International Journal of Electrical, Electronics and Computer Engineering **1**(2): 1-6(2012)

ISSN No. (Online) : 2277-2626

Election Administration Algorithm for Distributed Computing

S.K. Gandhi* and Pawan Kumar Thakur*

***Department of Computer Science and Engineering, AISECT University, Bhopal, (M.P.)*

(Recieved 10 June, 2012 Accepted 25 July, 2012)

ABSTRACT : An election algorithm is an algorithm for solving the coordinator election problem. Various algorithms require a set of peer processes to elect a leader or a coordinator. It can be necessary to determine a new leader if the current one fails to respond. Provided that all processes have a unique identification number, leader election can be reduced to finding the non crashed process with the highest identifier. Garcia-Molina's Bully Algorithm is a classic solution to leader election in synchronous systems with crash failures. In this paper, we will presented an efficient version of bully algorithm to minimize the redundancy in electing the coordinator, to reduce the recovery problem of a crashed process in distributed systems and thus to maximize the effectiveness of traditional bully algorithm.

Keywords: Election Administration , Bully Algorithm, Election, Coordinator, Process Number, Turnaround Time, Election Message, Ok Message, Coordinator Message, Query Message, Answer Message passing.

I. INTRODUCTION

A distributed computing system is basically a collection of autonomous processors or node interconnected by a communication network in which each processor or node has its own local memory and other peripherals, and the communication between any two processors of the system takes place by message passing over the communication network to corporate a common goal. When a node fails or when the communication subsystem which allows nodes to communicate fails. It is often necessary to recognize the active nodes so that they continue to perform a useful task for their common goal. If enough nodes fails the remaining node may decide that they just cannot perform the assigned task and may select a new or better suited job for themselves. After the failure occurs in a distributed computing system the first step is to elect a coordinator node to manage the operation. A process is used to coordinate many tasks. It is not an issue which process is doing the task, but there must be a coordinator that will work at any time. So electing a coordinator or a leader is very fundamental issue in distributed computing [1, 2, 3, 4]. And there are many algorithms that are used in election process. Bully election algorithm is one of them.

This works represents a modified version of bully algorithm using a new concept Election Administration. This approach will not only reduce redundant elections but also minimize total number of elections and hence it will minimize message passing, network traffic, and complexity of the existing system. In section 2 represents original bully assumptions and limitations. Methodology, algorithm and comparison of our proposed work is given in section 3, 4, and 5.

II. BULLY ALGORITHM BY GARCIA-MOLINA

Bully algorithm is one of the most famous election Algorithms which was proposed by Garcia-Molina in 1982. The assumptions and limitation briefly described in this section.

A. Assumption of Bully Algorithm

This algorithm is established on some basic assumptions which are:

(*a*) It is a synchronous system and it uses timeout mechanism to keep track of coordinator failure detection.

(*b*) Each process has a unique number to distinguish them.

(*c*) Every process knows the process number of all other processes.

(*d*) Processes do not know which processes are currently up and which processes are currently down.

(*e*) In the election, a process with the highest process number is elected as a coordinator which is agreed by other alive processes.

(*f*) A failed process can rejoin in the system after recovery.

In this algorithm, there are three types of message and there is an election message (inquiry) which is sent to announce an election, an answer (OK) message is sent as response to an election message and a coordinator victory message is sent to announce the new coordinator among all other alive processes.

B. Limitation of Bully Algorithm

Bully algorithm has following limitations :

1. **Network Traffic.** The main limitation of bully algorithm is the highest number of message passing during the election and it has order $O(n2)$ which increases the network traffic [9, 7, 6]. When any process that notices coordinator is down then holds a new election. As a result, there may n number of elections can be occurred in the system at a same time which imposes heavy network traffic.

2. **No guarantee on message delivery.** This algorithm is not guaranteed to meet the safety condition *E*l if processes that have crashed are replaced by processes with the same number. A process that replaces a crashed process p may decide that it has the highest number just as another process (which has detected p's crash) has decided that it has the highest number. Two processes will announce themselves as the coordinator concurrently. Unfortunately, there are no guarantees on message delivery order, and the recipients of these messages may reach different conclusions on which is the coordinator process [10].

3. **Redundant Election.** If the coordinator is running unusually slowly (say system is not working properly for some reasons) or the link between a process and a coordinator is broken for some reasons, any other process may fail to detect the coordinator and initiates an election. But the coordinator is up, so in this case it is a redundant election.

   Again, if a process p with lower process numbers a coordinator itself. If any process with the highest priority number is up, it will run the algorithm again than the current coordinator, crashes and recovers again, it will initiate an election where the current coordinator will win again. This is also a redundant election.

4. **Failure detector is unreliable.** Condition *E*l may be broken if the assumed timeout values turn out to be accurate - that is, if the processes' failure detector is unreliable [10]. Taking the example just given, suppose that process 5 either had not failed bit running unusually slowly (that is, the assumption that the system is synchronous incorrect) or that process 5 had failed but is then replaced. Just as process 4 sends its coordinator message, process 5 (or its replacement) does the same. Process 4 receives process 5's coordinator message after it sent its own and so sets elected 4 = process 5. Due to variable message transmission delays, process 3 receives process 4's coordinator message after process 5's and so eventually elected 3 = process 4. Condition *E*l has been broken.

5. Other drawbacks. From the above example, we can find out some other drawbacks :

(*i*) When process 5 recovers from failure, it becomes new coordinator whereas process 6 with the highest process number is still alive. This is a violation of the assumption.

(*ii*) At the time, when process 4 is coordinator, process 1 recovers from failure and it initiates an election. Then process 4 becomes coordinator again. This election is redundant

(*iii*) When process 2 notices the failure of the coordinator process 6, it sends election messages to processes 3, 4, 5 and 6. In reply of the election messages, it gets ok messages from processes alive among processes 3, 4, 5 and 6. Process 2 can elect the coordinator itself instead of holding elections by processes 3, 4 and 5. These elections are also redundant [4, 5].

III. METHODOLOGY OF OUR PROPOSED WORK

Our purposed work is based on Election Administration approach. It is an election administrative body well-known to deal with leader election mechanism in a distributed computing system. The Election Administration works as following :

1. Election Administration (E_A)

Election Administration made up with group of special processes in distributed system. It is certified to handle the whole election process. It defines the rules and regulations for attending in an election process in a distributed computing system. It has one Chief Election Admin (CEC) and four Election Admins. If any of the Admins failed, Election Administration E_A will recover that Admins without delay and other processes do not have concern of that. An Election Administration has a unique group ID. Other processes in the system communicate with Election Administration using this group ID. As a result, if any of the commissioners is down, there will be not any problem in election. It has a reliable failure detector (F_D). If maximum message transmission delay is *Tmsg* and maximum message processing delay is *Tpos* then maximum time required to get a reply after sending a message to any process from Election Administration is $T = 2T_{msg} + T_{pos}$. If Election Administration does not get any reply from a process within T time, then FD of Election Administration will report that requested process is down. As like as FD, Election Administration has another component named helper (H_P), the function of HP is to find out the process with the highest process number

using sending alive message. It knows process number of all processes of the system. Fig. 3 represents the architecture of an E_A.

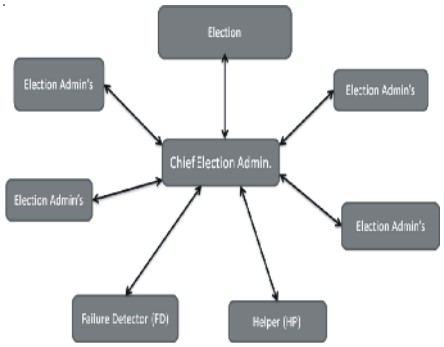


Fig. 1. Architecture of Election Administration (E_A).

2. Chief Election Admin

Chief Election Admin is the principal of Election Administration EA. The process with the highest priority in Election Administration group will be the Chief Election Admin. It controls other Election Admins and handles FD and HP.

3. Election Admin

Election Admin is a member of Election Administration. It is a special kind of process. Any Election Administration in a distributed system will have a few numbers of Election Admins (say three or four). All of them consult with the Chief Election Admin under the rules and regulation while there will be a need of an election

As the system is synchronous and Election Administration has a failure detector FD and helper HP to solve limitations which is mentioned in section 2, we have proposed a modified version of bully algorithm using Election Administration concept. This algorithm not only reduces redundant elections but also reduces message passing between processes and hence traffic in network will be decreased dramatically.

A. Algorithm

Our proposed algorithm has the flowing steps :

Step 1 : When process P_i notices that the coordinator process *Pj* is down [5], it sends an election message to Election Administration E_A.

Step 2 : Failure detector F_D of Election Administration E_A verifies election message sent by P_i. If the sending notice of P_i is not correct, then Election Administration E_A will send a coordinator message to P_i with process number Pn of the current coordinator.

Step 3 : If the sending notice of P_i is correct and if the highest process number is P_i, then Election Administration E_A will send a coordinator message to all processes with process number P_n of P_i as a new coordinator. If the highest process number is not P_i, Election Administration E_A will simply find out the alive process with the highest process number using helper H_P and sends a coordinator message to all processes with the process number P_n of that process as a new coordinator.

Step 4 : If any process P_n including last crashed coordinator P_j is up, it will send a Query message to the Election Administration E_A. If the process number P_n of the newly entranced process is higher than the process number of the current coordinator P_j, Election Administration E_A will send a coordinator message to all processes having the process number P_n of new coordinator. It not, Election Administration will simply send a coordinator message to newly entranced process having process number of the current coordinator.

Step 5 : If more than one process sends coordinator message to Election Administration E_A at the same time, then Election Administration E_A will consider the process with higher process number P_n which ensure less message passing to find out the highest process number using helper *Hp*.

B. Procedure

The step by step Election procedure of our proposed algorithm represents in Fig. 2(*a*), 2(*b*), 2(*c*), 2(*d*).

Step 1 : The system consists of six processes with process number 1 to 6. Current coordinator is the process 6. But it has just crashed and process 2 first notices this. So it sends an election message to the E_A in Fig. 2(*a*).

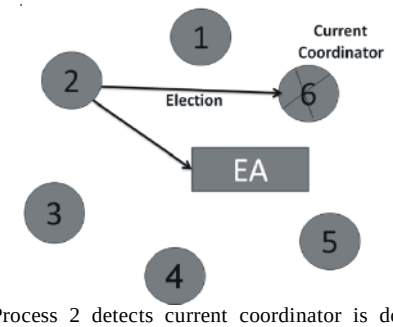


Fig. 2(*a*). Process 2 detects current coordinator is down and sends an election message to E_A.

Step 2 : EA ends verify message to the current coordinator to be sure about the election message sent by process 2. After verification as shown in Fig. 2(*b*).

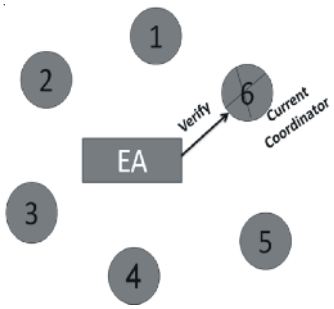


Fig. 2(*b*).

Step 3 : E_A sends alive message to process 5 (the next highest process number) to check either the current highest process is alive or not. And EA gets a reply message from EA gets a reply message from 5 as shown in Fig. 2(*c*).

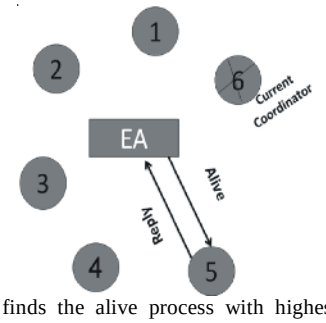
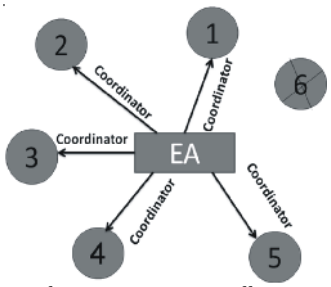


Fig. 2(*c*). E_A finds the alive process with highest number using alive message.

Step 4 : E_A select 5 as new coordinator and sends coordinator message to all processes having 5 as a new coordinator of the system as shown in Fig. 2(*d*).



2(*d*). EA sends coordinator message to all process having process number of currently won.

2. Query after Recovery : Fig. 3 represents the steps when a crashed process is up :

Step 1 : The last crashed coordinator 6 is up and sends a query message to EA. As process number of 6 is higher than the current coordinator of the system as shown in Fig. 3(*a*).

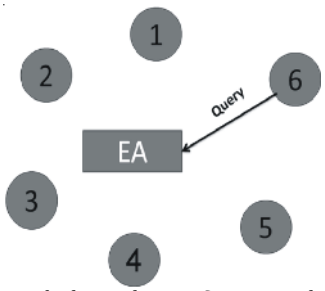
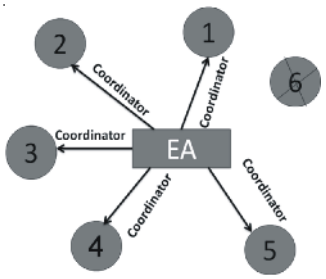


Fig. 3(*a*) Last crashed coordinator 6 is up and sends a query message to the EA.

Step 2 : EA sends coordinator message to all processes with process number 6 as new coordinator as shown in Fig. 3(*b*).



3(*b*) EA selects 6 as new coordinator and sends coordinator message to all processes.

Step 3 : Process 1 is now just Crashed as shown in Fig. 3(*c*).

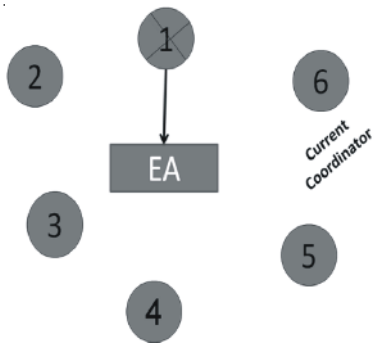


Fig. 3(*c*) Process 1 is now just Crashed.

Step 4 : Process 1 is just up after crashed, and it sends a query message to EA. It checks that process number of newly entranced is lower than the current coordinator as shown in Fig. 3(*d*).

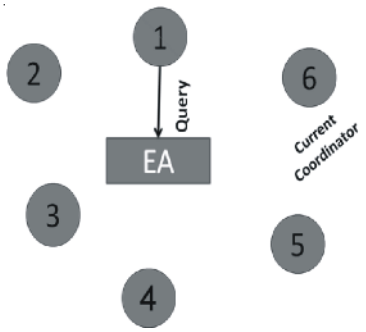


Fig. 3(*d*). Again process 1 is up and sends query message to E_A.

Step 5 : E_A sends coordinator message to only process 1 having the process number of current coordinator of the system as shown in Fig. 3(*e*).

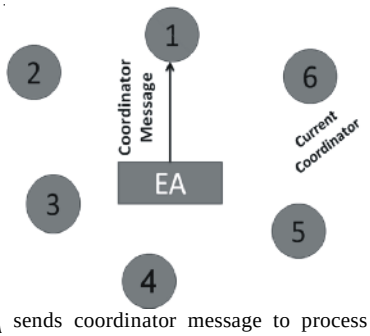


Fig. 3(*e*). E_A sends coordinator message to process 1 having the current coordinator.

At any time, if more than one processes notice that coordinator is down, they will send election message to E_A. After verification, E_A will consider election request of the process having higher process number. In Fig. 6, process 4 and 5 detect that coordinator 6 is down, so 4 and 5 send election message to E_A. After verification, E_A only consider election message of process 5. It ensures less message passing to find out the highest process number.

Say if E_A considers election message of 4, then according to our algorithm, E_A will have to send alive message to 5 to find higher process number. But if E_A considers election message of 5, it does not need to send alive message because, 5 is already the higher process number and E_A can select 5 as new coordinator. This was E_A can ensure less message passing.

IV. COMPARISON AND DISCUSSION

In this section, we present the comparison in different issues among our proposed algorithm, original bully algorithm. We consider message passing complexity and redundant election both of which increase network traffic.

1. **Message passing.** If there are n processes in the system and p is the process number which detects failure of coordinator, then :

(*a*) In original bully algorithm, there will be needed of message passing between processes. In the worst case, if process with the lowest process number detects coordinator as failed, then it requires message passing. In the best, case when p is the highest process number, it requires messages.

(*b*) For the case of modified bully algorithm there will be need of or messages passing between processes. In worst case that is the process with lowest process number detects coordinator as failed, it requires $3n - 1$ messages passing. In best case when p is the highest process number, it requires $(n - p) + n$ messages.

(*c*) For the case of modified bully algorithm there will be need of or $O(n)$ message passing between processes. In worst case that is the process with lowest process number detects coordinator as failed, it requires $3n - 1$ message passing. In best case when p is the highest process number, it requires $(n - p) + n$ messages.

For the case of our proposed algorithm there will be need of 1 election message to inform E_A, 2 verify message to ensure the failure of coordinator, and say r is the highest alive process then alive and reply message to find out the highest alive process and so total or $O(n)$ message passing between processes.

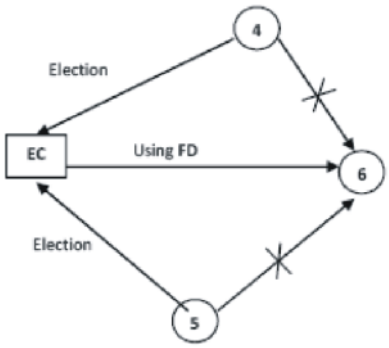


Fig. 4. More than one election Message to E_A.

If the process with lowest process number detects coordinator as failed it will not change total message. In worst case it may happen that our algorithm needs to check up process to $pn + 1$ to find out highest alive process. Only at that case it requires message passing between processes.

However, in best case, our algorithm may find the highest alive process with only one alive and one reply message that is highest alive process in the system is process with process number $n - 1$. In that case, our algorithm requires only $1+ 2 + 2 + n$ messages. When p is the highest process number, it requires only $1 + 2 + n$ messages.

(*d*) If a process crashes and recovers again, it sends a query message to all processes higher than that process to know the current coordinator which requires $2*(n - p)$ message passing. But in our algorithm, any process after recovery will only send query message to E_A and E_A will send a coordinator message having process number of current coordinator which requires only 2 messages passing.

2. Redundant election

(*a*) In original bully algorithm and modified bully algorithm , if coordinator is running unusually slowly say (system is not working properly for some reasons) or the link between a process and coordinator is broken for some reasons, there will be redundant election, although current coordinator is up. But in our algorithm, as E_A verifies either current coordinator is really up or down when E_A receives any election message from any process, it ensures that there will be no redundant election in the system.

(*b*) If a process p crashes and recovers again, it initiates an election where the current coordinator wins again which is redundant election. But in our algorithm, after recovery, any process will send query message to E_A and E_A will reply with coordinator message having process number with current coordinator which reduces unnecessary election.

3. Multiple Coordinators. As there is no guarantee on message delivery that may happen more than one coordinator exist in the system at a time in bully algorithm and modified bully algorithm. But in our algorithm there is no possibility of this as E_A handles whole election process.

V. CONCLUSION

In this work, we modified bully algorithm using a new concept Election Administration (EA). We tried to overcome limitations of original bully algorithm and modified bully algorithm. Our comparison and discussion section prove that our algorithm is more efficient than bully algorithm and modified bully algorithm in respect of message passing, redundant election and network traffic.

REFERENCES

[1] Coulouris, G., Dollimore, J., Kindberg, T.,"Distributed Systems Concepts and Design", Pearson Education, pp. 431-436 (2003).

[2] Tanenbaum A.S, "Distributed Operating System, Pearson Education", (2007).

[3] Sinha P.K, Distributed Operating Systems Concepts and Design, Prentice-Hall of India private Limited, (2008).

[4] H. Garcia-Molina, "Elections in Distributed Computing System", *IEEE Transaction Computer*, Vol. **C-31**, pp.48-59, Jan. (1982).

[5] Kshemkalyani A.D. and Singhal M., "Distributed Computing principles Algorithms, and Systems", Cambridge University Press, (1987).

[6] Quazi Ehsanul Kabir Mamun, Salahuddin Mohammad Masum,Mohammad Abdur Rahim Mustafa, "Modified Bully Algorithm for Electing Coordinator in Distributed System", 3rd WSEAS international conference on Software Engineering, Parallel Distributed Systems (SEPADS) February 13-15 (2004), Salzburg, Austria.

[7] 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).

[8] A new approach for election algorithm in distributed systemc. Second International Conference on Communication Theory, Reliability, and Quality of Service (2009).

[9] Thakur P. Kumar , Kumar Ram, Ali Ruhi and Malviya Rajendra, "A New Approach of Bully Election Algorithm for Distributed Computing", *Int. J. of Electrical, Electronics and Computer Engineering (IJEECE)* Vol **1**(1): 72-79(2011).

[10] Deepali P. Gawali, "Leader Election Problem in Distributed Algorithm", *IJCST* Vol. **3**, Iss ue 1, Jan. - March (2012).