A Comparative Study of Various Election Algorithms with Election Administration Approach for Distributed System

Dr. S.K Gandhi[1], Pawan Kumar Thakur[2]
[1]VYAPAM, GoMP, Bhopal (M.P) India.
[2]AISECT University, Bhopal (M.P) India.

ABSTRACT

In distributed computing systems when coordinator node fails the other nodes of the system need to elect another leader or coordinator to run the system smoothly. The bully algorithm is a classical approach for electing a leader in a distributed computing system. The improvement of the bully algorithm and other modify algorithms requiring less time complexity and minimum message passing over network. The performance analysis of bully and other modify election algorithm with our new proposed algorithm election administration would be appropriate to determine which algorithm performs better than the others. This paper represents the normal case of election in which we compare and discuss our approach with original bully algorithm and existing modified versions of bully algorithm.

Keywords: Election algorithm, Election administration, Distributed computing, failure, lowest priority, best case, detects, coordinator, worst case.

I INTRODUCTION

A distributed computing system is a collection of independent processors or node interconnected by a communication network. When a node fails it is often necessary to recognize the active nodes so that they continue to perform a useful task for their common goal. After the failure occurs in a distributed computing system the first step is to elect a coordinator node to manage the operation[1]. A process or node is used to coordinate many tasks. It is not an issue which process is doing the task, but there must be a coordinator that will work at any time. So electing a coordinator or a leader is very fundamental issue in distributed computing. There are many algorithms that are used in election process. Bully election algorithm is one of them. This works represents a modified version of bully algorithm using a new concept Election Administration. This approach will not only reduce redundant elections but also minimize total number of elections and hence it will minimize message passing, network traffic, and complexity of the existing system. In section 2 represents motivation and need of election algorithm section 3 objectives of study, section 4 represents objectives of study section 5 represents election administration approach, section 6 represents procedure of election, section 7 represents comparison and discussion of election administration with various election algorithm in normal case , section future work finally section 9 conclude the paper.

II MOTIVATION AND NEED OF ELECTION ALGORITHM

In distributed systems that involve multiple processes often utilize critical regions. When a process has to read or update certain shared structures it enters a critical section, performs its operation and leaves the critical section. We use special constructs to serialize access to critical sections (semaphores, monitors). These techniques do not support distributed systems because there is no single shared memory image. We need new techniques to achieve mutual exclusion. There are many algorithm used for distributed mutual exclusion. The main points to consider when evaluating of these algorithms are:

(a) What happens when messages are lost?
(b) What happens when a process or node crashes?

None of the algorithms have described to tolerate the loss of messages, process or node crashes. An algorithm for choosing a unique process to play a particular role is called an election algorithm. An election algorithm is needed for this choice.

III OBJECTIVES OF STUDY

The objects of study:

(a) To design a new Election Administration Approach to elect new coordinator for distributed system.

(b) To analysis of the Election Administration Approach.
(c) To compare Election Administration Approach with Bully Election Algorithm and different modify election algorithm.

IV LITERATURE REVIEW

Bully algorithm is one of the most famous election Algorithms which was proposed by Garcia-Molina in 1982. When a process P determines that the current coordinator is crashed because of message timeouts or failure of the coordinator to initiate a handshake, it executes bully election algorithm using the following sequence of actions[2]:

Step I. Process p sends an election message to all higher-numbered processes in the system. If no process responds then p becomes the coordinator. It wins the election and sends a coordinator message to all alive processes as shown in Fig.1.1 (a). Process 4 detects coordinator is failed and holds an election.

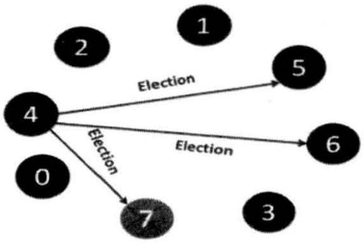


Fig.1.1 (a) Hold an Election

Step II. Process 5 and 6 respond to 4 to stop election they have higher priority than process 4as shown in Fig. 1.1(b).

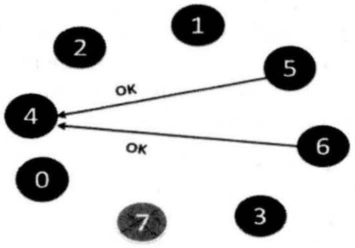


Fig. 1.2(b) Respond To Stop Election

Step III. Each of 5 and 6 process holds election now as shown in Fig.1.1(c). Process 6 responds to 5 to stop election.

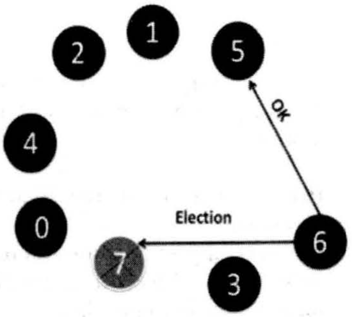


Fig.1.1(c) Election Message

Step V. The winner or new coordinator sends a message to other processes announcing itself as the new coordinator as shown in Fig. 1.1(d).

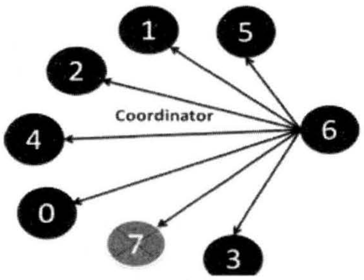


Fig.1.1(d) Coordinator Message

Immediately after the recovery of the crashed process is up, it runs bully algorithm. Bully algorithm has following limitations:

(a) The main limitation of bully algorithm is the highest number of message passing during the election and it has order $O(N^2)$ which increases the network traffic.

(b) As there is no guarantee on message delivery two processes may declare themselves as a coordinator at the same time.

(c) If the coordinator is running unusually slowly (say system is not working properly for some reasons) or the link between a process and a coordinator is broken for some reasons, any other process may fail to detect the coordinator and initiates an election. But the coordinator is up, so in this case it is a redundant election.

(d) Failure Detector is Unreliable Kordafshari et al. discussed the drawback of synchronous Garcia Molina's Bully Algorithm and modified it with an optimal message algorithm. They showed that their algorithm is more efficient than Garcia Molina's Bully algorithm, because of fewer message passing and fewer stages. Modified Bully algorithm by M.S. Kordafshari use following election process:

Step I. Process 2 detects coordinator process 6 is failed and holds an election by sending a election message to all the process who has higher priority than her as shown in Fig.1.2 (a).

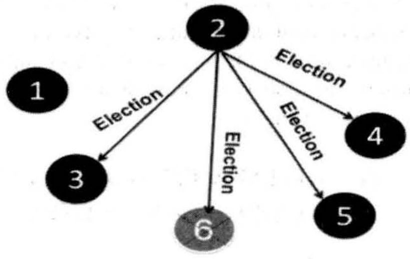


Fig.1.2(a) Hold a Election

Step II. Process 3, 4 and 5 respond with their process number as shown in the below Fig.4.2 (b).

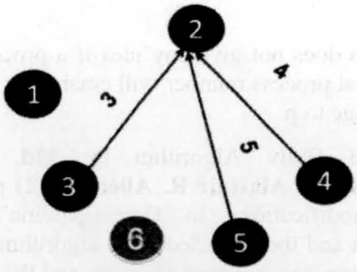


Fig.1.2 (b) Process Respond

Step III. Process 2 selects highest process number 5 and sends a grant message to 5 that means process 5 must be next leader as shown in Fig. 1.2(c).

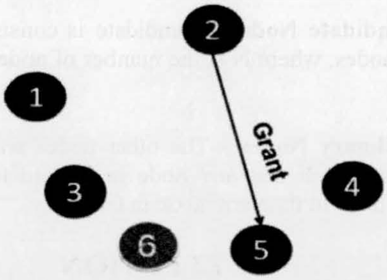


Fig.1.2(c) Grant Message

Step IV. Process 5 sends coordinator message to all processes coordinator will win again. This algorithm has following drawbacks.

(a) If a process p crashes after sending ELECTION message to higher processes or crashes after receiving priority number from higher processes, higher processes will wait for 3D (D is average propagation delay) time for coordinator broadcasting and if they do not receive any coordinator message, they will initiate modified algorithm again . If there are q different higher processes, then there will be q different individual instance of modified algorithm at that moment in the system. Those are redundant election[3].

(b) If process p sends GRANT message to the process with the highest priority number, and p does not receive COORDINATOR message from that process with in D time, p will repeats the algorithm, which is redundant election. As after any process with higher priority number compare to coordinator is up, it runs the algorithm, it increases redundant elections.

(c) Every redundant election takes resources, increases total message passing and increases network traffics.

Kabir Mamun et al. described an efficient version Bully algorithm to minimize redundancy in electing the coordinator and to reduce the recovery problem of a crashed process[4]. The operation of this algorithm is shown in Fig. 1.3.

Step I. Process 2 is the first one to notice this, so it sends election messages to all the processes higher than it, namely processes 3, 4, 5 and 6, as shown in Fig. 1.3(a).

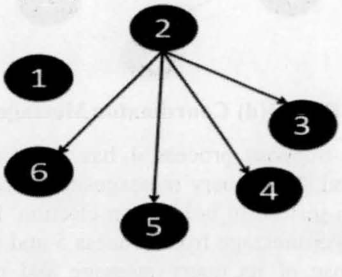


Fig.1.3 (a) Election Message

Step II. Processes 3, 4 and 5 are all response with ok message, as shown in Fig. 4.3(b).

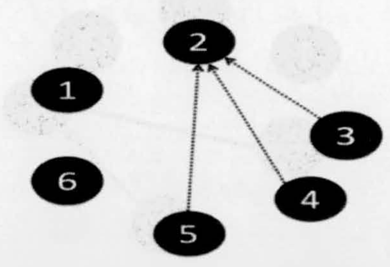


Fig.1.3 (b) Ok Message

Step III. Process 2 now knows who the alive process with highest process number is. So it elects process 5 as the new coordinator and sends coordinator messages to all other processes, as shown in Fig. 1.3 (c). The election is finished at this point. Every process knows process 5 as the new coordinator.

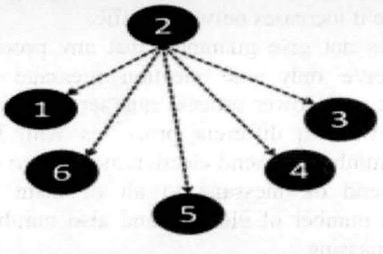


Fig.1.3(c) Coordinator Message

Step IV. Now process 4 has just crashed and process 6 has recovered from failure. As process 6 knows that it is the process with highest process number, it just sends coordinator message to all processes as shown in Fig. 1.3 (d) and becomes the new coordinator.

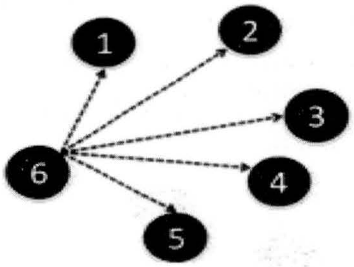


Fig.1.3(d) Coordinator Message

Step V. Suppose process 4 has recovered from failure and sends query messages to process 5 and process 6 instead of holding an election. Process 4 gets answer message from process 5 and process 6 in response of its query message and process 4 comes to know process 6 as the new coordinator.

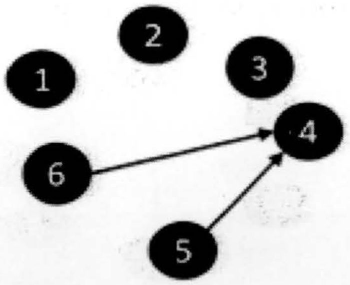


Fig.1.3(e) Answer Message

Although this algorithm reduces redundant election on some extent, it still has some redundant elections and also has high message complexity.

(a) On recovery, it sends query message to all processes with higher process number than it, and all of them will send answer message if they alive. Which increases total number of message passing and hence it increases network traffic.

(b) It does not give guarantee that any process p will receive only one election message from processes with lower process number. As a result there may be q different processes with lower process number can send election message to p and p will send ok message to all of them. This increases number of election and also number of message passing.

(c) It does not give any idea if p will crash after sending an election message to all processes with higher process number.

(d) It also does not give any idea if a process with the highest process number will crash after sending ok message to p.

Enhanced Bully Algorithm by **Md. Golam Murshed and Alastair R. Allen** (2012) proposes some modifications to Garcia-Molina's bully algorithm and the modified bully algorithm[6]. The basic system assumptions are as in, and the types of message used are very similar to the modification proposed in kazi kabir Mamun algorithm. In proposed they are proposing a set division. According to this concept, all the nodes of a synchronous distributed system are divided into two sets: Candidate nodes and Ordinary nodes.

(a) Candidate Nodes - Candidate is consisting of [N/2] nodes, where N is the number of nodes in the system.

(b) Ordinary Nodes - The other nodes will be in Ordinary, such that any node in Candidate has a higher node id than any node in Ordinary.

V ELECTION ADMINISTRATION APPROACH

Analyzing the shortcomings of bully and other modified bully algorithm we purposed a new approach. According to this concept, all the nodes of a synchronous distributed system are divided into two group: admin's and ordinary nodes or process. Admin's is comprised of (20: 1) nodes, where N is the number of nodes in the system. The other nodes will be in Ordinary, such that any node in Admin's has a higher node id than any node in Ordinary[7]. The election Administration architecture as shown in Fig. 1.4 made up with following type of processes: The election administration architecture consist following type of process.

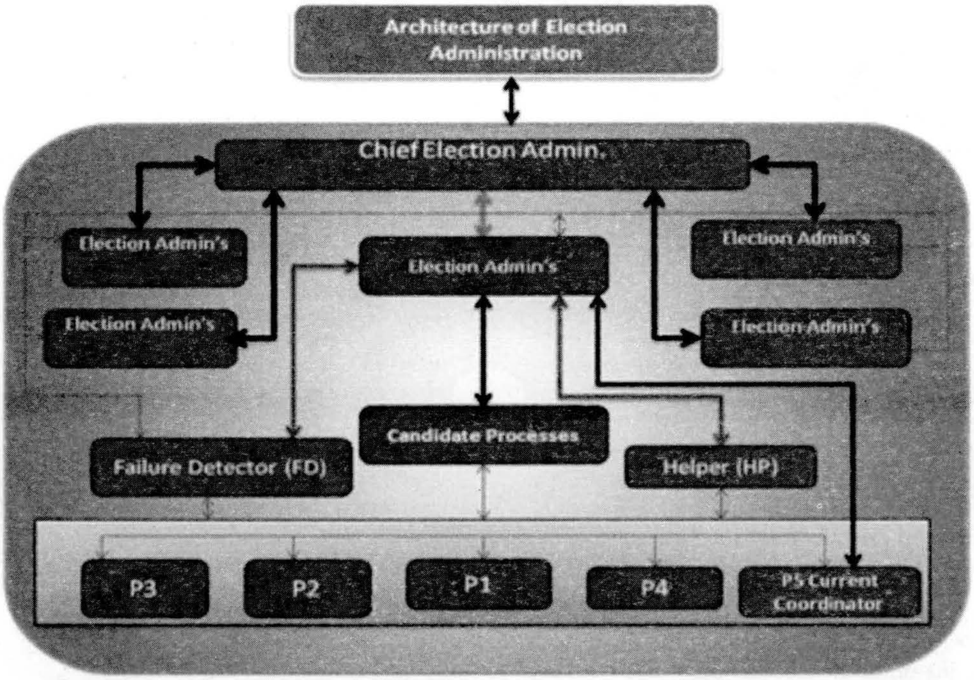


Fig.1.4 Architecture of Election Administration

(a) Chief Election Admin - Chief Election Admin CE_A is the coordinator process of Election Administration. It administrates other Election Admin's and handles F_D and H_P. This body works with one Chief Election Admin (*CEA*) and a few Election admin's. The process with the highest priority in Election Admin's will be the Chief Election Admin.

(b) Election Admin - Election Admin is a member of Election Administration. It is a special kind of process. Election Administration in a distributed system will have a few numbers of Election Admin's. All of them consult with the Chief Election Admin under the rules and regulation while there will be a need of an election. If any of the Admin's failed, Chief Election Administration will recover that admin's immediately and other processes (admin's) do not have concern of that. An Election Admin's has a unique group ID. Other processes in the system communicate with Election Admin's using this group ID.

(c) Ordinary Processes - These are general processes of the distributed system that performs a useful task to achieve a common goal. As we know that the system is synchronized, all these processes agreed to work with a coordinator which is select by the Election Administration. If the current coordinator fails then the process first knows can hold election for current coordinator.

(d) Failure Detector (FD) and Helper (HP) - As a result, if any of the admin's is down, there will be not any problem in election. It has a reliable failure detector (F_D). If maximum message transmission delay is Ttrans and maximum message processing delay is Tprocess then maximum time required to get a reply after sending a message to any process from Election Administration is T = 2Ttrans + Tprocess. If Election Administration does not get any reply from a process within T time, then F_D of Election Administration will report that requested process is down. Election Administration has another component named helper (H_P), the function of H_P is to find out the process with the highest process number using sending alive message. It knows process number of all processes of the system. As the system is synchronous and Election Administration has a failure detector F_D and helper H_P to solve limitations which is mentioned in previous chapter. Our proposed algorithm is briefly described below. There are total five types of message use in our approach for election process.

(i) Election Message. An *election message* is sent to announce an election.

(ii) Verify Message. A *verify message* to the current coordinator.

(iii) Alive Message. An *alive message* to the next highest process number if the current coordinator is fail.

(iv) Coordinator Message. *The Coordinator message* is send to all processes as a new coordinator of the system.

(v) Query Message. A *query message* is send when a crashed process is up. The up process can query to the election admin about the current coordinator.

VI PROCEDURE OF ELECTION

In normal case when a process normally detects the failure of the coordinator process it sends election message to the *EA* and waits for to receive coordinator message. *EA* sends verify message to the current coordinator to be sure about the election and sends alive message to the next highest process number to check either the current highest process is alive or not and gets a reply message. *EA* selects that process new coordinator of the system and sends coordinator message to all processes.

Step 1: The system consists of six processes with process number 1 to 6. Let the current coordinator be process with id 6. 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 Fig.1.5(a). Now Process 2 sends an election message to the EA about the current coordinator failure.

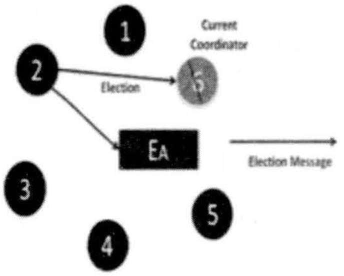


Fig.1.5 (a) Election Message

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. 1.5(b).

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

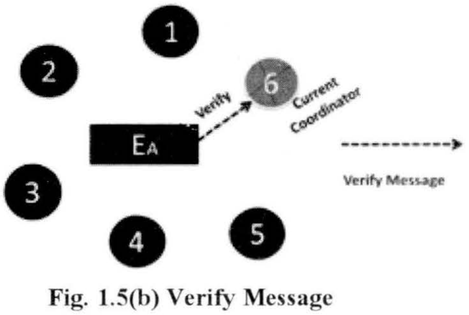


Fig. 1.5(b) Verify Message

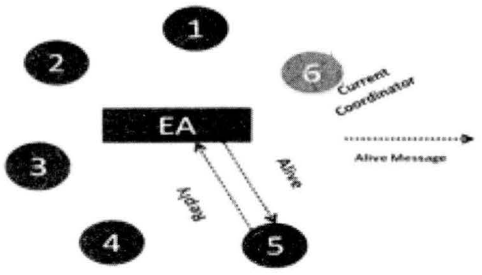
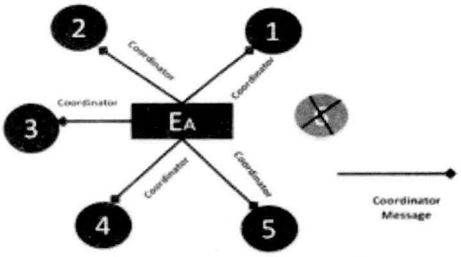


Fig.1.5(c) Alive Message

Step IV: A new message is broadcasted to all the processes informing about the new coordinator. *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.1.5 (d). 1.5 (d) *EA* sends coordinator message to all process having process number of currently won.



1.5 (d) Coordinator Message

In Query After Crash Recovery Case (QCRC) case when a process ordinary recover from failure it send a query message to the *EA*.. If the query processes number is higher than *EA* elect that process as current coordinator and send the coordinator message to all other process of the system. If a process with lower number it sends coordinator message of current coordinator of the system. The following Fig. 1.6(a), (b), (c), (d) and (e) represents the test execution of election administration approach in query after crash recovery case (QCRC).

Step I. The status of last coordinator is failed. Now the last crashed coordinator 6 is up and sends a query message to *EA*. If the last crashed coordinator 6 is up and sends a query message to *EA* as shown in Fig.1.6 (a).

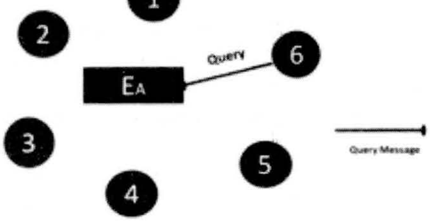


Fig.1.6 (a) Query Message

Step II. As process number of 6 is higher than the current coordinator of the system. The status of the process 6 will be coordinator. *EA* sends coordinator message to all processes with process number 6 as new coordinator [Fig.1.6 (b)].

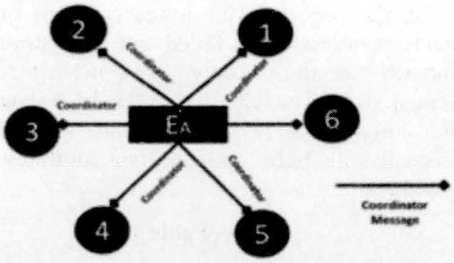


Fig.1.6 (b) Coordinator Message

1.6 (b) *EA* selects 6 as new coordinator and sends coordinator message to all processes.

Step III. Process 1 is now just Crashed [Fig. 1.6(c)].

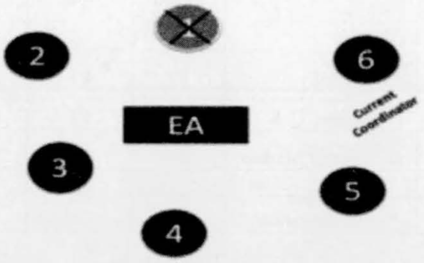


Fig.1.6(c) Crash Node

Step IV. Process 1 is just up after crashed, and it sends a query message to *EA*. *EA* checks that process number of newly entranced is lower than the current coordinator as shown in Fig. 1.6(d). Again process 1 is up and sends query message to *EA*.

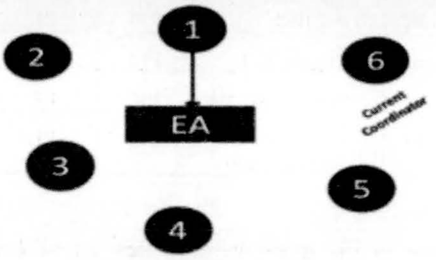


Fig.1.6(d) Query Message

Step V. *EA* sends coordinator message to only process 1 having the process number of current coordinator of the system [Fig. 1.6(e)].

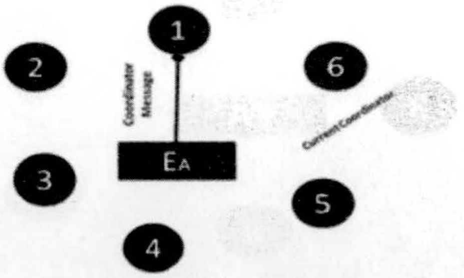


Fig.1.6 (e) Coordinator Message

Fig.1.6 (e) *EA* sends coordinator message to process 1 having the current coordinator.

In concurrent election case (CEC) case when more than one process may detect that the coordinator process has crashed. They will send election message to *EA* After verification will consider election request of the process having higher process number.

Step I. For example there are five processes in the system and the coordinator is process 5. In Fig. 1.7(a-d) represents the test execution of election administration approach in concurrent election case (CEC). Process 4 and 1 detect that coordinator 5 is down.

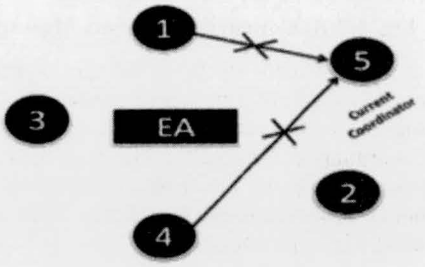


Fig. 1.7(a) Concurrent Process Detect Coordinator Down

Step II. To election a new coordinator 4 and 1 will send election message to *EA* as shown in Fig. 1.7(b).

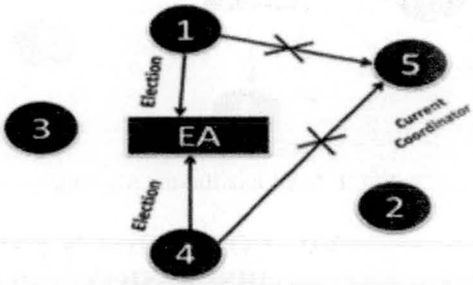


Fig. 1.7 (b). Election Message

Step III. *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. 1.7(c).

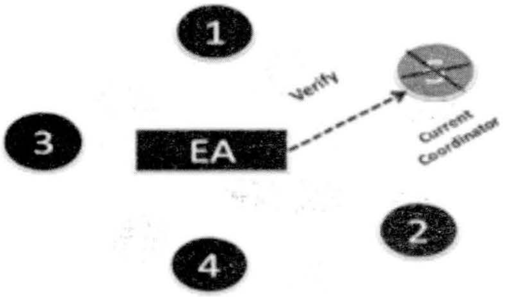


Fig. 1.7(c) Verification Message

Step IV. After verification EA only consider election message of process 4. It ensures less message passing to find out the highest process number. If *EA* considers election message of 1, then according to our algorithm, *EA* will have to send alive message to 4 to find higher process number.

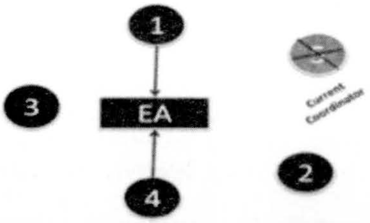


Fig. 1.7(d) Concurrent Process Message

But if *EA* considers election message of 4 it does not need to send alive message because 4 is already the higher process number and *EA* can select 4 as new coordinator. *EA* sends coordinator message to all processes having 4 as a new coordinator of the system as shown in above Fig. 1.7(e). This was *EA* can ensure less message passing.

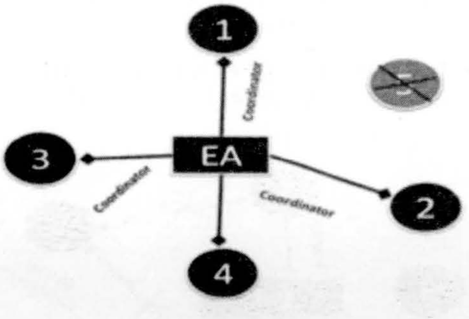


Fig. 1.7(e) Coordinator Message

VII COMPARISON AND DISCUSSION

In the normal case when a process with the lowest process number detects coordinator as failed, then it requires message passing. Garcia-Molina's bully algorithm requires O(N^2) messages to elect a coordinator node. The modified Bully Algorithm proposed in Mamun, Q. K. gains a significant improvement in the worst case. It requires only O(n) messages to elect a new coordinator node in the worst case[2]. The algorithm proposed by Md. Golam Murshed and Alastair R. Allen also requires at most 2(N – 1) messages in the worst case if at least one node in Candidate is live. In worst case that is the process with lowest process number detects coordinator as failed our new developed approach requires only 1+2+n-1 messages passing[5,6,7]. The table 1.1, 1.2 and 1.3 shown the comparative NC performance analysis of our approach with bully and different modifies bully algorithms.

Table 1.1

NC Performance Analysis in Worst Case: Number of Node / Process = 5

Case	Algorithms	Number of Node / Process = 5		
		Node Failed	Detector Node	No. of Messages
Worst	Bully	P5	P1	20
Worst	Mamun, Q. K.	P5	P1	11
Worst	Golam and Alastair	P5	P1	9
Worst	Election Administration	P5	P1	7

Table 1.2

NC Performance Analysis in Worst Case: Number of Node / Process = 10

Case	Algorithms	Number of Node / Process = 10		
		Node Failed	Detector Node	No. of Messages
Worst	Bully	P10	P1	20
Worst	Mamun, Q. K.	P10	P1	11
Worst	Golam and Alastair	P10	P1	9
Worst	Election Administration	P10	P1	7

Here in the example of tables 1.1, 1.2 and 1.3 node/process 1 detects that the current coordinator nodes node P5, P10 and P20 has crashed.

Table 1.3

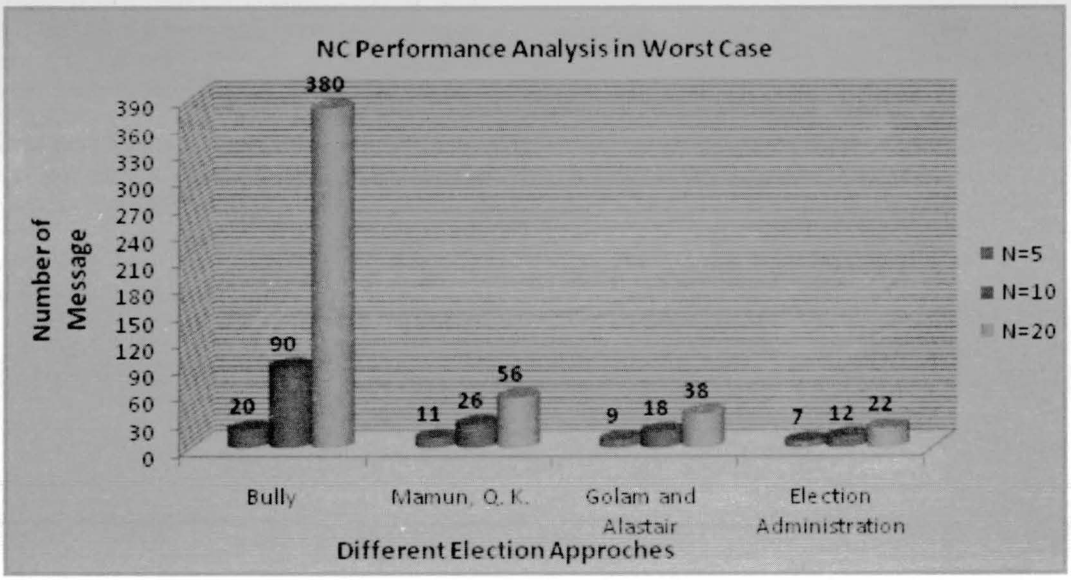
NC Performance analysis in Worst Case: Number of Node / Process = 20

Case	Algorithms	Number of Node / Process = 20		
		Node Failed	Detector Node	No. of Messages
Worst	Bully	P20	P1	380
Worst	Mamun, Q. K.	P20	P1	56
Worst	Golam and Alastair	P20	P1	38
Worst	Election Administratio n	P20	P1	22

The traditional Bully algorithm elects a new coordinator node in this case by performing a series of redundant elections and ends up when nodes are P5, P10 and P20. It producing 20, 90 and 380 messages in total. Mamun, Q. K. Algorithm needs 11, 26, 56 messages to elect a new coordinator. Md. Golam Murshed and Alastair R. Allen algorithm needs 9, 18, 38 messages to elect a new coordinator. Our proposed algorithm needs 7, 12 and 22 fewer messages. Our algorithm is fast and guarantees correctness and robustness, and the results show that it requires fewer messages to elect a new coordinator. The following graph 1.1 shown the comparative performance analysis of our approach with bully and different modifies bully algorithms in worst case.

In best case of election Garcia-Molina's bully algorithm requires N − 1 messages to elect a coordinator node in the best case, where N is the number of nodes. It will send election messages to N −1 nodes having higher id than itself. Each of the nodes eventually initiates a separate election one by one. Hence, it requires N − 1 messages in the best case. The modified Bully Algorithm proposed in Mamun, Q. K. also requires N − 1 messages in the best case. The algorithm proposed by Md. Golam Murshed and Alastair R. Allen also requires N − 1 messages in the best case. For the best case of our proposed algorithm there will be need of 1 election message to inform EA, 1 verify message to ensure the failure of coordinator, and n-1 messages to inform about new coordinator. In that case, our algorithm requires only 1+1+ n-1 messages.

In this case our new approach needs two extra messages to elect the coordinator: Election Message to inform the EA and verify message by EA to current coordinator but remove the problem of redundant election. In original bully algorithm and modified bully algorithm if coordinator is running unusually slowly say system is not working properly for some reasons or the link between a process and coordinator is broken for some reasons there will be redundant election, although current coordinator is up. But in our algorithm, as EA verifies either current coordinator is really up or down when EA receives any election message from any process, it ensures that there will be no redundant election in the system.



Graph 1.1 NC Performance analysis in Worst Case

Table 1.4

NC Performance Analysis in Best Case: Number of Node / Process = 5

Case	Algorithms	Number of Node / Process = 5		
		Failure Node	Detector Node	No. of Messages
Best	Bully	P5	P4	4
Best	Mamun, Q. K.	P5	P4	4
Best	Golam and Alastair	P5	P4	4
Best	Election Administration	P5	P4	6

Here in the example of Table 1.4 node/process P4 first detects that the current coordinator node/process P5 has crashed and declares itself as the new coordinator. The number of messages for this case is 4 the same for all three algorithms. In our approach it requires two extra messages one inform the *EA* and second is verified message by *EA*. It required 6 messages to elect new approach.

Table 1.5

NC Performance Analysis in Best Case: No. of Node / Process = 10

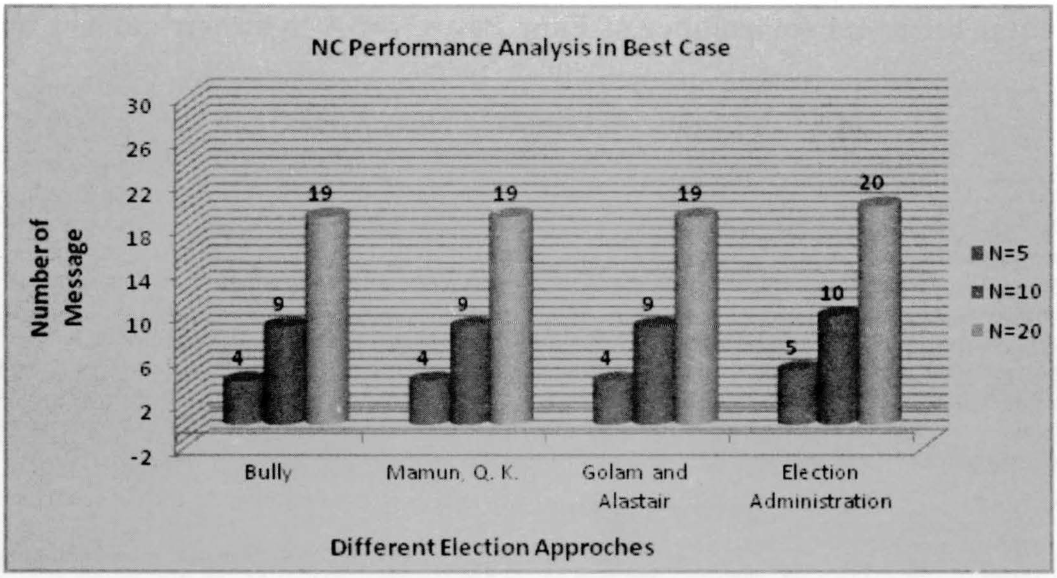
Case	Algorithms	Number of Node / Process = 10		
		Failure Node	Detecto r Node	No. of Messages
Best	Bully	P10	P9	9
Best	Mamun, Q. K.	P10	P9	9
Best	Golam and Alastair	P10	P9	9
Best	Election Administration	P10	P9	11

Table 1.6

NC Performance Analysis in Best Case: No. of Node / Process = 20

Case	Algorithms	Number of Node / Process = 20		
		Failure Node	Detecto r Node	No. of Messages
Best	Bully	P20	P19	19
Best	Mamun, Q. K.	P20	P19	19
Best	Golam and Alastair	P20	P19	19
Best	Election Administration	P20	P19	21

Here in the example of Table 1.5and Table 1.6, node P9 and P19 first detects that the current coordinator node, node P10 and P20 has crashed and declares itself as the new coordinator. The number of messages for this case is 9 and 19 the same for all three algorithms. In our approach it require on extra message that inform the election administration. It sends 11 and 21 coordinator messages. The following graph 1.2 shown the comparative performance analysis of our approach with bully and different modifies bully algorithms in best case..



Graph 1.2 NC Performance Analyses in Best Case

VIII CONCLUSION

This work presents some modifications to the classical bully algorithm which overcome the limitations of this algorithm and make it efficient and fast to elect a leader in synchronous distributed systems. The performance of the proposed algorithm has been compared with the original bully algorithm, Mamun, Q. K. approach Md. Golam Murshed and Alastair R. Allen algorithm and our proposal produces a better outcome. The algorithm is fast and guarantees correctness and robustness, and the results show that it requires fewer messages to elect a new leader.

REFERENCES

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

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

[3] Thakur P. Kumar , Kumar Ram, Ali Ruhi and Malviya Rajendra (2011)*"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.

[4] Garcia-Molina, H, (1982). *"Elections in a Distributed Computing Systems"*, *IEEE Transactions on Computers,* Vol. C–13, No. 1, pp. 48 – 59

[5] Mamun,Q.K., (2004) *"Modified Bully Algorithm for Electing Coordinator in Distributed System", 3rd WSEAS international conference on Software Engineering, Parallel Distributed Systems (SEPADS)"*, Salzburg, Austria,.

[6] Md. Golam Murshed and Alastair R. Allen, *"Enhanced Bully Algorithm for Leader Node* Election in Synchronous Distributed Systems", computers, www.mdpi.com/journal/computers, ISSN 2073-431X , June -2012.

[7] Dr. Gandhi S.K. and Thakur P. Kumar (2012) *"Election Administration Algorithm for Distributed Computing"* , IJE ECE, Vol 1(2): pp. 1-6.