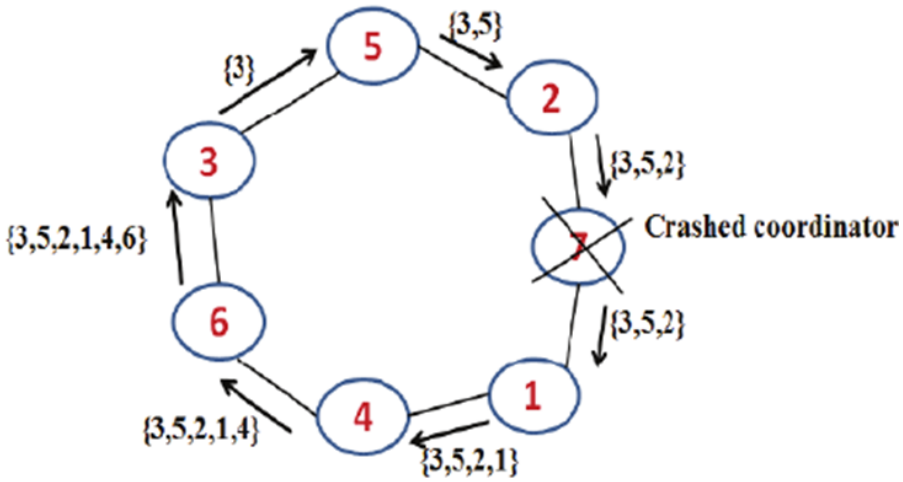

Fig 4.2: Working of Ring Algorithm

Program

```
import random

n = int(input("Enter the number of processes: "))

processes = []
prev_request = -1
for i in range(n):
  processes.append(bool(random.getrandbits(1)))

processes[n-1] = False
print("\nCo-ordinator Process: "+ str(n) +"\nCo-ordinator Process Now Dead: "+
str(n) )

request = random.randrange(n)
print(processes)
print(request)

def bully(request):
  for j in range(request + 1, n):
    print("\n\tReply from process: " + str(j+1) + " OK" if processes[j] == 1
```

```
else "")

    if processes[j]:
      request = j
      break

while( request < n - 1 and prev_request != request ):
  print("\n\nRequesting Process : " + str(request + 1))
  for i in range(request + 1, n):
    print("\n\tSending Election To : " + str((i + 1)))
  print()
  prev_request = request
  bully(request)

print("\n\nElected Co-ordinator Process : " + str((request + 1)) + "\nSending
Message To All Other Process...")
for i in range(n):
  print("\n\tProcess " + str((request + 1)) + " Elected Sending To Process : "
+ str((i + 1)))
  print("\nAll Messages Sent!!!")
```

Output:

```
Enter the number of processes: 8

Co-ordinator Process: 8
Co-ordinator Process Now Dead: 8
[False, False, False, False, False, False, True, False]
7


Elected Co-ordinator Process : 8
Sending Message To All Other Process...

        Process 8 Elected Sending To Process : 1

All Messages Sent!!!

        Process 8 Elected Sending To Process : 2

All Messages Sent!!!

        Process 8 Elected Sending To Process : 3

All Messages Sent!!!

        Process 8 Elected Sending To Process : 4

All Messages Sent!!!

        Process 8 Elected Sending To Process : 5

All Messages Sent!!!

        Process 8 Elected Sending To Process : 6

All Messages Sent!!!

        Process 8 Elected Sending To Process : 7

 All Messages Sent!!!

          Process 8 Elected Sending To Process : 8

 All Messages Sent!!!
```

Conclusion:

In this experiment we learnt about the election algorithm which is basically a process to elect a central coordinator from a group of processors. Then we learnt about the Bully Algorithm and Ring algorithm which are two election algorithms for two different configurations of distributed systems. Then Bully Algorithm was implemented successfully and the output was recorded as well.