International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (02), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

Resource based Bully Leader Election Algorithm

Nisha Yadav[1], Dr. Amit Sharma[2]

[1]*M.Tech Scholar, Computer Science & Engineering Department,*
Vedant College of Engineering and Technology, Bundi, Rajasthan (India)
[2]*Professor, Computer Science & Engineering Department,*
Vedant College of Engineering and Technology, Bundi, Rajasthan (India)

ABSTRACT

Leader election method may be adopted according to topology and other application requirements, very less attention has been paid towards resource based election. Modern distributed applications like web services, WSN (Wireless Sensor networks) etc require the leader processing node to have certain characteristics. Electing a leader based on the properties and resources of the processor is research focus of this work. This paper proposes a resource based leader election algorithm in two variants: Interval based and event triggered bully leader election algorithm. Dual criteria are used to decide which processor will be elected as leader. Resource factor and is used to decide the Leader. ~equal within a range technique is used for comparison. Conflicts are resolved using process Ids. Proposal is compared against traditional bully leader election algorithm. Simulation results show that proposed algorithm is a practically successful bully algorithm.

Keywords: ***Leader election algorithm, Bully Algorithm, Resources, Distributed Systems, Resource Factor.***

I. INTRODUCTION

This Computing technology is in complete shift from individual systems to shared and distributed systems. Collaboration means more resources and hence heavy computing tasks become easier. The field of research related to distributed algorithms and synchronization among them is an ever growing one.

1.1 Distributed systems

Distributed systems are a collection of various independent machines or computers which appears as a single well organized system. The processors or computers in distributed system are autonomous. User must think that he is using a single system that means the components of system must communicate and collaborate in a very smooth and fast way. There is no rule regarding the type of computers used in a distributed system. In a single system, computers from a high performance mainframe computer to a single node of sensor network can be used.

There are many criteria over which distributed systems can be classified. One of them is Time. Distributed systems are of two types based on timing model.

Asynchronous Distributed systems [1]: a system is called asynchronous distributed system if there is no fixed limit or duration on how long it takes for a message to be delivered or how much time elapses between the

executions two consecutive processes of the system. There is no global time. Each processor performs tasks independent of time of another processor/node. There is no coordination among nodes. For eg: Internet.

Synchronous Distributed systems: In synchronous distributed systems there is a global clock and all the processors or nodes match their physical/Logical clocks according to global time from time to time. Coordination among the nodes of the distributed systems is set by delay of messages sending. There is a fixed upper limit for the message to receive. If that limit is crossed the message is discarded.

1.2 Distributed algorithms

Distributed algorithms are made to execute on distributed systems .i.e. the systems having multiple processors which are interconnected [2]. Distributed algorithms are a sub-type of parallel algorithm they executes concurrently, with separate parts of the algorithm being run simultaneously on different independent processors, and having limited information about what the other parts of the algorithm are doing. For ex: Consensus algorithms: Consensus algorithms try to solve the problem of many processes agreeing on a common decision.

Leader election is one of the most critical tasks in distributed systems. Also it is necessary since in synchronization of processes usually requires a process to act as a leader and a leader is not the same every time because it can crash or failed due to some technical problem. That is why there is a need to make a mechanism for electing a new leader to take the position of the old one.

1.3 Approaches for leader election

Depending on the topology of the system (inter-connection of nodes) or event-detection capabilities, the leader election methods can be different. Though the categories are not all-inclusive or exhaustive, yet the broad classes of leader lection mechanisms are:

1. **Ring based leader election algorithms** – The communication topology of the nodes is assumed to be a ring. From the perspective of theoretical analysis and predicting behaviour of a leader election method, the ring topology is most convenient. Hence, literature mostly contains such methods.
2. **Bully based leader election algorithms** – These approaches work under event-detection settings. Whenever the leader node fails, the node that detects the failure can bully over others by assuming its ID or triggering a re-election to check if there exists a competition.
3. **Best Node based leader election algorithms** – These techniques decide the candidature of nodes or processes to become a leader based on certain criteria combined into a function. The winner is the node that has best value of such criteria function.

1.4 Problem Statement

We are proposing a Bully algorithm with best node based approach. Leader election algorithm with best node based has not been proposed so far in literature.

An issue with best node based approach is that the criteria required to select the best node is application specific. Hence our objective is to design a generalized best node bully approach for leader election in distributed systems.

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (02), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

II. BACKGROUND

The first bully leader election algorithm is proposed by Garcia Molina [3] in 1982 which is a very popular election algorithm. Author used timeout mechanism to keep record of coordinator/leader failure detection. Three types of messages: election message (inquiry generated by node which holds election), answer message (ok, acknowledgement of election message by nodes that received election message) and a coordinator message (victory, announcement of leader message generated by elected leader node) are communicated during the process of leader election. In 2005, Kordafshari et al[7] used optimal message algorithm to overcome the drawback of molina's algorithm. This algorithm is limited to non-redundant election.

In 2004, Mamun et al[10] again modified bully leader election algorithm to overcome the problem of redundancy of election of the coordinator and to minimize the recovery problem of the crashed process. This algorithm efficiently used the message communication to eliminate the redundancy of election procedure. In 1987, Frederickson[9] presented a leader election algorithm in synchronous unidirectional ring. This algorithm work under the assumption that a unique integer id is assigned to each processor is integer and the communication is in counter-clockwise direction. This algorithm focuses on two main thoughts: One is message processes generated on different processors have different transmission rates (delay) and a fast message process if overtakes a slower message process then slower message process is killed. Second is that each message processor has a preliminary phase in which each message process is travelled at same transmission rate.

Rahman[4] introduces the concept of election commission to modify the bully election algorithm. Rahman discussed the bully algorithm and its different modified versions proposed before. The drawbacks of these modified versions are reduced at an extent by the approach of election commission. In this scheme the efficiency of election commission is shown by the comparisons between different modified versions of bully algorithm and election algorithm used bully algorithm. Biswas et al[5] in 2017, proposed a ring leader election algorithm for distributed systems which is based on resource factor. In this algorithm, available resources of a node in a distributed system are taken into consideration for leader election. Each node calculates its resource factor value. A Process priority status queue (PPSQ) is maintained and passes in the message to each node in the ring. The node/process having higher RF value is at the top of the PPSQ. Soundarabaiet al[6] proposed a modified version of ring election algorithm called message efficient ring election algorithm (MEREA). Their main focus was on fault tolerance since node failure and process crash are very common in a distributed system. It has low time and cost complexity as compared to traditional ring algorithms. Thus, provides better performance.

Mohammadreza et al[8] proposed modifies version of bully and ring leader election algorithm as well a heap tree mechanism for leader election. The heap tree mechanism is based on max-heap data structure. Their modified bully algorithm is based on a sort mechanism for the minimization of the message passed with fault tolerance capabilities. Its disadvantage is that the leader election process is slow due to sort mechanism.

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (02), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

III. PROPOSED METHOD

This research work proposes a leader election technique in two variants. It is basically a best node based approach. The criteria to decide which node is best are combined together into single function. First section describes this idea. Thereafter, the proposed techniques are described.

3.1 Computation of resource factor

We propose to have a generalized leader election algorithm that can suit a variety of practical applications. Since many applications have a distributed system where individual processors are of different capabilities, it is impractical to treat them equally. Hence, we suggest introducing a Resource factor that would serve as a criterion to decide the best node. Every application or distributed system may differ in the priority of the resources or the no. of the resources important for the application. Hence we propose a general formula where the resource factor is a weighted linear combination of the various factors being considered by the application. Let denotes the factor and denote its weight, then the cumulative Resource factor is computed as:

$$R = \sum_{i=1}^{m} \alpha_i f_i \qquad (1)$$

Where, = weight of factor having value 0-1

= number of factors to be considered

The values of decide the relative priority of the Resource factor, i.e. if $\alpha_i >$ for any two resources and it implies that resource is more important than resource .

To be more explicit, we list down here some of the possible factors:

1. Processing Power
2. Memory Capacity
3. Shared Resource
4. Residual Energy

Also, we include the Resource Factor to be 0 when the application has to treat every node as equal (Homogeneous systems).Since, the resources may change over time i.e. the value of may change. The collision may occur at any point of time and leader may change whether a failure event occur or not.

3.2 Criteria to select best node

For a given set of nodes the node with the highest RF is considered to be the best candidate to be the leader. In case of collisions i.e., equal values of resource factor of two or more nodes, then the node with higher ID is considered to be the winner. This criterion can be formulated as:

$$(x_1, y_1) \prec (x_2, y_2) \equiv (x_1 <_c x_2) \vee ((x_1 = x_2) \wedge (y_1 < y_2))$$

Where every processing node is characterized by a pair $(x,$ where denotes the resource factor and is the process id. Since the resource factor is a numerical quantity an order can be established to compare two values. We propose the comparison to be

$$x_1 <_c x_2 \Rightarrow (x2 - x1) > \delta \qquad (3)$$

That is two numeric values of resource factor are always comparable to each other if within delta quantity of each other. If the difference is greater than delta then the larger value is considered as larger in the lexical context of our protocol. The threshold quantity delta is decided according to the factors and application .This quantity is used for regulation purposes so that minor drop or increase in resource factor does not affect the process of leader election.

3.3 Proposed Leader election algorithm

In applications where a leader has more responsibilities than other nodes it is undesirable to have a leader with fewer resources. Generally leader election is done based on process IDs, electing the node with highest id as leader. Here the process ids are randomly generated values having no correspondence to the resources available at the node. Thus, it is very likely that a node with very less resources may get elected as leader. We propose to use resource factor as a criteria for leader election. The basic difference or challenge while considering resource factor instead of process id is that process id is fixed for a process while the value of resource factor may change. So the challenge here is to include the changing factors within the protocol. Our proposed method can be simply outlined as follows:

1. Each process has a pair associated with it containing its process id and value of resource factor.
2. All processes in the system will become eventually aware of the best node. The best node is decided as discussed in section 3.2.
3. In case the leader node fails election mechanism is repeated only among the nodes which have higher or comparable resource factor than the node that detects failure.

3.4 Analysis of proposed technique

1. The number of messages exchanged among processes/nodes for electing the leader should be ideally low. The message complexity of our proposed technique is listed in Table 3.1. The best case of the proposed technique is that the node that detects failure has the highest value of resource factor among remaining nodes. The margin by which its factor is high is such that no other node is candidate for re-election. Hence, only a broadcast message from winner to all remaining nodes is sent. This is of $O(n)$. The worst case is when the node that detects failure has lowest value of resource factor. In this case, entire election process is to be repeated. One round of election requires exchange of message between all pairs of nodes, amounting to message complexityof $O(n^2)$.
2. For the iterative variant of the proposed scheme, the message complexity per round is same as the one round of election. That is $O(n^2)$ as the messages are exchanged between all pairs of "alive" nodes.
3. Thus, proposed modifications to the bully style leader election are to adjust resource-based requirements of an application does not incur any extra message overhead. While it may be possible that the resources are so distributed among the nodes that the average case is very much near to the best case. The number of candidates that takes part in election after failure events are much less than the number of "alive" nodes.

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (02), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

IV. EXPERIMENT AND RESULTS

4.1 Experimental Setup

For this simulation we used CPU power, memory and energy as resources for the calculation of resource factor and $\alpha_1 = 0.4, \alpha_2 = 0.2 \text{ and } \alpha_3 =$ as their resource weightage respectively.

Resource factor is calculated as

$$R = 0.4 * cpupower + 0.2 * memory + 0.7 * energy \quad (4)$$

for each node. Number of nodes taken into consideration is 10, 15, 20 and 25 during simulation.

We have taken CPU resource value in the range of 0 to 1, memory resource value in the range of 1-100 and energy resource value in the range of 10-100.

We have proposed two algorithms.

1. Interval based RF bully algorithm
2. Event Trigger based RF bully algorithm

These are compared with traditional interval based and trigger based bully algorithms respectively. Memory and energy are being depleted with time. While the CPU power remains constant.

4.2 Comparison of interval based algorithms

Interval based bully election algorithms means that the election will occur after a particular time interval whether the leader is crashed or not. For simulation purpose we have set leader election for both traditional bully election algorithm and proposed RF based bully algorithm at the same time interval/same iteration so that we can observe and compare the election process and its result and number of message communicated more effectively.

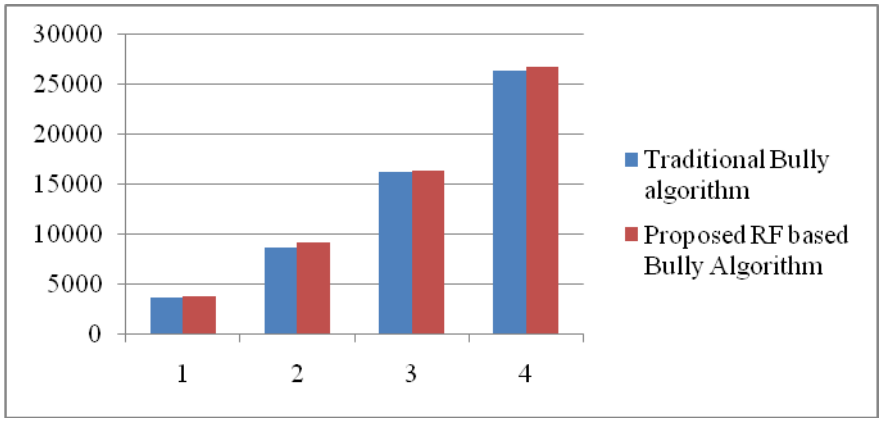

Fig 1 Comparison of traditional interval based bully algorithm and proposed RF interval based bully algorithm

Practically, the number of messages getting exchanged among the nodes is recorded in the experiments. The values are taken as average of 50 runs. The numbers of messages in the proposed algorithm are more than the traditional one. Fig 4.1 shows that the message comparatively for various simulations of both the algorithms.

The increase is not more than 10% in any case. This slight increase can be explained as the number of nodes participating in the election (alive) at any iteration is more than that in traditional algorithm. Due to more alive nodes, the messages are more. Fig 4.2 shows the number of dead nodes in the simulation of both the algorithms. It is clear that the proposed algorithm has lesser number of dead nodes (crashed nodes).

International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No. (02), January 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

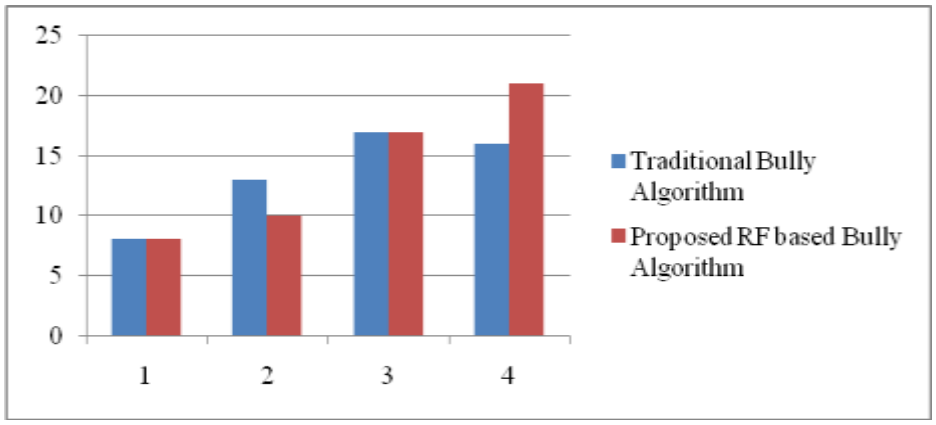

Fig 2 Comparison of number of dead nodes in traditional bully algorithms and proposed RF based bully algorithms

4.3Event triggered based algorithms

Event trigger based bully election algorithm means that the election will occur only when the leader node/process is crashed. For simulation purpose we have crashed both traditional bully election algorithm and proposed RF based bully algorithm at the same time/same iteration so that we can observe and compare the election process and its result more effectively.

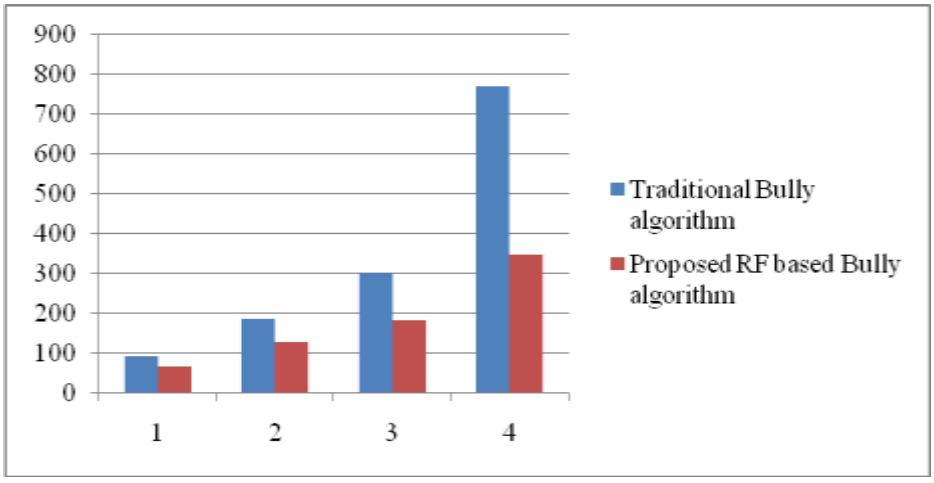

Fig3 Comparison Message exchanged in traditional event trigger based bully algorithm and proposed RF interval based bully algorithm

The number of messages getting exchanged among the nodes is recorded in the experiments. The values are taken as average of 50 runs/executions. The numbers of messages in the proposed algorithm are less than the traditional one. Fig 4.3 shows that the message comparatively for various simulations of both the algorithms. As the number of nodes involved in the experiment increase the no. of messages increase exponentially as expected in both the algorithms. But the number of messages for the proposed algorithm is almost 50% less than the traditional bully algorithm. The explanation for such good performance is that whenever the leader node fails the re-election takes place among the nodes with higher id than the node that detects failure in the traditional bully algorithm. Hence on an average the number of candidates for re-election can be many. While in the

proposed algorithm every node maintains a queue of resource factor and process ids. In this case the candidates for re-election are much less irrespective of the node that detects failure.

4.4Interpretations of results

The proposed algorithms perform better in following situations:

1. For applications that are energy critical like sensor networks or resource critical like web services.
2. Applications where the resources available at processing nodes are highly dynamic in nature.
3. Applications where the elected leader node has certain designated functions to perform which require large amount of resources.

The aspects in which the proposed algorithm is better than others are

1. The numbers of messages exchanged among the nodes for election purpose are reduced to half for event triggered settings. This is due to reduced number of nodes eligible to participate in election when the leader crashes.
2. The added information of resource factor and its changing values impose only a 10% increase in the message overhead in the interval based settings.

V. CONCLUSION AND FUTURE SCOPE

Modern computing environments are distributed in nature and leader election is one of the most needed fundamental algorithms for distributed systems. Though there have been several leader election algorithms proposed in literature and many of them are practically accepted; very less attention has been paid to the criteria for electing a leader process basically leader election is performed under the assumption of equality of all processes yet there are applications which have inherently different characteristics of the processing nodes. Moreover these characteristics are crucial for the application itself. We have proposed a leader election algorithm that considers the resources of the processors as a criterion for leader election. A generalized method to compute resource factor according to the requirements of the application is proposed. It is a weighted linear combination of the values associated with resources of a node. A comparison technique called equal-within-a-range is suggested to use while comparing resources of two nodes. The node with the highest resource factor is elected as leader and the conflicts are resolved using the process ids.

Two variants are proposed in this report the interval based variant performs re-election after regular intervals as it considers the system to be dynamic with changing resources. The second variant is for event triggered systems where a re-election is performed only when the current leader node is failed or crashes. Simulation experiments are performed to compare with traditional leader election algorithms. The interval based variant shows 10% increase in message complexity but the numbers of failed nodes are less as compared to traditional leader election technique. The event triggered variant shows as much as 50% decrease in message complexity as compared to traditional bully leader election algorithm. Thus our proposal is effective for resource critical applications in distributed systems.

Theoretical and empirical analysis of the proposed algorithm for various inter-connection topologies can be performed to observe its applicability in general. Practical application of the proposed leader election like

wireless sensor networks or cloud computing can be designed to observe the gain factor against classical leader election. Our proposed work has assumed independent resource availability and no interdependence or resource sharing has been considered. For the distributed systems where processes have shared resources the computation of resource factor needs to be adapted.

REFERENCES

[1] B. Awerbuch, "Complexity of network synchronization". *Journal of the ACM*, vol.-32, issue-4, pp. 804-823, 1985.

[2] Lynch, and Nancy, "Distributed Algorithms", *ISBN 978-1-55860-348-6*, Morgan Kaufmann Publishers, 1996.

[3] H. Garcia-Molina, "Elections in Distributed Computing System". *IEEE Transaction Computer*, vol.C-31, pp.48-59, 1982.

[4] M. M. Rahman and A. Nahar, "Modified Bully Algorithm using Election Commission", 2010.

[5] T. Biswas, A. K. Ray, P. Kuila, and S. Ray, "Resource Factor-based Leader Election for Ring Networks", 2017.

[6] P. B. Soundarabai, J. Thriveni,K. R. Venugopal, and L. M. Patnaik,"An Improved Leader Election Algorithm for Distributed Systems", *IJNGN*, vol.-5, 2013.

[7] M. S. Kordafshari,M. Gholipour, M. Jahanshahi,and A.T. Haghighat,"Modified Bully Election Algorithm in Distributed Systems".

[8] M. Effatparvar, N. Yazdani, M. Effatparvar,A. Dadlani, and A. Khonsari,"Improved Algorithms for Leader Election in Distributed Systems",*IEEE Conference*, 2010.

[9] Fredrickson, and N. Lynch,"Electing a leader in a synchronous ring",*Journal of ACM*,vol.-34, pp. 98-115,1987.

[10] Q.E.K. Mamun, S.M. Masum, M.A.R. Mustafa. "Modified Bully Algorithm for Electing Coordinator in Distributed Systems". In *Proceedings of the 3rd WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, Salzburg, Austria, 2004.