ALGORITHM AND EXECUTION OF ELECTION ADMINISTRATION APPROACH IN SIMPLE CASE (SC) TO ELECT COORDINATOR FOR DISTRIBUTED SYSTEM

Dr. S.K Gandhi[*]

Pawan Kumar Thakur[**]

ABSTRACT

In fault-tolerant distributed systems the coordinator node has to perform some specific controlling tasks and this node is well known to the other nodes. When a crash failure of the coordinator node, it is urgently needed to reorganize the existing active nodes to call for an election and to elect a coordinator in order to continue the operation of the entire system. Election Administration is a collection of special processes in distributed system. It is an election administrative body. This body is authorized to handle the whole election process. In a distributed computing system, it defines the system and regulations for attending in an election process. The purposes of this paper are to presents algorithm and procedure to elect coordinator in simple case of election administration approach for distributed system.

Keywords: *Failure Detector, transmission delay, processing delay, Helper, verify message, alive message, Coordinator message, Process table.*

[*] Research Guide, Department of Computer Science & Engineering, AISECT University, Bhopal (M.P), India

[**] Ph. D Research Scholar, Department of Computer Science & Engineering, AISECT University, Bhopal (M.P), India

1. INTRODUCTION

When a process normally detects the failure of the coordinator process it sends election message to the Election Administration and waits for to receive coordinator message. Election Administration sends verify message to the current coordinator to be sure about the election message[5]. Election Administration sends alive message to the next highest process number to check either the current highest process is alive or not and Election Administration gets a reply message. Election Administration selects that process new coordinator of the system and sends coordinator message to all processes. In this paper we first section 2 present the requirements of election algorithm. In section 3 we present the algorithm and selection 4 execute our approach. Finally section 5 analysis and 6 conclude the paper.

2. REQUIREMENTS

Any Election Algorithm should satisfy the following two properties.

1. Safety. Any process *P*, has *ldr = NULL* if it is participating in the election, or its *ldr=P*, where P is the highest PID and it is alive at present.

2. Liveness. All the processes should agree on the chosen leader P after the election. That is, *ldr = Pn. Pi* where *i=1, 2,..., N[1,2,3]*.

3. ALGORITHM FOR ELECTION IN NORMAL CASE (NC)

The normal case approach of Election Administration is described by the pseudo-code in algorithm 1.1. The normal case of election is that where Election Administration using ELECTN_NC algorithm to elect the coordinator process.

Where:

N = Number of process

n= Process number[6,7]

ldr = Local variable containing the process id of the current coordinator process[4]

EA= Election Administration[5]

FD= Failure Detector[8]

Hp= Helper[5]

Ttrans= Maximum message transmission delay

Tprocess= Maximum message processing delay

T = 2Ttrans + Tprocess = Maximum(turnaround) time required to get a reply after sending a message to any process from *EA[5]*

Algorithm 1.1 ELECTN_NC

```
procedure ELECTN_NC
By default, the state of a process is ELECTION-OFF
Pre: process $i$ recognizes that coordinator process has crashed
$ldri \leftarrow NULL$
$State_i \leftarrow elect$
Post: New Coordinator process elected
$State_i \leftarrow normal$
/* performed by a process $Pi$, which triggers the election procedure */
$ldri \leftarrow NULL$
$Pi$ sends an election message to $E_A$
$State_i \leftarrow elect$
Wait($T = 2Ttrans + Tprocess$)
 $F_D$ of $E_A$ verifies election message sent by $P_i$.
 If ($P_i$ is not correct) then
     /*$E_A$ send current coordinator message to $P_i$ with $Pn$*/
       Broadcast($coordinator$, $j$)
       Procedure Update($ldr$, $i$)
       $State_i \leftarrow normal$
  If (($P_i$ is correct) && (highest $Pn$ is $P_i$)) then
       /*$E_A$ send a coordinator message to $N$ (all process)with $P_n$ of $P_i$
       Broadcast($coordinator$, $i$)
       Procedure Update($ldr$, $i$)
       $State_i \leftarrow normal$
 Else
/*$E_A$ find alive process with the highest process number using helper $H_{P*/}$
Find highest $Pn$ process $k$ from the $Hp$
        $E_A$ sends a coordinator message to all processes with the $P_n$ of new coordinator.
       Broadcast(coordinator, k)
       Procedure Update(ldr, i)
       $State_i \leftarrow normal$
 End if
 End if
End Procedure
```

Algorithm 2 UPDATE

```
Procedure UPDATE(var, val)
$Vari \leftarrow Val \mid \forall i$
End procedure
```

In the above algorithm 1.1 ELECTN_NC By default, the state of a process is ELECTION-OFF. When process *Pi* normally detect the failure of the coordinator process then the variable called *ldri*, which contains the process *id* of the current leader sets this lead to *NULL*. The state variable $state_i$ of *Pi* contains the value election. Process *Pi* sends an election message to Election Administration $E_{A.}$ Now the state variable $state_i$ of *Pi* contains the value *wait* (*T = 2Ttrans + Tprocess)* where T is the maximum (turnaround) time required to get a reply after sending a message to any process from Election Administration *EA*. Failure detector F_D of Election Administration E_A verifies election message sent by process $P_{i.}$ If the election message of process P_i is not correct and the current coordinator process is *Pj* then E_A broadcast current coordinator message to process P_i with process number *Pn* of process *Pj*. The procedures update change *ldr* (the Local variable containing the process id of the current coordinator process) value of process *Pi and* sets it process number *Pn* of process *Pj*. The state variable of *statei* contains the value *normal*. If the election message of process P_i is correct and the Process number *Pn* of *Pi* is highest then Election Administration E_A sends a coordinator message to *N* (all process) with process number *Pn* of process $P_{i.}$ Election Administration EA broadcast current coordinator message to all the process of the system and procedure update *ldr* value of and sets its process number *Pn* process *Pi*. The state variable of *statei* contains the value *normal*. Finally Election Administration E_A find alive process with the highest process number using helper H_P and find highest *Pn* process *k* from the helper *Hp*. The Election Administration *EA* sends a coordinator message to all processes of the system with the P_n of new coordinator. The procedure update change the *ldr* value of process *Pi* and sets it process number *Pn* of process *Pj*. the state variable of *statei* contains the value *normal.* Algorithm 5.3 UPDATE contain procedure update which sets *ldr* (the Local variable containing the process id of the current coordinator process) value of process any process.

4. EXECUTION

Election administration approach uses four types of message in normal case to elect any coordinator for distributed system. These are:

1. Election message. An *election message* is sent to announce an election.

2. Verify message. A *verify message* to the current coordinator.

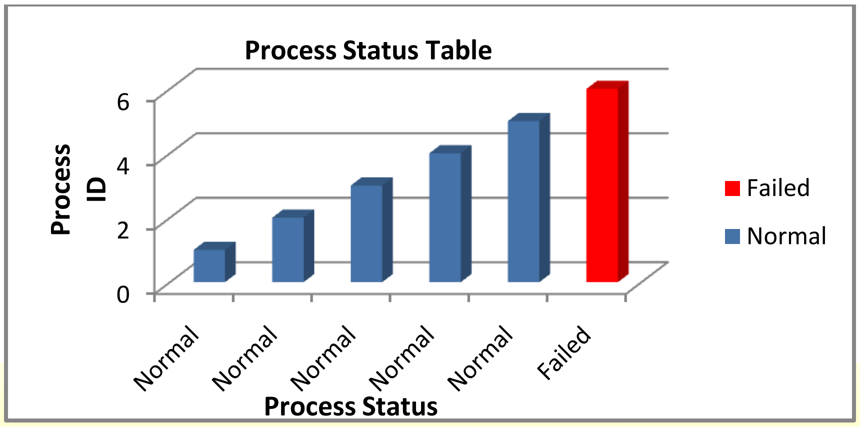
3. Alive message. An *alive message* to the next highest process number if the current coordinator is fail.

4. Coordinator message. *The Coordinator message* is send to all processes as a new coordinator of the system. The below Fig. 1.1(a), (b), (c) and (d) are represents normal election case of the proposed algorithm.

Step 1: The system consists of six processes with process number 1 to 6. Let the current coordinator be process with id 6 and the status of the other processes is as shown in table 1.1 and Graph 1.1. The status of all the processes except process 6 is NORMAL, which means they are alive in the system. Process 6 is the current coordinator.

Table 1.1. Process Status

Process ID	**Status**
1	NORMAL
2	NORMAL
3	NORMAL
4	NORMAL
5	NORMAL
6	COODINATOR

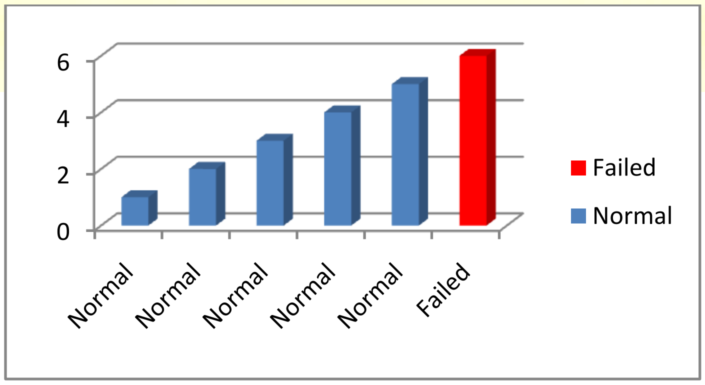


Graph 1.1 Process Status table

Now suppose at some point in time, a process 2 sends a REQUEST message to process 6, but does not receive a reply till its time out. Thus process 2 discovers that the coordinator with process id 6 has crashed /failed and so it is the time for an election as shown in process status table 1.2, process status Graph 1.2 and Fig.1.1 (a) .

Table 1.2. Process Status

Process ID	Status
1	NORMAL
2	NORMAL
3	NORMAL
4	NORMAL
5	NORMAL
6	FAILED



Graph 1.2 Process Status table

Now Process 2 sends an election message to the E*A* about the current coordinator failure.

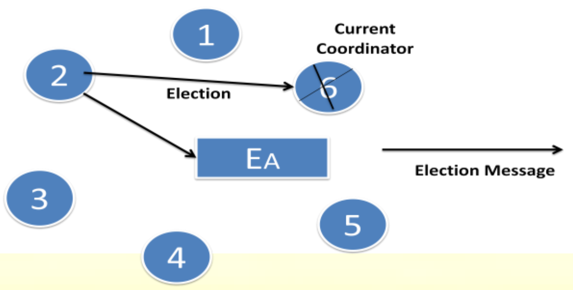


Fig 1.1(a) *Election Message*

Step II. EA sends verify message to the current coordinator to be sure about the election message sent by process 2. After verification as shown in Fig. 1.1(b).

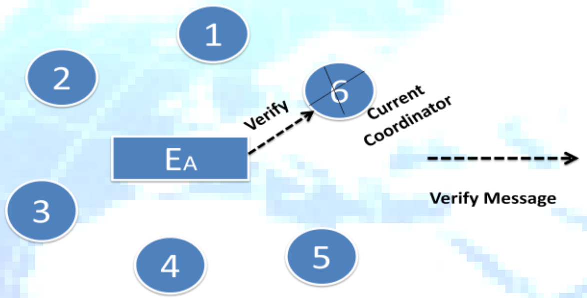


Fig. 1.1(b) Verify Message

Step III: *EA* sends alive message to process 5 (the next highest process number) to check either the current highest process is alive or not. And *EA* gets a reply message from *EA* gets a reply message from 5 as shown in Fig.1.1(c).

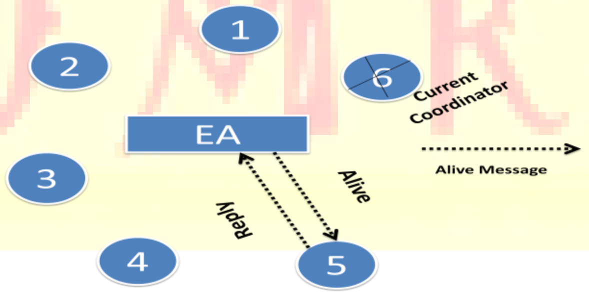


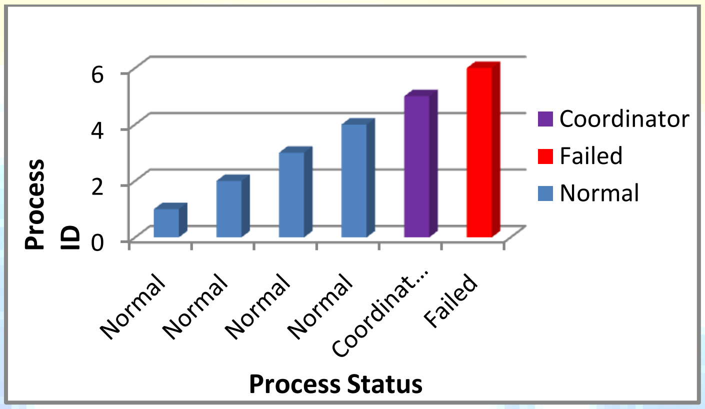
Fig. 1.1(c) Alive Message

1.1 (c) EA finds the alive process with highest number using alive message.

Step IV: A new message is broadcasted to all the processes informing about the new coordinator as shown in table 1.3 and process graph status table 1.3:

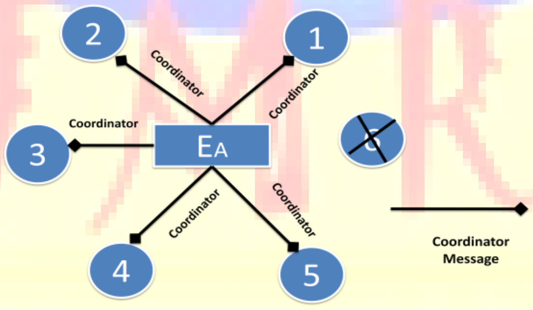
Table 1.3 Process status Table

Process ID	Status
1	NORMAL
2	NORMAL
3	NORMAL
4	NORMAL
5	COODINATOR
6	FAILED



Graph 1.3 Process Status table

EA select 5 as new coordinator and sends coordinator message to all processes having 5 as a new coordinator of the system as shown in Fig.1.1 (d).



1.1 (d) Coordinator Message

1.1 (d) EA sends coordinator message to all process having process number of currently won.

5. ANALYSIS

In **worst case** it may happen that our algorithm needs to check up process to p+1 to find out highest alive process. Only at that case it requires message passing between processes. However, in **best case**, our algorithm may find the highest alive process with only one alive and one reply message that is highest alive process in the system is process with process number n-1. In that case, our algorithm requires only 1+2+2+n messages.

6. CONCLUSION

In normal case of our approach there will be need of 1 election message to inform *EA*, 2 verify message to ensure the failure of coordinator, and say r is the highest alive process then alive and reply message to find out the highest alive process and so total or O(n) message passing between processes[9,10]. If the process with lowest process number detects coordinator as failed it will not change total message.

REFERENCES

[1] Sinha P.K, Distributed Operating Systems Concepts and Design, Prentice-Hall of India private Limited, (2008).

[2] H. Garcia-Molina, *"Elections in Distributed Computing System"*, IEEE Transaction Computer, Vol. C-31, pp.48- 59, Jan. (1982).

[3] Thakur P. Kumar , Kumar Ram, Ali Ruhi and Malviya Rajendra, *"A New Approach of Bully Election Algorithm for Distributed Computing"*, Int. J. of Electrical, Electronics and Computer Engineering (IJEECE) Vol 1(1): pp. 72-79(2011).

[4].Dr. Gandhi S.K. and Thakur P. Kumar, *Designing Issues for Distributed Computing System: An Empirical View,*" International Journal of Innovative Research & Development, Vol 1 Issue 11, pp. 326-40, Dec. (2012).

[5]. Dr. Gandhi S.K. and Thakur P. Kumar, *"Election Administration Algorithm for Distributed Computing"* International Journal of Electrical, Electronics and Computer Engineering 1(2): pp. 1-6(2012).

[6]. Dr. Gandhi S.K. and Thakur P. Kumar, *"Designing a Chat room application using peer-to-peer and client-server approach of Distributed Systems"*, Anusandhan: Science Technology & Management Journal of AISECT University, Vol II Issue III, pp. 95-100, March(2013).

[7]. Dr. Gandhi S.K. and Thakur P. Kumar, "*Analysis of Mutual Exclusion Algorithms with the significance and need of Election Algorithm to solve the coordinator problem for Distributed System"* International Journal on Emerging Technologies 4(1): pp.17-25(2013).

[8].Thakur P. Kumar, *"Reliable failure detector for Distributed System"*, IC-GISM: International Conference on Global Innovation in Science & Management, 2013, Swami Vivekanand University Sagar M.P. India.

[9]. Dr. Gandhi S.K. and Thakur P. Kumar, *"A comparative study of Bully and Ring Election Algorithm in the Terms of Complexity & Message passing"*, AERA: National Conference on Advances in Engineering Research & Application, March 14-16, 2013, Oriental College of Technology, Bhopal.

[10]. Dr. Gandhi S.K. and Thakur P. Kumar, *"Performance Analysis of Various Election Algorithms in Distributed System"* , BSS Journal of Computer Application, 2013.