

Study on Election Algorithm in Distributed System

Hetal Katwala¹, Prof. Sanjay Shah²

¹(PG Student of Computer Engineering, Merchant Engineering College, Gujarat, India)

² (Professor, Department of Computer Science and Engineering, Government Engineering College, Gandhinagar, Gujarat, India)

Abstract: In distributed system, electing a leader for the various coordination activities is an important issue. This paper present different election algorithm with different approach. The coordinating activities can be a directory search, balancing the load of the distributed system etc. The major coordinating activity is to manage the use of a shared resource in an optimal manner. The goal of a leader election in distributed system of autonomous processes is to select one of the currently alive processes as a leader so as to manage the coordination activities of the other processes in the system. This paper proposes a comparative analysis of the various election algorithms in distributed system.

Keywords: Election, Coordinator, Priority, Status table, Resource characteristics

I. Introduction

In recent years, distributed systems are growing rapidly. Therefore, managing and controlling these systems becomes a challenging issue.

A distributed system is a collection of processors interconnected by a communication network in which each processor has its own local memory and other peripherals and the communication between them is held by message passing over the communication network [1].

Distributed algorithms require that there be a leader process in the entire system that performs some type of coordination activity needed for the smooth running of other processes in the system. As the processes in the system need to interact with the leader process, they all must know who the current leader is.

The important case to be dealt is when the leader fails, so there is a requirement for a temporary new leader to take the job of coordination. Hence an election is done to find out the next leader in the system.

Election algorithms are based on the following assumptions:

1. Each process in the system has a unique priority number.
2. Whenever an election is held, the process having the highest priority number among the currently active process is elected as the coordinator.
3. On recovery, a failed process can take appropriate actions to rejoin the set of active processes.

II. Existing Algorithms

Many algorithms have been proposed for electing leaders in distributed systems

1. Bully algorithm proposed by Hector Garcia Molina in 1982.
2. Ring algorithm proposed by Silberschatz and Galvin in 1994.
3. Sandipan Basu(2010)
4. Dynamic Election Strategy in Distributed System

2.1 Hector Garcia-Molina, (1982; also known as bully algorithm) [2]

2.1.1 Assumptions

Every node in the system has a unique priority number.

- Every node in the system knows the priority of the other nodes.
- Whenever an election is held. The node having the highest priority number among the currently live nodes is elected as the coordinator.
- On recovery, a failed process can take appropriate actions to rejoin the set of active processes.

2.1.2 Algorithm

When a node (say n1) sends a request message to the coordinator and does not receive a reply within a fixed timeout period, it assumes that the coordinator has failed. It then initiates an election by sending an election message to every other node with a higher priority number than itself. If node n1 does not receive any response to its election message within a fixed timeout period, it assumes that among the currently active nodes it has the highest priority number. Therefore it takes the job of the coordinator and sends a broadcast

message: coordinator message to all the nodes in the system that has lower priority than itself declaring that it is the new coordinator. If n1 receives a response for its election message, this means that there are nodes live that have higher priority than itself, so n1 does not take any action and waits to receive the final result of the election.

When a node n2 receives an election message from a node with lower priority than it, it sends a response message: alive message to the sender informing that it is alive and will take over the election activity. Now n2 holds an election if it is not already holding one. In this way, the election activity moves on to the nodes that has the highest priority number among the currently active processes and eventually wins the election and becomes the new coordinator.

A failed node n must initiate an election after a recovery. If the current coordinators priority number is higher than the node n then the current coordinator will win the election initiated by node n. On the other hand, if n's priority is higher than the current coordinator, it will not receive any response for its election message. So it wins the election and takes over the coordinator's job from the currently active coordinator. Therefore, the active process having the highest priority number always wins the election. Hence the algorithm is called the "bully" algorithm.

2.1.3 Operation

Initially there are 6 alive nodes in the system and node 6 with the highest priority is the coordinator. But node 6 has crashed which is realized by node 2, so it sends an election message to nodes 3,4,5,6 with higher priority than node 2.

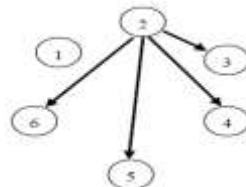


Fig 1.1

As node 6 has crashed, so node 2 receives OK message only from nodes 3, 4, 5 and discovers that there are nodes which are live with higher priority than itself.

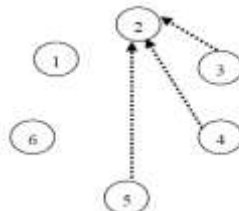


Fig 1.2

Now node 3 sends election message to nodes 4, 5, 6. Similarly, node 5 and 5 sends message to nodes with higher priority than theirs.

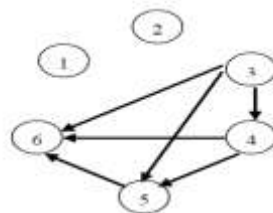


Fig 1.3

Nodes 4, 5 sends OK message to node 3 and 3,4 respectively. Node 5 discovers that among the currently live nodes, it has the highest priority.

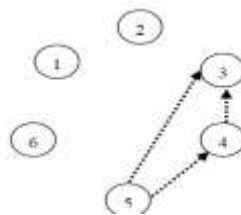


Fig 1.4

Node 5 broadcasts coordinator message to all the nodes.

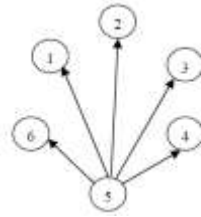


Fig 1.5

2.2 Silberschatz And Galvin (1994; known as Ring algorithm)

2.2.1 Assumptions

1. All the nodes in the system are organized as a logical ring.
2. The ring is unidirectional in the nodes so that all the messages related to election algorithm are always passed only in one direction.

2.2.2 Algorithm

While the message circulates over the ring, if the successor of the sender nodes is down the sender can skip over successor, or the one after that until an active member is located.

When a node n_1 sends a request message to the current coordinator and does not receive a reply within a fixed timeout period, it assumes that the coordinator has crashed. So it initiates an election by sending an election message to its successor. This message contains the priority of node n_1 . On receiving the election message, the successor appends its own priority number to the message and passes it on to the next active member in the ring.

In this manner, the election message circulates over the ring from one active node to another and eventually returns back to node n_1 . Node n_1 recognizes the message as its own election message by seeing that in the list of priority numbers held within the message the first priority number is its own.

Among this list, it elects the node with the highest priority as the new coordinator and then circulates a coordinator message over the ring to inform the other active nodes. When the coordinator message comes back to node n_1 , it is removed by node n_1 .

When a node n_2 recovers after failure, it creates an inquiry message and sends it to its successor. The message contains the identity of node n_2 . If the successor is not the current coordinator it simply forwards the enquiry message to its own successor. In this way, the inquiry message moves forward along the ring until it reaches the current coordinator. On receiving the inquiry message, the current coordinator sends a reply to node n_2 informing that it is the current coordinator.

2.3 Sandipan Basu Algorithm [3]

2.3.1 Assumptions

The following assumptions are made for this algorithm:-

- (1) All nodes in the system are assigned a unique identification numbers from 1 to N .
- (2) All the nodes in the system are fully connected.
- (3) On recovery, a failed process can take appropriate actions to rejoin with the set of active processes.
- (4) When a process wants some service from the coordinator, the coordinator is bound to response within the fixed time out period; besides its other tasks.
- (5) We assume that a failure cannot cause a node to deviate from its algorithm and behave in an unpredictable manner.
- (6) Lamport's concept of logical clock is used in distributed system that we are considering.

2.3.2 Algorithm

When a process (say) P_i sends a message (any request) to the coordinator and does not receive a response within a fixed timeout period, it assumes that the coordinator has somehow failed. Process P_i refers to its process status table, to see who is process having the second highest priority number. It then initiates an election, by sending an *ELECTION* message to the process (say) P_j , having priority just below the failed Coordinator; i.e. process with the second highest priority number.

2.3.2.1 Case 1

When P_j receives an election message (from P_i), in reply, P_j sends a response message *OK* to the sender, informing that it is alive and ready to be the new coordinator. Therefore, P_j will send a message *COORDINATOR* to all other live processes (having priority less than P_j) in the system. Hence, P_i starts its execution from the point where it was stopped.

Number of messages in this case = $2 + (n-1)$

2.3.2.2 Case 2

If P_i does not receive any response to its election message, within a fixed timeout period; it assumes that process P_j also has somehow failed. Therefore, process P_i sends the election message to the process (say, P_k) having the priority just below the process P_j . This process continues, until P_i receives any confirmation message *OK* from any of the process having higher priority than P_i . It may be the case that, eventually P_i has to take the charge of the coordinator. In that case, P_i will send the *COORDINATOR* message to all other processes having lower priority than P_i .

2.3.2.3 Case 3

Consider process P_m recovers from its failed state. Immediately, it sends a *REQUEST* message to any of its live neighbors. The purpose of the *REQUEST* message is to get the process status table from its neighbor. So, as soon as any of P_m 's live neighbors receives a *REQUEST* message, it sends a copy of the current process status table to P_m . After receiving the process status table, P_m checks whether its own priority number is less than the process having the highest priority (i.e. current coordinator's priority) or not.

Number of messages in this case = 2

2.3.2.3.1 Case1

If the current coordinator's priority is higher than P_m 's priority, in that case, P_m will send its priority number and an *UPDATE* messages to all other processes in the system, to tell them to update P_m 's status (from *CRASHED* to *NORMAL*) in their own process status table.

Number of messages in this case = $(n-1)$

2.3.2.3.2 Case 2

If P_m 's priority is higher than the current coordinator's priority; then P_m will be the new coordinator and update the process status table and sends the *COORDINATOR* message to all other processes in the system, and takes over the coordinator's job from the currently active coordinator.

Number of messages in this case = $(n-1)$

So the efficiency of the algorithm in any case is $O(n)$

2.4 A Dynamic Election Strategy In Distributed System [4]

This paper presents an election strategy that is based on the resources currently available on the machine where the processes are running.

The strategy is to find out a process which is running on a machine with the richest set of resource characteristics available at the time of the election. The resource characteristics of a machine may include the level of security provided by the machine, number of processors in a multiprocessor system, speed of each processor, available RAM on the machine, etc. Thus in addition to reducing the messaging overhead among the processes, the resource characteristics is an important criteria in electing a process as a leader.

The main goals of the proposed algorithm strategy are:

- 1) To achieve a greater overall improvement in system performance at a reasonable cost.
- (2) To utilize the available resources most efficiently.
- (3) To have the ability to modify the system itself in accordance with any changes.
- (4) To find an optimum leader whose machine is enriched with the best resource characteristics.

2.4.1 Resource Factor of a machine

Resource factor (RF)
 $= r_0 * \alpha_0 + r_1 * \alpha_1 + \dots + r_n * \alpha_n \dots \dots \dots (i)$

Thus for every such j machine out of m machines of the system, the resource factor RF for each machine j is :
 for $j=0$ to m , $i = 0$ to n ,

$RF_j = \sum_i [(r_i * \alpha_i)] \dots \dots \dots (ii)$

Resource factor (RF)
 $= r_0 * \alpha_0 + r_1 * \alpha_1 + \dots + r_n * \alpha_n$

The resource characteristic with the highest weighted value is considered to be of highest priority and then with the decreasing weighted values, decreases the priorities.

2.4.2 Dynamic election strategy

All the resources along with their resource characteristics are registered with a GIS, which keeps track of all the machines. As and when the processes are created, they are sent for execution according to the time shared system under consideration.

Now, among the currently running processes, one process is designated as a coordinator for a special role to coordinate the processes of the system. The selection of this coordinator process is important as it has an important role of the working system.

Consider that a process P_i is the coordinator at time t_1 , $0 \leq i \leq k$, out of total k processes. At a later time t_2 , a process P_j $0 \leq j \leq k$ sends a REQUEST message to the coordinator process P_i . But P_j does not receive a reply from coordinator P_i till P_j 's timeout period.

At this point, process P_j discovers that the coordinator P_i has failed due to some reason and it is the time to elect a new coordinator. Thus, the proposed algorithm is executed which runs as follows:

The resource characteristics of each and every resources in the system are fetched from the GIS along with the current status of every process running on the which PE. The resource factor of all the machines is calculated. The machines with the highest resource factor say m_k is chosen. If the only process running on the machine m_k is process P_i which has just crashed, then the machine with 2nd highest resource factor is chosen, otherwise one of the processes except process P_i is chosen on machine m_k .

For multiple processes running on the chosen machine, the process's CPU time requirement is calculated and the one with the least CPU is elected as the new coordinator. This message is broadcasted to all the other processes by process P_j about the new status. Thus the process running on the richest set of resource characteristics and the least CPU time requirement is chosen as the new leader.

Table 1. Comparison in term of message passing when election initiates

Algorithm	Best case	Worst case	In Recovery
Bully Algo.	$O(n)$	$O(n^2)$	Best case- $O(n^2)$ Worst case- $(n-1)$
Ring Algo.	$2(n-1)$	$2(n-1)$	$n/2$
Sandipan Basu Algo.	$(n-1)$	2	2

In Bully algorithm, when the process having the lowest priority number detects the coordinator's failure and initiates an election, in a system of n processes, altogether $(n-2)$ elections are performed. All the processes except the active process with the highest priority number and the coordinator process that has just failed perform elections. So in the worst case, the bully algorithm requires $O(n^2)$ messages. When the process having the priority number just below the failed coordinator detects failure of coordinator, it immediately elects itself as the coordinator and sends $n-2$ coordinator messages. So in the best case, it has $O(n)$ messages.

During recovery, a failed process must initiate an election in recovery. So once again, Bully algorithm requires $O(n^2)$ messages in the worst case, and $(n-1)$ messages in the best case.

In ring algorithm, on the contrary, irrespective of which process detects the failure of coordinator and initiates an election, an election always requires $2(n-1)$ messages. $(n-1)$ messages needed for one round rotation of the ELECTION message, and another $(n-1)$ messages for the COORDINATOR message. The algorithm proposed by Sandipan Basu has $O(n)$ message efficiency.

During recovery, a failed process does not initiate an election on recovery, but just searches for the current coordinator. So ring algorithm only requires $n/2$ messages on average during recovery.

In the algorithm proposed by Sandipan Basu, the number of ELECTION messages made when the coordinator fails is 2 in the worst case. And it

requires $(n-1)$ coordinator messages. In the best case, it requires only $(n-2)$ coordinator messages as there is no need to make any ELECTION message.

During recovery, in the best and the worst case, a failed process requires 2 ELECTION message are required to know the current coordinator and $(n-1)$ messages to send its own priority to other nodes. So in all, $2 + (n-1)$ messages are requires. Thus it requires $O(n)$ messages.

In dynamic election strategy in order to elect a leader when a leader of a coordinator fails. The main goal of this strategy is to utilize the resources of the distributed system in its most efficient way. Thus the

process running on a machine that has the best resources available at the time of election will be designated as a new leader. This process can then coordinates and manage the activities in the network with its rich set of resources.

III. Conclusion

The paper makes an analysis of different election algorithms and discussed efficiency in terms of number of messages exchanged in each case. In dynamic strategy for choose coordinator consider the highest resources factor and cpu time.

References

- [1] Sinha P.K, Distributed Operating Systems Concepts and Design, Prentice-Hall of India private Limited, 2008.
- [2] Tanenbaum A.S Distributed Operating System, Pearson Education, 2007.
- [2] H.Garcial Molina "Election in a distributed Computing System". IEEE Trans. Comp, 1982, vol31, no. 1, pp. 48-59. 1982
- [3] An Efficient Approach of Election Algorithm in Distributed Systems" - Sandipan Basu / Indian Journal of Computer Science and Engineering (IJCSE), Vol 20, No 1, 2010
- [4] Fredrickson and Lynch, Fredrickson, and Lynch, "Electing a Leader in a synchronous Ring", journal of the ACM, Vol 34, pp 98-115, 1987.
- [5] "A Dynamic Election Strategy in Distributed System" - Prof. Sanjay Shah & Heta Jasmin Jhaveri/ International Journal of Engineering Research and Technology (IJERT), Vol. 1 Issue 3, May - 2012
- [6] GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing – Rajkumar Buyya and Manzur Murshed