# Optimized Bully Election Method for Selection of Coordinator Process and Recovery of Crashed Process

**Rachna Gajre[*], Dr. Leena Ragha[**]**

[*]Department Of Computer Engineering, RAIT
[**] Department Of Computer Engineering, RAIT

***Abstract***- In distributed systems one process or a node is required in such a way that it can act as leader node or a coordinator. Election algorithms are meant for electing process or node that acts as leader node also called as coordinator from among currently alive processes such that at any instance of time there will be single coordinator for all the processes in the system. So, election algorithms are momentous in any distributed system. Bully algorithm is one of the standard approaches for electing the coordinator in distributed systems. In this paper, we have presented a bully algorithm that minimizes the number of messages while electing the new coordinator and when a process recovers from a crashed state in distributed systems and thus reduces the network traffic caused.

***Index Terms***- Bully Election algorithm, Coordinator, Election message, OK message, and Process Status table

## I. Introduction

A distributed system is a collection of processors interconnected by a communication network in which each processor has its own local memory and other peripherals and the communication between them is done by message passing over the communication network [1]. Distributed systems require that there should be a coordinator node in the entire system that will perform coordination activity which is needed for the smooth running of other nodes in the system. Many leader election algorithms such as the Bully algorithm, Ring algorithm, Chang and Robert's algorithm, Peterson's algorithm, etc. have been proposed over the years. We will be discussing bully leader election algorithm in detail. In this paper initially we have described concept of bully election algorithm in section I and then its related work done by various authors to reduce number of messages in section II. In section III we will discuss our proposed modification and compare the modified recovery mechanism with one of initially modified recovery method. In section IV we will discuss number of messages required while electing coordinator and when a process recovers from failure. An algorithm for choosing a coordinator to play a distinct role in the system is called as election algorithm. The assumptions on which bully election algorithm is based are as follows [1, 2]:

1. It is a synchronous system and it uses timeout mechanism to keep track of coordinator failure detection.
2. Each process has unique number to distinguish them.
3. Every process knows the process number of all other processes.
4. Processes do not know which processes are currently up and which processes are currently down.
5. In election a process with highest process number is elected as coordinator which is agreed by all other live processes.
6. A failed process can rejoin in the system after recovery.
7. The communication subsystem does not fail.

The algorithm works as follows, when process P notices that the coordinator is crashed, it initiates an election algorithm and sends election message to all the processes having higher priority number than itself. If process P doesnot receive any response from processes having higher priority number than it then process P declares himself as coordinator by sending coordinator message to all the processes having priority number lower than it. If process P receives response from any process having higher priority number than it then process P now knows that higher process is alive and waits for final result. The receiver of election message gives response to sender by sending OK message to it to indicate that it is alive. Now it will initiate election unless it is already holding one. Finally all give up except one which is the new coordinator. The new coordinator announces it victory by sending coordinator messages to all process having process number lower than it. The working is as shown in figure 1.

As a part of recovery action as shown in figure 2, this method requires that a failed process (say P*k*) must initiate an election on recovery. If the current coordinator's priority number is higher than that of P*k*, *then* current coordinator will win the election initiated by P*k* and will continue to be the coordinator. On the other hand, if of P*k*'s priority number is higher than that of current coordinator, it will not receive any response for its election message. So it wins the election and takes over coordinator's job from current coordinator. Therefore, the active process having the highest priority number always wins the election. Hence the algorithm is called **bully algorithm** [1].

## II. RELATED WORK

As the basic well known bully election algorithm proposed by Garcia Molina large numbers of messages are exchanged due to which traffic in network is increased. So to decrease this number of messages various authors have suggested modifications in this bully algorithm to reduce number of messages.

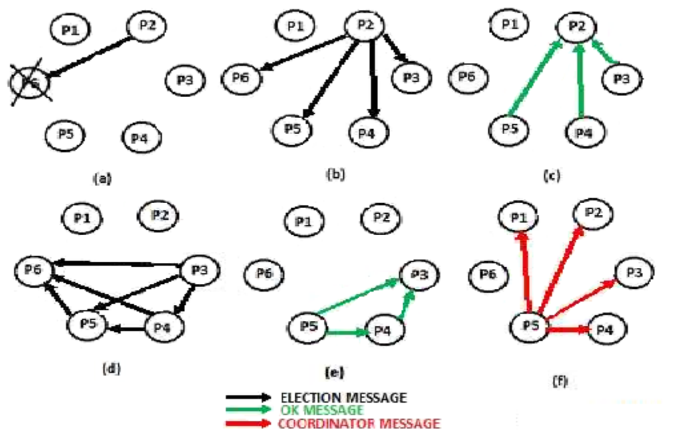

**Figure 1: Election of Coordinator by Garcia**
**(a)P2 request service from P6 (b)P2 sends election message to P3,P4,P5 and P6 (c)P3,P4 and P5 send OK message to P2 (d)P3,P4 and P5 initiate election (e)P4 sends OK message to P3, P5 sends OK message to P3 and P4 (f)P5 sends coordinator messages to P1,P2,P3 and P4.**

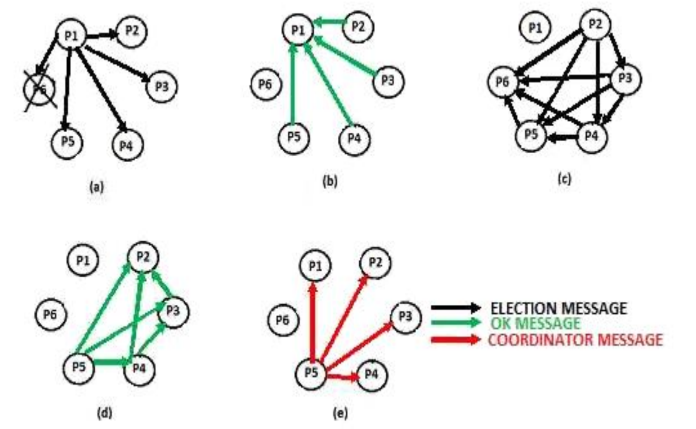

**Figure 2: Recovery Process by Garcia**
**(a) P1 sends election message to P2,P3,P4,P5 and P6 (b) P2,P3,P4 and P5 send OK message to P1 (c) P2,P3,P4 and P5 initiate election (d)P3 sends OK message to P2 ,P4 sends OK message to P2 and P3, and P5 sends OK message to P2,P3 and P5 (e)P4 sends OK message to P3, P5 sends OK message to P3 and P4.**

In [3] the number of messages that should be exchanged between the processes is reduced and furthermore the number of stages required to elect the coordinator is decreased from at most five stages to four stages. According to this paper, when the process P finds that coordinator has crashed, sends election message to all other processes with higher priority number for which it will receive response as OK message with its unique priority number to process P. If no process responses to process P, it will broadcast one coordinator message to all processes, declaring itself as a coordinator. If some process response to process P by comparing the priority numbers, the process P will select the process with the highest priority number as coordinator and then sends to it the grant message. At this stage the coordinator process will broadcast a message to all other processes and informs itself as a coordinator. Now suppose process P1 recovers from its failed state and is now unaware about who is the coordinator. So P1 holds the election by same procedure mentioned above. So if we look at traditional bully algorithm then number of stages and number of messages being passed is reduced to some extent. In [4], when a process (say) Pi finds that coordinator has somehow failed it refers to its process status table, to see who is process having the second highest priority number. It then initiates an election, by sending an election message to the process (say) Pj, having priority just below the failed coordinator; i.e. process with the second highest priority number. When Pj receives an election message (from Pi), in reply, Pj sends a response message OK to the sender, informing that it is alive and ready to be the new coordinator. Therefore, Pj will send a message coordinator to all other live processes (having priority less than Pj) in the system. Hence, Pi starts its execution from the point where it was stopped. If Pi does not receive any response to its election message, within a fixed timeout period; it assumes that process Pj also has somehow failed. Therefore, process Pi sends the election message to the process (say, Pk) having the priority just below the process Pj. This process continues, until Pi receives any confirmation message OK from any of the process having higher priority than Pi. It may be the case that, eventually Pi has to take the charge of the coordinator. In that case, Pi will send the coordinator message to all other processes having lower priority than Pi. According to [5], when a process P comes to know that coordinator has crashed it sends election messages to all the processes with higher process numbers. If process P doesnot receive any response then P wins the election and if process P receives the response (i.e. ok message along with process number of the responder) then process P will compare process numbers of all message and select highest process number as coordinator and process P will send coordinator messages to all the processes informing who is the new coordinator .Now suppose process P1 recovers from its failed state and is now unaware about who is the coordinator. So instead of holding an election process P1 will send query message to process P2 and process P3. Now P2 and P3 will give response by sending answer message to process P1.Now P1 will come to know that current coordinator is process P5. Now suppose process P6 recovers from its crashed state and process P6 knows that it is the process with highest process number so it will directly send coordinator messages to all the processes in the system. In [6] the process of coordinator election is proposed but on recovery how the process can rejoin itself is not specified. In this algorithm whenever a process finds the coordinator is dead, it sends an election message to a process which has the biggest number. With considering that the biggest process will be new coordinator, so it’s not necessary that other processes to be busy for this problem. Whenever a process receives the election message, it should introduce itself as a new coordinator. The receiver of message process may be dead such as the coordinator. So if the sender doesn’t receive the response, initiator process sends the election message to the next biggest process. This procedure maybe repeated for several times.

## III. PROPOSED METHOD

As we are considering distributed systems, hence, some assumptions also need to make about the communications network. This is very important because nodes communicate only by exchanging messages with each other. The following aspects about the reliability of the distributed communications network should be considered [4].

1. Messages are not lost or altered they are correctly delivered to their destination in a fixed amount of time; i.e., no communication failure occurs.
2. Messages reach their destination in a fixed amount of time, but the time of arrival is not fixed.
3. Nodes know the physical layout of all nodes in the system and know the path to reach each other.
4. A node never pauses and always responds to incoming messages with no delay.

This research tries to reduce network traffic present in distributed systems during leader election and process recovery. Suppose process Pi detects coordinator has failed so it checks the status table and sends election message to second highest priority message (say Pj).On receiving message from Pi, process Pj immediately sends coordinator messages to every live process. After receiving coordinator message from Pj each live process would update its process status table.

Consider the example in figure 3, suppose there are six processes P1, P2, P3, P4, P5 and P6 respectively in the system. Among these six processes P6 is considered as highest priority and P1 is with lowest priority. So P6 is the coordinator as it has highest process number and let process P1 is down. Suppose P2 wants some service from coordinator. So P2 sends a request to the coordinator P6.Now if process P2 does not receive a response within a fixed period of time, then process P2 assumes that the coordinator has somehow crashed. Having a look at the current process table, process P2 will send an ELECTION message to the process having priority just below the failed coordinator's priority(P5 in this case). On receiving election message from P2 process P5 sends coordinator messages to all live processes. The process status table when new coordinator P5 is elected is shown in table I.

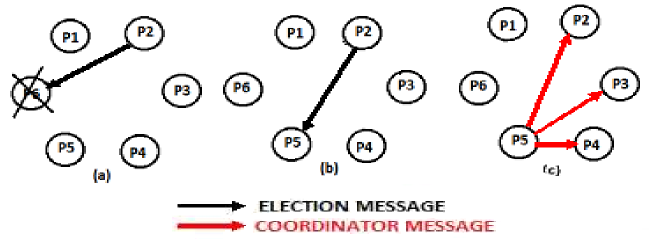

Figure 3: **Election of Coordinator in Proposed Method**
**(a)P2 request service from P6 (b)P2 sends election message to P5 (c)P5 sends coordinator message to P2,P3 and P5.**

Now suppose process Pm recovers from failure so there can be two cases:

**Case1:**
If the current coordinator's priority is higher than Pm's priority, in that case, Pm will send its priority number and an UPDATE messages to all other live processes in the system, to tell them to update Pm's status (from CRASHED to NORMAL) in their own process status table.

**Case 2:**
If Pm's priority is higher than the current coordinator's priority; then Pm will be the new coordinator and update the process status table and sends the COORDIANTOR message to all other live processes in the system, and takes over the coordinator's job form the currently active coordinator.

Table I: Process Status Table When P5 Is Elected As Coordinator

| **Process Priority** | **Status** |
|---|---|
| P1 | CRASHED |
| P2 | NORMAL |
| P3 | NORMAL |
| P4 | NORMAL |
| P5 | COORDINATOR |
| P6 | CRASHED |

Now suppose in example above if process P1 recovers from its failed state and is now unaware about who is the coordinator and status of processes. So it immediately, sends a REQUEST message to any of its live neighbors (in this case Process P2). So, as soon as any of P1's live neighbors receives a REQUEST message, it sends a copy of the current process status table to P1. After receiving the process status table, P1 checks whether its own priority number is less than the process having the highest priority (i.e. current coordinator's priority) or not. Since P1 is smaller than current coordinator so it will send its priority number and an UPDATE messages to all other live processes in the system, to tell them to update P1's status (from CRASHED to NORMAL) in their own process status table as shown in figure 4. The process status table when P1 recovers from failure is shown in table II.

Table II: Process Status Table When P1 Is Recovers From Failure

| **Process Priority** | **Status** |
|---|---|
| P1 | NORMAL |
| P2 | NORMAL |
| P3 | NORMAL |
| P4 | NORMAL |
| P5 | COORDINATOR |
| P6 | CRASHED |

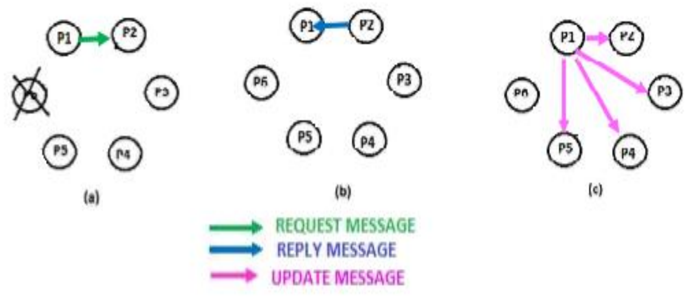

**Figure 4: Proposed Recovery Process**
**(a)P1 sends Request message to P2 (b)P2 sends Reply message to P1 (c)P1 sends update message with its process number to P2,P3,P4 and P5**

Suppose Pm recovers and we send Query messages to process having process number higher than itself then in response it will get answer message in which process Pm will understand who the current coordinator according to algorithm is in [5].

In our example now process P1 recovers from failure so it will send query messages to process P2, P3, P4, P5 and P6. As response P1 will receive answers from the live processes and will understand that the current coordinator is P5 as shown in figure 5.

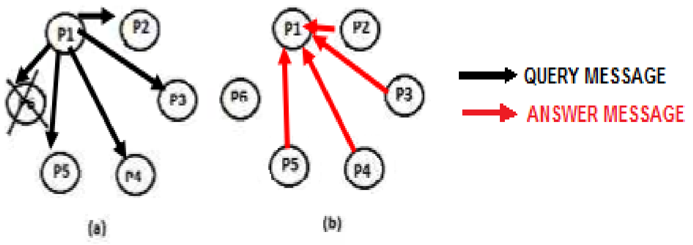

**Figure 5: Recovery Method of Pawan Kumar Thakur**
**(a)P1 sends query messages to P2, P3, P4, P5 and P6 (b) Process P2, P3, P4 and P5 send answer message to process P2.**

If we compare both methods of recovery we can see that in first method we maintain a status table which is requested by process when it recovers by neighbor through which it also comes to know about which processes are alive and the current coordinator by sending just one request message whereas in second method we send query message to all the processes having process number higher than itself and then everyone replies with answer message through which it understands who is the current coordinator. From both methods less number of messages is sent in first method. So first method i.e. our proposed (first) method is better than this second method.

## IV. RESULTS

The analysis of bully algorithm's proposed by various authors with best case and worst case and also with respect to different network size is analyzed in this chapter.

1. According to algorithm in [1] the number of messages required for various numbers of nodes is as shown in table III.

Table III: No. of messages required for various numbers of nodes according to algorithm in [1]

| No. Of Nodes | No. Of messages in electing a coordinator | No. Of messages when process recovers from failure |
|---|---|---|
| 6 | 20 | 29 |
| 10 | 72 | 89 |
| 15 | 178 | 205 |

2. According to algorithm in [3] the number of messages required for various numbers of nodes is as shown in table IV.

Table IV: No. of messages required for various numbers of nodes according to algorithm in [3]

| No. Of Nodes | No. Of messages in electing a coordinator | No. Of messages when process recovers from failure |
|---|---|---|
| 6 | 13 | 15 |
| 10 | 25 | 27 |
| 15 | 40 | 42 |

3. According to algorithm in [4] the number of messages required for various numbers of nodes is as shown in table V.

Table V: No. of messages required for various numbers of nodes according to algorithm in [4]

| No. Of Nodes | No. Of messages in electing a coordinator | No. Of messages when process recovers from failure |
|---|---|---|
| 6 | 5 | 7 |
| 10 | 9 | 11 |
| 15 | 14 | 16 |

4. According to algorithm in [5] the number of messages required for various numbers of nodes is as shown in table VI.

Table VI: No. of messages required for various numbers of nodes according to algorithm in [5]

| No. Of Nodes | No. Of messages in electing a coordinator | No. Of messages when process recovers from failure |
|---|---|---|
| 6 | 12 | 9 |
| 10 | 24 | 17 |
| 15 | 39 | 27 |

5. According to algorithm in [6] the number of messages required for various numbers of nodes is as shown in table VII.

Table VII: No. of messages required for various numbers of nodes according to algorithm in [6]

| No. Of Nodes | No. Of messages in electing a coordinator | No. Of messages when process recovers from failure |
|---|---|---|
| 6 | 5 | - |
| 10 | 9 | - |
| 15 | 14 | - |

6. In our proposed method the number of messages required for various numbers of nodes is as shown in table VIII.

Table VIII: No. of messages required for various numbers of nodes according to Proposed Modification

| No. Of Nodes | No. Of messages in electing a coordinator | No. Of messages when process recovers from failure |
|---|---|---|
| 6 | 4 | 6 |
| 10 | 8 | 10 |
| 15 | 13 | 15 |

From above results we have analyzed number of messages required to be exchanged for various numbers of nodes and can say that in each paper number of message is reduced.

## V. CONCLUSION

In original bully algorithm the number of messages to be exchanged is very large. To overcome this drawback we have proposed an optimized method by combining ideas from initially modified algorithms. From analysis we can say that our proposed method requires less number of messages than from all other algorithms and also we compared our recovery method with initially modified recovery method.

We measure performance of our algorithm by number of messages passed in system. From the above discussions about original bully algorithm and modified bully algorithm we can say that our proposed method is better since it requires less number of messages to be sent in system in both cases when electing coordinator and on recovery of any process.

## REFERENCES

[1] Sinha P.K, "Distributed Operating Systems Concepts and Design", Prentice-Hall of India private Limited, 2008.
[2] H.Gracia-Molina, "Elections in a distributed computing system", IEEE Trans. on Computers, vol. C-31, no. 1, Jan.1982.
[3] M.S.Kordafshari, M.Gholipour, M.Jahanshahi, A. T. Haghighat, "Modified bully election algorithm in distributed system", SEAS Conferences, Cancun, Mexico, 2005.
[4] Sandipan Basu, "An Efficient Approach of Election Algorithm in Distributed Systems", Indian Journal of Computer Science and Engineering (IJCSE), Vol 20, No 1, 2010.
[5] Pawan Kumar Thakur, Ram Kumar, Ruhi Ali and Rajendra kumar Malviya ,"A New Approach of Bully Election Algorithm for Distributed Computing" Int. J. of Electrical, Electronics and Computer Engineering (IJEECE) Vol 1(1): 72-79,2011.
[6] S. Mahdi Jameii "A Novel Coordinator Selection Algorithm in Distributed Systems",(IJAEST) International Journal Of Advanced Engineering Sciences And Technologies Vol No. 9, Issue No. 2, 2011.s

## AUTHORS

**First Author** – Rachna Gajre, M.E.Computer (pursuing), RAIT, e-mail: rachi.g12@gmail.com.
**Second Author** – Dr. Leena Ragha, Ph.D (Computer Science& Engineering), RAIT and email: leena.ragha@gmail.com.