A new proposed Bully Algorithm For Leader Election Process In Distributed Systems

Mohammad Mosleh
Department of Computer Engineering
Islamic Azad University
Dezful.Khouzestan. Iran
Mohammad.mosleh@gmail.com

Eman Zamane
Department of Computer Engineering
Islamic Azad University
Dezful,Khouzestan, Iran
Eman_zamane@yahoo.com

Abstract **Coordinator selection is a vital issue not only in distributed computing but also in communication network, centralized mutual exclusion algorithm, centralized control IPC, etc. A leader is required to make synchronization between different processes. Different election algorithms are used to elect a coordinator among the available processes in the system such a way that there will be only one coordinator at any time. Bully election algorithm is one of the classical and well-known approaches in coordinator election process. This paper will present a modified version of bully election algorithm using a new concept called election commission. This approach not only reduces redundant elections but also minimizes total number of elections and hence it will minimize message passing, network traffic, and complexity of the existing system.**

Keywords- Bully election algorithm; Coordinator; Distributed Systems; Message passing.

I. INTRODUCTION

Election of a leader is an essential problem in distributed computing. It has been the subject of intensive research since its importance was first articulated by Gerard Le Lann[1] in 197. In a distributed computing system, a process 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 and there are many algorithms that are used in election process. Bully election algorithm is one of them. This paper represents a modified version of bully algorithm using a new concept Election Commission. It reduces redundant elections, minimizes message passing and network traffic. In section 2, it is given an introduction to election algorithms; section 3 represents methodology of our proposed algorithm. In Section 4, an overall Comparison of our algorithm with Bully and MBA algorithms is given.

II. Election Algorithms

An election algorithm is an algorithm for solving the coordinator election problem. Various algorithms require a set of peer processes to elect a leader or a coordinator. It can be necessary to determine a new leader if the current one fails to respond. Provided that all processes have a unique identification number, leader election can be reduced to finding the non-crashed process with the highest identifier. An election algorithm is an algorithm for solving the coordinator election problem. Various algorithms require a set of peer processes to elect a leader or a coordinator. It can be necessary to determine a new leader if the current one fails to respond.

A. Bully Algorithm

Bully algorithm is one of the most famous election algorithms which were proposed by Garcia-Molina [2] in 1982. This algorithm is established on some basic assumptions which are:

- It is a synchronous system and it uses timeout Mechanism to keep track of coordinator failure detection [3]
- Each process has a unique number to distinguish them [1-3].
- Every process knows the process number of all other processes [2-5].
- Processes do not know which processes are currently up and which processes are currently down [1,6,7].
- In the election, a process with the highest process number is elected as a coordinator which is agreed by other alive processes [3].

The procedure is shown in fig 1.

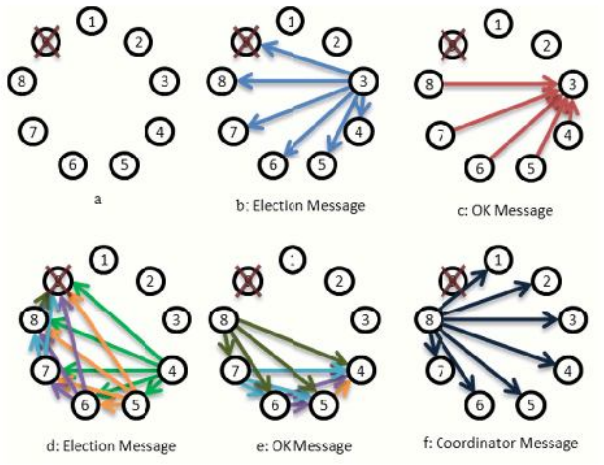

Figure 1. Bully Algorithm: (a) process 3 detects coordinator is failed, (b) process 3 sends election message to processes 4 9. (c) Processes 4 9 respond to 3 to stop election, (d) each of 4 9 send election message, (e) process 8 responds to 4 7 and process 7 responds to 4-6 and etc. to stop election, (f) Process 8 winds and announces to all.

- A failed process can rejoin in the system after recovery [

In this algorithm, there are three types of message and there is an election message (inquiry) which is sent to announce an election, an answer (ok) message is sent as response to an election message and a coordinator (victory) Message is sent to announce the new coordinator among all other alive processes [

When a process P determines that the current coordinators 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 (figure 1).

- P sends an election message (inquiry) to all other processes with higher process numbers respect to it. If P doesn't receive any message from processes with a higher process number than it, it wins the election and sends a coordinator Message to all alive processes.
- If P gets answer message from a process with a higher process number; P gives up and waits to get coordinator message from any of the process with higher process number. Then new process initiates an election and sends election message to processes with higher process number than that one. In this way, all processes will give up the election except one which has the highest process number among all alive processes and it will be elected as a new coordinator. New Coordinator broadcasts itself as a coordinator to all alive processes in the system.
- Immediately after the recovery of the crashed process is up, it runs bully algorithm.

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 $O(n^2)$ which increases the network traffic.
- When any process that notices coordinator is down then holds a new election. As a result, there May n 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 a redundant election.

Again, if process P with lower process number than the current coordinator, crashes and recovers again, it will initiate an election from current state.

B. Modified Bully algorithm by M.S. Kordafshari et al.

M. S. 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.

According to M. S. Kordafshari et al. [4], their proposed algorithm is briefly described below in fig 2.

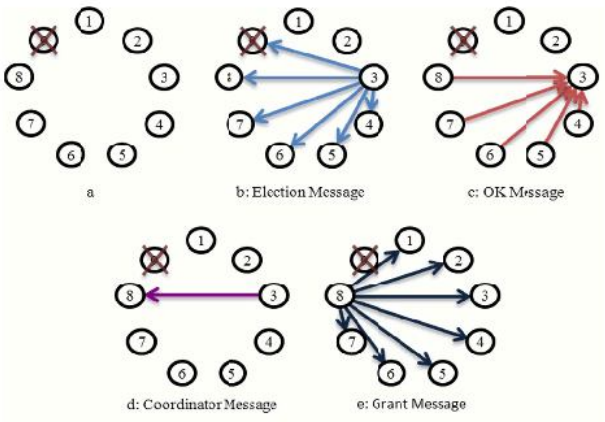

Figure 2. Modified Bully Algorithm (MBA) by M.S. Kordafshari et al: (a) process 3 detects coordinator is failed, (b) process 3 send election message to processes 4-9, (c) Processes 4 9 respond with their process number, (d) Process 3 selects highest process number and send a grant message to it, (d) Process 8 sends coordinator message to all processes. Coordinator will win again. This is also a redundant election.

- ✓ When process P notices that the coordinator is down, it initiates an election by sending Election message to all processes with higher priority number. If no process responses to P, it declares itself as a new coordinator. If some processes response to P, it will select the process with highest priority number and send back a GRANT message to that selected process. Finally, selected process broadcast a coordinator message to all others as a coordinator itself. If any process with the highest priority number is up, it will run the algorithm again.
- ✓ To reduce concurrence election, when process P notices that the coordinator is down, it initializes election. If process Q (Q may be P) receives an ELECTION message from any process with lower priority number, it waits for a short time and replies to the process with lowest priority number and stop its own algorithm. But if P neither receives any response nor any ELECTION message from other processes with lower priority number, it declares itself as a coordinator.

This algorithm has following drawbacks[4].

- ❖ 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 don t receive any coordinator message, they will initiate modified algorithm again. If there are Q different high processes, then there will be Q different individual instance of modified algorithm at that moment in the system. Those are redundant election.
- ❖ If process P sends GRANT message to the process with the highest priority number and P doesn t 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.
- ❖ Although Q sends stop message to P, if any other process R lower than Q sends ELECTION message to Q with this condition R<P<Q, it takes network resources to send stop messages and increases network traffic.

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

III. PROPOSED METHOD

We intend to present a method which is an enhanced version of bully algorithm. 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. In other way the receiver of message process maybe 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. (Pi means the process which its number is i). If Pi finds the coordinator is dead, it begins the following election algorithm:

I. Pi sends an election message to Pj that Pj has biggest number. (If Pi=Pj: Pi becomes coordinator)
II. If Pi receives the response message from Pj, then Pj becomes new coordinator.
III. If Pj doesn t reply to the message (Pj is dead) then two over steps is repeated.

If a process receives an election message from the other processes, it sends the coordinator message to all processes and introduces itself as new coordinator. Our proposed algorithm is shown in fig 3.

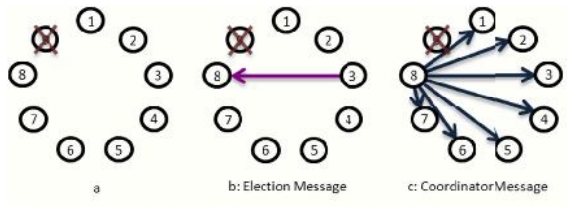

Figure 3. Proposed Algorithm

In this example, P3 finds the coordinator is dead. Because P8 has the most priority it sends only an election message for P8. Then P8 sends the coordinator message for all processes.

But if P8 was dead, it wouldn t response, Then P3 would send election message to P7, and according to P7 wouldn t response too, it would send the message to the next process until inform the process which has the biggest number among all alive processes.

When a process gains victory in election, it must inform others about it. So it must send message to other processes. We tried to decrease the number of messages during the election not the messages which must be sent after it.

Probably by considering 1 p new coordinator can be found only by one message and the probability that it may be found with two massages is p*(1 p) and the probability that it may be found with three massages is p2 *(1 p) and the probability that it may be found with i massages is p_{i-1}*(1-

We can show m (the average of necessary massage number for selecting new coordinator) as the following:

$$m=(1-p)+2p(1-p)+3p2(1-p)+\cdots+(n-1)pn-2(1-p)$$

$$= (1-p)\sum_{i=1}^{n-1} ip^{i-1} \qquad (1)$$

$$\lim_{n\to\infty}\sum_{i=1}^{n} ip^{i-1} = \frac{1}{(1-p)^2} \qquad (2)$$

According to the formula (1) & (2) and with the considering this fact that we have many processes:

$$m = (1-p)\sum_{i=1}^{n-1} ip^{i-1} = \frac{1-p}{(1-p)^2} = \frac{1}{1-p} \qquad (3)$$

So we concluded that the average number of necessary transmission messages for coordinator to be $\frac{1}{1-p}$. That whatever p value is less than the number of transmission massages and their average also is less. In table (1) per varying amounts of p, m value has shown.

Table 1. Calculation of m by p

p	m
0.05	1.05
0.1	1.11
0.2	1.25
0.3	1.43
0.4	1.67
0.5	2
0.7	3.33
0.8	5
0.9	10
0.95	20

When p is small we can select a new coordinator just by one or two massages. Meanwhile the maximum number of message that needs to be sent is n-1, which happens only while the smallest process find the coordinator dead and all other process have dead. In all last methods, if a big process finds that the coordinator is dead, the less number of messages is send. If two or more processes find that the coordinator is dead, concurrency, and they act independently so they arrive to a same result finally.

IV. Conclusion

In this paper, we discussed the drawbacks of Garcia-Molina's bully algorithm and modified bully algorithm. Then we presented an optimized method. Our algorithm is more efficient rather than the bully algorithm and modified bully algorithm, because the complexity decreased from O (n2) and O (n) to O (1) (formula 1-3) and fewer message passing and the fewer stages are required in our algorithm.

References

[1] G. Le Lan, "Distributed System Towards a Formal Approach", In Information Processing 77, B. Gilchrist, Ed.Amsterdam, Thenetherlands: North-Holland, 1977, pp.155- 160.

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

[3] S. Singh, J.F. Kurose, "Electing good leaders (election leader algorithm)", Journal of Parallel and Distributed Computing, Vol. 21, No. 2, , May 1994, pp. 184-201.

[4] M. S. Kordafshari, M. Gholipour, M. Jahanshahi, A. T. Haghighat, "Modified bully election algorithm in distributed system", WSEAS Conferences, Cancun, Mexico, 2005, May 11-14.

[5] D.MenasceR.Muntz and J.Popek, "A locking protocol for resource coordination in Distributed databases", ACM TODS, VOL.5, NO.2, JUNE 1980, pp.103-138.

[6] S.H. Park, Y. Kim, and J. S. Hwang "An Efficient Algorithm for Leader- Election in Synchronous Distributed Systems", IEEE Transaction on Computers, vol. 43, no. 7, 1991-1994, 1999 pp.

[7] G. Singh, "Leader Election in the Presence of Link Failures", IEEE Transaction on Parallel and Distributed Systems, vol. 7, no. 3, March 1996, pp.231-236.