Election algorithm and distributed processing

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 another machine. Many algorithms used in the 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 a 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 the 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 a 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 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 messages 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.

- (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 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 a 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 neighbour 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:
   - (I) If message received does not contain 1 in 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 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 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.

C++

```
#include <iostream>
#include <vector>
#include <algorithm>

struct Pro {
  int id;
  bool act;
  Pro(int id) {
    this->id = id;
    act = true;
  }
};

class GFG {
 public:
  int TotalProcess;
  std::vector<Pro> process;
  GFG() {}
  void initialiseGFG() {
    std::cout << "No of processes 5" << std::endl;
    TotalProcess = 5;
    process.reserve(TotalProcess);
    for (int i = 0; i < process.capacity(); i++) {
      process.emplace_back(i);
    }
  }
  void Election() {
    std::cout << "Process no " << process[FetchMaximum()].id << " fails" << std::endl;
    process[FetchMaximum()].act = false;
```

```
    int old = initializedProcess;
    int newer = old + 1;

    while (true) {
      if (process[newer].act) {
        std::cout << "Process " << process[old].id << " pass Election(" << process[old].id << ") to"
        old = newer;
      }

      newer = (newer + 1) % TotalProcess;
      if (newer == initializedProcess) {
        break;
      }
    }

    std::cout << "Process " << process[FetchMaximum()].id << " becomes coordinator" << std::endl;
    int coord = process[FetchMaximum()].id;

    old = coord;
    newer = (old + 1) % TotalProcess;

    while (true) {

      if (process[newer].act) {
        std::cout << "Process " << process[old].id << " pass Coordinator(" << coord << ") message to
        old = newer;
      }
      newer = (newer + 1) % TotalProcess;
      if (newer == coord) {
        std::cout << "End Of Election " << std::endl;
        break;
      }
    }
  }
  int FetchMaximum() {
    int Ind = 0;
    int maxId = -9999;
    for (int i = 0; i < process.size(); i++) {
      if (process[i].act && process[i].id > maxId) {
        maxId = process[i].id;
        Ind = i;
      }
    }
    return Ind;
  }
};

int main() {
  GFG object;
  object.initialiseGFG();
  object.Election();
  return 0;
}
```

Java

```
import java.util.Scanner;

public class GFG {
```

```
        boolean act;
        Pro(int id)
        {
            this.id = id;
            act = true;
        }
    }
    int TotalProcess;
    Pro[] process;
    public GFG() { }
    public void initialiseGFG()
    {
        System.out.println("No of processes 5");
        TotalProcess = 5;
        process = new Pro[TotalProcess];
        int i = 0;
        while (i < process.length) {
            process[i] = new Pro(i);
            i++;
        }
    }
    public void Election()
    {
        System.out.println("Process no "
                           + process[FetchMaximum()].id
                           + " fails");
        process[FetchMaximum()].act = false;
        System.out.println("Election Initiated by 2");
        int initializedProcess = 2;

        int old = initializedProcess;
        int newer = old + 1;

        while (true) {
            if (process[newer].act) {
                System.out.println(
                    "Process " + process[old].id
                    + " pass Election(" + process[old].id
                    + ") to" + process[newer].id);
                old = newer;
            }

            newer = (newer + 1) % TotalProcess;
            if (newer == initializedProcess) {
                break;
            }
        }

        System.out.println("Process "
                           + process[FetchMaximum()].id
                           + " becomes coordinator");
        int coord = process[FetchMaximum()].id;

        old = coord;
        newer = (old + 1) % TotalProcess;

        while (true) {

            if (process[newer].act) {
                System.out.println(
                    "Process " + process[old].id
                    + " pass Coordinator(" + coord
                    + ") message to process "
```

```
            }
            newer = (newer + 1) % TotalProcess;
            if (newer == coord) {
                System.out.println("End Of Election ");
                break;
            }
        }
    }
    public int FetchMaximum()
    {
        int Ind = 0;
        int maxId = -9999;
        int i = 0;
        while (i < process.length) {
            if (process[i].act && process[i].id > maxId) {
                maxId = process[i].id;
                Ind = i;
            }
            i++;
        }
        return Ind;
    }

    public static void main(String arg[])
    {
        GFG object = new GFG();
        object.initialiseGFG();
        object.Election();
    }
}
```

Python3

```
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 = newer
```

```
        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) + '
                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()
```

C#

```
using System;

class GFG
{
  class Pro
  {
    public int id;
    public bool act;
    public Pro(int id)
    {
      this.id = id;
      act = true;
    }
  }

  int TotalProcess;
  Pro[] process;

  public GFG() { }

  public void initialiseGFG()
  {
    Console.WriteLine("No of processes 5");
    TotalProcess = 5;
    process = new Pro[TotalProcess];
    int i = 0;
```

```
    process[i] = new Pro(i);
    i++;
  }
}

public void Election()
{
  Console.WriteLine("Process no "
                    + process[FetchMaximum()].id
                    + " fails");
  process[FetchMaximum()].act = false;
  Console.WriteLine("Election Initiated by 2");
  int initializedProcess = 2;

  int old = initializedProcess;
  int newer = old + 1;

  while (true)
  {
    if (process[newer].act)
    {
      Console.WriteLine(
        "Process " + process[old].id
        + " pass Election(" + process[old].id
        + ") to" + process[newer].id);
      old = newer;
    }

    newer = (newer + 1) % TotalProcess;
    if (newer == initializedProcess)
    {
      break;
    }
  }

  Console.WriteLine("Process "
                    + process[FetchMaximum()].id
                    + " becomes coordinator");
  int coord = process[FetchMaximum()].id;

  old = coord;
  newer = (old + 1) % TotalProcess;

  while (true)
  {

    if (process[newer].act)
    {
      Console.WriteLine(
        "Process " + process[old].id
        + " pass Coordinator(" + coord
        + ") message to process "
        + process[newer].id);
      old = newer;
    }
    newer = (newer + 1) % TotalProcess;
    if (newer == coord)
    {
      Console.WriteLine("End Of Election ");
      break;
    }
  }
}
```

```
  {
    int Ind = 0;
    int maxId = -9999;
    int i = 0;

    while (i < process.Length)
    {
      if (process[i].act && process[i].id > maxId)
      {
        maxId = process[i].id;
        Ind = i;
      }
      i++;
    }
    return Ind;
  }

  static void Main(string[] args)
  {
    GFG obj = new GFG();
    obj.initialiseGFG();
    obj.Election();
  }
}
```

Javascript

```
class Pro {
  constructor(id) {
    this.id = id;
    this.act = true;
  }
}

class GFG {
  constructor() {
    this.TotalProcess = 0;
    this.process = [];
  }

  initialiseGFG() {
    console.log("No of processes 5");
    this.TotalProcess = 5;
    for (let i = 0; i < this.TotalProcess; i++) {
      this.process[i] = new Pro(i);
    }
  }

  Election() {
    console.log("Process no " + this.process[this.FetchMaximum()].id + " fails");
    this.process[this.FetchMaximum()].act = false;
    console.log("Election Initiated by 2");
    let initializedProcess = 2;

    let old = initializedProcess;
    let newer = (old + 1) % this.TotalProcess;

    while (true) {
      if (this.process[newer].act) {
        console.log(
          "Process " + this.process[old].id +
```

```
        old = newer;
      }

      newer = (newer + 1) % this.TotalProcess;
      if (newer === initializedProcess) {
        break;
      }
    }

    console.log("Process " + this.process[this.FetchMaximum()].id + " becomes coordinator");
    let coord = this.process[this.FetchMaximum()].id;

    old = coord;
    newer = (old + 1) % this.TotalProcess;

    while (true) {
      if (this.process[newer].act) {
        console.log(
          "Process " + this.process[old].id +
          " pass Coordinator(" + coord +
          ") message to process " +
          this.process[newer].id
        );
        old = newer;
      }
      newer = (newer + 1) % this.TotalProcess;
      if (newer === coord) {
        console.log("End Of Election ");
        break;
      }
    }
  }

  FetchMaximum() {
    let Ind = 0;
    let maxId = -9999;
    for (let i = 0; i < this.process.length; i++) {
      if (this.process[i].act && this.process[i].id > maxId) {
        maxId = this.process[i].id;
        Ind = i;
      }
    }
    return Ind;
  }
}

const object = new GFG();
object.initialiseGFG();
object.Election();
// this code is contributed by writer
```


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 0 pass Coordinator(3) message to process 1
Process 1 pass Coordinator(3) message to process 2
End Of Election
```

Time Complexity : O(n^2) in the worst case scenario, where n is the number of processes.

Space complexity : O(n)

Feeling lost in the world of random DSA topics, wasting time without progress? It's time for a change! Join our DSA course, where we'll guide you on an exciting journey to master DSA efficiently and on schedule.
Ready to dive in? Explore our Free Demo Content and join our DSA course, trusted by over 100,000 geeks!

- DSA in C++
- DSA in Java
- DSA in Python
- DSA in JavaScript

Recommended Problems

Frequently asked DSA Problems

Solve Problems

Last Updated : 08 Mar, 2023

26

Previous

Wave and Traversal Algorithm in Distributed System

Next

Client-Server Software Development | Introduction to Common Object Request Broker Architecture (CORBA)

Similar Reads

Components of Load Distributing Algorithm | Distributed Systems

Difference between Greedy Algorithm and Divide and Conquer Algorithm

Signal Processing and Time Series (Data Analysis)

Minimum difference between the highest and the smallest value of mines distributed

Components of Image Processing System

Satellite Image Processing

Introduction to Processing | Java

Using Vectors in Processing Language

Creative Programming In Processing | Set 2 (Lorenz Attractor)

MPI - Distributed Computing made easy

Complete Tutorials

Learn Algorithms with Javascript | DSA using JavaScript Tutorial

DSA Crash Course | Revision Checklist with Interview Guide

Learn Data Structures and Algorithms | DSA Tutorial

Mathematical and Geometric Algorithms - Data Structure and Algorithm Tutorials

Learn Data Structures with Javascript | DSA using JavaScript Tutorial

Article Contributed By :

N

NishuAggarwal

NishuAggarwal

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : devendrasalunke, nmkiniqw7b, modalaashwin41, dnyaneshwarsssss, davuluriaakashnag, mrtutrial

Article Tags : Algorithms-Misc , Algorithms , DSA , Misc

Practice Tags : Algorithms, Misc

Improve Article | Report Issue

A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

OpenCV Python Tutorial

Python Interview Question

DevOps

Git

AWS

Docker

Kubernetes

Azure

GCP

DevOps Roadmap

System Design

What is System Design

Monolithic and Distributed SD

High Level Design or HLD

Low Level Design or LLD

Crack System Design Round

System Design Interview Questions

Grokking Modern System Design

NCERT Solutions

Class 12

Class 11

Class 10

Class 9

Class 8

Complete Study Material

Commerce

Accountancy

Business Studies

Indian Economics

Macroeconomics

Microeconimics

Statistics for Economics

UPSC Study Material

Polity Notes

Geography Notes

History Notes

Science and Technology Notes

Economy Notes

Ethics Notes

Digital Logic Design

Engineering Maths

Competitive Programming

Top DS or Algo for CP

Top 50 Tree

Top 50 Graph

Top 50 Array

Top 50 String

Top 50 DP

Top 15 Websites for CP

JavaScript

TypeScript

ReactJS

NextJS

AngularJS

NodeJS

Express.js

Lodash

Web Browser

School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

Management & Finance

Management

HR Managament

Income Tax

Finance

Economics

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

SSC CGL Practice Papers