Comparison of Bully Election Algorithms in Distributed System

Vaibhav P. Gajre[*]

[*]Department Of Computer Engineering, RAIT

Abstract- Many distributed algorithms require one process or node to act as leader node in distributed systems. Election algorithms are meant for electing such process or node that acts as leader node also called as coordinator from among currently alive processes in such a way that at any instance of time there will be single coordinator for all the processes in the system. Therefore, election algorithms are very important in any distributed system. Bully algorithm is one of the standard approaches for electing the coordinator in distributed systems. In this paper, we have compared base and efficient version of bully algorithm to minimize the number of messages when electing the coordinator and when a process recovers from a crashed state in distributed systems.

Index Terms- Bully Election algorithm, Coordinator message, 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 held by message passing over the communication network [1]. Several distributed algorithms require that there be a coordinator node in the entire system that performs some type of coordination activity needed for the smooth functioning of other nodes in the system. As the nodes in the system need to interact with the coordinator node, they all must unanimously agree on who the coordinator is. In addition, if the coordinator node fails due to some reason (e.g. link failure) then there is a need for electing a new coordinator.

1.1 Election Algorithm:

An algorithm for choosing a coordinator to play a distinct role is called an election algorithm. Leader election is a fundamental problem in the distributed systems, [8]. Election algorithms are meant for electing a coordinator process from among the currently running processes in such a manner that at any instance of time there is single coordinator for all processes in the system. Election algorithms are based on the following assumptions:

1. Each process in the system has a unique priority number.
2. Whenever an election is held, the process having the highest priority number among the currently active processes is elected as the coordinator.
3. On recovery, a failed process can take appropriate actions to rejoin the set of active processes.

Therefore, whenever initiated, an election algorithm finds out which of the currently active processes has highest priority number and then informs this to all other active processes. Leader election is the process of determining a process as the manager of some task distributed among several processes [7]. Every leader election algorithm must be satisfied by the safety and liveness condition for an execution to be admissible. The liveness condition states that every node will eventually enter an elected state or a non-elected state. The safety condition for leader election requires that only a single node can enter the elected state and eventually, become the leader of the distributed system. Information is exchanged between nodes by transmitting messages to one another until an agreement is reached. Once a decision is made, a node is elected as the leader and all the other nodes will acknowledge the role of that node as the leader.

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, [3].

1. Messages are not lost or altered and are correctly

delivered to their destination in a finite amount of time; i.e., no communication failure occurs.

2. Messages reach their destination in a finite amount of time, but the time of arrival is variable.
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.

1.2 Bully Election Algorithm:

This algorithm was proposed by Garcia-Molina. In this algorithm, it is assumed that every process knows the priority number of every other process in the system. The algorithm works as follows:

- When a process (say P*i*) sends a request message to the coordinator and does not receive a reply within a fixed timeout period; it assumes that the coordinator has failed.
- It then initiates an election by sending an election message to every process with a higher priority number than itself. If P*i* does not receive any response to its election message within a fixed timeout period, it assumes that among the currently active processes it has the highest priority number. Therefore it takes up the job of the coordinator and sends a coordinator message to all the processes having lower priority numbers than itself, informing that from now, it is the new coordinator.
- On the other hand, if P*i* receives a response for its election message, this means that some other process having higher priority number is alive. Therefore, P*i* does not take any further action and just wait to receive the final result of the election it initiated.
- When a process (say P*j*) receives an election message, it sends response message to sender informing that it is alive and will take over the election activity. Now P*j* initiates an election if it is not already holding one. In this way, election activity gradually moves on to the process that has the highest priority number among the currently active processes, eventually wins the election, and becomes the coordinator.
- As a part of recovery action, 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 priority number of P*k* is higher than that of current coordinator, it will not receive any response for its election message. Therefore, it wins the election and takes over coordinator's job from currently active coordinator. Therefore, the active process having the highest priority number always wins the election. Hence, the algorithm is termed as **bully algorithm** [1].

The Bully Algorithm proposed by Garcia Molina is based on assumptions that are as follows [1, 7]:

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.

Consider the example in figure 1.1, suppose there are six processes P1, P2, P3, P4, P5 and P6 respectively. Among these six processes let P1 is down and P6 is the coordinator as it has highest process number. Suppose P2 wants some service from coordinator and P6 is crashed. So P2 comes to know that coordinator is failed due to some reason so it initiates an election. Process P2 sends election messages to all the processes with higher process number than itself. The live processes with high process number reply with OK message to process P2. Now P2 stops and waits to receive coordinator message. Now processes P3, P4 and P5 make elections and among them P5 wins the election. Now P5 is the new coordinator so P5 sends coordinator message to all processes having lower priority.

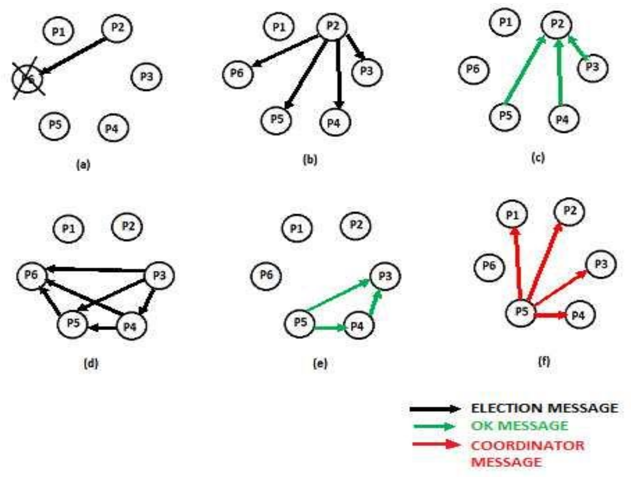

Figure 1.1 Election of Coordinator by Garcia (a)P2 request service from P6 (b)P2 sends election message to P3,P4,P5 and P6 (c)P3,P4 and P5 send OK message to P2 (d)P3,P4 and P5 initiate election (e)P4 sends OK message to P3, P5 sends OK message to P3 and P4 (f)P5 sends coordinator messages to P1,P2,P3 and P4.

Now suppose process P1 recovers from its failed state and is now unaware about who is the coordinator. As shown in figure 1.2, P1 holds the election by same procedure of algorithm above and P5 wins the election again as shown in figure below. Now if process P6 recovers then P6 knows that it is the process with highest process number so it will simply bully every one and send coordinator messages to all the processes in the system.

The algorithm we discussed above is the base algorithm in which new modifications are made by some authors. The various modifications made to these algorithms are described in section II and in section III, we will be comparing all the methods.

II. LITERATURE REVIEW

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.

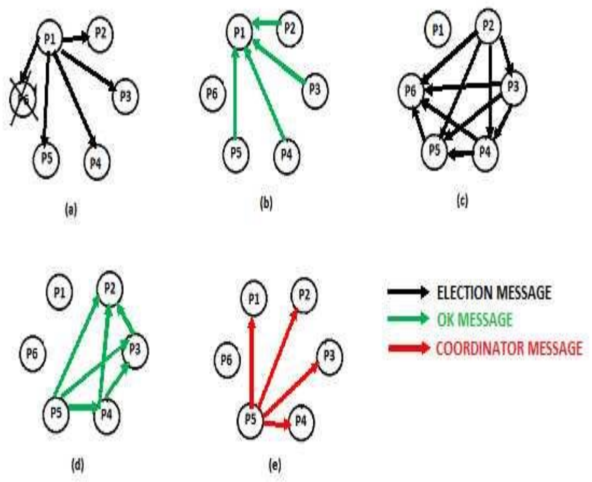

Figure 1.2 Recovery Process by Garcia (a)P1 sends election message to P2,P3,P4,P5 and P6 (b)P2,P3,P4 and P5 send OK message to P1 (c)P2,P3,P4 and P5 initiate election (d)P3 sends OK message to P2, P4 sends OK message to P2 and P3, and P5 sends OK message to P2, P3 and P5 (e)P4 sends OK message to P3, P5 sends OK message to P3 and P4.

In paper [2], M. S. Kordafshari, M. Gholipour, M.Jahanshahi, A.T. Haghighat, have proposed optimized method by modifying the Bully algorithm that intensively decreases the number of messages that must be exchanged between processes. Furthermore, the number of stages is decreased from five stages to four stages.

The algorithm works as follows:

- When the process P finds out that coordinator is crashed, sends ELECTION message to all other processes with higher priority number.
- Each process that receives ELECTION messages (with higher process than P) sends 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. it will send election message to all the processes and by comparing their priority numbers it will send GRANT message to higher process number i.e.to process P5 in this case.
- Now if process P6 recovers from it crashed state it will also follow the same procedure. So if we look at traditional bully algorithm then number of stages and number messages being passed is reduced to some extent.

In paper [3], Sandipan Basu has proposed an election algorithm that can be used in distributed systems to elect a coordinator. This is a betterment of an already existing Election algorithm (also known as **Bully Algorithm**) proposed by Hector Garcia-Monila.

The algorithm works as follows:

- When a process (say) Pi sends a message (any request) to the coordinator and does not receive a response within a fixed timeout period; it assumes that the coordinator has somehow failed. Process Pi 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.

Here two cases may appear:

Case 1:

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.

Case 2:

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.

In paper [4], Pawan Kumar Thakur, Ram Kumar, Ruhi Ali and Rajendra kumar Malviya have presented a modified version of existing bully algorithm. It is based on assumptions made bully algorithm.

The algorithm works as follows:

- 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 but 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. Then 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 paper [5], S. Mahdi Jameii, Loghman Kiani Shahvandi, Fatemeh Najafi and Payam Kiani Shahvandi have presented a method, which is an enhanced version of bully algorithm.

In this, how coordinator can be elected is proposed but on recovery, how the process would rejoin is not specified.

This algorithm works as follows:

Whenever a process finds the coordinator is dead, it sends an election message to a process, which has the biggest number. 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. If Pi finds the coordinator is dead, it begins the election algorithm by sending an election message to Pj that Pj has biggest number. (If Pi=Pj: Pi becomes coordinator). If Pi receives the response message from Pj, then Pj becomes new coordinator and if Pj doesn't reply to the message (Pj is dead) then election message is send to process having highest number below Pj and so on.

In paper [6], Rachna Gajre and Dr. Leena Ragha have presented a method, which is an enhanced version of bully algorithm. This paper tries to reduce network traffic present in distributed systems during leader election and process recovery.

The algorithm works as follows:

- ➢ uppose Process Pi wants some service from coordinator, therefore it sends a request to the coordinator Pk.
- ➢ ow if process Pi does not receive a response within a fixed period of time, then process Pi assumes that the coordinator has somehow crashed. Having a look at the current process table, process Pi will send an ELECTION message to the process having priority just below the failed coordinator's priority(i.e below Pk).
- ➢ n receiving election message from P_i process Pj (let Pj be the process having second highest priority i.e. below Pk) sends coordinator messages to all live processes. After receiving coordinator message from Pj, each live process would update its process status table.

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.

III. COMPARISON

The comparison of workdone by various authors is as shown in table I.

Table I: Comparison of workdone by various authors

Authors	**Complexity**		**No. stages required**	**Maintains status table**	**Types of messages**
	Best Case	**Worst Case**			
Garcia Molina	O (n)	O (n^2)	5	No	Election, Ok and Coordinator S
M. S. Kordafshari	O (n)	O (n)	4	No	Election, (Ok+ Priority), Grant and Coordinator N
Sandipan Basu	O (n)	2(constant)	3	Yes	Election, Ok, Coordinator, Request and (Priority + Update) O
Pawan	O(n)	O(n)	3	No	Election,

Kumar Thakur					Ok, Coordinator, Query and Answer
S. Mahdi Jameii	O(constant)	O(constant)	2	No	Election and Coordinator
Rachna G.	O(constant)	O(constant)	2	Yes	Election, Coordinator, Request, Reply and Update

From above comparison, as we can see the traditional bully algorithm has been improved to greater extent. We can say that a method proposed by S. Mahdi Jameii and Rachna G. has less complexity and require less number of stages. The method proposed by Sandipan Basu and Rachna G. both maintain status table due to which we are able to keep track of live and dead(failed) processes.

IV. CONCLUSION

Among several election algorithms, we have studied bully election algorithm in detail. In original bully algorithm, the numbers of messages exchanged are very high. To overcome this drawback, various authors have proposed modified algorithms. All methods measure performance of election algorithm by number of messages sent in system.

REFERENCES

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

[2] M.S.Kordafshari, Gholipour, M.Jahanshahi, A.T.Haghighat, “Modified bully election algorithm in distributed system”, SEAS Conferences, Cancun, Mexico, 2005.

[3] Sandipan Basu, “An Efficient Approach of Election Algorithm in Distributed Systems”, Indian Journal of Computer Science and Engineering (IJCSE), Vol 20, No 1, 2010.

[4] 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.

[5] 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.

[6] Rachna Gajre and Dr. Leena Ragha, “Optimized Bully Election Method for Selection of Coordinator Process and Recovery of Crashed Process”, International Journal of Scientific and Research Publications, Volume 3, Issue 5, May 2013.

[7] H. Gracia-Molina, “Elections in a distributed computing system”, IEEE Trans. on Computers, vol. C-31, no. 1, Jan.1982.

[8]http://lindajmiller1969.hubpages.com/hub/Distributed-Systems-Election-Algorithms.

AUTHORS

First Author – Vaibhav P. Gajre, B.E.Computer, RAIT, vaibhav.gajre@gmail.com.