AN IMPROVED LEADER ELECTION ALGORITHM FOR DISTRIBUTED SYSTEMS

P BeaulahSoundarabai▫, Thriveni J▫▫, K R Venugopal▫▫, L M Patnaik▫▫▫

*DepartmentofComputerScience,ChristUniversity,Bangalore
beaulah.s@christuniversity.in
**DepartmentofComputerScienceandEngineering
UniversityVisvesvarayaCollegeofEngineering,BangaloreUniversity,Bangalore
▫▫▫ Honarary Professor, IISc., Bangalore

ABSTRACT

Leader Election Algorithm , not only in distributed systems but in any communication network, is an essential matter for discussion. Tremendous amount of work are happening in the research community on this Election, because many network protocols are in need of a coordinator process for the smooth running of the system. These socalled Leader or Coordinator processes are responsible for the synchronization of the system. If there is no synchronization, then the entire system would become inconsistent which intern makes the system to lose its reliability. Since all the processes need to interact with the leader process, they all must agree upon who the present leader is. Furthermore, if the leader process crashes, the new leader process should take the charge as early as possible. New leader is one among the currently running processes with the highest process id. In this paper we have presented a modified version of ring algorithm. Our work involves substantial modifications of the existing ring election algorithm and the comparison of message complexity with the original algorithm. Simulation results show that our algorithmminimizes the number of messages being exchanged in electing the coordinator..

KEYWORDS

Election, Coordinator,Message Complexity, Ring Algorithm,Distributed System.

1. INTRODUCTION

Leader Election is a vital and fundamental problem in distributed systems and in any communication network. Distributed Systems is a collection of heterogeneous systems which interact with each other through messages. The main objective of Distributed System is , though there are heterogeneous systems in the network, it creates a single system image or uniprocessor image to the user, through various transparency metrics. The communication between the processes is achieved by exchanging messages. The software of the Distributed System is tightly coupled and the processes of the system coordinate with each other. They have lots of resources in common and so mutual exclusion algorithms are used to take care of the critical regions. While they wait for the common resources, they might end up in a deadlock. Deadlock detection and prevention algorithms should keep an eye on the resources and if there is a deadlock wound wait or waitdie algorithms are used to kill the eldest or the youngest process to remove the deadlock.

The replicated data management, group communication, atomic commit protocols, etc needs the process coordination. All the above stated protocols need a particular process among the group, to be the leader to have the control over the situation. In general, the process with the highest processid is the coordinator or the leader. Any process, which satisfies the rule can become the

DOI : 10.5121/ijngn.2013.5102

leader and the only issue in distributed system is that at any point of time there should be a unique process available as the leader to do the coordination and all the other processes should agree up on the present leader, without any confusion.

Motivation: If the processes of the distributed system never fail, then the leader process can be decided at the time of the process group gets generated. But, there are systems where processes keep coming and leaving in the group and processes do crash. Specially, in Wireless networking like Wireless LAN, Satellite oriented Services cellular phones, the mobile systems are subject to loss of messages or the data and the mobile host can crash or can be down for some time. Electing a leader process is a basic operation which happens in the system very often. Many researchers have contributed a lot of paradigms to simply this task. Different kinds of leader election algorithms have been proposed and most of them are widely in usage.

Contribution: In this paper, we have proposed a modified ring algorithm with minimized number of messages .The main objective of the algorithm is the fault tolerance. In mobile ad hoc network, we always face node failure or the process crash. Even during the time of leader process crash, there should be a new leader process available, without much wasting of time and the number of messages which are exchanged.

Organization: The remainder of this paper is organized as follows: Section II reviews the related work; Background of Election and Ring Election Algorithms is available in Section III. The Efficient Ring Election Algorithm is proposed in Section IV; Section V deals with the Performance Evaluation; Conclusions are presented in Section VI.

2. RELATED WORK

Sung Hoon Park [1] has proposed a concept called Failure Detector which works as an independent module with a 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. And as the Failure Detector is the centralized component, it has the problem of single source failure and also creates bottleneck situation to access the module.

Sandipan Basu [2] has addressed the issue of bully algorithm and proposed his 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 neighbour. 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 tern 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.

ChangYoung 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] discussed the drawback of synchronous bully algorithm and modified it with an optimal message algorithm. 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

Cuibo Yu et al., [7] have proposed a different idea on SN(Super Node) election algorithm based upon district partition which divided the whole overlay into k small districts and using distributed and parallel computing in these small units was brought forward. This algorithm would decrease the message complexity to O(n2 k) and increase the electing speed. At the same time, the elected SN would be evenly distributed in the whole overlay.

Mehdi EffatParvar et al., [8] have proposed a new approach with fault tolerant mechanisms base on heap for coordinator finding in wireless environment. The authors created the new algorithm called Heap tree algorithm , by modifying the bully and the ring election algorithms. They have also compared the algorithm's running time and message complexity with the existing algorithms.

3. BACKGROUND

A. Election Algorithm

Election algorithm is a special purpose algorithm, which is run for selecting the coordinator process among N number of processes. These coordinator or leader process plays an important role in the distributed system to maintain the consistency through synchronization. For example, in a system of client server, the mutual exclusion algorithm is preserved by the server process Ps, which is chosen from among the processes Pi where i=1,2,...,N that are the group of processes which would use the critical region. Election Algorithm is needed in this situation to choose the server process among the existing processes. Eventually all the processes must agree upon the leader process. If the coordinator process fails due to various reasons, then immediately the election should happen to choose a new leader process to take up the job of the failed leader.

Any process can initiate the election algorithm whenever it encounters that the failure of leader process. There can be situations that all N processes could call N concurrent elections. At any time, process Pi is one among the following two states, when the election happens: Participant refers to the process is directly or indirectly involved in election algorithm, Nonparticipant refers to the process in not engaged with the election algorithm currently. The goal of Election Algorithm is to choose and declare one and only process as the leader even if all processes participate in the election. And at the end of the election, all the processes should agree upon the new leader process without any confusion. Without loss of generality, the elected process should be the process with the largest process identifier. This may be any number representing the order / birth/ priority/ energy of the process. Each process has a variable called LEAD, which contains the process id of the current leader. When the process participates in the election, it sets this lead to NULL.

Any Election Algorithm should satisfy the following two properties [9].

1) **Saf ety :** Any process P, has LEAD = NULL if it is participating in the election, or its LEAD =P, where P is the highest PID and it is alive at present.
2) **Likeness :** All the processes should agree on the chosen leader P after the election. That is, LEAD = PID Pi where i=1,2,...,N.

The Bully Election Algorithm [10] of Garcia Molina in 1982, elects the leader process uniquely which satisfies the safety and likeness requirements. Depending on a network topology, many algorithms have been presented for electing leader in distributed systems.

Depending on a network topology, many algorithms have been presented for electing leader in distributed systems. The numbers may be allocated in simply numerical order of the Ethernet address or some other numbers such as priority, the mere process id, etc. The Ring Election Algorithm [11] is based on the ring topology with the processes ordered logically and each process knows its successor in an unidirectional way, either clockwise or anticlockwise.

When any process notices that the coordinator process has crashed, it creates an EL MSG by inserting its own PID and sends the message to its successor. If the successor is also down, the message would skip that process and goes to the next process of the successor or to the next etc., till it reaches a process which is not dead, along the ring network. When the EL MSG is received by any process, it adds its PID to the list in the message. Like this, all the available processes in the ring would insert their respective PID in the list. Finally, the EL MSG comes back to the process which initiated the message and the process too would recognize that it only had intiated that message, by identifying its own PID in the list.gh

The Election initiator process analyses and finds the highest PID among the available processes, converts the ELMSG into COMSG and removes all the PIDs from the list but the highest PID. This COMSG message is circulated along the ring for one circulation to inform the running processes about who the new COORDINATOR is. When this message is circulated once, it is discarded and the Election Algorithm ends here.

When the message comes back to the process that started it:

(i) TheprocessseesitsIDinthelist.
(ii) It checks all the PIDs and decides the coordinator (the process with the hightest ID).
(iii) It changes the message type COMSG and enters the LEAD process in the message.
(iv) COMSGiscirculatedagain.
(v) When it comes back to the process that started it, and it gets discarded there.

Limitation of Ring Election Algorithm: Multiple election messages may happen in parallel when more than one process detects the failure of the coordinator process. This creates a lot of overhead of creating and servicing each and every election message. This causes heavy traffic and sometimes congestion in the network. In the best case, when a single process detects the crashed leader, Ni is obtained with O(n) as follows:

$$Ni = ne + nc. \quad (1)$$

Where ne refers to the number of EL MSG and nc refers to the number of CO MSG. In the average and worst case, when all the N processes start the election message, , Ni is obtained with O(n2) from the following equation.

$$Ni = n(ne + nc). \quad (2)$$

This message complexity will drastically bring down the entire systems performance, as all the processes spend quite a lot of amount of time in servicing these messages or processing these messages. In order to bring up the performance even during election, we need to have exactly only one complete election happening instead of simultaneous redundant elec-tions. All the other redundant election messages need to be killed. For solving this problem an improved election algorithm is proposed in the following section.

4. MESSAGE EFFICIENT RING ELECTION ALGORITHM (MEREA)

In the previous section we have seen that in the average and the worst case scenarios, the number of messages that are exchanged between processes is high in the original Ring Algorithm. Therefore it imposes heavy traffic on the network . The proposed algorithm tries to intensively decrease the redundant Election messages.

Assumptions

1. All the processes in the distributed group should have their clocks synchronized to each other. We have logical clock and physical clock synchronization algorithms namely Lamports algorithm for the logical clock synchronization and Cristians and Bekeley algorithms for the real time clocks or the logical clocks.

2. The Network is perfect. (i.e.) when any message is sent, it wont be lost/modified . It would reach the destination. If the destination process is alive, it can see the message which was sent to it. Here too, we have reliable primitives to keep the network perfect.

Table 1. MEREA Algorithm

```
begin
Step1:call Build EL_MSG
Step2: for k: = 1 to n - 1
        call update EL_MSG
        Send the EL_MSG to SUCC_i
       endfor
Step3: Build COMSG
Step4: Circulate COMSG
end
```

The proposed Message Efficient Ring Election Algorithm is given in Table 1 through 4 . When process P realizes that the coordinator has crashed, it initiates the Message Efficient Election Algorithm by creating the EL MSG. It inserts its ID, and the time of creation of the and circulates the message in the ring by throwing it to its immediate neighbour.

According to our assumption, all the processes in the group have their clocks synchronized, and so all the alive processes have the same time in their clock.When any process Q receives the EL MSG n the ring, it reads its log, to check whether it has created any EL MSG recently. Any one of the following 3 scenarios is possible.

(i) Q would have initiated the ELMSG before P.

(ii) Q would have initiated the ELMSG just after P..

(iii) Q did not create any ELMSG at all.

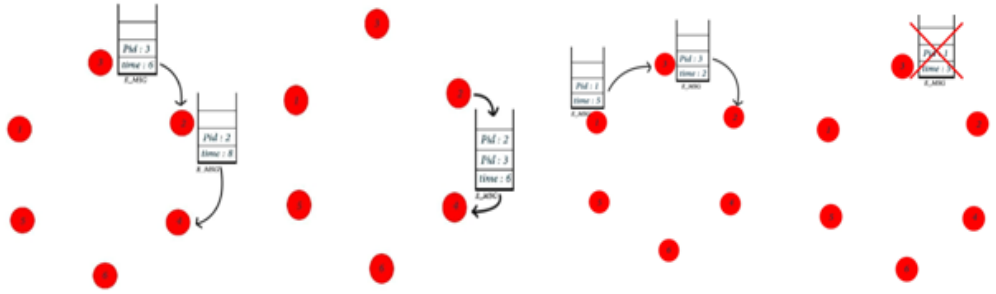

Figure 1. Passing and Destruction of Election Message

5. MATHEMATICAL ANALYSIS

A. Best Case Analysis

Let N be the number of processes, In the Best case, only one process detects that the the coordinator has crashed. Then, altogether, there will be 2n messages sent, one full circulation EL MSG for a maximum of N-1 processes and one full circulation CO MSG for a maximum of N-1 processes.

$$n_e + n_c = 2n \qquad (3)$$

leading to O(n) message complexity. The Ring Election Algo-rithm and the our proposed Ring Election Algorithm have the same time complexity as in equation (1).

B. Worst Case Analysis

Ring Algorithms worst case message complexity is $O(n^2)$ from equation (2). In our Algorithm, the worst case of the modified Election Algorithm is further divided into three more cases of Best, Average and Worst case.

1) Best Case: The best case is when, the processes initiate the Election in the anticlockwise direction according to time when the ring network is of clockwise direction. (ie) when the Election Algorithm gets invoked in the opposite direction of the ring according to time. The Election Messages get deleted in the very first go itself, that is when Process i send s election message EMi to its immediate neighbour Process j, and since the time of Election Message of Process j, EMj ¡ Emi. Emi is killed by process j. Like this, except the very first or very earlier election message, all the other election messages would be killed by the immediate successor processes. The number of EL MSG and CO MSG = 1 + 1+ ...+ 1 (n-1 times) and EL MSG destroyed by SUCC ≤ n − 1T herefore, T otal Messages = ne + nc + n-1y e Where ye represents the number of young election messages. Here, the time complexity is O(n).

2) Average and Worst Case: It is when the processes initiate the Election in the clockwise direction according to time when the ring network is of clockwise direction. (i.e.) when the Election Algorithm gets invoked along the direction of the ring according to time. Let Process a creates the first election message and passes it on along the ring. Then let its successor creates the Election message and passes to its successor and so on. In this case, only Process (a) can kill the election messages created by all the other processes. The duplicate election messages would take n-1, n-2, n-3, ...(3,2,1 hops to reach Process a right from the successor to the immediate predecessor). The number of messages are,

$$T\,otalmessages = 1 + 2 + ... + n - 1 + n + n$$

$$\sum_{K=1}^{N} k = (1/2)n2 + (1/2)n + n$$

$$= n + n + n\ (n-1)\ /2$$

$$= n2 - n/2 + 2n \qquad (4)$$

Bringing the time complexity to O(n2) But in average case, the EL MSG would be destroyed somewhere in the middle and most of the times by the SUCC itself. When the processes receive the EL MSG by other processes, they wont initiate the Election at all. And so, which would be O(n). The probability of having O(n2)time complexity is very rare.

Table 2. Function: Build EL _MSG(T, PID)

```
begin
Create EL MSG

Set T = Current Time
Set F = TRUE
Append PID

Return
      end
```

Table 3. Function: Update EL_ MSG

```
begin
        if (my F = TRUE)
    then  if (my T ¡ Ti )
   then Destroy EL _MSG
    Return
       endif
           else
              Append PID in the list
       endif
end
```

Table 4. Function: Build CO _MSG

```
begin
    LEAD = highest PID from the list
    Delete all the PID except LEAD
     Change EL_MSG to CO _MSG
    Return
end
```

We have focused more on the worst case as processes in the real world keep communicating with each other and try sharing the common resources, many processes would identify the Leader's

crash and intern initiate the Election. Therefore, best case would be very rare and the average case to worst case are only possible. Our algorithm tries to reduce the number of messages in the worst case to a great extent.

6. EXPERIMENTS AND SIMULATION RESULTS

We simulated Ring election algorithm and MEREA in Java. We created number of processes such as 100, 200 and so on. The clock time of each process was synchronized. We kept 3 seconds message propogation time to reach from one process to another and 1 second to process the message. We also made the process with the highest PID to be down which in turn invoked the Election activity automatically. We focused only on the worst case scenario because in the real time there is no possibility of having best case. Processes in the network with critical applications keep on sharing the common resources and exchange messages and this needs to be monitored and controlled by the coordinator. Worst case only is possible during these scenarios. In MEREA, the Worst case is again classified into two sub cases like Worst-Worst case and Worst-Best case. In Worst-Worst case, the message complexity is still $O(n^2)$ but it drastically reduces the number of messages into one forth of the messages as in Ring Algorithm. Our algorithm's Worst-Best case's performance is O(n). Figure 5 represents the number of messages being shared by all these processes duing the simulation. The MEREA Worst-Worst case has shown little lesser number of messages comparing to its mathematical analysis. That is because when we destroy messages, it depends on the time of Election Message and the time it takes to propogate. Figure 1 gives the comparison of the two algorithms namely the existing Ring Election Algorithm and our Proposed Algorithm MEREA, in terms of messages being passed during the best and the worst case.

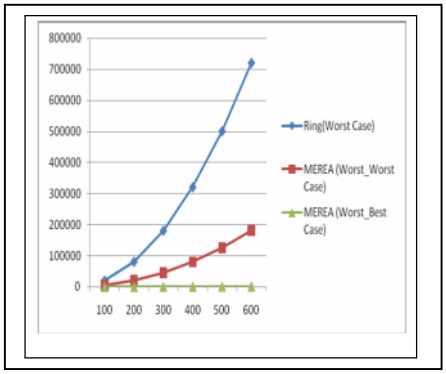

Figure 1. Comparison of Ring and MEREA

7. CONCLUSIONS

The duplicate election messages are destroyed and so the respective COORDINATOR messages are avoided. There is only one election message which completes its circulation and only one COORDINATOR message. This scenario is applicable for all the best, average and worst cases. N number of COORDINATOR messages are circulated in worst case of existing Ring Algorithm and N-1 duplicate election messages were also present in the existing algorithm, but our algorithm destroys all of them as early as possible by the help of synchronized clock time. These two factors gives better performance in terms of time and messages. The idea can be applied in

Bully's Election algorithm. In future we would work on choosing the right cluster leaders in distributed sensor network.

REFERENCES

[1] Sung-Hoon Park, "A Stable Election Protocol based on an Unreliable Failure Detector in Distributed Systems", Proceedings of IEEE Eighth International Conference on Information Technology: New Generations, pp. 976-984, 2011.

[2] Sandipan Basu, "An Efficient Approach of Election Algorithm in Distributed Systems", Indian Journal of Computer Science and Engineering (IJCSE), vol. 2, No. 1, pp. 16-21.March 2011.

[3] Muhammad Mahbubur Rahman, Afroza Nahar , "Modified Bully Algorithm using Election Commission", MASAUM Journal of Computing(MJC),Vol.1 No.3,pp.439-446,October 2009, ISSN 2076-0833.

[4] Chang-Young Kim, Sung-Hoon Bauk, " The Election Protocol for Reconfigurable Distributed Systems" , ICWN, pp. 295-301, 2006.

[5] M. S. Kordafshari, M. Gholipour, M.Jahanshahi, A.T. Haghighat, "Modified Bully Election Algorithm In Distributed System", Wseas Conferences, Cancun, Mexico, May 11-14, 2005.

[6] Sepehri M , Goodarzi M , "Leader Election Algorithm Using Heap Structure", Proceedings Of The 12th Wseas International Conference On Computers(Iccomp'08), 2008.

[7] Cuibo Yu, Xuerong Gou, Chunhong Zhang, Yang Ji, "Supernode Election Algorithm In P2p Network Based Upon District Partition", Dcta: International Journal Of Digital Content Technology And Its Applications, Vol. 5, No. 1, Pp. 186 -194, 2011.

[8] Ben Ari, "Principles Of Concurrent And Distributed Programming", Pearson Education, 2nd Edition, 2006.

[9] H. Garcia-Molina, "Elections in Distributed Computing System", IEEE Transaction Computer, Vol. C-310, pp. 48- 59, 1982.

[10] Andrew S and Tanenbaum, "Distributed Systems Principles and Paradigms", Beijing: Tsinghua University Press, pp.190–192, 2008.