CHAPTER 4
IMPROVED BULLY ELECTION ALGORITHM FOR
DISTRIBUTED SYSTEMS

4.1 INTRODUCTION
In Distributed Computing Systems, the collaborating processes in a system are often identical. One of the central problems of these coordinating processes is election of a leader. Given a network of processes, all of them should exactly decide on one process as their leader. It is usually required that all non-leader processes are informed or involved in the process of electing the leader. A leader election algorithm is one of the main actions of distributed systems, as it acts as a basis for more complex and high level algorithms and applications. An important research issue in this regard in distributed systems is to find the better algorithm which does the task perfectly without affecting the systems performance as this algorithm might often run automatically at any duration of time even if the system is completely overloaded. We cannot treat this algorithm as a overhead as it plays an important role in the synchronization of the system which in turn maintains the coherency of the system even during partial failures.

4.2 BULLY ALGORITHM
Bully algorithm was first presented by Garcia Molina [13] 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.
4.2.1 Aim of Bully Algorithm
The aim of an election Algorithm execution is selecting one process as leader 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 without other processes contradicting this decision.

4.2.2 Assumptions of Bully Algorithm
- Each process has a unique and not null number identifier to distinguish from each other and each process knows other process’ identification number.
- Processes do not know about which are the processes currently up and down.
- The entire system is synchronous. Timeouts are used for deciding process failure and the network is perfect that when messages are sent, they would be definitely received and the order of sending would be the order of receiving also. There is no failure or damage of messages in the system.
- Processes can crash at any point of time.
- Message delivery between processes is reliable and time bound.

4.2.3 Methodology of Bully Algorithm
When the failure of the existing coordinator is encountered by any process in the group, the distributed system invokes the bully election algorithm to dynamically decide the next eligible coordinator process. Suppose that process, say P, finds out the coordinator’s crash, according to bully algorithm, P immediately holds an election. The following steps explain the bully algorithm.

When P notices that the coordinator has crashed recently, it initiates an election process by

- Sending an ELECTION message to all processes with higher identification number (ID) than of itself.
There are three possibilities.

- If no process responds within the time limit, $P$ wins the election and becomes a coordinator.

- If there are few processes with higher ID available in the system, then each of them reply the initiator by an OK message to indicate that it is alive and will take over the election of leader activity.

- Each responder process holds an election, by sending the ELECTION message to its higher ups by following the first two steps.

- This election continues till all processes give up except one and that one process will be the one with the highest ID available in the system and it becomes the new coordinator.

- The new coordinator announces its victory by broadcasting a COORDINATOR message to all, with which the election algorithm comes to an end. When the crashed leader process comes back to action, it immediately broadcasts another COORDINATOR message by which it becomes the leader once again without holding any election and so this algorithm is named as Bully Election Algorithm.

Figure 4.1 [97] explains the various stages of Bully election algorithm. Here, there are a group of 7 processes and process 4 identifies the failure of the leader process 7 by initiating the election.

- Process 4 sends ELECTION message to its higher ups. (processes 5, 6 and 7)

- Process 5 and 6 respond by sending OK messages, telling process 4 that they would take over the execution of electing the leader.

- Now processes 5 and 6 hold an election individually leading to two simultaneous elections.
• The reply of OK message from process 6 to process 5 tells that it would continue the process of electing the leader.
• Process 6 waits for a clock slice time for a reply from process 7. Since process 7 does not send the reply due to its failure, after the timeout, process 7 wins the election and informs all the processes about it by sending the COORDINATOR message by which the election algorithm terminates.

![Figure 4.1: Stages of Bully Algorithm](image)

4.2.4 Advantages and limitations

The advantages of Bully algorithm are that it is distributed in nature rather than following a centralized approach and its simple implementation is another benefit. [9][10][11]

This method requires minimum of 4 stages if there the initiator is the coordinator or takes many recursive stages to find the coordinator process. The
probability of detecting a failure process during the execution of algorithm is lowered in contrast to other election algorithms. Therefore other algorithms impose heavy traffic in the network than this Bully algorithm. [12] Another advantage of this algorithm is that only the processes with higher ID respect to process ID that detects the crash of the coordinator will be involved in election, and the lower ones are not disturbed with this election process.

However the two major limitations of Bully algorithm are the number of stages to decide the new leader leading to lots of time wastage and huge number of messages exchanged due to the sending and receiving of ELECTION, OK, COORDINAGOR messages. [13] The elected coordinator process might crash and initiating another election would take such a lot of time that some process might timeout waiting for the coordinator.

4.3 MODIFIED BULLY ALGORITHM

In this chapter, we propose our modified bully algorithm, to eliminate the limitations faced by the bully algorithm. It is essential to invoke election algorithm by an active participant to elect a leader so as to continue the normal function of the whole system. [98]

4.3.1 Assumptions of Modified Bully Algorithm

Besides having all the assumptions of the existing algorithm, we assume the following two factors.

- All processes hold an election flag, if this flag is true then election cannot be initiated by any process.
- All processes have a variable to store coordinator information.

Following are the steps for our proposed algorithm:

*Step 1*
Initially all election flag are set to false. When a process, say P, notices that the coordinator has recently crashed, it initiates an election algorithm by setting its election flag to true.

- P broadcasts an ELECTION message to all processes.
- All processes set their election flag to true, so that none of the process can start parallel election until current election come to the end.
- Coordinator variable is reset to zero in all processes.
- If no node responses, P wins the election and becomes a coordinator.

**Step2**

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

- The receiver sends an OK message back to the sender to indicate that it is alive and it is participating in the election.
- The sender process waits and collects the OK replies from the alive processes for a period of one clock cycle time.
- Finally, all active processes respond to the sender and the sender analyzes all the process IDs and selects the highest process ID among them; Stores that in the coordinator variable.
- The sender P informs the selected coordinator process to be the leader by sending a message.
- The elected coordinator process cross checks with its higher processes for their availability by another election message. If any such higher process is alive, then that will take over, else currently elected process will be the coordinator.
- If the selected coordinator is down during or after receiving this message, the sender process comes to know about it as there is no broadcast from the new coordinator; it informs the second highest process to be the
leader. This gets repeated until there is a broadcast of a coordinator message.

- The new coordinator announces its victory by sending a message to all processes telling them, it is the new coordinator.
- All the alive processes set their coordinator variable with the new coordinator’s ID and set their election variable to false again with which the election comes to an end.

**Step 3**

When the old coordinator process comes alive after its crash, it immediately sends a coordinator message to take over the leadership by bullying all the processes and so this algorithm is called modified bully algorithm.

Figure 4.2 shows the stages involved in modified Bully election algorithm.

- Process 4 initiates an election after discovering the failure of the coordinator process 7. It broadcasts the ELECTION message to processes in the group.
- Process 5 and 6 respond, informing 4 about their presence in the system by OK message.
- Process 4 inform 6 to become coordinator.
- Process 6 checks if process 7 is alive by sending another election message.
- Since no reply from process 7, process 6 wins and broadcasts the Coordinator Message to all the processes.

### 4.3.2 Advantages of Modified Bully Algorithm

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 by verifying the failure of the existing leader before taking over the leadership. Because we have the election variable, we tried avoiding the parallel elections unlike the case in the bully algorithm.
Since we have removed the duplicate or parallel elections, their respective coordinator message broadcasts are also avoided which leads to reduced number of messages. As processes may fail in any time, the initiator process keeps a watch on the entire election process to check whether there is any coordinator message broadcast within the time slice. If not it keeps on calling for the next highest process to takeover repeatedly till the election is successful.

![Diagram](image)

**Figure 4.2:** Stages of Modified Bully Algorithm

### 4.3.3 Limitations of Modified Bully Algorithm

If the initiator process itself fails, then there is no other process to monitor the fail-safe mechanism of the election. However, the probability of the initiator and the new coordinator process to be down at the very same time might be minimum, there would be very less failure of our election. Deciding of the time of waiting of the election initiator to get response from all higher processes is very crucial. The limitation here 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.

### 4.4 SIMULATION AND COMPARISION
4.4.1 Election Algorithm Simulator

We have used a GUI based java simulator for simulating the two election algorithms. 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).

4.4.2 Bully Algorithm Simulation

Simulation Setup

- Started with 10 process nodes participating in election.
- Node 4 started election.

Bully Simulation Logs

Election Algorithm Simulator Started !!!!
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
Message sent : ElectionMessage : 5:5->6
Message Received : ElectionMessageResponse : 6.....6->5
Message sent : ElectionMessage : 5:5->7
Message Received : ElectionMessageResponse : 7.....7->5
Message sent : ElectionMessage : 5:5->8
Message Received : ElectionMessageResponse : 8.....8->5
Message sent : ElectionMessage : 5:5->9
Message Received : ElectionMessageResponse : 9.....9->5
Message sent : ElectionMessage : 5:5->10
Message Received : ElectionMessageResponse : 10.....10->5
Message sent : ElectionMessage : 6:6->7
Message Received : ElectionMessageResponse : 7.....7->6
Message sent : ElectionMessage : 6:6->8
Message Received : ElectionMessageResponse : 8.....8->6
Message Received : ElectionMessageResponse : 9.....9->6
Message sent : ElectionMessage : 6:6->10
Message Received : ElectionMessageResponse : 10.....10->6
Message sent : ElectionMessage : 7:7->8
Message Received : ElectionMessageResponse : 8.....8->7
Message sent : ElectionMessage : 7:7->9
Message Received : ElectionMessageResponse : 9.....9->7
Message sent : ElectionMessage : 7:7->10
Message Received : ElectionMessageResponse : 10.....10->7
Message sent : ElectionMessage : 8:8->9
Message Received : ElectionMessageResponse : 9.....9->8
Message sent : ElectionMessage : 8:8->10
Message Received : ElectionMessageResponse : 10.....10->8
Message sent : ElectionMessage : 9:9->10
Message Received : ElectionMessageResponse : 10.....10->9

Node 10 is elected as the Coordinator
Figure 4.3 represents the simulation of Bully algorithm in which Figure 4.3 (a) and Figure 4.3 (b) shows that, process 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
conclude the election by the coordinator message where process ID 10 is the new leader. These activities are depicted in Figure 4.3 (c) and Figure 4.3 (d).

4.4.3 Modified Bully Algorithm Simulation

Simulation Setup

- Started with 10 process nodes participating in election.
- Node 4 started election.

Modified Bully Simulation Logs

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 coordiID : 10
Message sent : ElectedAsCoordinator : 4:4->10
Largest coordiID (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
msg_count : 25

The simulation of Bully algorithm is represented Figure 4.4. Where Figure 4.4 (a) and Figure 4.4 (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. Unlike the Bully algorithm, all these processes reply to the initiator process 4 instead of starting their own election. in Figure 4.4(c) and Figure 4.4(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.
4.5 PERFORMANCE ANALYSIS

Table 4.1 shows the number of messages exchanged during the execution of both algorithms.

The original bully algorithm has used lots of messages to decide the new leader as it does not have a control on the unnecessary parallel elections. Through this message complexity, it is very clear that the numbers of messages are increasing drastically in the Bully algorithm compared to the modified Bully algorithm due to the inclusion of data structure election variable in each process and through which the system tries to limit the number elections happening in the system.

Table 4.1: Message Comparison of Bully and Modified Bully algorithms

<table>
<thead>
<tr>
<th>No. of Processes</th>
<th>No. of Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bully Algorithm</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>99</td>
</tr>
<tr>
<td>15</td>
<td>224</td>
</tr>
<tr>
<td>20</td>
<td>399</td>
</tr>
<tr>
<td>25</td>
<td>624</td>
</tr>
</tbody>
</table>

Figure 4.5 shows a comparison graph where both Bully and modified Bully are highlighted in different colors. It presents the comparison of the number of
messages exchanged in both the algorithms, where number of nodes represented by horizontal axis and number of messages represented by vertical axis. This graph shows that bully algorithm is having curve shape that describes the message complexity as $O(n^2)$ and modified Bully algorithm is having linear growth described by an almost straight line which of $O(n)$.

![Graph showing message complexity vs number of nodes and messages exchanged in both algorithms](image)

Figure 4.5: Number of messages used during the election

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

### 4.5.1 Analytical Comparision

The message complexity of the existing bully algorithm is $O(n^2)$. [10]

Let, $N$ represent the number of processes, $P$ the priority number of processes that find out the crashed coordinator and $T_m$ represent the number of messages passing between processes when the process $P$ detects the crashed coordinator.

In modified bully algorithm the number of messages passing between processes for performing election is obtained from the following equation:

$$T_m = 2 \times (N - P) + N$$  \hspace{1cm} (1)
Which has Order O(n). In the worst case that is P = 1 (process with lowest priority number finds out crashed coordinator):

\[ T_1 = 2 \times (N - 1) + 1 = 3N - 1 \]  

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

\[ T_m = (N - P + 1) (N - P) + N - 1 \]  

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

\[ T_1 = N^2 - 1 \]  

Which has Order O(n²). 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 (P1 is the lowest process).

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

- Total number of election message sent to set (S) of n processes (\{P_1, P_2, P_3, ..., P_n\}) are (n - 1).
- Total response message received by P_1 is (n - 1).
- Now P_2 will send election message to n – 2 processes.
- Total response message received by P_2 is (n - 2).
- Similarly for P_3, P_4... and P_n.
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:

\[ T_m = (n - 1) + (n - 2) + (n - 3) + \ldots + (n - n - 3) + (n - n - 2) + (n + n - 1) + (n - 1) \]

Simplifying the above formula, we get

\[ T_m = n (n + 1) / 2 \] (5)

which is of O(n^2).

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

- Total number of election message sent to set (S) of n processes ({P_1, P_2, P_3, \ldots, P_n}) are (n - 1).
- Total response message received is (n - 1).
- Informing to coordinator and coordinator to check with past coordinator involve two messages, and
- 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:

\[ T_m = (n - 1) + (n - 1) + 1 + 1 + (n - 1), \text{ or} \]

\[ T_m = 3n - 1 \] (6)

which is of O(n).

4.6 SUMMARY

In this chapter, we have reviewed the existing Bully algorithm election algorithm and presented its limitations. It has a problem of high number of message passing and it is not fault tolerant. As its message complexity has order
O(n^2), it increases the network traffic drastically and brings down the overall performance of the system during the time of election. It also has many stages to decide the next leader which would waste lots of time for the processes to resume to their normal execution. We have proposed and implemented our 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, reducing the number of stages and fault tolerant during the election of the next leader. Our algorithm is has proved better performance over the existing algorithm and thus will definitely increase the overall performance of the system.