International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 11, November - 2013

Amelioration on Coordinator Election Algorithm in synchronous distributed system

Anamika Datley (Soni) [1], Brajesh Patel [2], Shailesh Soni [3]

[1,3] Deptt. of Master of Computer Application, SRIT, Jabalpur, M.P., India
[2] Deptt. of Computer Science , SRIT, Jabalpur, M.P., India

Abstract-An important challenge confronted in distributed systems is the adoption of suitable and efficient algorithms for coordinator election. The main role of an elected coordinator is to manage the use of a shared resource in an optimal manner. Among all the algorithms reported in the literature, the Bully and Ring algorithms have gained more popularity. This paper, presents an enhancement of the bully algorithm requiring less time complexity and minimum message passing. This algorithm also maintains fault tolerant behaviour of the system**.**

Keywords: Coordinator, Synchronous Distributed System, Process node , Campaigner set, General set

1. Introduction

In a distributed system, each process must operate accurately to cooperate with other process. There are mechanisms such as file server, time server, and central lock coordinator in the distributed system. In generally, these servers are called coordinators. But there are problems that one of them is the failure of process node on networks. Algorithms which select a coordinator are called the coordinator election algorithms. In these algorithms, when the coordinator is crashed, other process must elect another coordinator. Many algorithms have been presented for electing coordinator in distributed systems on networks such as Bully and Ring .

Coordinator election is a technique that can be used to break the symmetry of distributed systems In order to determine a central controlling process in a distributed system, a process is elected from the group of processes as the coordinator to serve as the centralized controller for that decentralized system .

The main focal point of an efficient and robust election algorithm is to minimize the number of messages generated for the election procedure and to reduce the time complexity of the overall execution time. Based on analysis, this paper shows that Garcia-Molina's bully algorithm [1] can be modified to enhance it's performance in message passing and subsequently demand less time complexity that results in electing a new coordinator faster.

2.Features of a leader election algorithm

Coordinator election is a procedure that is embedded in every process of the distributed system. Any process which detects the failure of the leader can initiate a leadership election. The election concludes its operation when a coordinator is elected and all the processes are aware of the new coordinator.

In leader election algorithm following assumptions are made:
1) Each process must have a unique id .
2) Each process knows the id of the others.
3) Process are not aware of the current state of other processes.

Along with above, the following requirements must be met:

1.SAFETY: All the live processes must agree on the elected coordinator. Each process must contains a two local variables:

a) cdr denotes the current leader of the system.
b) State which represents the current state of the process. It may be normal, elect or hold state. When process is in normal operation and there is an active leader in the system then the state is normal. State elect means the system is in the process of electing a new leader. State hold is used when a process actively takes part in the election and is waiting for the result.

1) **LIVENESS**: after completion of the election all the live process and the elected must come into a position where all of them are in state normal .

3. Our proposed algorithm

Here we draw some specifications from Garcia-Molina's bully algorithm [1] and new approach algorithm[6] and then proposes some modifications. As it has been mentioned, the number of messages exchanged between process in Bully algorithm [1] is very high. It was modified to present a new approach based on a sort mechanism to reduce the number of messages [6]. This new approach algorithm[6] may, however, consume more time with regard to the actual Bully algorithm[1] in finding and electing the leader. In this section, we introduce another approach to facilitate the algorithm with fault tolerant capabilities.

IJERT

Analysing the shortcomings of these algorithms ,here we are proposing a set partitioning. All processes of the process group are partitioned into two sets: campaigner set and the general set. Campaigner set consist of [N/2] processes , where N is the number of processes in the system. The others will be in the general set such that id of every process of campaigner set is higher than that of every process of General set. The number of process in each set are fixed now and no reshuffle is needed for the time being as it would add extra overhead to the system.

Our proposed election algorithm is capable of handling some exceptional situations that could arise in a synchronous distributed system and are described as below:

IDEAL CASE

When a process from the general set identifies that the coordinator is crashed, it sends an election message to all the processes of the Campaigner set and waits for a particular time. When it receives response from all other processes ,its easy to decide which is the next higher priority process among the live processes. Now the electioneer process sends a coordinator message to all the processes of the system about the new coordinator. If a process from Campaigner set detects the crash of coordinator then it sends an election messages to its higher process nodes of its set only, waits for a particular time for reply messages and then broadcast the coordinator message to all processes.

CAMPAIGNER FAILURE CASE

When no process from the Campaigner set responds to the detector process of the general set within a particular time the detector process comes to know that no process in the Campaigner set is alive. So it sends an election message to the higher id processes of the general set only and waits for the response and after that it broadcasts a coordinator message to all the live process nodes. Here in this case the coordinator is from the general set.

CRASHED COORDINATOR REVIVAL CASE

When a previous crashed coordinator recovers it simply broadcasts (n-1) messages, where n is the total number of process to the (n-1) processes of the process group about its revival and declares itself as the new coordinator of the group.

4. Example of proposed algorithm

Let us assume:

The total number of processes in the system are N=12

Campaigner set have higher (N/2) processes i.e. 12, 11,10, 9,8,7,

General set have lower (N/2) processes i.e. `1, 2,3,4,5,6

Current coordinator is: process 12

Now each process knows its set, other processes set and the current coordinator id.

This election process is done in three steps:

A) Election initialization

Currently the coordinator process 12 is crashed and process 2 had detected the failure(Fig 1.1).process 2 sends election message to Campaigner set processes i.e. 12,11,10,9,8,7 as shown in fig.1.2 below

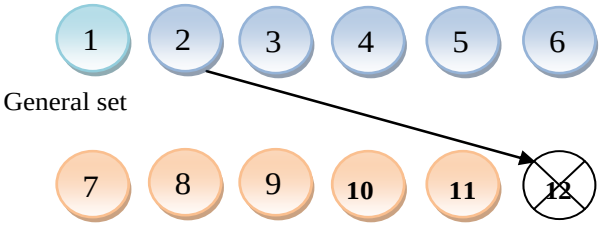


Figure 1.1

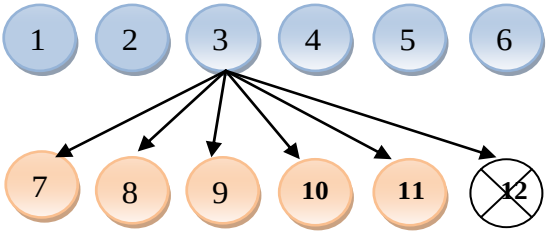


Figure 1.2

B) Response message

The processes of Campaigner set sends the reply message to the detector process 2 as in fig. 2 below

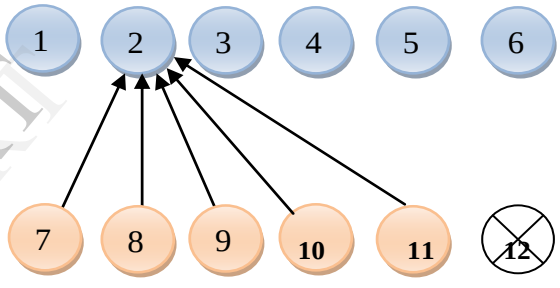
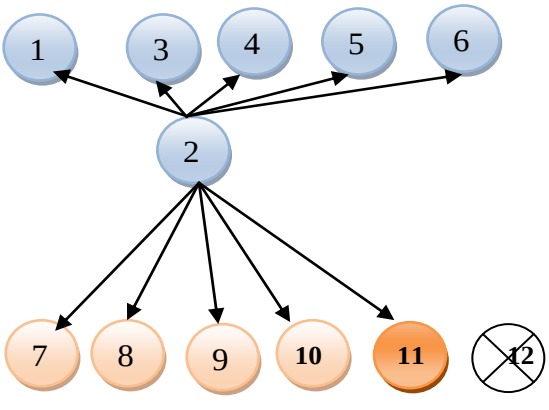


Figure 2

C) Coordinator message

Process 2 now knows the highest id process among all live processes hence process 2 declares the highest process id i.e. process 11 as the new coordinator to all the process nodes of the system as depicted in fig. 3



IJERT

5. Performance analysis

Based on message generation in the system, a comparative analysis of the Garcia-Molina bully algorithm [1], new approach bully algorithm [6] and our proposed algorithm would be appropriate to determine which algorithm performs better than the others. We had analysed these three algorithms in three cases as best case, worst case and the coordinator revival case. Let us assume that the Total processes in the system are N.

- BEST CASE: The best case happens when the process having the next highest id number detects the failure of the coordinator and initiates the election.
  - A) In bully algorithm [1] (N-1) messages are required to elect the new coordinator.
  - B) In new approach bully algorithm [6] also (N-1) messages are required to elect the new coordinator.
  - C) In our proposed algorithm also (N-1) messages are required to elect the new coordinator.

The time complexity in the best case for all the three algorithms is O (N).
Hence there is no improvement found in best case.

- **WORST CASE:** The worst case happens when the process having the lowest id number detects the failure of the coordinator and initiates the election.
  - A) In bully algorithm [1] n^2 messages are required to elect a new coordinator.
  - B) In new approach bully algorithm [6] 2 N messages are required to elect a new coordinator.
  - **C)** In our proposed algorithm (2N-3) messages are required to elect a new coordinator**.**

The time complexity in new approach bully algorithm [6] is less than from bully algorithm [1] and complexity of our proposed is very much less than above two.
Hence , there is a drastic improvement in worst case in our proposed algorithm.

- **COORDINATOR REVIVAL CASE:** This is the case when a previous crashed coordinator recovers back to join the group.
  - A) In Garcia-Molina bully algorithm [1] (N-1) messages are required to elect a new coordinator.
  - B) In new approach bully algorithm [6] (N-1) messages are required to elect a new coordinator.
  - C) In our proposed algorithm (N-1) messages are required to elect a new coordinator.

  The time complexity in all the above algorithms is O(N) Hence, no improvement is obtained in coordinator revival case.

6)Results and Discussion

Table 1. For N=5

Case	Bully[1]	M.Bully[6]	My Algo
Best	4	4	4
Worst	20	11	9
NRC	4	4	4

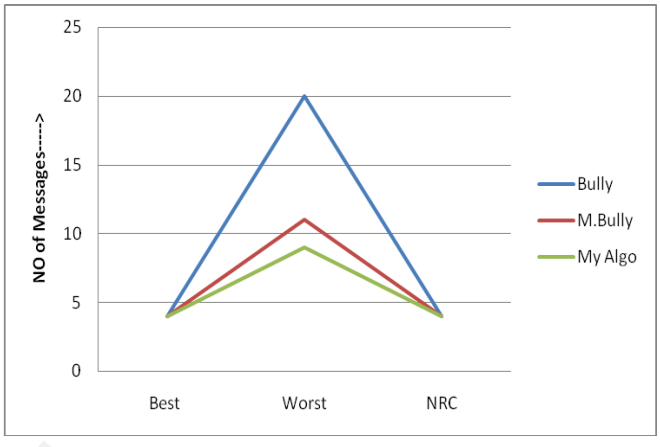
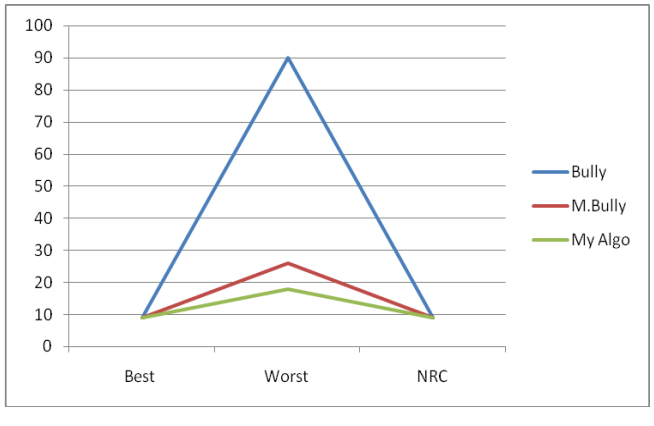


Table 2 For process =10

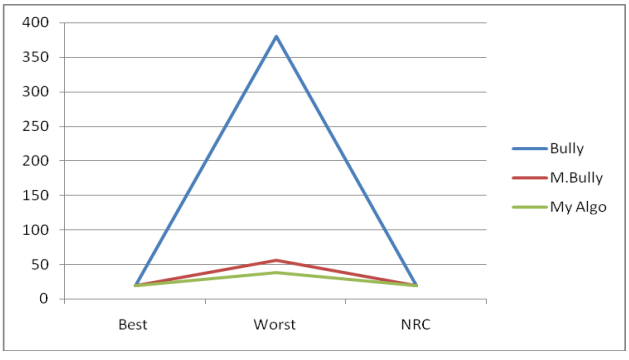
Case	Bully [1]	M.Bully[6]	My Algo
Best	9	9	9
Worst	90	26	18
NRC	9	9	9



IJERT

Table 3 For process =20

Case	Bully [1]	M.Bully[6]	My Algo
Best	19	19	19
Worst	380	56	38
NRC	19	19	19



8)Referances

[1] H. Garcia Molina, "Elections in a Distributed Computing System." *IEEE Trans*. Comp, 1982, vol.31, no. 1, pp.48-59.

[2] N. Fredrickson and N. Lynch, "Electing a Leader in a Synchronous Ring." *J.ACM*, 1987, vol.34, no.1, pp.98-115.

[3] S.Singh and J.Kurose, "Electing 'Good' Leaders." *J.Par.Distr.Comput*, 1994, vol. 21, pp.184-201.

[4] N. Fredrickson and N. Lynch, "Electing a Leader in a Synchronous Ring." *J.ACM*, 1987, vol.34, no.1, pp.98-115.

[5] Le Lann, G., "Distributed Systems – Towards a Formal Approach", in Information Processing 77, B. Gilchrist, Ed. Amsterdam, The Netherlands: North-Holland, pp. 155-160, 1977.

[6] M.gholipour,M.F kodafshari for A New Approach For Election Algorithm in Distributed System
2009 Second International Conference

[7] Sung-Hoon-Park,"A Probabilistically Correct Election Protocol in Asynchronous Distributed System ", APPT, LNCS 2834, pp.177-183, 2003

[8] Kim, J. L. and Belford, Geneva G., "A robust, distributed election protocol", Proc. of seventh IEEE Computer Soc. Symp. Reliable Distributed Systems, pp. 54-60, Columbus, Ohio, Oct. 1988.

[9] Kim, W., "AUDITOR: A Framework for high availability of DB/DC systems", IBM Res. Rep. RJ3512, 1982.

[10] S.Park, Y.Kim and J.S.Hwang, "An Efficient Algorithm for Leader-Election in Synchronous Distributed Systems," 1999 IEEE TENCON.

7) Conclusion and future scope

This paper presents some modifications to the classical bully algorithm which overcome the limitations of this algorithm, and make it efficient and fast to elect a leader in synchronous distributed systems. The performance of the proposed algorithm has been compared with the original bully algorithm and its modification and our proposal produces a better outcome. The algorithm is fast and guarantees correctness and robustness and the results shows that it requires fewer messages to elect a new coordinator.

There is also a scope to propose an algorithm for an asynchronous system and this will be a future work.

IJERT