Volume 5, Issue 10, October-2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Analysis and Comparison of Different Election Algorithm

Swati Patel, Girish Tere
Dept. of Computer Science, Thakur College of Science and Commerce
Maharashtra, India

Abstract- This paper proposes a comparative analysis of the various available election algorithms in distributed system on the basis of their structure, assumptions and its complexity. It also discusses limitations of different algorithms for coordinator selection. The comparison is done on the basis of number of messages passed and time complexity parameters of algorithms.

Keywords- Election, Coordinator, priority.

I. INTRODUCTION

A Distributed system is an application that executes a collection of protocols to coordinate the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks and helps entire application to run on heterogeneous devices. Leader Election is an important and major issue in Distributed system. Various algorithms are available to cope with this issue. Most two well known algorithms are Bully Algorithm and Ring Algorithm. Among this Bully Algorithm has gain more popularity to elect a leader. Election Algorithm provides a technique to pick a coordinator which will take the responsibility of the system. If the coordinator get fails then the system needs to elect a new leader by initializing a leader election through various algorithm strategies.

II. DIFFERENT ELECTION ALGORITHMS

A. Bully Algorithm by Garcia Molina

Assumptions:

- The system is synchronous and uses timeout for identifying process failure.
- Allows processes to crash during execution of algorithm.
- Message delivery between processes should be reliable.
- Prior information about other process id's must be known.

Algorithm:

When a process P determines that the current coordinator is down because of message timeouts or failure of the coordinator to initiate a handshake, it performs the following sequence of actions:

1. P broadcasts an election message (inquiry) to all other processes with higher process IDs, expecting an "I am alive" response from them if they are alive.
2. If P hears from no process with a higher process ID than it, it wins the election and broadcasts victory.
3. If P hears from a process with a higher ID, P waits a certain amount of time for any process with a higher ID to broadcast itself as the leader. If it does not receive this message in time, it re-broadcasts the election message.
4. If P gets an election message (inquiry) from another process with a lower ID it sends an "I am alive" message back and starts new elections.

Consider 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.

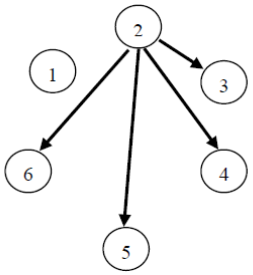

Fig. 1. Bully Algorithm by Garcia Molina

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.

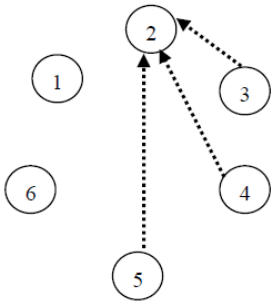

Fig. 2. Bully Algorithm by Garcia Molina

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.

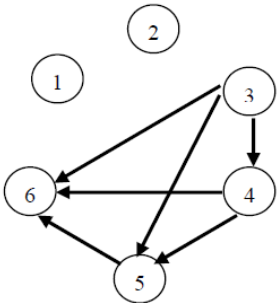

Fig. 3. Bully Algorithm by Garcia Molina

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.

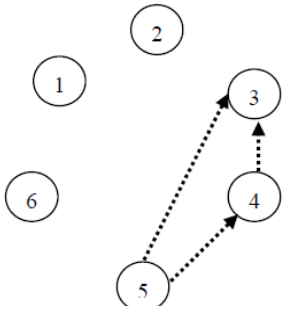

Fig. 4. Bully Algorithm by Garcia Molina

Node 5 broadcasts coordinator message to all the nodes.

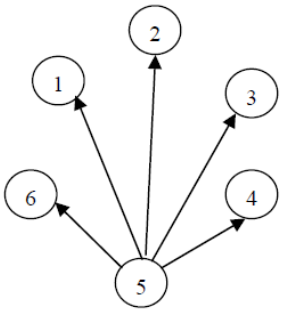

Fig. 5. Bully Algorithm by Garcia Molina

Limitations:

1. 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.
2. When any node 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.
3. As there is no guarantee on message delivery, two nodes may declare themselves as a coordinator at the same time. Say, P initiates an Election and didn't get any reply message from Q, where Q has a higher node number than P. At that case, p will announce itself as a coordinator and as well as Q will also initiate new election and declare itself as a coordinator if there is no node having higher node number than Q.

4. Again, if the coordinator is running unusually slowly (say system is not working properly for some reasons) or the link between a node and a coordinator is broken for some reasons, any other node 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 node P with lower node number than the current coordinator, crashes and recovers again, it will initiate an election from current state.

B. Ring algorithm by Silberschatz and Galvin

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.

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. Algorithm using the following sequence of actions:

When a node n1 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 n1. 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 n1. Node n1 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 n1, it is removed by node n1.

When a node n2 recovers after failure, it creates an inquiry message and sends it to its successor. The message contains the identity of node n2. 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 n2 informing that it is the current coordinator.

Example:

- Two processes, 2 and 5, discover simultaneously that the previous coordinator, process 7, has crashed.
- Each of these builds an ELECTION message and starts circulating it.
- Eventually, both messages will go all the way around, and both 2 and 5 will convert them into COORDINATOR messages, with exactly the same members and in the same order.
- When both have gone around again, both will be removed.
- It does no harm to have extra messages circulating –at most it wastes a little bandwidth.

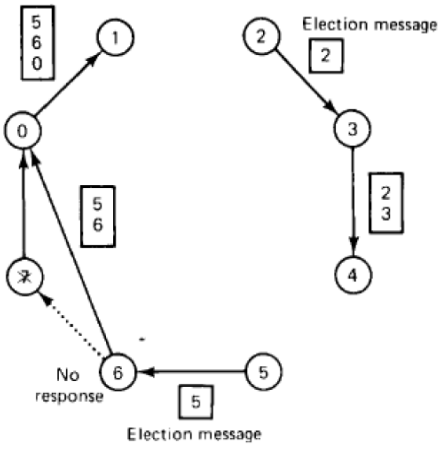

Fig. 6. Ring algorithm by Silberschatz and Galvin

C. Sandipan Basu Algorithm

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.

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.

Case 1

When Pj receives an election message (from Pi), in reply, Pj sends a response message OK to the sender, informing that it is alive and ready to be the new coordinator. Therefore, Pj will send a message COORDINATOR to all other live processes (having priority less than Pj) in the system. Hence, Pi starts its execution from the point where it was stopped. Number of messages in this case = 2 + (n-1)

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**.

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

1. 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. 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 COORDIANTOR 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).

D. Modified Bully algorithm by Quazi Ehsanul Kabir Mamun

In this algorithm described an efficient version Bully algorithm to minimize redundancy in electing the coordinator and to reduce the recovery problem of a crashed process

Assumption:

There are five types of message. An election message is sent to announce an election, an ok message is sent in response to an election message, on recovery, a process sends a query message to the processes with process number higher than it to know who the new coordinator is, a process gets an answer message from any process numbered higher than it in response to a query message and a coordinator message is sent to announce the number of the elected process as the new coordinator.

Algorithm:

a) When a process p notices that coordinator is down, it sends an election message to all processes with higher number. If no response, p will be the new coordinator.
b) If p gets ok message, it will select the process with highest process number as coordinator and send a coordinator message to all process.
c) When a crashed process recovers, it sends query message to all process with higher process number than it.
d) And if it gets reply then it will know the coordinator and if it doesn't get any reply it will announce itself as a coordinator.

Limitations:

Although this algorithm reduces redundant election on some extent, it still has some redundant elections and also has high message complexity. Some of the limitations are given below:

1. On recovery, it sends query message to all processes with higher process number than it, and all of them will send answer message if they alive. Which increases total number of message passing and hence it increases network traffic
2. It doesn't give guarantee that any process p will receive only one election message from processes with lower process number. As a result there may be q different processes with lower process number can send election

essage to p and p will send ok message to all of them. This increases number of election and also number of message passing.

3. It doesn't give any idea if p will crash after sending an election message to all processes with higher process number.
4. It also doesn't give any idea if a process with the highest process number will crash after sending ok message to p.

III. COMPARISONS OF DIFFERENT ALGORITHMS

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(n2) 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(n2) 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. 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 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.

IV. CONCLUSION

In this paper, we have shown the comparison between various election algorithms in a distributed system. The comparison is done on the basis of number of messages passed and time complexity parameters of algorithms. Some algorithms overcome the overhead the sending of number of messages. This paper also discusses limitations of different algorithms for coordinator selection.

REFERENCES

[1] H Garcia-Molina. Elections in Distributed Computing System, IEEE Transaction on Computers,310:48-59, 1982.

[2] Heta Jasmin Javeri, Sanjay Shah, A Comparative Analysis of Election Algorithm in Distributed System IP multimedia communication, a special issue from IJCA ?????

[3] S. Mahdi Jameii, A Novel Coordinator Selection Algorithm in Distributed Systems‖ Department of Computer-Engineering, Islamic Azad University, Shahr-e-Qods Branch, Tehran, Iran IJAEST Vol No. 9, Issue No. 2, 310 - 313

[4] Quazi Ehsanul Kabir Mamun, Salahuddin Mohammad Masum, "Modified Bully algorithm for electing coordinator in distributed system‖ . CD proceedings of 3rd WSEAS international conference on software engg. , parallel & distributed system (SEPADS) 2004 feb 13-15, Australia

[5] Tanenbaum A.S, Distributed Operating System, Pearson Education, 2007.

[6] An Efficient Approach of Election Algorithm in Distributed Systems" - Sandipan Basu / Indian Journal of Computer Science and Engineering (IJCSE), Vol 20, No 1, 2010.

