International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2015): 6.391

Modified Bully Election Algorithm for Crash Recovery in Distributed Systems

Sindhu Daniel

Assistant Professor of Mount Zion College of Engineering

Abstract: ***Many distributed algorithms require one process to act as coordinator, initiator or otherwise perform some special role. The main role of an elected coordinator is to manage the use of shared resource in an optimal manner. An election algorithm is an algorithm for solving the coordinator election problem. The coordinator election problem is to choose a process from among a group of processes on different processors in a distributed system to act as the centre coordinator. Therefore, election algorithms are very important in any distributed systems. Bully election algorithm is one of the classical approaches in distributed computing for dynamically electing a coordinator with highest priority number or highest process ID number. In this paper, we are compared base and efficient version of bully algorithm to minimize the number of messages during the election and when a process recovers from a crashed state in distributed systems.***

Keywords: Bully Election algorithm, Coordinator, Election message, OK message, and Process Status table

1. Introduction

Distributed system is a collection of independent computers that appears to its user as a single coherent system. A distributed system is a collection of processors that do not share memory or a clock. Each processor has its own memory, and the processors communicate via communication networks. These computers communicate and cooperate with each other only by passing the messages over a communication network. To the users, this collection of computers appears to be a single coherent system. Users can communicate easily with this system without knowing the physical location of the system. In distributed computing, an election algorithm is used for choosing a single process to perform a particular task which will play the role of the server. But it is important that all the processes have the same opinion about the choice. When the process completes and the server does not want to continue any more, then some other process will perform the role of the server or leader and the old server is replaced by a new one to lead the collection of processors. 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 election algorithm is an algorithm for solving the coordinator election problem. Various algorithms require a set of processes to elect a leader or a coordinator. Election algorithms elect a coordinator process from among the currently running processes.

These algorithms have two major goals:
- They attempt to locate the process with the highest process number and designate it as the coordinator, and inform all the active process about this coordinator.
- The second goal of an election algorithm is to allow a recovered leader to reestablish control.

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.

Election algorithms are based on the following assumptions:
- Provide each process with a unique process ID/system number.
- Elect a process using a total ordering on the required set.
- All processes know the process number of members.
- All processes agree on the new coordinator.
- All processes hold an election to determine if the new coordinator is up or crashed

1.2 Bully Election Algorithm

The Bully Algorithm proposed by Garcia Molina is based on assumptions that are as follows:

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.

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].

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 new coordinator so P5 sends coordinator message to all processes having lower priority.

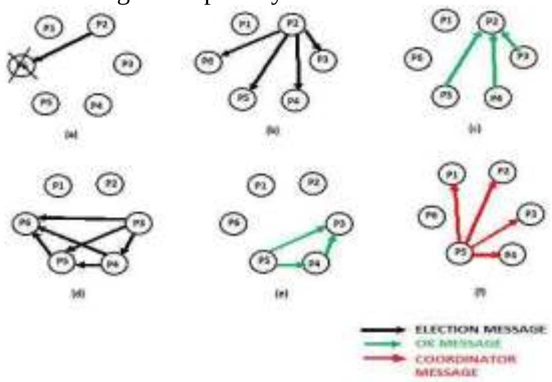

Figure 1.1: Election of Coordinator by Garcia

(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.

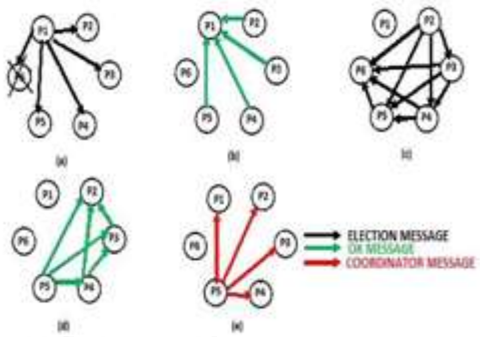

Figure 1.2: Recovery Process by Garcia

(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.

1.3Limitations

Bully algorithm has following limitations:

- The main limitation of bully algorithm is the highest number of message passing during the election and it has order O(n2) which increases the network traffic.
- When any process that notices coordinator is down then holds a new election. As a result, there may number of elections can be occurred in the system at a same time which imposes heavy network traffic.
- As there is no guarantee on message delivery, two Processes may declare themselves as a coordinator at the same time. Say, p initiates an election and didn't get any reply message from Q, where Q has a higher process number than p. At that case, p will announce itself as a coordinator and as well as Q will also initiate new election and declare itself as a coordinator if there is no process having higher process number than Q.
- Again, 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 redundant election.
- Again, if a process p with lower process number than the current coordinator, crashes and recovers again, it will initiate an election where the current coordinator will win again. This is also a redundant election.

2. Modified Bully Algorithms

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].

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 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.

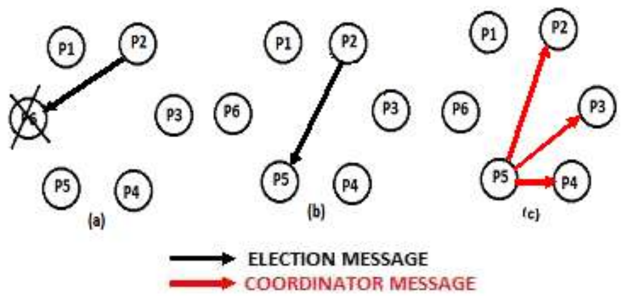

Figure **3:** Election of Coordinator in Proposed Method (a)P2 request service from P6 (b)P2 sends election message to P5 (c)P5 sends coordinator message to P2,P3 and P5.

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

- 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.
- If Pm's priority is higher than the current 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 1: 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 2: Process Status Table When P1 Is Recovers From Failure

Process priority	status
P1	Normal
P2	Normal
P3	Normal
P4	Normal
P5	Coordinator
P6	Crashed

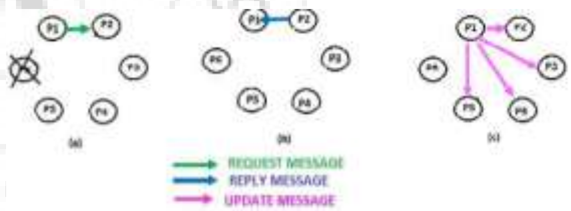

Figure 4: Proposed Recovery Process
a) P1 sends Request message to P2 (b)P2 sends Reply message to P1 (c)P1 sends update message with its process number to P2,P3,P4 and P5

We have analyzed number of messages required to be exchanged for various numbers of nodes and can say that in our paper number of message is reduced.

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

Table 1: 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) In our proposed system the number of messages required for various numbers of nodes is as shown in table below

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

3. Conclusion

In 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. 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.

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] 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.
[4] " Comparisons of bully election algorithms in distributed systems". Vaibhav P. Gajre*International Journal of Scientific and Research Publications, Volume 3, Issue 9, September 2013 ISSN 2250-3153
[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 Science and Technologies Vol No. 9, Issue No. 2, 2011.s