Improved Bully Election Algorithm for Distributed Systems

P Beaulah Soundarabai[a], Ritesh Sahai[a], Thriveni J[b], K R Venugopal[b], L M Patnaik[c]

[a]Department of Computer Science , Christ University, Bangalore 560 029 India,
Contact: beaulah.s@christuniversity.in

[b]University Visvesvaraya College of Engineering, Bangalore University, Bangalore.

[c]Honorary Professor, Indian Institute of Science, Bangalore.

Electing a leader is a classical problem in distributed computing system. Synchronization between processes often requires one process acting as a coordinator. If an elected leader node fails, the other nodes of the system need to elect another leader without much wasting of time. The bully algorithm is a classical approach for electing a leader in a synchronous distributed computing system, which is used to determine the process with highest priority number as the coordinator. In this paper, we have discussed the limitations of Bully algorithm and proposed a simple and efficient method for the Bully algorithm which reduces the number of messages during the election. Our analytical simulation shows that, our proposed algorithm is more efficient than the Bully algorithm with fewer messages passing and fewer stages.

Keywords: Bully Algorithm, Distributed Systems, Leader Election, Synchronization.

1. INTRODUCTION

Distributed computing is a decentralized and parallel computing, using two or more computers communicating over a network to accomplish a common task. Centralized control in distributed systems helps to achieve some specific goals such as mutual exclusion, synchronization, load balancing, and time scheduling. This type of distributed system often requires a unique node to play the role of leader or coordinator of the other nodes to take care of synchronization. As node crash failure is very common in distributed systems. Failure of a leader node requires special attention and needs extra tasks to elect another one to act as leader.

The collaborating processes are often identical. One of the central problems is election of a leader. Given a network of processes, exactly one process should take the decision that it is the leader. It is usually required that all non-leader processes are informed or involved in the process of the leader election. A leader election algorithm is one of the basic activities of distributed systems, as it acts as a basis for more complex and high level algorithms and applications. An important challenge in distributed systems is the adoption of suitable and efficient algorithms for coordinator election. The main role of an elected coordinator is to manage the use of a shared resource in an optimal manner which in turn maintains the coherency of the system even during partial failures.

1.1. Motivation

The main drawback of Bully algorithm is more number of message passing. As it is mentioned before the message passing has order O(n^2) that increases traffic in network. It also has five stages to decide the next leader which would waste a lots of time for the processes to resume their normal execution. Bully algorithm is a safe way for election; however its traffic is relatively high.

1.2. Contribution

In this paper, we have proposed a modified Bully algorithm which preserves all the advantages of the existing algorithm and at the same time eliminates the limitations of it by reducing the number of messages and the the number of stages to elect the next leader.

1.3. Organization

The remainder of this paper is organized as follows: Section 2 reviews the related work, Section 3 describes the Problem Definition and Methodology of Bully algorithm, Modified Bully algorithm is given in Section 4, Section 5 details the simulation and the comparison of the two algorithms, Conclusions are presented in Section 6.

2. LITERATURE SURVEY

Effat Parvar M R [1] described novel approaches towards improving the Bully and Ring algorithms and also proposed the heap tree mechanism for electing the coordinator. The higher efficiency and better performance with respect to the existing algorithms was also validated through simulation.

Sandipan Basu [2] has discussed the limitations of bully algorithm and proposed a modified algorithm. In the original bully algorithm, when the leader process is crashed, immediately the new leader is elected. But, if the old leader process comes back, it once again initiates the election. The author suggests that there need not be another election, instead, the old leader process can accept the new leader process by sending the new request of who the leader is?, to its neighbor. In the next round of election, it can try becoming the leader.

Muhammad Mahbubur Rahman *et al.*, [3] have also proposed a modified bully election algorithm. In their paper, they say that the bully algorithm has $O(n^2)$ messages which increases the network traffic. In the worst case, n number of elections can occur in the system which again in turn will yield in a heavy network traffic. They have proposed the same algorithm but with Failure Detector, Helper processes to have unique election with the Election Commission.

Chang-Young Kim *et al.*, [4] have proposed the election protocol for reconfigurable distributed systems which again was based on bully election algorithm. The actual election is run by the base stations making the protocol, energy efficient. The protocol is also independent from the overall number of mobile hosts and the data structures required by the algorithm are managed at the base station, making the protocol scalable as well.

M S Kordafshari *et al.*, [5] have done a survey of synchronous bully algorithm and modified it with an optimal message algorithm and also discussed the limitations of these algorithms. The authors have tried to reduce the number of elections happening in the classical bully algorithm. The proposed algorithm has only one election at any point of time, which brings down the number of messages being exchanged drastically. Sepehri M *et al.*, [6] have dealt with the distributed leader election algorithm for a set of processes connected by a tree network. The authors have proposed a linear time algorithm using heap structure using reheap up and reheap down algorithms. They also have analysed the algorithm and reached a logarithmic number of message complexity.

Sung Hoon Park [7] has proposed Failure Detector which has an overhead module with a specialised function that detects crash and recovery of a node in a system. This report can be given to any process at request. The author modified the bully algorithm using the failure detector. The performance of the system goes down because of the overhead of Failure Detector. Failure Detector is the centralized component, which leads to the traditional problems of single point of failure and bottleneck scenario of single queue access the resource.

Zargarnataj [8] has developed an algorithm which is based on Bullys election algorithm with an additional feature of an assistant to the new leader. If the present leader node crashes, the assistant leader would become the new leader without any overhead of election. Whenever any process realises the absence of the leader, it immediately sends a message to the assistant leader to alert it. When the assistant leader receives any such message, it confirms the unavailability of the leader by timeout message and if it is true, it broadcasts the leader message to all the processes. But the limitation of this algorithm is that if the assistant leader is also not available then there is a lot of messages sent and time de-

lays to invoke the election algorithm once again.

3. BULLY ALGORITHM

Distributed systems require some special capabilities of a good and efficient leader election algorithm, such as leader longevity, low communication overhead, low complexity in terms of time and messages, and providing uniqueness to the elected leader. Several algorithms have been proposed to deal with the leader node failure problem, and the Bully Algorithm is the classical one amongst them for electing a leader node in synchronous systems, although this algorithm demands a large number of messages between the nodes.

Bully algorithm was first presented by Garcia Molina in 1982. The Bully algorithm in distributed computing system is used for dynamically electing a leader by using the process ID number. The process with the highest process ID number is elected as the leader process.

1. Assumptions

(i) Each process has a unique and not null number to distinguish them and each process knows other process number.

(ii) Processes dont know which ones are currently up and down.

(iii) The entire system is synchronous. Timeouts are used for deciding process failure.

(iv) Processes can crash even during execution of algorithm.

(v) Message delivery between processes is reliable and time bound.

2. Aim:

The aim of election Algorithm execution is selecting one process as leader (Coordinator) that all processes agree with it. In other words, electing a process with the highest priority or highest ID number as a leader or coordinator.
Suppose that the process P finds out the coordinator crashed, P immediately holds an election. This algorithm has the following steps:

a) Step1:
When a process, P, notices that the coordinator crashed, it initiates an election algorithm.

(i) P sends an ELECTION message to all processes with higher numbers respect to it.

(ii) If no one responses within the time limit, P wins the election and becomes a coordinator.

b) Step2:
When a process receives an ELECTION message from one of the processes with a lower number response to it:

(i) The receiver sends an OK message back to the sender to indicate that it is alive and will take over.

(ii) The receiver holds an election, unless it is already holding one.

(iii) Finally, all processes give up except one that is the new coordinator.

(iv) The new coordinator announces its victory by sending a message to all processes telling them, it is the new coordinator.

c) Step3:
Immediately after the process with higher number compare to coordinator is up, bully algorithm is run.

Figure 1 explains the steps involved in the Bully election algorithm.

(i) Process 4 holds an election

(ii) Process 5 and 6 respond, telling 4 to stop

(iii) Now 5 and 6 each hold an election individually leading to two simultaneous elections.

(iv) Process 6 tells 5 to stop by sending OK to it.

(v) Process 6 wins the election because no higher processes responded to it and it informs all the processes that it is the Coordinator from now on.

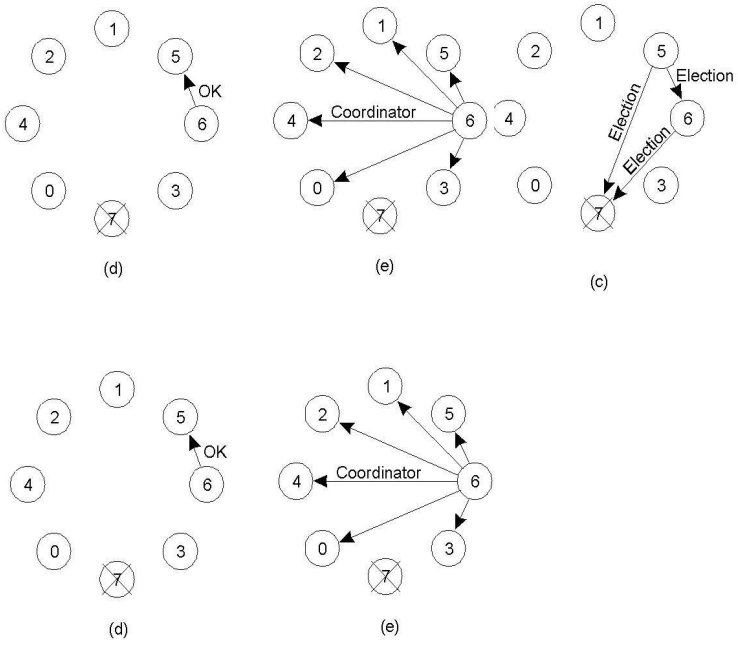


Figure 1. Bully Algorithm.

3.1. Advantages and limitations.

The advantages of Bully algorithm are that this algorithm is a distributed method with simple implementation [9][10][11]. This method requires at most five stages, and the probability of detecting a crashed process during the execution of algorithm is lowered in contrast to other algorithms. Therefore other algorithms impose heavy traffic in the network in contrast to Bully algorithm [12]. Another advantage of this algorithm is that only the processes with higher priority number respect to the priority number of process that detects the crash coordinator will be involved in election, not all process are involved. However the two major limitations of Bully algorithm are the number of stages to decide the new leader and the huge number of messages exchanged due to the broadcasting of election and OK messages [13].

4. MODIFIED BULLY ALGORITHM

Generally, in fault-tolerant distributed systems the leader node has to perform some specific controlling tasks and this node is well known to the other nodes. This node does not necessarily possess any extra processing feature to become elected, but having the highest process-ID. Election algorithms need a special mechanism to elect the leader. After crash failure of the leader node, it is urgently needed to reorganize the existing active nodes to call for an election and to elect a leader in order to continue the operation of the entire system.

4.1. Modified Bully Algorithm Details

1. Assumption

Besides having all the assumptions of the existing algorithm, we assume

(i) All processes hold an election flag, if this flag is true election cannot be initiated by any process.

(ii) All processes have a variable to store coordinator information.

a) Step1

Initially all election flag are set to false. When a process, P, notices that the coordinator crashed, it initiates an election algorithm.

(i) P sends an ELECTION message to all processes.

(ii) All processes set their election flag to true, so that none of the process can start

(iii) Coordinator variable reset to zero.

(iv) If no one responses, P wins the election and becomes a coordinator.

b) Step2

When a process receives an ELECTION message from one of the processes with lower numbered response to it:

(i) The receiver sends an OK message back to the sender to indicate that it is alive and will take over.

(ii) The sender P extracts process ID of receiver and store it in coordinator variable. Only IDs greater than the stored ID can override the coordinator ID variable value.

(iii) Finally, all processes responded and higher process ID among them is stored in coordinator variable.

(iv) The sender P collects coordinator ID from variable and informed him (coordinator process ID) that he is coordinator.

(v) The elected coordinator process cross check with his higher processes, if any higher process is alive he will take over, else currently elected process will be coordinator.

(vi) The new coordinator announces its victory by sending a message to all processes telling them, it is the new coordinator.

(vii) All processes set the coordinator ID in coordinator variable and reset election flag to false.

c) Step3

Immediately after the process with higher number compare to coordinator is up, bully algorithm is run.

Figure 2 shows the steps involved in Modified Bully Election Algorithm.

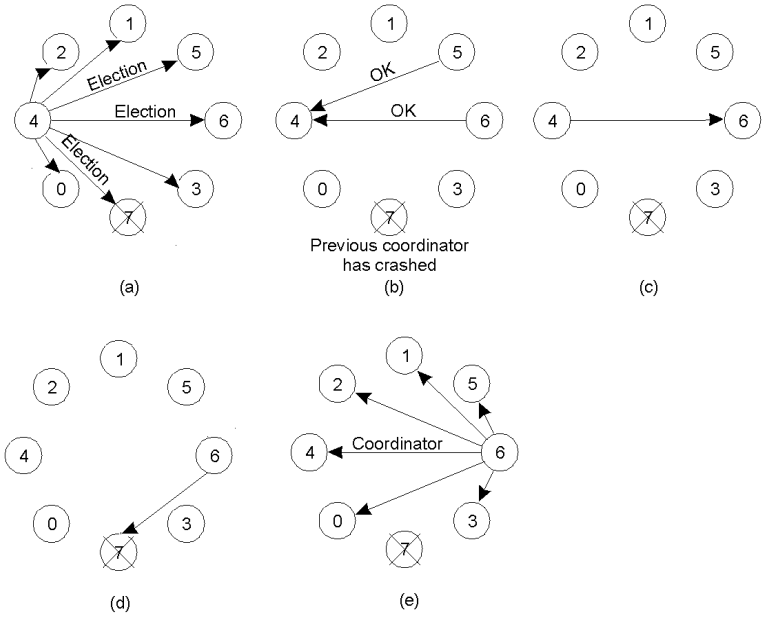


Figure 2. Modified Bully Algorithm.

(i) Process 4 holds an election

(ii) Process 5 and 6 respond, informing 4 about their presence in the system by OK message.

(iii) Processes 4 informs 6 to become coordinator.

(iv) Process 6 checks with process 7 if it is come back.

(v) Since no reply from process 7, process 6 wins and broadcasts the Coordinator Message to all the processes.

4.2. Advantages

Modified Bully algorithm is having all advantages of Bully algorithm. The additional advantages of modified Bully algorithm are that this algorithm is a very simple, having fail-safe mechanism, no parallel election, and reduced number of messages.

4.3. Limitations

How long the election initiator should wait to get response from all higher processes. If we keep a timer then the limitation could be the timeout value. Higher timeout will raise performance issue and lower timeout may miss responses from higher processes due to busy network traffic. However failsafe mechanism will be very helpful in this case.

5. SIMULATION AND COMPARISION

5.1. Election Algorithm Simulator

We used a GUI based simulator for simulating election algorithm. This simulator was capable of creating process node, creating distributed process network, message passing and electing the coordinator. The simulator is enhanced with GUI capability, allowing users to save and load the distributed network, display messages, selecting start node (green) and recognizing coordinator node by changing the color (red).

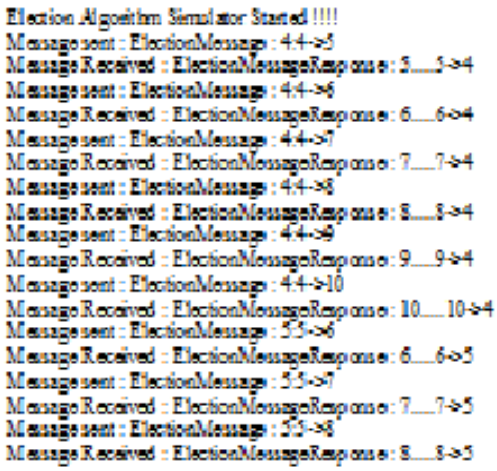


Figure 3. Simulation Log1

Figure 4. Simulation Log2

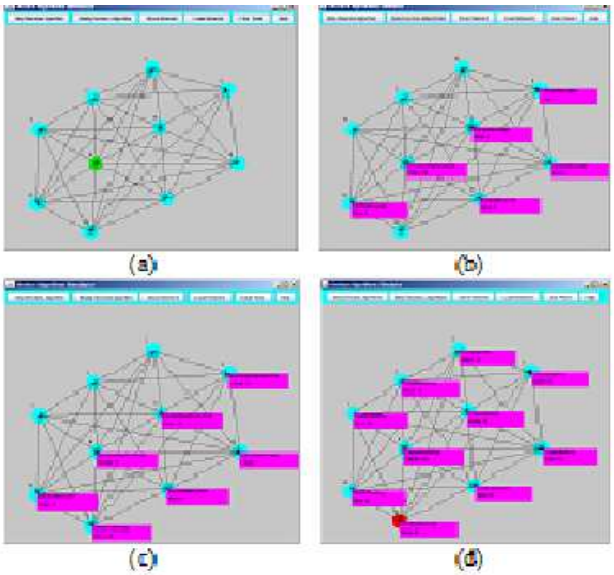


Figure 5. Bully Algorithm Simulation

5.2. Bully Algorithm Simulation

(i) Simulation Setup:
Started with 10 process nodes participating in election. Node 4 started election.

(ii) Bully Simulation Logs

Figure 3 and Figure 4 shows the the Simulation logs of Bully Algorithm. Figure 5 represents the simulation of Bully algorithm in which Figure 5(a) and Figure 5(b) shows that, process ID 4 identifies the absence of the leader and initiates the election by sending the election message to its higher ups namely to processes 5, 6,. . . , 10. All these processes in turn start their own election and concludes the election by the coordinator message where process ID 10 is the new leader. These activities are depicted in Figure 5(c) and Figure 5(d).

5.3. Modified Bully Algorithm Simulation

(i) Simulation Setup:
Started with 10 process nodes participating in election. Node 4 started election.

(ii) Modified Bully Simulation Logs

The Figure 6 shows the Simulation logs of the Modified Bully Algorithm.The simulation of Bully algorithm is represented Figure 7, where Figure 7(a) and Figure 7(b) show that, process ID 4 identifies the absence of the leader and initiates the election by sending the election message to its higher ups namely to processes 5, 6, . . . , 10. Unlike the Bully algorithm, all these processes reply to the initiator process 4 instead of starting their own election. In Figure 7c) and Figure 7(d) show that process 4 decides the new coordinator (which is 10 in our simulation) and informs process 10 to take over and the election gets concluded by the broadcast of coordinator message where process ID 10 is the new leader.

5.4. Message Comparison

Table 1 shows the comparison for both algorithms. In this table we represented the message growth following by corresponding number of processes in the distributed network. Table 1 shows that numbers of messages are increasing drastically in the Bully algorithm compare to the modified Bully algorithm.

```
Election Algorithm Simulator Started !!!!
Message sent : ElectionMessage : 4:4->1
Message sent : ElectionMessage : 4:4->2
Message sent : ElectionMessage : 4:4->3
Message sent : ElectionMessage : 4:4->5
Message Received : ElectionMessageResponse : 5:---5->4
Message sent : ElectionMessage : 4:4->6
Message Received : ElectionMessageResponse : 6:---6->4
Message sent : ElectionMessage : 4:4->7
Message Received : ElectionMessageResponse : 7:---7->4
Message sent : ElectionMessage : 4:4->8
Message Received : ElectionMessageResponse : 8:---8->4
Message sent : ElectionMessage : 4:4->9
Message Received : ElectionMessageResponse : 9:---9->4
Message sent : ElectionMessage : 4:4->10
Message Received : ElectionMessageResponse : 10:---10->4
Largest coordID : 10
Message sent : ElectedAsCoordinator : 4:4->10
Largest coordID (after checking for 2nd time) : 10
Message sent : CoordinatorMsg : 10:10->1
Message sent : CoordinatorMsg : 10:10->2
Message sent : CoordinatorMsg : 10:10->3
Message sent : CoordinatorMsg : 10:10->4
Message sent : CoordinatorMsg : 10:10->5
Message sent : CoordinatorMsg : 10:10->6
Message sent : CoordinatorMsg : 10:10->7
Message sent : CoordinatorMsg : 10:10->8
Message sent : CoordinatorMsg : 10:10->9
```

Figure 6. Modified Bully Algorithm Simulation Logs

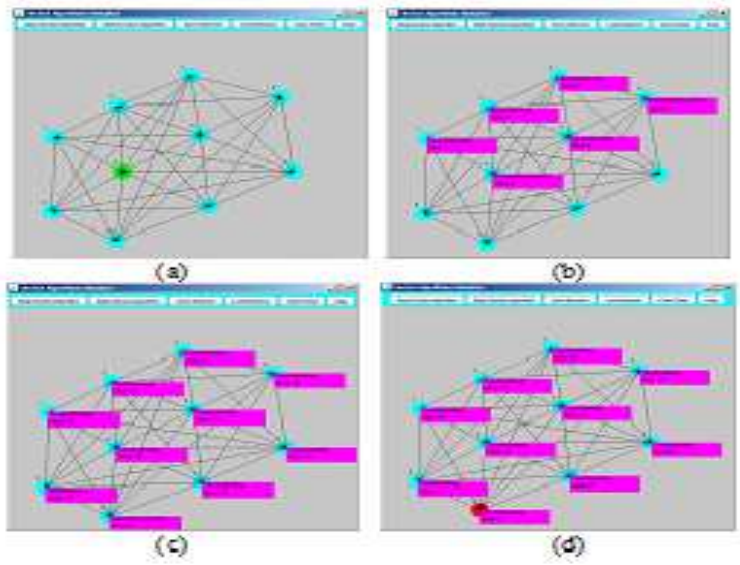


Figure 7. Modified Bully Algorithm Simulation

Table 1
Message Comparison of Bully and Modified Bully Algorithms

	Messages	
Processes	**Bully Algorithm**	**Modified Bully Algorithm**
5	24	13
10	99	28
15	224	43
20	399	58
25	624	73

Figure 8 shows a comparison graph where both Bully and modified Bully are highlighted in different colors. Graph presents the comparison where number of nodes represented by horizontal axis and number of messages represented by vertical axis. Graph shows that Bully is having curve shape that describe O(n^2) and modified Bully algorithm is having linear growth described by a straight line or O(n).

5.5. Simulation Result

Our simulation result shows that modified election algorithm is more efficient as it reduces the number of messages, also avoided any parallel election process. The comparative results are well explained by simulation logs, comparison graph and table.

5.6. Analytical Comparision

If only one process detects crashed coordinator
N: The number of processes
P: The priority number of processes that find out the crashed coordinator
T_m: The number of messages passing between processes when the Pth member detects the crashed Coordinator.
In Bully Modified algorithm the number of massages passing between processes for performing election is obtained from the following formula:

$$T_m = 2 * (N - P) + N \tag{1}$$

Which has Order O(n). In the worst case that is $P = 1$ (process with lowest priority number finds out crashed coordinator):

$$T_1 = 2 * (N - 1) + 1 = 3N - 1 \tag{2}$$

Whereas the number of massage passing between processes in the Bully algorithm for performing election is obtained from the following formula:

$$T_m = (N - P + 1)(N - P) + N - 1 \tag{3}$$

In the worst case that is P= 1 (process with lowest priority number detects crashed coordinator):

$$T_1 = N^2 - 1 \tag{4}$$

Which has Order O(n^2). Number of messages in proposed Bully algorithm will be equal to $3n$ -1 that obviously means this modified algorithm is better than bully algorithm. Now assume that the set of processes in S= {P_1, P_2, P_3, ..., P_n} from processes find out the crashed coordinator concurrently (P_1 is the lowest process).

In Bully algorithm, considering worst case and assuming lowest process start election, then:

(i) Total number of election message sent to set (S) of n processes ({P_1, P_2, P_3, , P_n}) are (n - 1).

(ii) Total response message received by P_1 is (n - 1).

(iii) Now P_2 will send election message to n - 2 processes.

(iv) Total response message received by P_2 is (n - 2).

(v) Similarly for P_3, P_4 and P_n.

(vi) Finally P_n informing to every process by sending coordinator message is again (n - 1) message.

The number of message passing between processes for performing election is obtained from the following formula:

T_m = (n - 1) + (n - 2) + (n - 3) + ... + (n - n - 3) + (n - n - 2) + (n + n - 1) + (n - 1)

Simplifying the above formula, we get

$$T_m = n(n + 1)/2 \tag{5}$$

which is of O(n^2).

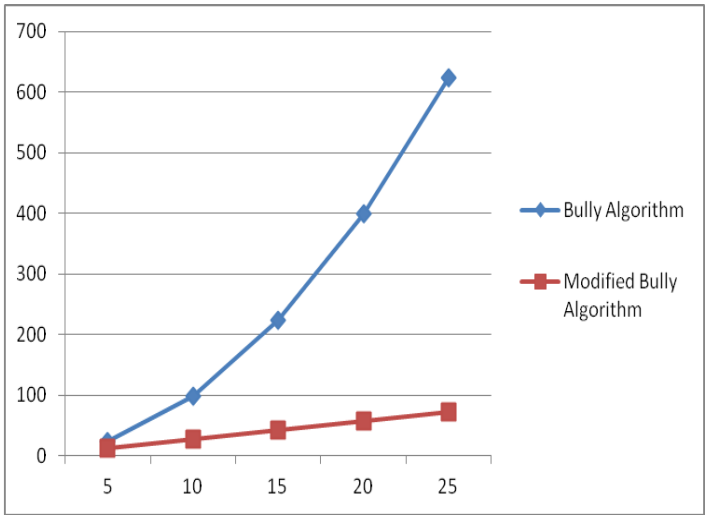


Figure 8. Number of Messages used During the Election

In our modified algorithm, considering worst case and assuming lowest process start election, then:

(i) Total number of election message sent to set (S) of n processes ($\{P_1, P_2, P_3, \ldots, P_n\}$) are ($n$ - 1).

(ii) Total response message received is (n - 1).

(iii) Informing to coordinator and coordinator to check with past coordinator involve two messages, and

(iv) Finally informing to every process by sending coordinator message is again (n - 1) message.

The number of message passing between processes for performing election is obtained from the following formula:
$T_m = (n\text{ -}1) + (n\text{ - }1) + 1 + 1 + (n\text{ - }1)$, or

$$T_m = 3n - 1 \text{ or } 3n \tag{6}$$

which is of O(n).

6. CONCLUSIONS

In this paper, we discussed the drawbacks of Bully algorithm and then we presented an optimized method for the Bully algorithm called modified bully algorithm. Modified Bully algorithm shows improved performance than the Bully algorithm. The additional advantages of modified Bully algorithm are that this algorithm is a very simple, having fail-safe mechanism, no parallel election, and reduced number of messages.

Our analytical simulation shows that our algorithm is more efficient rather than the Bully algorithm, in both number of message passing and the number of stages, and when only one process runs the algorithm message passing complexity decreased from O(n^2) to O(n). In this analysis we consider the worst case in modified algorithm. Result of this analysis clearly shows that modified algorithm is better than bully algorithm with fewer message passing and the fewer stages.

REFERENCES

1. M R Effat Parvar. Improved Algorithms for Leader Election in Distributed Systems, *in 2nd International Conference on Computer Engineering and Technology (ICCET)*, 2010.

2. Sandipan Basu. An Efficient Approach of Election Algorithm in Distributed Systems, *Indian Journal of Computer Science and Engineering (IJCSE)*, 2:16–21, 2011.
3. Muhammad Mahbubur Rahman and Afroza Nahar. Modified Bully Algorithm using Election Commission, *MASAUM Journal of Computing(MJC)*, 1(3):439–446, October 2009.
4. Chang-Young Kim and Sung-Hoon Bauk. The Election Protocol for Reconfigurable Distributed Systems, in *International Conference on Wireless Networks (ICWN)*, pages 295–301, 2006.
5. M S Kordafshari, M Gholipour, M Jahanshahi and A T Haghighat. Modified Bully Election Algorithm in Distributed System, in *WSEAS Conferences*, 2005.
6. M Sepehri and M Goodarzi. Leader Election Algorithm using Heap Structure, *Proceedings of the 12th WSEAS international conference on Computers (ICCOMP08)*, 2008.
7. Sung-Hoon Park. TA Stable Election Protocol based on an Unreliable Failure Detector in Distributed Systems, *Proceedings of IEEE Eighth International Conference on Information Technology: New Generations*, pages 976–984, 2011.
8. M Zargarnataj. New Election Algorithm based on Assistant in Distributed Systems, *IEEE International Conference on Computer Systems and Applications*, pages 324–331, 2007.
9. H Garcia-Molina. Elections in Distributed Computing System, *IEEE Transaction on Computers*, 310:48-59, 1982.
10. Chang Ben Ari. Principles of Concurrent and Distributed Programming, *Pearson Education*, 2nd edition, 2006.
11. Andrew S Tanenbaum. Distributed Systems Principles and Paradigms, *Beijing: Tsinghua University Press*, 2008.
12. G Le Lan. Distributed System Towards a Formal Approach, *Information Processing, The Netherlands: North-Holland*, pages 155–160, 1977.
13. Sung-Hoon-Park. A Probablistically Correct Election Protocol in Asynchronous Distributed System, *APPT, LNCS*, 310, pages 177–183, 2003.

P Beaulah Soundarabai is presently an Assistant Professor in the Department of Computer Science in Christ University, Bangalore. She obtained her Bachelors, Masters Degrees in Computer Applications from Madurai Kamarai University, Madurai. She also has completed M. Phil. in Computer Science. Presently she is pursuing Ph.D in the field of Distributed Systems in Christ University.

Thriveni J has completed Bachelor of Engineering, Masters of Engineering and Doctoral Degree in Computer Science and Engineering. She has 4 years of industrial experience and 16 years of teaching experience. Currently she is an Associate Professor in the Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore. Her research interests include Networks, Data Mining and Biometrics.

Venugopal K R is currently the Principal, University Visvesvaraya College of Engineering, Bangalore University, Bangalore. He obtained his Bachelor of Engineering from University Visvesvaraya College of Engineering. He received his Masters degree in Computer Science and Automation from Indian Institute of Science Bangalore. He was awarded Ph.D in Economics from Bangalore University and Ph.D in Computer Science from Indian Institute of Technology, Madras. He has a distinguished academic career and has degrees in Electronics, Economics, Law, Business Finance, Public Relations, Communications, Industrial Relations, Computer Science and Journalism. He has authored and edited 39 books on Computer Science and Economics, which include Petrodollar and the World Economy, C Aptitude, Mastering C, Microprocessor Programming, Mastering C++ and Digital Circuits and Systems *etc.*. During his three decades of service at UVCE he has over 400 research papers to his credit. His research interests include Computer Networks, Wireless Sensor Networks, Parallel and Distributed Systems, Digital Signal Processing and Data Mining.

L M Patnaik is currently Honorary Professor, Indian Institute of Science, Bangalore, India. He was a Vice Chancellor, Defense Institute of Advanced Technology, Pune, India and was a Professor since 1986 with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. During the past 35 years of his service at the Institute he has over 700 research publications in refereed International Journals and refereed International Conference Proceedings. He is a Fellow of all the four leading Science and Engineering Academies in India; Fellow of the IEEE and the Academy of Science for the Developing World. He has received twenty national and international awards; notable among them is the IEEE Technical Achievement Award for his significant contributions to High Performance Computing and Soft Computing. His areas of research interest have been Parallel and Distributed Computing, Mobile Computing, CAD for VLSI circuits, Soft Computing and Computational Neuroscience.

Improved Bully Election Algorithm for Distributed Systems

P Beaulah Soundarabai[a], Ritesh Sahai[b], Thriveni J[c], K R Venugopal[d], L M Patnaik[e]

[a]Department of Computer Science , Christ University, Bangalore 560 029 India, Contact: beaulah.s@christuniversity.in

[b]Department of Computer Science, Christ University, Hosur Main Road, Bangalore.

[c]University Visvesvaraya College of Engineering, Bangalore University, Bangalore.

[d]University Visvesvaraya College of Engineering, Bangalore University, Bangalore.

[e]Honorary Professor, Indian Institute of Science, Bangalore.

Electing a leader is a classical problem in distributed computing system. Synchronization between processes often requires one process acting as a coordinator. If an elected leader node fails, the other nodes of the system need to elect another leader without much wasting of time. The bully algorithm is a classical approach for electing a leader in a synchronous distributed computing system, which is used to determine the process with highest priority number as the coordinator. In this paper, we have discussed the limitations of Bully algorithm and proposed a simple and efficient method for the Bully algorithm which reduces the number of messages during the election. Our analytical simulation shows that, our proposed algorithm is more efficient than the Bully algorithm with fewer messages passing and fewer stages.

Keywords : Leader Election; Distributed Systems; Bully algorithm; Synchronization; distributed systems;

1. INTRODUCTION

Distributed computing is a decentralized and parallel computing, using two or more computers communicating over a network to accomplish a common task. Centralized control in distributed systems helps to achieve some specific goals such as mutual exclusion, synchronization, load balancing, and time scheduling. This type of distributed system often requires a unique node to play the role of leader or coordinator of the other nodes to take care of synchronization. As node crash failure is very common in distributed systems, failure of a leader node requires special attention and needs extra tasks to elect another one to act as leader.

The collaborating processes are often identical. One of the central problems is election of a leader. Given a network of processes, exactly one process should take the decision that it is the leader. It is usually required that all non-leader processes are informed or involved in the process of the leader election. A leader election algorithm is one of the basic activities of distributed systems, as it acts as a basis for more complex and high level algorithms and applications. An important challenge in distributed systems is the adoption of suitable and efficient algorithms for coordinator election. The main role of an elected coordinator is to manage the use of a shared resource in an optimal manner which in turn maintains the coherency of the system even during partial failures.

1.1. Motivation

The main drawback of Bully algorithm is the high number of message passing. As it is mentioned before the message passing has order $O(n^2)$ that increases traffic in network. It also has five stages to decide the next leader which would waste lots of time for the processes to resume their normal execution. Bully algorithm is a safe way for election; however its traffic is relatively high.

1.2. Contribution

In this paper, we have proposed a modified Bully algorithm which preserves all the advantages of the existing algorithm and at the same time eliminates the limitations of it by reducing the number of messages and the the number of stages to elect the next leader.

1.3. Organization

The remainder of this paper is organized as follows: Section 2 reviews the related work; Section 3 describes the Problem Definition and Methodology of Bully algorithm; Modified Bully algorithm is given in Section 4; Section 5 details the simulation and the comparison of the two algorithms; Conclusions are presented in Section 6.

2. LiteratureSurvey

Effat Parvar, M.R.[1] described novel approaches towards improving the Bully and Ring algorithms and also proposed the heap tree mechanism for electing the coordinator. The higher efficiency and better performance with respect to the existing algorithms was also validated through simulation.
Sandipan Basu [2] has discussed the limitations of bully algorithm and proposed a modified algorithm. In the original bully algorithm, when the leader process is crashed, immediately the new leader is elected. But, if the old leader process comes back, it once again initiates the election. The author suggests that there need not be another election, instead, the old leader process can accept the new leader process by sending the new request of who the leader is?, to its neighbor. In the next round of election, it can try becoming the leader.

Muhammad Mahbubur Rahman et al., [3] have also proposed a modified bully election algorithm. In their paper , they say that the bully algorithm has $O(n^2)$ messages which increases the network traffic. In the worst case, there will be n number of elections can occur in the system which again in turn will yield in a heavy network traffic. They have proposed the same algorithm but with Failure Detector, Helper processes to have unique election with the Election Commission.

Chang-Young Kim et al., [4] have proposed the election protocol for reconfigurable distributed systems which agai n was based on bully election algorithm. The actual election is run by the base stations making the protocol, energy efficien t. The protocol is also independent from the overall number of mobile hosts and the data structures required by the algorithm are managed at the base station, making the protocol scalable as well.
M S Kordafshari et al., [5] have done a survey of synchronous bully algorithm and modified it with an optimal message algorithm and also discussed the limitations of these algorithms. The authors have tried to reduce the number of elections happening in the classical bully algorithm. The proposed algorithm has only one election at any point of time, which brings down the number of messages being exchanged drastically. Sepehri M et al., [6] have dealt with the distributed leader election algorithm for a set of processes connected by a tree network. The authors have proposed a linear time algorithm using heap structure using reheap up and reheap down algorithms. They also have analysed the algorithm and reached a logarithmic number of message complexity.

Sung Hoon Park [7] has proposed Failure Detector which has an overhead module with a specialised function that detects crash and recovery of a node in a system. This report can be given to any process at request. The author modified the bully algorithm using the failure detector. The performance of the system goes down because of the overhead of Failure Detector. Failure Detector is the centralized component, which leads to the traditional problems of single point of failure and bottleneck scenario of single queue access the resource.
Zargarnataj [8] has developed an algorithm which is based on Bullys election algorithm with an additional feature of an assistant to the new leader. If the present leader node crashes, the assistant leader would become the new leader without any overhead of election. Whenever any process realises the absence of the leader, it immediately sends a message to the assistant leader to alert it. When the assistant leader receives any such message, it confirms the unavailability of the leader

by timeout message and if it is true, it broadcasts the leader message to all the processes. But the limitation of this algorithm is that if the assistant leader is also not available then there is lot of messages sent and time delays to invoke the election algorithm once again.

3. Bully Algorithm

Distributed systems require some special capabilities of a good and efficient leader election algorithm, such as leader longevity, low communication overhead, low complexity in terms of time and messages, and providing uniqueness to the elected leader. Several algorithms have been proposed to deal with the leader node failure problem, and the Bully Algorithm is the classical one amongst them for electing a leader node in synchronous systems, although this algorithm demands a large number of messages between the nodes.
Bully algorithm was first presented by Garcia Molina in 1982. The Bully algorithm in distributed computing system is used for dynamically electing a leader by using the process ID number. The process with the highest process ID number is elected as the leader process.

1. Assumption

(i) Each process has a unique and not null number to distinguish them and each process knows other process number.

(ii) Processes dont know which ones are currently up and down.

(iii) The entire system is synchronous. Timeouts are used for deciding process failure.

(iv) Processes can crash even during execution of algorithm.

(v) Message delivery between processes is reliable and time bound.

2. Aim:

The aim of election Algorithm execution is selecting one process as leader (Coordinator) that all processes agree with it. In other words, electing a process with the highest priority or highest ID number as a leader or coordinator.
Suppose that the process P finds out the coordinator crashed, P immediately holds an election. This algorithm has the following steps:
a) Step1:
When a process, P, notices that the coordinator crashed, it initiates an election algorithm.

(i) P sends an ELECTION message to all processes with higher numbers respect to it.

(ii) If no one responses within the time limit, P wins the election and becomes a coordinator.

b) Step2:
When a process receives an ELECTION message from one of the processes with a lower number response to it:

(i) The receiver sends an OK message back to the sender to indicate that it is alive and will take over.

(ii) The receiver holds an election, unless it is already holding one.

(iii) Finally, all processes give up except one that is the new coordinator.

(iv) The new coordinator announces its victory by sending a message to all processes telling them, it is the new coordinator.

c) Step3:
Immediately after the process with higher number compare to coordinator is up, bully algorithm is run.

Figure 1. explains the steps involved in the Bully election algorithm.

(i) Process 4 holds an election

(ii) Process 5 and 6 respond, telling 4 to stop

(iii) Now 5 and 6 each hold an election individually leading to two simultaneous elections.

(iv) Process 6 tells 5 to stop by sending OK to it.

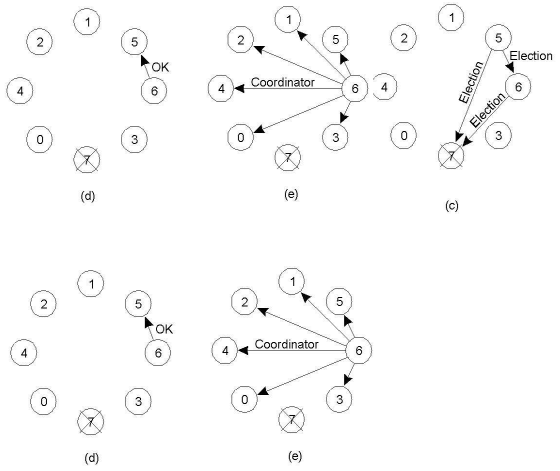


Figure 1. Bully Algorithm.

(v) Process 6 wins the election because no higher processes responded to it and it informs all the processes that it is the Coordinator from now on.

3.1. Advantages and limitations.

The advantages of Bully algorithm are that this algorithm is a distributed method with simple implementation. [9][10][11]. This method requires at most five stages, and the probability of detecting a crashed process during the execution of algorithm is lowered in contrast to other algorithms. Therefore other algorithms impose heavy traffic in the network in contrast to Bully algorithm[12]. Another advantage of this algorithm is that only the processes with higher priority number respect to the priority number of process that detects the crash coordinator will be involved in election, not all process are involved. . However the two major limitations of Bully algorithm are the number of stages to decide the new leader and the huge number of messages exchanged due to the broadcasting of election and OK messages[13].

4. Modified Bully Algorithm

Generally, in fault-tolerant distributed systems the leader node has to perform some specific controlling tasks and this node is well known to the other nodes. This node does not necessarily possess any extra processing feature to become elected, but having the highest process-ID. Election algorithms need a special mechanism to elect the leader. After crash failure of the leader node, it is urgently needed to reorganize the existing active nodes to call for an election and to elect a leader in order to continue the operation of the entire system.

4.1. Modified Bully Algorithm Details

1. Assumption

Besides having all the assumptions of the existing algorithm, we assume

(i) All processes hold an election flag, if this flag is true election cannot be initiated by any process.

(ii) All processes have a variable to store coordinator information.

a) Step1

Initially all election flag are set to false. When a process, P, notices that the coordinator crashed, it initiates an election algorithm.

(i) P sends an ELECTION message to all processes.

(ii) All processes set their election flag to true, so that none of the process can start

(iii) Coordinator variable reset to zero.

(iv) If no one responses, P wins the election and becomes a coordinator.

b) Step2

When a process receives an ELECTION message from one of the processes with lower numbered response to it:

(i) The receiver sends an OK message back to the sender to indicate that it is alive and will take over.

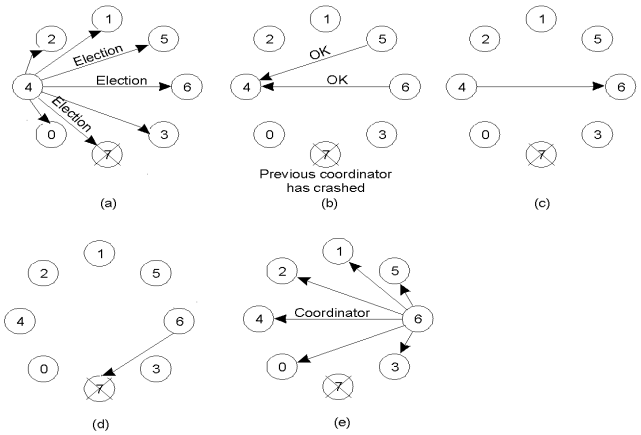


Figure 2. Modified Bully Algorithm.

(ii) The sender P extracts process ID of receiver and store it in coordinator variable. Only IDs greater than the stored ID can override the coordinator ID variable value.

(iii) Finally, all processes responded and higher process ID among them is stored in coordinator variable.

(iv) The sender P collects coordinator ID from variable and informed him (coordinator process ID) that he is coordinator.

(v) The elected coordinator process cross check with his higher processes, if any higher process is alive he will take over, else currently elected process will be coordinator.

(vi) The new coordinator announces its victory by sending a message to all processes telling them, it is the new coordinator.

(vii) All processes set the coordinator ID in coordinator variable and reset election flag to false.

c) Step3
Immediately after the process with higher number compare to coordinator is up, bully algorithm is run.

Figure 2. shows the steps involved in Modified Bully Election Algorithm.

(i) Process 4 holds an election

(ii) Process 5 and 6 respond, informing 4 about their presence in the system by OK message.

(iii) Processes 4 informs 6 to become coordinator.

(iv) Process 6 checks with process 7 if it is come back.

(v) Since no reply from process 7, process 6 wins and broadcasts the Coordinator Message to all the processes.

4.2. Advantages

Modified Bully algorithm is having all advantages of Bully algorithm. The additional advantages of modified Bully algorithm are that this algorithm is a very simple, having fail-safe mechanism, no parallel election, and reduced number of messages.

4.3. Limitations

How long the election initiator should wait to get response from all higher processes. If we keep a timer then the limitation could be the timeout value. Higher timeout will raise performance issue and lower timeout may miss responses from higher processes due to busy network traffic. However failsafe mechanism will be very helpful in this case.

5. Simulation and Comparision

5.0.1. Election Algorithm Simulator

We used a GUI based simulator for simulating election algorithm. This simulator was capable of creating process node, creating distributed process network, message passing and electing the coordinator. The simulator is enhanced with GUI capability, allowing users to save and load the distributed network, display messages, selecting start node (green) and recognizing coordinator node by changing the color (red).

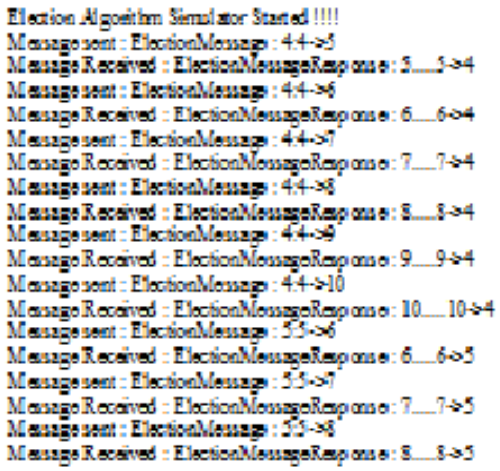


Figure 3. Simulation Log1

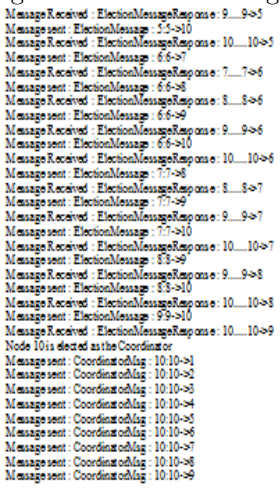
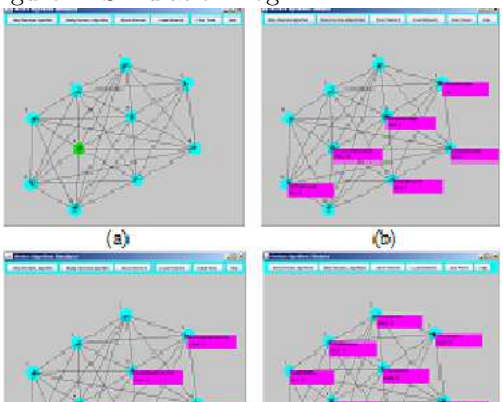


Figure 4. Simulation Log2



5.0.2. Bully Algorithm Simulation

(i) Simulation Setup:
Started with 10 process nodes participating in election.Node 4 started election.

(ii) Bully Simulation Logs

Figure 3 and Figure 4 shows the the Simulation logs of Bully Algorithm. Figure 5 represents the simulation of Bully algorithm in which Figure 5.(a) and Figure 5.(b) shows that, process ID 4 identifies the absence of the leader and initiates the election by sending the election message to its higher ups namely to processes 5, 6, , 10. All these processes in turn start their own election and concludes the election by the coordinator message where process ID 10 is the new leader. These activities are depicted in Figure 5.(c) and Figure 5.(d).

5.1. Modified Bully Algorithm Simulation

(i) Simulation Setup:
Started with 10 process nodes participating in election.Node 4 started election.

(ii) Modified Bully Simulation Logs

The Figure 6 shows the Simulation logs of the Modified Bully Algorithm.The simulation of Bully algorithm is represented Figure 7, where Figure 7. (a) and Figure 7. (b) show that, process ID 4 identifies the absence of the leader and initiates the election by sending the election message to its higher ups namely to processes 5, 6, , 10. Unlike the Bully algorithm, all these processes reply to the initiator process 4 instead of starting their own election. in Fig.ure 7. (c) and Figure 7. (d) show that process 4 decides the new coordinator (which is 10 in our simulation) and informs process 10 to take over and the election gets concluded by the broadcast of coordinator message where process ID 10 is the new leader.

5.2. Message Comparison

Table 1 shows the comparison for both algorithms. In this table we represented the message growth following by corresponding number of processes in the distributed network. Table shows that numbers of messages are increasing

Figure 6. Modified Bully Algorithm Simulation Logs

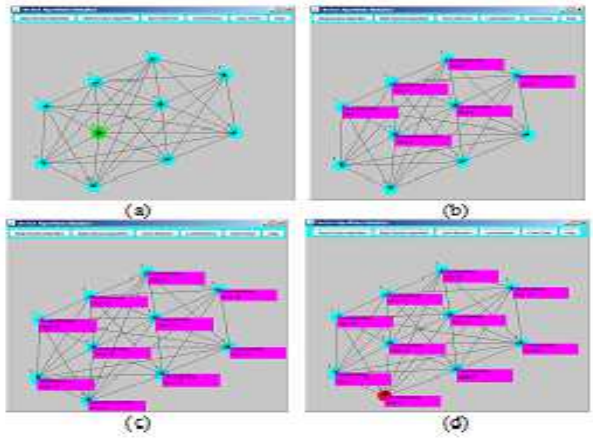


Figure 7. Modified Bully Algorithm Simulation

Table 1
Message Comparison of Bully and Modified Bully Algorithms

	Messages	
Processes	**Bully Algorithm**	**Modified Bully Algorithm**
5	24	13
10	99	28
15	224	43
20	399	58
25	624	73

drastically in the Bully algorithm compare to the modified Bully algorithm.

Figure 8. shows a comparison graph where both Bully and modified Bully are highlighted in different colors. Graph presents the comparison where number of nodes represented by horizontal axis and number of messages represented by vertical axis. Graph shows that Bully is having curve shape that describe $O(n^2)$ and modified Bully algorithm is having linear growth described by a straight line or O(n).

5.3. Simulation Result

Our simulation result shows that modified election algorithm is more efficient as it reduces the number of messages, also avoided any parallel election process. The comparative results are well explained by simulation logs, comparison graph and table.

5.4. Analytical Comparision

If only one process detects crashed coordinator
N: The number of processes
P: The priority number of processes that find out the crashed coordinator
T_m: The number of messages passing between processes when the Pth member detects the crashed Coordinator.
In Bully Modified algorithm the number of massages passing between processes for performing election is obtained from the following formula:

$$T_m = 2 * (N - P) + N \quad (1)$$

Which has Order O(n). In the worst case that is P = 1 (process with lowest priority number

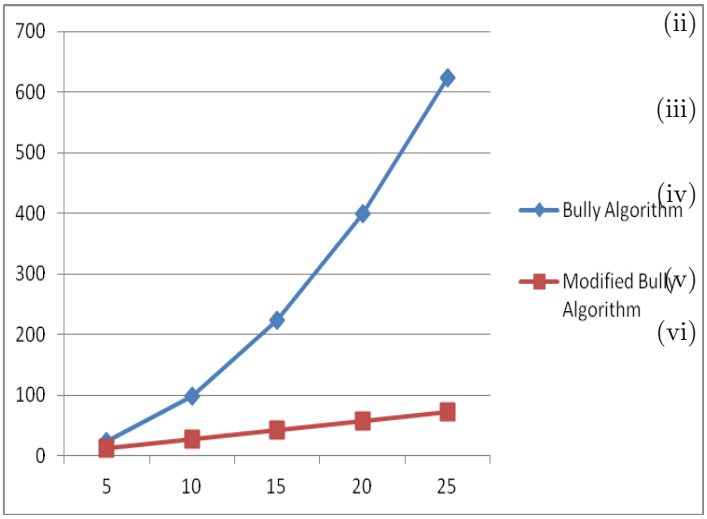


Figure 8. Number of messages used during the election

finds out crashed coordinator):

$$T_1 = 2 * (N - 1) + 1 = 3N - 1 \quad (2)$$

Whereas the number of massage passing between processes in the Bully algorithm for performing election is obtained from the following formula:

$$T_m = (N - P + 1)(N - P) + N - 1 \quad (3)$$

In the worst case that is P = 1 (process with lowest priority number detects crashed coordinator):

$$T_1 = N^2 - 1 \quad (4)$$

Which has Order O(n^2). Number of messages in proposed bully algorithm will be equal to 3n 1 that obviously means this modified algorithm is better than bully algorithm. Now assume that the set of processes in S = P1, P2, P3, ... Pn from processes find out the crashed coordinator concurrently (P1 is the lowest process).

In Bully algorithm, considering worst case and assuming lowest process start election, then:

(i) Total number of election message sent to set (S) of n processes (P_1, P_2, P_3, , , P_n) are (n - 1).

(ii) Total response message received by P_1 is (n - 1).

(iii) Now P_2 will send election message to n - 2 processes.

(iv) Total response message received by P_2 is (n - 2).

(v) Similarly for P_3, P_4 and P_n.

(vi) Finally Pn informing to every process by sending coordinator message is again (n - 1) message.

The number of message passing between processes for performing election is obtained from the following formula:

T_m = (n - 1) + (n - 2) + (n - 3) + + (n - n - 3) + (n - n - 2) + (n + n - 1) + (n - 1)

Simplifying the above formula, we get

$$T_m = n(n + 1)/2 \quad (5)$$

which is of O(n^2).

In our modified algorithm, considering worst case and assuming lowest process start election, then:

(i) Total number of election message sent to set (S) of n processes (P_1, P_2, P_3, , , P_n) are (n - 1).

(ii) Total response message received is (n - 1).

(iii) Informing to coordinator and coordinator to check with past coordinator involve two messages, and

(iv) Finally informing to every process by sending coordinator message is again (n - 1) message.

The number of message passing between processes for performing election is obtained from the following formula:

$$T_m = (n-1)+(n-1)+1+1+(n-1), or T_m = 3n-1 or 3n \quad (6)$$

which is of O(n).

6. Conclusions

In this paper, we discussed the drawbacks of Bully algorithm and then we presented an optimized method for the Bully algorithm called modified bully algorithm. Modified Bully algorithm shows improved performance than the Bully algorithm. The additional advantages of modified Bully algorithm are that this algorithm is a very simple, having fail-safe mechanism, no parallel election, and reduced number of messages.

Our analytical simulation shows that our algorithm is more efficient rather than the Bully algorithm, in both number of message passing and the number of stages, and when only one process runs the algorithm message passing complexity decreased from $O(^2)$ to O(n). In this analysis we consider the worst case in modified algorithm. Result of this analysis clearly shows that modified algorithm is better than bully algorithm with fewer message passing and the fewer stages.

REFERENCES

1. M.R Effat Parvar, "Improved Algorithms for Leader Election in Distributed Systems," *in 2nd International Conference on Computer Engineering and Technology (ICCET)*, 2010.
2. Sandipan Basu, "An Efficient Approach of Election Algori thm in Distributed Systems," *Indian Journal of Computer Science and Engineering (IJCSE)*, vol. 2, pp. 16–21, 2011.
3. Muhammad Mahbubur Rahman and Afroza Nahar, "Modified Bully Algorithm using Election Commission," *MASAUM Journal of Computing(MJC)*, vol. 1, no. 3, pp. 439–446, October 2009.
4. Chang-Young Kim and Sung-Hoon Bauk, "The Election Protocol for Reconfigurable Distributed Systems," in *International Conference on Wireless Networks (ICWN)*, pp. 295–301, 2006.
5. M. S. Kordafshari, M. Gholipour, M. Jahanshahi, A.T. Haghighat, "Modified bully election algorithm in distributed system," in *WSEAS Conferences*, 2005.
6. M. Sepehri and M. Goodarzi, "Leader election algorithm using heap Structur," *Proceedings of the 12th WSEAS international conference on Computers (ICCOMP08)*, 2008.
7. Sung-Hoon Park, "TA Stable Election Protocol based on an U nreliable Failure Detector in Distributed Systems," *Proceedings of IEEE Eighth International Conference on Information Technology: New Generations*, pp. 976–984, 2011.
8. M. Zargarnataj, "New Election Algorithm based on Assistant in Distributed Systems," *IEEE International Conference on Computer Systems and Applications*, pp. 324–331, 2007.
9. H.Garcia-Molina, "Elections in Distributed Computing System," *IEEE Transaction Computer*, vol. 310, pp. 48-59, 1982.
10. Chang Ben Ari, "Principles of Concurrent and Distributed Programming," *Pearson Education*, 2nd edition, 2006.
11. Andrew S Tanenbaum, "Distributed Systems Principles and Paradigms," *Beijing: Tsinghua University Press*, 2008.
12. G. Le Lan, "Distributed System Towards a Formal Approach," *Information Processing, The netherlands: North-Holland*, pp. 155–160, 1977.
13. Sung-Hoon-Park, "A Probablistically Correct Election Protocol in Asynchronous Distributed System," *APPT, LNCS*, vol. 310, pp. 177–183, 2003.

P Beaulah Soundarabai is presently an assistant professor in the department of Computer Science in Christ University, Bangalore. She obtained her Bachelors, Masters and degrees in Computer Applications from Madurai Kamaraj University, Madurai she also has complete M. Phil. in Computer Science. Presently she is pursuing Ph. D. in the field of Distributed Systems in Christ University.