Asian Journal of Information Technology 15 (22): 4482-4488, 2016
ISSN: 1682-3915
© Medwell Journals, 2016

A New Improved Election Algorithm for Achieving High Availability in Distributed Systems

N. Gopikarani, G. Sudha Sadasivam and G. Kavitha
Department of Computer Science and Engineering, PSG College of Technology,
Coimbatore, Tamil Nadu, India

Abstract: In distributed environment, the leader election process is to main system consistency and to make the system a reliable one. Leader election process is to select a single node as leader among other members. High availability is a system design approach. It ensures that a pre arranged level of operational performance will be met during a contractual measurement period. High availability is essential for mission critical applications. The term highly available refers that the systems can continue providing services even when master node (i.e., coordinator) fails. Goal of the proposed work is to use a suitable distributed election algorithm to improve high availability features. Stable leader election is implemented to increase robustness and efficiency. Election commission concept is used to minimize message passing. This algorithm is used to identify a temporary master when there is any failure in existing master node and reduces message passing. At the back-end the implementation will be invoked inside a cluster to achieve scalability in addition to high availability features.

Key words: Coordinator, leader election process, message passing, stable leader election, election commission

INTRODUCTION

It is paradoxical that the larger a system is, the more critical is its availability and the more difficult it is to make it highly-available. It is possible to build small ultra-available modules but building large systems involving thousands of modules and millions of lines of code is still an art. These large systems are a core technology of modern society, yet their availability is still poorly understood. In a distributed computing system, 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.

Need of election in distributed system: Several distributed algorithms require that there be a coordinator node in the entire system that performs some type of coordination activity needed for the smooth running of other nodes in the system (Park *et al*., 1999). As the nodes in the system need to interact with the coordinator node, they all must unanimously who the coordinator is. Also if the coordinator node fails due to some reason (e.g., link failure) then a new coordinator node must be elected to take the job of the failed coordinator (Garcia, 1982).

Leader election process: Distributed systems are used to increase the computational speed of problem solving. These systems use a number of computers which cooperate with each other to execute some task. Control of distributed algorithms requires one process to act as a controller (leader) (Garcia, 1982). Leader is responsible to maintain the stability all time overall the network. If the leader fails for any reason, new leader should be elected directly to recover from instability (Tanenbaum, 2007). Electing leader is a vital issue not only in distributed computing but also in communication network, centralized mutual exclusion algorithm, centralized control IPC, etc. A leader is required to make synchronization between different processes by Attiyae and Welch (Chandra and Toueg, 1996).

Leader election process is a program distributed over all nodes, it starts when one or more processors discover leader has failure, it terminates when remaining processors know who the new leader is. Leader Election Algorithms (LEAs) are widely used in centralized systems to solve single point failure problem (Aguilera *et al*., 2001). For example, in client server, LEAs are used when the server fails and the system needs to transfer the leadership to another station. The LEAs are also used in

Corresponding Author: N. Gopikarani, Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, Tamil Nadu, India

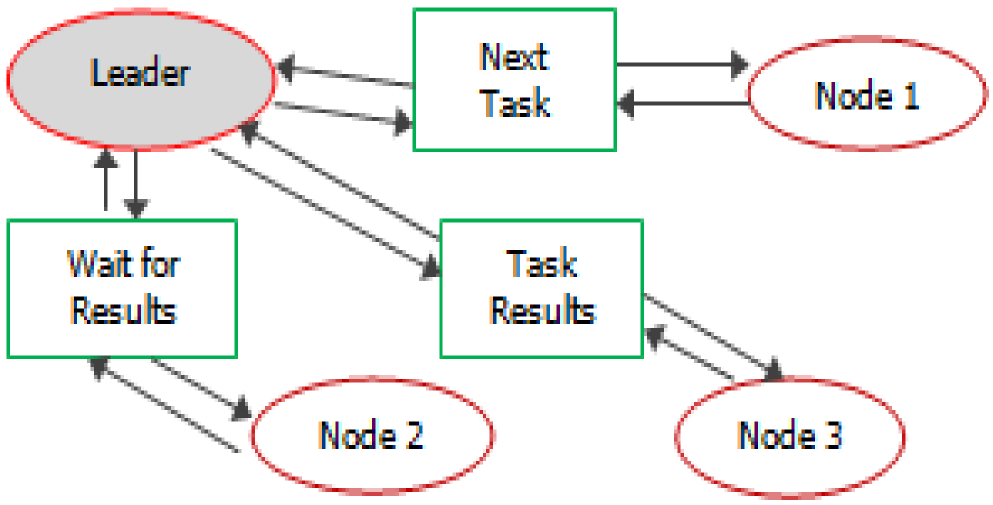

Fig. 1: Typical leader coordination

token ring. When the node that has the token fails, the system should select a new node to have the token (Basu, 2011).

A leader election algorithm is stable if it ensures that once a leader is elected, it remains the leader for as long as it does not crash and its links have been behaving well, irrespective of the behavior of other processes as shown in Fig. 1.

Election commission: Election commission is an electoral administrative body established to deal with leader election mechanism in a distributed computing system. It is constructed by a group of special processes in distributed system. It is authorized to handle the whole election process (Sinha, 2008). It defines the rules and regulations for attending in an election process in a distributed computing system. It has one Chief Election Commissioner (CEC) and four election commissioners. If any of the commissioners failed, Election commission will recover that commissioner immediately and other processes do not have concern of that.

An election cmmission has a unique group ID. Other processes in the system communicate with eection cmmission 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 (FD) (Sinha, 2008). If election commission does not get any reply from a process within the specified time, then FD of election commission will report that requested process is down. As like as FD, Election commission has another component Named Helper (HP), 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.

Chief election commissioner is the principal of Election Commission. The process with the highest priority in election commission will be the chief election commissioner. It administrates other election commissioners and handles FD and HP. Election commissioner is a member of election commission. It is a special kind of process. Any election commission in a distributed system will have a few numbers of election commissioners (say four). All of them consult with the chief election commissioner to elect a coordinator in a distributed system as shown in Fig. 2.

Bully algorithm: Leader election is a procedure that is embedded in every node of the distributed system (Park *et al*., 1999). Any node which detects the failure of the leader node can initiate a leadership election. The election concludes its operation when a leader is elected and all the nodes are aware of the new leader and agree on that as in Fig. 3a-e.

Drawbacks: Every time a node (former leader or ordinary node) recovers from a crash failure, it initiates an election which consumes significant system resources (Dolev *et al*., 1991).

Although, this algorithm ensures liveness, it sometimes fails to meet the safety condition. This can happen when a former leader node is replaced by a node with the same id number while the election procedure is in progress. The newly elected node and the former leader node (which was down for a while) will both announce themselves as leaders simultaneously (Dolev *et al*., 1991).

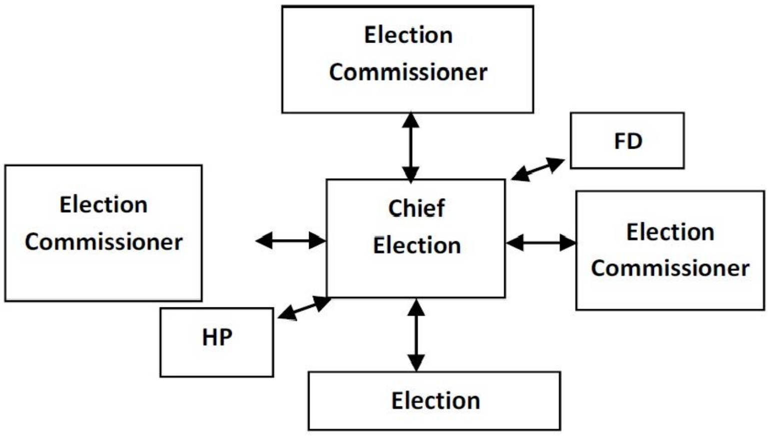

Fig.2: Architecture of election commission

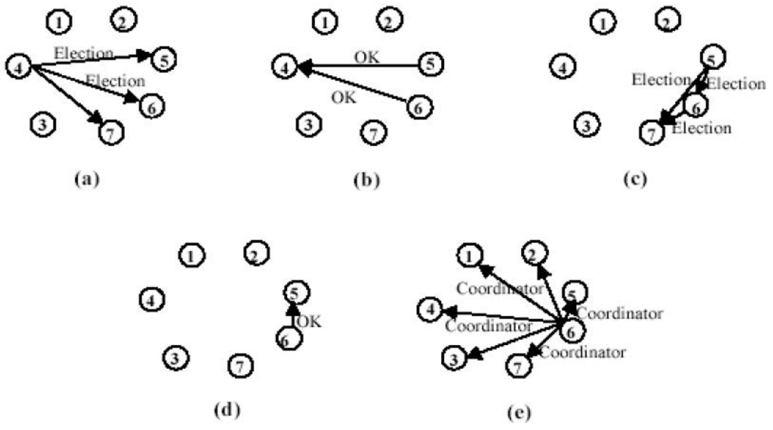

Fig. 3: Election procedure in bully algorithm

This algorithm elects a new leader node with the help of a number of redundant elections. In the worst case, it can require a large number of messages to elect a leader node (Rahman and Nahar, 2010). The worst case occurs when a node with the lowest id initiates an election: the role for initiating election is handed over to a node with next higher id and this continues till the node with the highest id takes over the role. As a result, at least n-2 redundant elections take place in the entire system where n is the number of nodes. This algorithm does not provide an efficient solution for the simultaneous detection of leader node failure by more than one node. More than one election may take place at the same time which imposes a heavy load on the network (Garcia, 1982).

METERIALS AND METHODS

System design

Modified leader election algorithm: The proposed leader election algorithm uses the concepts of election commission and stable leader election. The reasons for adopting these concepts in a distributed system are:

- To reduce redundant elections
- To minimize message passing
- To improve the complexity

Our proposed algorithm has the following steps; when process P notices that the coordinator is down, it sends an ELECTION message to election commission. FD of election commission verifies ELECTION message sent by P. If the sending notice of P is not correct, then election commission will send a COORDINATOR message to P with process number of the current coordinator. If the sending notice of P is correct and if the highest process number is P, then election commission will send a COORDINATOR message to all processes with process number of P as a new coordinator. If the highest process

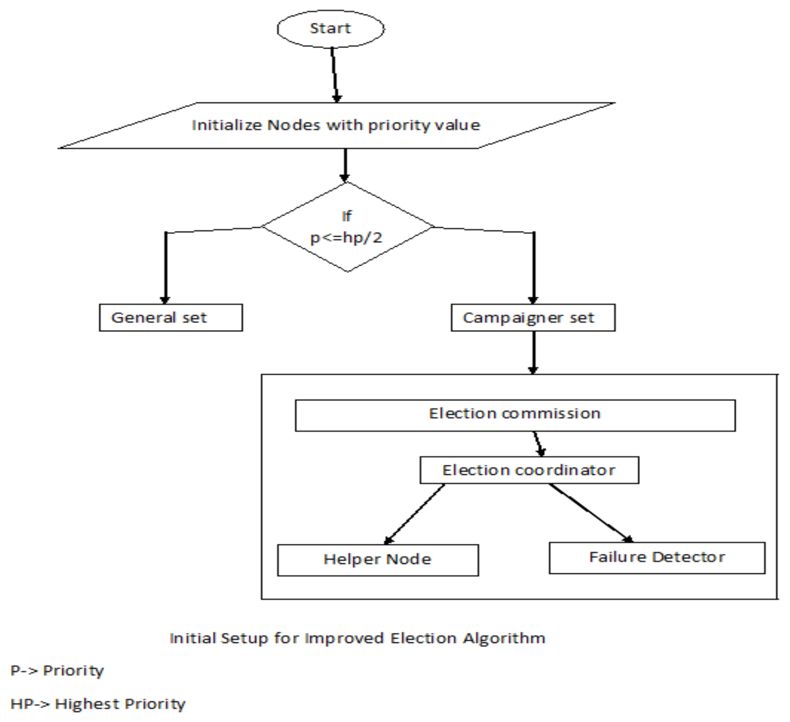

Fig. 4: Initial setup

number is not P, Election commission will simply find out the alive process with the highest process number using HP(HP checks the priority based sorted list of the candidate set) and sends a coordinator message to all processes with the process number of top-most process after the crashed coordinator as a new coordinator as in Fig. 4.

If any process excluding last crashed coordinator is up, it will send a query message to the election commission. Election commission will simply send a coordinator message to newly entranced process having process number of the current coordinator. The recovered coordinator will have to wait till the current low priority coordinator finishes the on-going election process. If more than one process sends election message to election commission at the same time, then election commission will consider the process with higher process number which ensure less message passing to find out the highest process number using HP. The overall election process is shown in Fig. 5.

Procedure: Let's assume, the system consists of five processes with process number 1-5.

Case 1-current coordinator crashes: Current coordinator is the process 5. But it has just crashed and is first noticed by process 2. So it sends an election message to the EC. The EC sends verify message to the current coordinator about the election message sent by process 2. After verification, Helper node (HP) of EC sends alive message to process 4 (the next highest process number from the MAP table) to check if it is alive or not. And HP gets a reply message from 4. The EC then selects 4 as new coordinator and sends coordinator message to all processes.

Case 2-Normal process recovery: Assume, process 1 has just recovered after being crashed. It sends a query message to EC. EC checks that process number of newly entranced is lower than the current coordinator. So, EC sends coordinator message to only process 1 having the process number of current coordinator of the system.

Case 3-Multiple election messages: At any time, if more than one processes notices that the coordinator is down, they will send election message to EC. After verification, EC will consider election request of the process having higher process number. For example, process 4 and 5 detect that coordinator 6 is down, So 4 and 5 send election message to EC. After verification, EC only considers the election message of process 5. It ensures less message passing to find out the highest process

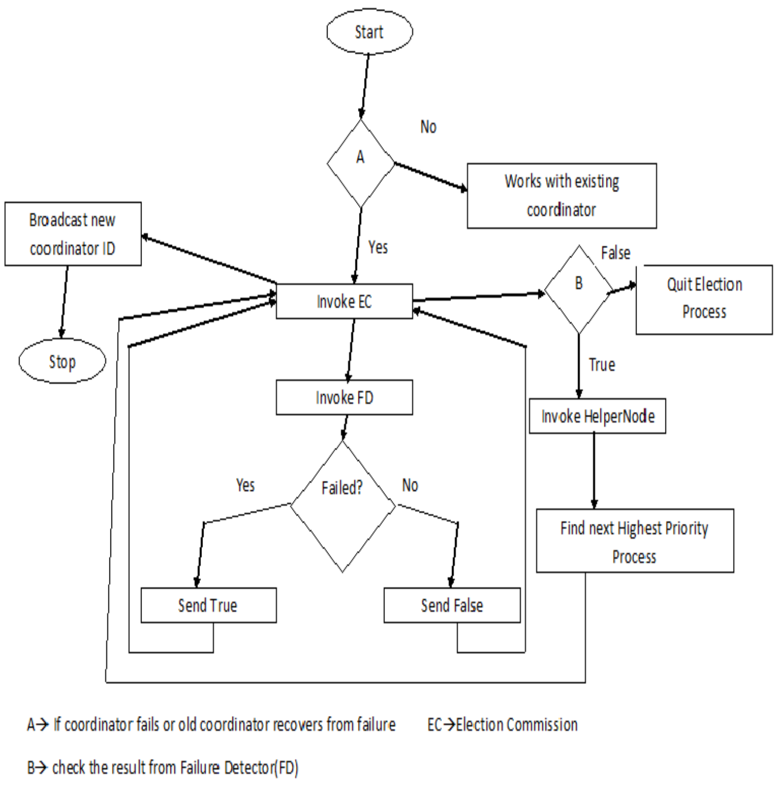
Start
A
No
Works with existing coordinator
Yes
Broadcast new coordinator ID
Stop
Invoke EC
B
False
Quit Election Process
True
Invoke HelperNode
Find next Highest Priority Process
Invoke FD
Failed?
Yes
No
Send True
Send False
A→ If coordinator fails or old coordinator recovers from failure
EC→Election Commission
B→ check the result from Failure Detector(FD)

Fig. 5: Overall election process

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

Case 4-Crashed coordinator recovery: When the crashed coordinator recovers, the EC verifies the process ID. If the process recovered has a higher priority than the current coordinator, then EC implements the stable leader concept. The recovered process is requested to wait for the current coordinator to complete its election cycle. Hence, unlike in the bully algorithm, the number of messages needed to terminate an existing election is avoided here.

RESULTS AND DISCUSSION

Experimental setup: Experiments are done on a cluster of about 5 systems with pentium 4 processor: Intel® Core™2 Duo, 2.54 Ghz sprocessor, 1GB RAM, 160GB hard disk interconnected with ethernet LAN capable of transmitting at a maximum speed of 4 Mbps. Results are analyzed based on time complexity.

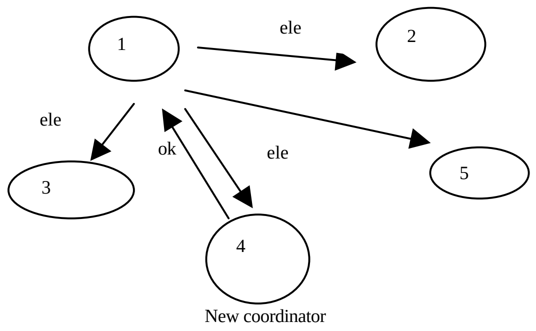

Fig. 6: Coordinator failure and re-election

Implementation of existing bully algorithm

Coordinator failure and re-election: Figure. 6, there are five processes labeled as process 1, process 2, process 5. Process 5 is considered as the highest priority process among the given set of processes. The coordinator crash is noticed by Process 1 and it initiates the election. Process 4 is elected as the new coordinator from the group.

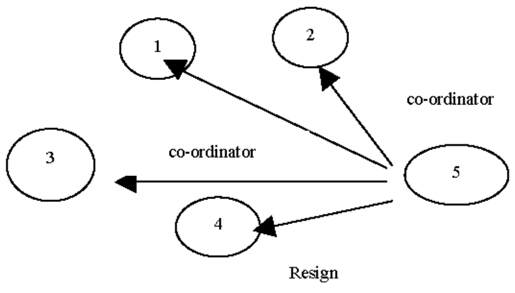

Fig. 7: Crashed coordinator recovery

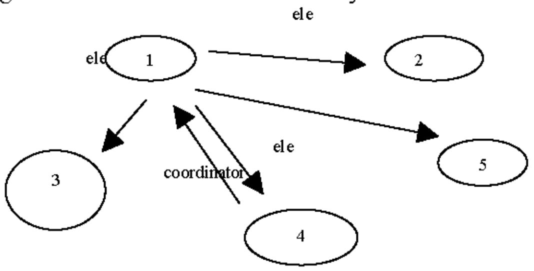

Fig. 8: Normal process recovery

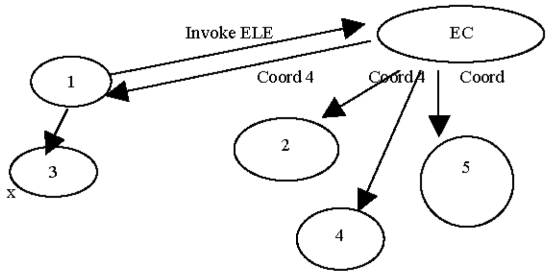

Fig. 9: Coordinator failure and re-election

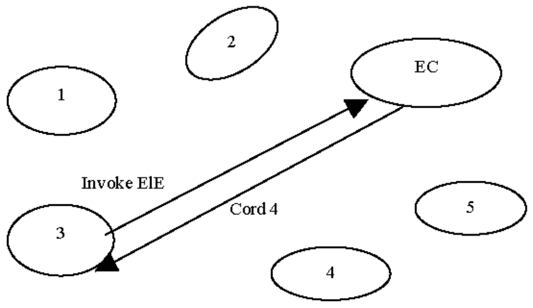

Fig. 10: Crashed coordinatorr recovery

Crashed coordinator recovery: Figure 7, process 5 recovers from the crash. Process 5 bullies the current coordinator and asks Process4 to resign.

Normal process recovery: Figure 8, process 2 recovers from crash and sends election message to all higher priority processes. Process 4 sends back a reply message.

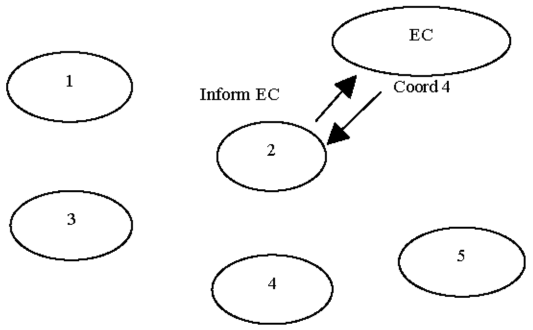

Fig. 11: Normal process recovery

Table 1: Performance analysis of bully algorithm

Case number	No. of messages
1	13
2	4
3	5

Implementation of the proposed algorithm

Coordinator failure and re-election: Figure 9 there are five processes with random priorities. The coordinator is initially process 3. Process 1 checks the status of the coordinator. As process 3 is no longer alive, election commission initiates the election. The new coordinator is process 4. EC informs all the processes that process 4 is the new coordinator.

Crashed coordinator recovery: Figure 10, the crashed coordinator process3 recovers and invokes the election. Election commission requests the recovered coordinator to wait till the current coordinator fails.

Normal process recovery: Figure 11 process 2 recovers from failure and invokes the election commissioner. The EC notifies the recovered process of the current coordinator.

Performance analysis: If there are n processes in the system and p is the process number which detects failuref coordinator, then. In original bully algorithm, there will be need 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 p1 is the highest process number, it requires messages. Total number of messages passed between the processes is more as shown in Table 1.

In the case of proposed algorithm there will be a need of 1 election message to inform EC, a verify message to ensure the failure of coordinator and say R1 is the alive process with highest priority then alive and reply message to find out the highest alive process, so a total of O (n) messages are needed between processes. If the process with lowest process number detects coordinator as failed

Table 2: Performance analysis of proposed algorithm

Case number	No of messages
1	8
2	2
3	2

there is no change in total message. In the worst case this algorithm needs to check process p1+1 to find out highest alive process. In this case message passing is required between processes. However, in best case, this 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 n1-1. In that case, this algorithm needs only 1+2+2+n1 messages. When p1 is the highest process number, then 1+2+n1 messages are required. The total number of messages needed for all the three cases is shown in Table 2.

CONCLUSION

Thus a modified leader election algorithm has been proposed to improve the availability of the system. The proposed algorithm has a better complexity O(n) when compared to the existing bully algorithm, O (n2). The performance of the proposed algorithm was measured in terms of the number of messages sent across the network. The proposed algorithm can be used to improve the availability of data in a distributed environment. The term highly-available refers that the system can continue providing services even when coordinator fails.

ACKNOWLEDGEMENTS

The researchers convey their heartfelt thanks to Dr. R. Rudramoorthy, Principal, PSG College of Technology and Dr. R. Venkatesan, Professor and Head, Department of computer Science and Engineering, PSG College of Technology. This work is performed in the grid/cloud computing lab at PSG College of Technology.

REFERENCES

Aguilera, M.K., G.C. Delporte, H. Fauconnier and S. Toueg, 2001. Stable leader election. Proceedings of the International Symposium on Distributed Computing, October 3-5, 2001, Springer Berlin Heidelberg, Berlin, Germany, ISBN: 978-3-540-42605-9,-pp: 108.

Basu, S., 2011. An efficient approach of election algorithm in distributed systems. Indian J. Comput. Sci. Eng. (IJCSE.), 2: 16-21.

Chandra, T.D. and S. Toueg, 1996. Unreliable failure detectors for reliable distributed systems. J. ACM., 43: 225-267.

Dolev, S., A. Israeli and S. Moran, 1991. Uniform Dynamic Self-Stabilizing Leader Election. In: Distributed Algorithms. Sam, T., G.S. Paul and K. Lefteris (Eds.). Springer Berlin Heidelberg, Berlin, Germany, ISBN: 978-3-540-55236-9, pp: 167-180.

Garcia,M.H., 1982. Electronics in a distributed computer system. IEEE. Trans. Comput., 31: 48-59.

Park, S.H. Y. Kim and J.S. Hwang, 1999. An efficient algorithm for leader-election in synchronous distributed systems. Proceedings of the 10th Conference on IEEE Region (TENCON 99), December 15-17, 1999, IEEE, New York, USA., ISBN: 0-7803-5739-6, pp: 1091-1094.

Rahman, M.M. and A. Nahar, 2010. Modified bully algorithm using election commission. Masaum J. Comput. (MJC.), 1: 439-446.

Sinha, P.K., 2008. Distributed Operating Systems Concepts and Design. Prentice-Hall of India Private Limited, Delhi, India,.

Tanenbaum, A.S., 2007. Distributed Operating System. Pearson Education, Upper Saddle River, New Jersey,.