A New Approach For Election Algorithm in Distributed Systems

M. Gholipour\textsuperscript{1} M. S. Kordafshari\textsuperscript{1} M. Jahanshahi\textsuperscript{2} A. M. Rahmani\textsuperscript{3}
gholipour@qazviniau.ac.ir kordafshari@qazviniau.ac.ir mjahanshahi@iauctb.ac.ir rahmani@sr.iau.ac.ir

1 Computer Engineering Department, Islamic Azad University, Qazvin Branch, Qazvin, Iran.
2 Computer Engineering Department, Islamic Azad University, Central Tehran Branch, Tehran, Iran.
3 Computer Engineering Department, Islamic Azad University, Science and Research Branch, Tehran, Iran.

Abstract - Leader election is an important problem in distributed computing systems. In this problem, when the leader is crashed, other nodes must elect another leader. Garcia-Molina’s Bully Algorithm is a classic solution to cope with this problem. In this paper we introduce a new method based on electing a leader and alternatives. If the leader crash, the alternative takes, care of the leader's responsibilities. Our results revealed that often, after a leader crash, leader alternative elect as a leader and continue to work. increases. We show that our proposed algorithm is more efficient than the Garcia Molina’s one in term of time, message complexity and in the fewer stages.

Keyword - Election algorithm

I. Introduction
Leader (coordinator) election is an influential problem in distributed computing systems since performance of all nodes in system depending on leader. Depending on a network topology, many algorithms have been presented for electing leader in distributed systems [1, 2, 3, 4, 5]. Gerard Le Lann [3] introduced first paper about election in 1977. AUDITOR [6] is one of the first protocols to coordinate the leader election problem. In this protocol, each node contains an "auditor" and there is an ordered ranking of auditors. The highest-ranking auditor select 'audit coordinator' which responsible for detection of failures. A disadvantage of this approach is that, if multiple node failures (or partitioning), it may take many sequential executions of the promotion protocol before a candidate is successful in reaching coordinator rank [5]. Bully algorithm is one of the most applicable elections algorithms that was presented by Garcia Molina in 1982[1].

In this paper, we initially explain bully algorithm and discuss the drawbacks of it and present new algorithm considering number of message passing. We show that our algorithm is more efficient than Garcia Molina’s Bully algorithm.

In Garcia Molina’s Bully algorithm, when a node (e. g. node N) realizes that the leader is crashed, it sends an ELECTION message to all nodes with higher numbers. If no one responds, node N wins the election and becomes leader. If one of the higher processes answers, it takes over. Node N’s job is done. In case of receiving a message, the receiver sends an OK message back to the sender to indicate that he is alive and will take over. The receiver then holds an election, unless it is already holding one. Eventually, all nodes give up but one and that one is the new leader [1].

The main drawback of Bully algorithm is the high number of message passing. In this method the message passing has order $O(n^2)$ that increases heavy traffic on the network. The advantage of Bully algorithm is its simplicity.

The rest of the paper is organized as follows: Section 2 describes improved method for solving Bully algorithm drawbacks. In Section 3, we analyze the algorithm and compare it with bully algorithm. Finally, Section 4 is conclusion.

II. Modified Election Algorithm
As it has been mentioned in Section 1 in Bully algorithm number of messages that should be exchanged between processes is high. Therefore this method imposes heavy traffic on the network. To cope with this drawback we will present an optimized method by modifying it such that our method intensively decreases the number of messages that should be exchanged between processes. Furthermore the number of stages is decreased from at most five stages to at most four stages.

In our algorithm, we use several ordered nodes composing coordination group \{coordinator, alternative1, alternative2,…alternativeK\}. These nodes are used for preventing global election between all nodes.

Initially using the modified election algorithm described in Section b-2, a process with greatest number is selected as coordinator. Similarly other (k-1) processes of coordination group which have the next priority numbers are selected as alternative1, alternative2,... alternativeK.
Then their process id(s) are sent to all processes. When process P notices that the coordinator is crashed, sends crash-leader message to alternative1 informing coordinator is crashed.

A) If alternative1 is alive:
Alternative1 sends ok message to process p informing it from receiving the message. Then it sends a message to coordinator in order to be sure from its crashing. In case of receiving ok from coordinator, it obtains that the process p made a mistake and the algorithm is finished. Otherwise alternative1 is selected as coordinator and broadcasts leader message and in this message introduce the process p as its substitution.

B) If alternative1 is not alive:
Process p waits for t time step for receiving ok from alternative1 (t=2d; d is average of propagation delay). If p does not receive any message, sends crash-leader to other alternatives in order. If all alternatives are crashed, then process p runs the modify election algorithm as follows:

**Step1-** When process P notices that all members of the coordinator group have crashed, it initiates an election algorithm.

**Step2-** When the process P finds out the coordinator is crashed, sends ELECTION message to all other processes with higher priority number.

**Step3-** Each process that receives ELECTION messages (with higher process than P) sends OK message with its unique priority number to process P.

**Step4-a)** If no process responses to process P,
In this case p is selected as coordinator. Then it sends a message to processes with lower number in order to select other members of the coordination group and waits 2d time steps. After receiving all process ids, it selects k-1 higher process as members of coordination group and it will broadcast one COORDINATOR message to all processes, declaring coordinator group member.

**Step4-b)** If some process response to process P:
The process P will select the coordinator group *i.e. process with the highest priority number as coordinator and k higher processes as alternative1 and alternative 2,...alternative k*, then it sends to the new coordinator the GRANT message.
Step5- at this stage the coordinator process will broadcast a message to all other processes that contains its priority number and alternative 1 and alternative 2,…alternative k numbers.(coordinator group numbers)

New algorithm not only has all advantages of Bully algorithm also it doesn’t has the drawback of Bully algorithm (high number of message passing). Furthermore maximum number of stages is decreased from five stages to four stages.

It is clear that if process P crashes after sending ELECTION message to higher processes, or crashes after receiving the priority numbers from process with higher priority number, higher process wait at most 3D time for coordinator broadcast. (D is average propagation delay). If it will not receive, this process runs the modified algorithm. If a process with higher priority number crashes after sending its priority number to P, process P sends GRANT message to it meaning that it is the highest process and P waits for broadcasting coordinator message. If after D time, process P doesn’t receives the COORDINATOR message, it repeats the algorithm again. Therefore we can use this algorithm as an efficient and safe method to selecting the coordinator.

If multiple processes simultaneously realize that the coordinator is crashed

When more than one process or all processes find out the coordinator has crashed simultaneously, there is no problem if at least one candidate is alive. But if coordination group is crashed all of them run in parallel election algorithm, therefore heavy traffic imposed to the network. For solving this problem in modified election algorithm we act as follow (Figure3).

Step1-When process P realizes that the all members of the coordinator group have crashed, it initiates modified election algorithm presented in Section 2-b.

Step2-When process P ' (P ' may be P) receives the ELECTION message from process or processes with lower priority number compare to itself, it waits a short time that can be specified perfectly and then answers to the process with lowest priority number only. In this situation if P=P' (This process initiates the algorithm and also received the ELECTION message from other processes), then stops the algorithm.

Step3-After process P' answered to P, if P ' receives an ELECTION message from process R(R<P<P'), P' answers to process R by sending its priority number and sends STOP message to process P.(Figure 4)
Figure 3. The modified election algorithm. (a) process 4 and 2 find out the crashed coordinator simultaneously and therefore each of which send ELECTION messages separately. (b) other processes that receive more than one ELECTION message should only send their own priority number to process with lowest id number (in this case 2). (c,d) process 3 stop the algorithm because receiving the ELECTION message from process 2. Process 2 continue the algorithm.

Figure 4. An example of use Stop message in modify election algorithm.
Step 4 - when a process receives the STOP message stops the algorithm immediately.
Step 5 - if process p neither receives any response from other process(es), nor does it receive any ELECTION message form processes with lower Priority number, then in this case it can inform other processes containing iT(P) as COORDINATOR.

III. Mathematical analysis
If only one process detects crashed coordinator: 
n : The number of processes
r : The priority number of processes that find out the crashed coordinator
N(r): The number of messages passing between processes when the r-th member detects the crashed Coordinator.

K and p are the number of candidates and probability of each node crashing respectively.

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

\[ N_r = (1 + 1 + 1_p) + P^*(1 + 1 + 1_p) + P^*(2 + 1 + 1_p) + P^*(3 + 1 + 1_p) + \ldots + P^{k-1}*(k - 1 + 1 + 1_p) + P^*(2*(n - r_j) + l_b + k) \]

Which has the following order:

\[ O(\sum_{i=0}^{k-1} P^*(1_b + 2 + i) + P^*(2*(n - r) + 1_b + k)) \]

In the worst case that is \( r = 1 \), i.e. process with lowest priority number finds out crashed coordinator. (In case of all members of coordinator group are crashed.)

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

\[ N_r = (n - r + 1)(n - r) + 1_b \]

That has Order \( O(n^2) \).

3.2 Analytical comparison of two algorithms if set of \( S = \{ r_1, r_2, \ldots, r_n \} \) run the algorithm simultaneously.

Now assume that the set of processes in \( S = \{ r_1, r_2, \ldots, r_n \} \) from processes find out the crashed coordinator concurrently (\( r_i \) is lowest process):

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

\[ T = (n - r_j + 1)(n - r_j) + 1_b \]

In our modified algorithm the number of message passing between processes for performing election is obtained from the following formula:

\[ N_r = (1 + 1 + 1_p) + P^*(1 + 1 + 1 + 1_p) + P^*(2 + 1 + 1 + 1_p) + P^*(3 + 1 + 1 + 1_p) + \ldots + P^{k-1}*(k - 1 + 1 + 1 + 1_p) + P^*(2*(n - r_j) + 1_b + k) \]

In bully algorithm the number of message passing is based on the process with lowest priority number. That means there isn’t any difference between states that only process \( r_j \) detects the crashed coordinator and state that in which the set of \( S = \{ r_1, r_2, \ldots, r_n \} \) detects crashed coordinator. But in modified algorithm set of \( S = \{ r_1, r_2, \ldots, r_n \} \) is also important. If the priority numbers of the processes that detects the crashed coordinator is higher, the number of message passing will be decreased considerably.

IV. Conclusion
Many algorithms related to the leader election are proposed. In this paper, we propose a new algorithm for leader election in ad hoc networks. The main idea of proposed algorithm is that it uses candidate nodes. The complexity of proposed algorithm is \( O(n) \) and computer simulation shows its number of messages is smaller than other important algorithms.

References