# Department of Computing

**Course Code: CS332**

**Class: BSCS-9ABC**

Lab 7: Synchronization, Election Algorithm

**Date: March 28, 2022**

**Time: 09:00-11:50am & 2:00-04:50 pm**

**Instructor: Shah Khalid, Dr. Farzana Jabeen**

|  |
| --- |
| **Submitted By:**  **Fatima Seemab**  **291310**  **BSCS 9B** |

Lab 7: Synchronization, Election Algorithm

Introduction

Distributed Algorithm is an 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.

**Objectives**

The objective of this lab is to understand the process of synchronization and learn the election algorithm by implementing the Bully and Ring algorithm.

**Tools/Software Requirement**

Select the programming language of your choice and choose the editor accordingly.

But you are encouraged to solve this lab using C/Java language.

Tools/Software: Visual Studio/ Eclipse

**Useful tutorial links:**

<https://www.youtube.com/watch?v=vKpc0y_Ik3E&ab_channel=Lastmomenttuitions>

**Description**

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.

If coordinator does not respond to it within a time interval T, then it is assumed that coordinator has failed.

Now process P sends election message to every process with high priority number.

It waits for responses, if no one responds for time interval T then process P elects itself as a coordinator.

Then it sends a message to all lower priority number processes that it is elected as their new coordinator.

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 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 –

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.

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.

**Task**

**Task 1**

|  |
| --- |
| Implement the Bully Algorithm. Enter the number of processes as seven “7”, assign the priorities to all the processes, show the sate of each process “active->1/inactive->0” , now assume process 3 initiate the election request and complete the task of selecting leader/ coordinator, now assume the chosen leader/coordinator fails and again repeat the process and now select the leader among the 6 remaining processes. |

**CODE:**

|  |
| --- |
| n=input("Enter the number of nodes: ") Nodes={} for i in range(int(n)):  Nodes[i]={}  priority = input("Enter the priority for the node with ID :" +str(i+1) +" ")  active=input("Enter the 1 for active status for the node with ID :"+ str(i+1)+" ")  Nodes[i]["node\_id"] = i+1  Nodes[i]["priority"] = priority  Nodes[i]["active"] = active  max\_pri=-1  for key,value in Nodes.items():  if max\_pri < int(Nodes[key]["priority"]) and Nodes[key]["active"]=="1":  id=Nodes[key]["node\_id"]  max\_pri = int(Nodes[key]["priority"])  if max\_pri == -1:  print("No node is active") else:  print("The coordinator chosen by the bully algorithm has ID:", max\_pri)  while True:  id = input("Enter the id of node going to run the election algorithm: ")  max\_pri = -1  for key,value in Nodes.items():  if int(Nodes[key]["node\_id"])>=int(id) :  print("message sent to id"+str(Nodes[key]["node\_id"]))  if max\_pri < int(Nodes[key]["priority"]) and Nodes[key]["active"] == "1":  max\_pri = int(Nodes[key]["priority"])   if max\_pri==-1:  print("No node is active")  else:  print("The coordinator chosen has ID:", max\_pri)   active\_update=input("Do you want to update the active status.Enter Y/N for Yes/No ")  if(active\_update=="Y"):  i=0  for key, value in Nodes.items():  active = input("Enter the 1 for active status for the node with ID " + str(i+1) + ":")  Nodes[i]["active"] = active  i += 1  else:  exit() |

**OUTPUTS:**

**For 3 nodes**

**Text

Description automatically generated**

**For 7 nodes:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Task 2**

|  |
| --- |
| Implement the Ring algorithm and perform the same tasks for this algorithm as mentioned in task 1. |

**CODE**

|  |
| --- |
| n=input("Enter the number of nodes: ") Nodes={}  for i in range(1,int(n)+1):  Nodes[i] = {}  Nodes[i]["node\_id"] = i  Nodes[i]["next"] = (i+1)%(int(n)+1)  if Nodes[i]["next"]==0:  Nodes[i]["next"]=1  for i in range(int(n)):  active = input("please enter the 1 for active status of node id # " + str(i+1))  Nodes[i+1]["active"] = active print(Nodes)  while True:  id = input("Enter the id of node going to run the election algorithm: ")  list=[]  for key,value in Nodes.items():  if int(Nodes[key]["node\_id"])==int(id) and Nodes[key]["active"]=="1":  list.append(Nodes[key]["node\_id"])  node=int(Nodes[key]["next"])  while node != int(id):  if Nodes[node]["active"] == "1":  list.append(node)  node = int(Nodes[node]["next"])  print("next",node)  print(list)  coordinator=max(list)  print("The cordinator chosen from election algorithm is:",coordinator)  active\_update = input("Do you want to update the active status.Enter Y/N for Yes/No")  if (active\_update == "Y"):  i = 1  for key, value in Nodes.items():  active = input("Enter the 1 for active status for the node with ID " + str(i) + ":")  Nodes[i]["active"] = active  i += 1 |

**OUTPUTS:**

**For 3 nodes:**

Text

Description automatically generated

**For 7 nodes:**

Graphical user interface, text

Description automatically generated

Useful Links:   
<http://www.c4learn.com/c-programs/c-program-for-bully-election-algorithm-election-algorithms-2.html>

**Deliverables**

Upload single word file with screenshots of your output and code on LMS.

In case of any confusion or query feel free to contact me at mehwish.kiran@seecs.edu.pk