An Announcer Based Bully Election Leader Algorithm in Distributed Environment

Minhaj Khan(✉), Neha Agarwal, Saurabh Jaiswal, and Jeeshan Ahmad Khan

Shri Ramswaroop Memorial University, Lucknow, Uttar Pradesh, India
minhajkhan7786@gmail.com, lko.neha@gmail.com, saurabhjaiswalcs@gmail.com, jeeshan.jak@gmail.com

Abstract. In distributed system, a job is divided into sub jobs and distributed among the active nodes in the network; communication happens between these nodes via messages passing. For better performance and consistency, we need a leader node or coordinator node. There is no compulsion that leader node should be same all the time because of out of services, crashed failure etc. Over past years, tremendous algorithms have been introduced to select a new leader when leader is dead or crashed. Bully algorithm is a well known traditional method for the same when leader or coordinator becomes crashed. In this algorithm the highest Id node is selected as a leader, but this algorithm has some drawbacks such as message passing complexity, heavy network traffic, redundancy etc. To overcome this problem, we are introducing an announcer based Bully election leader algorithm which is the modified version of original algorithm to overcome the above mentioned shortcomings. In our proposed algorithm we use an announcer who will decide the next leader or coordinator after current leader failure. Our analytical comparison presents that our proposed algorithm uses less messages passing with respect to the existing algorithms.

Keywords: Distributed systems · Bully algorithm · Announcer
Message passing

1 Introduction

In distributed computing various nodes connected to one another via a network to solve a common problem without having any concern who performed it. In a network each node communicates with each other to make an acceptable decision but problem arises when consistency needed among the active nodes. To determine consistency, a node is selected as a leader and act as a centralized controller node for that decentralized distributed system to achieve some specific goals like synchronization, time scheduling, load balancing, mutual exclusion etc. [10]. Several algorithms have been presented to select a leader or coordinator in a distributed system like bully algorithm, ring algorithm, LCR algorithm which use some specific topology such as spanning tree, fully connected graph, ring topology etc. [1, 14]. So here we got a chance to select any topology for designing the distributed system which reduces time and message passing during the leader selection [14].

P. Bhattacharyya et al. (Eds.): NGCT 2017, CCIS 827, pp. 664–674, 2018.
https://doi.org/10.1007/978-981-10-8657-1_51

Here we are presenting a new approach which uses the fully connected topology and extended version of bully algorithm by which we can reduce message passing, network traffic, redundancy as well as time to select a leader. This approach is based on below basic assumptions:

a. A synchronous timeout based system is in use.
b. Each node contains only information about its owned unique node id and announcer id as well as leader id.
c. During election, the highest id node will be the announcer.
d. Whoever will have noticed that leader is down, it will become a new leader and same information will broadcast by the announcer.
e. There are N nodes in the network and N^{th} node is announcer by default. If it will crash, then $(N - 1)^{th}$ node will be the new announcer.
f. If message will come from new announcer, then node will have to validate the announcer id and update its table accordingly.
g. After recovery, failed node can again join the system again.

The layout of the paper is organized as follows. This paper is separated into 6 sections. Section 1 is describing the basics of distributed system. Section 2 is introducing literature survey regarding leader selection in distributed system with its limitations. Section 3 presents the proposed algorithm and is briefly explained with an example. In Sect. 4 we have figured out the performance analysis of the proposed modified algorithm. In Sect. 5 there is a comparison between proposed and other existing algorithms. Section 6 concludes the paper along with future work.

2 Literature Survey

For electing a leader in distributed system, various algorithms have been proposed. In this section we are going to describe two notable leader election algorithm i.e. original bully algorithm which is one of the basic election algorithm [1], and second is modified bully election algorithm [4].

2.1 Original Bully Algorithm

Bully algorithm was proposed by Garcia Molina in 1982. In this algorithm, the node having the highest Id works as a leader [4, 8, 9]. If any node observes that the coordinator is not responding i.e. the coordinator failed then detector node will start an election and sending election message to all nodes which are having higher Ids than its own Id. If detector node doesn't receive any response from the receivers within certain time duration, then it elects itself as a leader and send leader message to all nodes in the network but if the detector node receive responses from the receivers, it means these nodes are alive and will take over the election. Afterwards all nodes give up except one node that means the last one node who wins is now work as a current leader and broadcast leader message to all nodes that have lower Id [12].

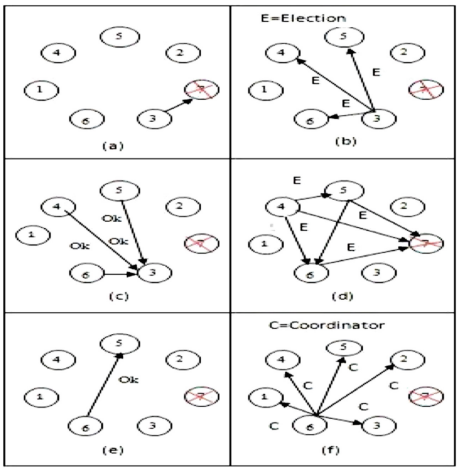

Fig. 1. Traditional Bully algorithm steps for electing a coordinator

In Fig. 1 node 7 has biggest Id and work as a coordinator.

a. Node 3 observes the coordinator is failed.
b. After observing coordinator is failed, node 3 send election message to their biggest Id nodes i.e. nodes 4, 5 and 6.
c. Node 3 received Ok messages from nodes 4, 5 and 6 that means these nodes now will take over the election.
d. After sending ok message to node 3, nodes 4, 5 and 6 will send election message to their biggest Id nodes.
e. Nodes 5, 6 send ok message to node 4 and node 6 send ok message to node 5.
f. Node 6 wins the election and elects as a new coordinator and send coordinator message to all nodes.

2.1.1 Limitations

a. This algorithm needs high number of messages passing for electing a new coordinator when current coordinator is failed and due to this heavy network traffic generates in the system.
b. At a time two nodes may broadcast as a coordinator if they are next biggest IDs nodes.
c. When N nodes observe that the coordinator is failed then accordingly number of nodes, election message will be started which will impose heavy network traffic.
d. There is no confirmation that the coordinator is exactly failed or not.

2.2 Modified Bully Algorithm Presented by M. S. Kordafshari et al.

M. S. Kordafshari et al. proposed a new algorithm which is the enhanced version of original Bully algorithm. This algorithm basically focuses on reducing message passing

and network traffic and ensured that only one leader remains in the system at a time. In this algorithm, when any node finds that the leader is crashed then it will send election message to its highest process number. If it will not receive any response messages from the receivers, then it elects itself as a leader and send coordinator message to all alive nodes. If it will receive response message from them, then it will send GRANT message to highest process number between them [4, 11]. After receiving the GRANT message, the highest process number will have to send coordinator message to all alive nodes [4, 11, 13].

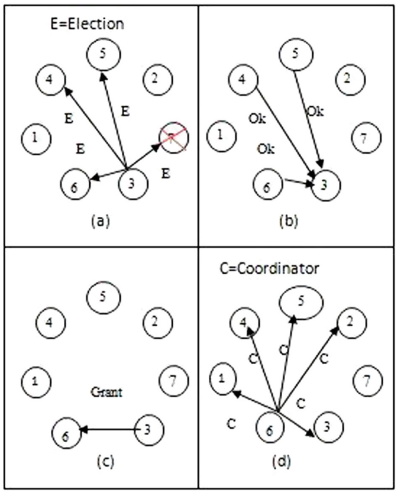

Fig. 2. Modified bully algorithm steps for selecting a coordinator

In Fig. 2, node 7 is working as a leader because it has highest process number.

a. Node 3 observes the coordinator is failed then send election message to their biggest Id nodes i.e. nodes 4, 5 and 6. If node 3 doesn't receive any responses from these nodes, then it will select itself as a leader and send coordinator message to all alive nodes.
b. After receiving the election message these nodes send responses (ok) to node 3.
c. After receiving the responses from these nodes, node 3 compares the Ids of these nodes and send grant message to biggest Id node. Here node 6 is biggest Id node.
d. After receiving GRANT message from node 3, node 6 will elect as a new coordinator and node 6 send coordinator message to all nodes.

2.2.1 Limitations

a. If a node failed after sending election message to biggest Id nodes or failed after getting priority of biggest Id nodes, the biggest Id nodes will wait for 3D time D is average propagation delay time) for coordinator message [11],if they don't get any coordinator message, they will start election again [4].

b. After getting the GRANT message if that node crashed (which send Grant message) then detector will have to start election again and send the GRANT message again to the biggest Id nodes between the remaining ones and this will create redundancy
c. Each redundant election consumes resources and generates more messages passing and network traffic.
d. There is no such guarantee of coordinator is exactly failed or not.

3 Proposed Algorithm

This paper proposes a new algorithm to elect a leader between nodes in the distributed environment. This algorithm overcomes the problem which is revealed in bully algorithm and modified bully algorithm. This algorithm is announcer based algorithm where announcer decide who will be the next leader when current leader is failed. In this algorithm the node which has the biggest Id is work as a announcer and if any node notices the leader is failed then send election message to announcer, the announcer decide who will be the next leader and also take confirmation of the old leader is exactly crashed or not.

The variables which are used in this algorithm are given below:
anp_id -> announcer process Id, this variable is used for announcer
slp_id -> store leader process Id, this variable is used to store the leader process Id
mcp_id -> message creator process Id, this variable create the message creator process Id
clp_id -> crashed leader process Id, this variable contains recently crashed leader process Id.

3.1 Algorithm

Here we are going to describe the algorithm which is used for leader node election (Figs. 3, 4 and 5).

```
int anp_id, slp_id, mcp_id, clp_id
//when any node N detect the leader is crashed, initiate an election
Create message msg (mcp_id, clp_id) and send to announcer
Start timer
//upon receiving message by announcer
If (slp_id = = clp_id)
{
        If (leader is failed)
          slp_id = mcp_id
broadcast leader msg (anp_id, mcp_id, clp_id)
else
broadcast  message msg (anp_id, slp_id, null)
}
else
discard the received message msg (mcp_id, clp_id)
}
```

Fig. 3. Pseudo code when any node finds that the leader is crashed

```
//when nodes N1,N2,N3.....Nn detect the leader is crashed, initiate an election
Create message msg (mcp_id, clp_id) and send to announcer
 Start timer
//upon receiving messages from N1,N2,N3.....Nn for i=1 to n
Find the node Ni whose mcp_id is highest and discard the messages transmitted by other
nodes If (slp_id = = clp_id) \\clp_id taken from node Ni {

   If (leader is failed)
       slp_id = mcp_id
broadcast leader msg (anp_id, mcp_id, clp_id)
else
broadcast message msg (anp_id, slp_id, null)
}
else
discard the received message msg (mcp_id, clp_id)
```

Fig. 4. Pseudo code when two or more nodes find the crash of the leader

```
//when announcer and leader both are crashed and any node X detect the leader is
crashed Initiate an election
Create message msg (mcp_id, clp_id) and send to announcer
Start timer
If (sender not receives any responses from announcer)
Send message to N-1 node \\N-1 node is the next highest Id node
//after receiving message by N-1 node Check (anp_id is crashed
or not)
If (anp_id is crashed)
anp_id(N) = anp_id(N-1)
Check (leader is failed or not)
If (leader is failed)
broadcast leader message (anp_id, mcp_id, clp_id)
else
broadcast message msg (anp_id, slp_id, null)
}
else
discard the received message msg (mcp_id, clp_id)
```

Fig. 5. Pseudo code when any node find announcer and leader both are crashed

Case1: When any node found that leader is crashed and it needs a leader then immediately it will start election by sending election message [(msg <mcp_id, clp_id>)] to the announcer. Announcer will check the slp_id with clp_id and if it matched then it will check whether stored leader is really crashed or not. If it is crashed then announcer will store the mcp_id as a leader and broadcast this message [(msg <anp_id, mcp_id, clp_id>)] to remaining nodes in the network to inform others about the new leader.

After getting message from the announcer, nodes will update their table and replace old leader id with new one.

In the example (shown in Fig. 6), node 4 founds that the current leader (node 5) is crashed, so node 4 create a message (msg <4, 5>) and send to announcer (node 7). After receiving the message (msg <4, 5>), announcer compare its store leader process

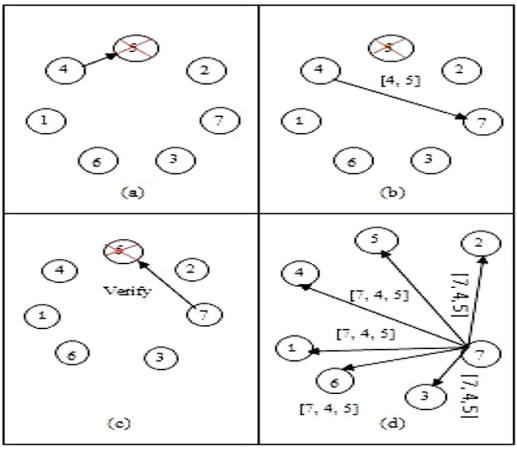

Fig. 6. Steps when one node detect leader is failed

id (slp_id) with the second field of the received message (clp_id). Here slp_id (5) = clp_id(5). According to our algorithm the announcer (anp_id) verify the leader is exactly crashed or not. If leader is crashed then it store mcp_id(4) as a new leader and broadcast this message (msg <7, 4, 5>) to remaining nodes in the network to inform others about the new leader. After getting message from the announcer, nodes will update their table and store mcp_id(4) as a slp_id(4).

Case2: If simultaneously two or more nodes noticed that leader is down, they will send message to the announcer. Now announcer have to elect leader between them based on whose node id is bigger.

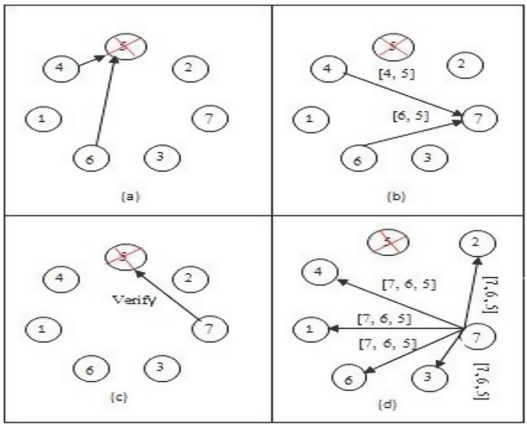

Fig. 7. Steps when two nodes detect leader is failed

In the example (shown in Fig. 7) when at a time node 4 and node 6 detect the leader is down or crashed. In this case node 4 create the message <4, 5> and node 6 create the message <6, 5> and send to announcer (node7). After getting message the announcer compare the message creator process id (mcp_id) of both nodes. Here mcp_id (6) > mcp_id(4). So announcer will discard the message <4, 5> and select the message <6, 5> . After that the announcer compares slp_id and clp_id. Here slp_id (5) = clp_id(5), so announcer check the slp_id is exactly crashed or not if crashed then it store mcp_id(6) as a slp_id and broadcast message <7, 6, 5> to all active nodes in the network to inform about the new leader. After receiving message the nodes update their table and store mcp_id(6) as a slp_id.

Case3: If leader and announcer both were crashed and any node notices it then it will send this message (msg < mcp_id, clp_id >) to $(N - 1)^{th}$ node considering it a new announcer. After receiving the message, $(N - 1)^{th}$ node will have to validate it and broadcast message (msg <anp_id, mcp_id, clp_id>) to the network.

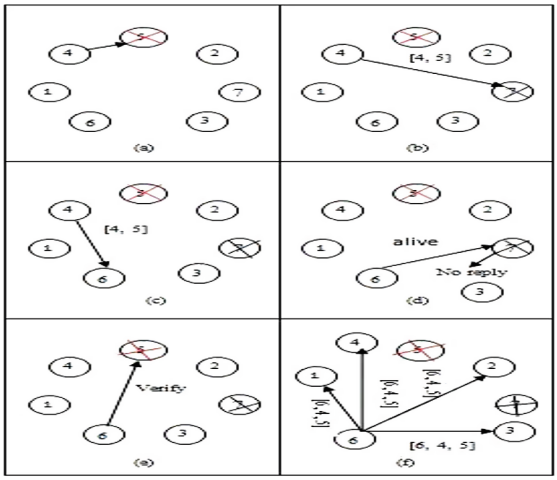

Fig. 8. When any node detect announcer and leader both are crashed

Here in example (shown in Fig. 8) node 4 detect that the leader 5 is crashed then it immediately create a message <4, 5> and send to announcer (node 7). After a certain period of time if node 4 not received any responses from the announcer. Then it sends the message <4, 5> to node 6. According to our algorithm after receiving the message node 6 will check the announcer (node 7) is exactly down or not if node 7 is down then node 6 updates their table and store as new anp_id. After that it compares slp_id and clp_id. Here slp_id(5) = clp_id(5). So node 6 store mcp_id(4) as a new slp_id and broadcast to all active nodes. After receiving message the nodes update their table and store the anp_id(6) as a new announcer and mcp_id(4) as new store leader process slp_id.

4 Performance Analysis

In Bully algorithm [11] for choosing a leader we need of huge number of messages passing and time. The message complexity of this algorithm in worst case is O(n2) and message complexity of modified Bully algorithm is O(n) in worst case. It is more efficient than Bully algorithm but it also require more messages to choose a leader but our proposed algorithm more efficient and better performance in comparison to these algorithm because it require less message passing for choosing a leader in distributed system which is shown in Table 1. The message complexity of our proposed algorithm is O(n) and mathematical analysis of this algorithm is shown in Sect. 4.1.

4.1 Mathematical Analysis

Best Case: If there are n nodes in a network and only one node observes leader failure then number of message passing (M) between the nodes for electing a leader will be

$$M = 2 + 1 + (n - 2) = 3 + (n - 2)$$

Average Case: If there are n nodes in a network and more than one node (assumed x) observes leader failure then number of message passing (M) between the nodes for electing a leader will be

$$M = 2 * x + 1 + (n - 2)$$

Worst Case: There are n nodes in a network and all nodes detect leader failure then number of message passing (M) between the nodes for electing a leader will be

$$M = 2 * (n - 2) + 1 + (n - 2) = 3 * (n - 2) + 1$$

5 Comparison with Other Algorithms

In this section, we compare our proposed algorithms with respect to the existing algorithms based on their message passing complexity. Table 1 shows the number of message passing to select a leader in worst case. Figure 9 shows a comparison graph in our proposed algorithm, Bully Algorithm and modified Bully Algorithm. Graph shows comparison where number of nodes denoted by horizontal axis and number of message denoted by vertical axis.

Table 1. Comparison among bully algorithm, modified bully algorithm and our proposed algorithm

No of nodes in a network	Leader election algorithms		
	Bully algorithm	Modified bully algorithm	Proposed algorithm
5	24	14	10
10	99	29	25
15	224	44	40
20	399	59	55
25	624	74	70

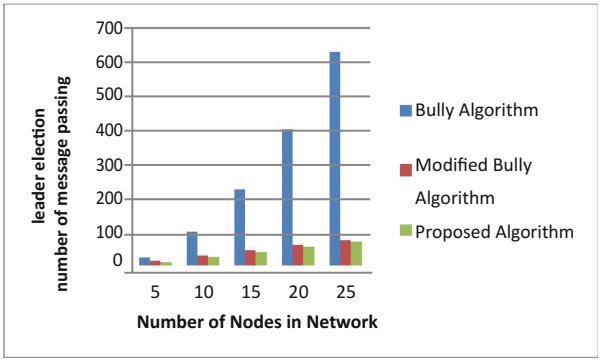

Fig. 9. Comparison among proposed algorithm, bully algorithm and modified bully algorithm

6 Conclusion and Future Works

After analysis of Bully algorithm and modified Bully algorithm, we propose a new algorithm for selection of a leader that gives overall less message passing during the selection of a leader. The main idea of over proposed algorithm is that it uses highest ID as an announcer node who will decide which one is the next leader in case of leader failed or crashed. And by analytical simulation, we have shown that in our propose algorithm the number of message passing for selecting a leader is less than with respect to entire leader election algorithm.

As a future work, we will study to overcome the problem of message complexity in worst case that will never affect the performance of our proposed algorithm.

References

1. Garcia-Molina, H.: Elections in a distributed computing system. IEEE Trans. Comput. **31**, 48–59 (1982)
2. Mirakhorli, M., Sharifloo, A.A., Abbaspour, M.: A novel method for leader election algorithm. In: The 7th IEEE International Conference on Computer and Information Technology (CIT 2007), pp. 452–456, October 2007
3. Kordafshari, M.S., Gholipour, M., Jahanshahi, M., Haghighat, A.T.: Modified bully election algorithm in distributed systems. In: WSEAS Conference (2005)
4. Park, S.H., Hwang, Y.K.: An efficient algorithm for leader-election in synchronous distributed systems. IEEE Trans. Comput. **43**(7), 1991–1994 (1999)
5. Gholipur, M., Kordafshri, M.S., Jahanshani, M., Rahmani, A.M.: A new approach for election algorithm in distributed systems. In: International Conference on Computer and Information Technology (2009)
6. Effat Parvar, M.R., Yazdani, N., Parvar, M.E., Dadlani, A., Khonsari, A.: Improved algorithms for leader election in distributed systems. In: 2nd International (IEEE) conference, vol. 2 (2010)
7. Yassein, M.B., Alslaity, A.N., Alwidian, S.A.: An efficient overhead-aware leader election algorithm for distributed systems. IJCA J. **49**(6) (2012)
8. Alhadidi, B., Baniata, L.H., Baniata, M.H.: AlSharaiah, M: Reducing massage passing and time complexity in bully election algorithms using two successors. Int. J. Electron. Electr. Eng. **1**(1), 1–4 (2013)
9. Soundarabai, P.B., Sahai, R., Thriveni, J., Venugopal, K.R., Patnaik, L.M.: Efficient Bully Election Algorithm in Distributed Systems, pp. 243–251. Elsevier Publications, Amsterdam (2013)
10. Rahman, M., Nahar, A.: Modified Bully algorithm using election commission. MASAUM J. Compu. **1**(3), 88–96 (2009)
11. Chhabra, S., Tyagi, G., Mundra, A., Rakesh, N.: Location based coordinator election algorithm in distributed environment. In: International Conference on Computer and Computational Sciences (2015)
12. Kordafshari, M.S., Gholipour, M., Jahanshahi, M., Haghighat, A.T.: Two novel algorithms for electing coordinator in distributed systems based on bully algorithm. In: 4th WSEAS International Conference (2005)
13. Biswas, A., Dutta, A.: A timer based leader election algorithm. IEEE (2016)
14. Yadav, D.K., Sharma, V.K.: Optimal distributed leader election algorithm. Int. J. Adv. Res. Comput. Sci. Technol. (IJARCST), **2** (2014)
15. Gawali, D.P.: Leader election problem in distributed algorithm. Int. J. Comput. Sci. Technol. (IJCT), **3** (2012)
16. Beaulah Soundarabai, P., Thriveni, J., Venugopal, K.R., Patnaik, L.M.: An improved leader election algorithm for distributed systems. Int. J. Next Gener. Netw. **5**, 21 (2013)
17. Gajre, V.P.: Comparison of bully election algorithms in distributed system. Int. J. Sci. Res. Publ. **3** (2013)
18. Katwala, H., Shah, S.: Study on election algorithm in distributed system. IOSR J. Comput. Eng. **7**, 34–39 (2012)
19. Shirali, M., Toroghi, A.H., Vojdani, M.: Leader election algorithms: history and novel schemes. IEEE Computer Society (2008)
20. Sathesh, B.M.: Optimized bully algorithm. Int. J. Comput. Appl. **121** (2015)
21. Mamun, Q.E., Moham, S.: Modified bully algorithm for electing coordinator in distributed systems. In: WSEAS International Conference on Software Engineering, Parallel and Distributed Systems, pp. 22–28