A Choice Based Selection Algorithm for Elect a Destination System in Distributed Environment

B.krishna Sagar\textsuperscript{1}, E. Madhusudhana Reddy\textsuperscript{2} and S.Ramakrishna\textsuperscript{3},
\textsuperscript{1}Computer Science and Engineering, J.N.T.U.K, Kakinada, India, krishna.sagar521@gmail.com,
\textsuperscript{2}Computer Science & Engineering, DRK College of Engineering & Technology, Hyderabad, India, e_mreddy@yahoo.com,
\textsuperscript{3}Computer Science, Sri Venkateswara University, Tirupati, India, drsramakrishna@yahoo.com.

Abstract

In our daily life, elections play a significant role for growing our improvements. Election is a process to elect a one leader in a group of persons. The same approach is follows in distributed systems to identify a system acts as a coordinator. If we select a system as a leader that system is taking care about entire system is in consistent state. In the scenario of message passing, if destination system is unable to handle server messages, then immediately those messages are discarded. If the same situation is happened with many systems, which affects wastage of network and server resources. To avoid this scenario, we proposed a novel choice based selection algorithm to select an appropriate system for delivering a server’s message. In this algorithm, we are considered past history about individual peers. This data plays a key role to elect a trusted associate destination system to transfer actual message to a right system.
1 Introduction

Election is a very useful process and fundamental problem to elect a leader in distributed environment. The process involved several parameters to choose a good coordinator in a system to maintain consistent state. In a distributed system, the leader system should be coordinate many tasks. The huge amount of man/system hours is necessary to for elect a leader in distributed environment why because several network protocols are involved in this process for synchronization of system. If the synchronization fails, then entire system may come under inconsistent state. To avoid this inconsistent state, we require efficient coordinator for maintaining the complete system.

In a distributed computing system, a process is used to coordinate many tasks. It is not a problem which systems are doing the work, but there should be a coordinator that will work at any time. So, electing a coordinator or a leader is very essential problem in distributed environment and there are several algorithms that are used in this process. Leader, Ring and Bully algorithms are some of them. In a group of communicating protocols, the elect a new coordinator is essential when a leader is crashes or away from the group. In this scenario in distributed environment, elections are conduct in appropriate situations. Election algorithm has a variety of applications such as key distribution, routing coordination, sensor coordination, and general control. When nodes are mobile, topologies can change and nodes may dynamically join/leave a network. In such networks, election algorithm can play a major role frequently, making it a particularly critical component of system operation. In this paper, we are used this novel election algorithm for delivering messages in absentia of destination system by choosing a nearby system or friend system through its past communication history.
The traditional statement of the election problem is to finally elect a distinct coordinator from a set of nodes from many sources.

2 Background – Election Algorithms

Election algorithm is very essential and special purpose algorithm for selecting the coordinator system among N number of system. This coordinator system plays a major role in distributed environment to the consistency of entire system through synchronization. For example, in a system of client server, the mutual exclusion algorithm is preserved by the server process Ps, which is chosen from among the processes Pi where i=1,2,...,N that are the group of processes which would use the critical region. Election Algorithm is needed in this situation to choose the server process among the existing systems. Definitely all the systems must approve upon the coordinator system. If the coordinator system fails due to any reasons, then immediately the election should happen to choose a new coordinator system to take up the job of the failed coordinator.

Any process can initiate the election algorithm whenever it encounters that the failure of coordinator system. There can be situations that all N processes could call N concurrent elections. At any time, process Pi is one among the following two states, when the election happens: Participant refers to the process is directly or indirectly involved in election algorithm, Non-participant refers to the process in not engaged with the election algorithm currently. The goal of Election Algorithm is to choose and declare only one system as the coordinator even if all the systems are participating in the election process. And at the end of the election, all the processes should agree upon the new leader process without any confusion. Without loss of generality, the elected process should be the process with the largest process identifier. This may be any number representing the order / birth/ priority/ energy of the process. Each process has a variable called coordinator, which contains the process id of the current leader. When the process participates in the election, it sets this lead to NULL.

Any Election Algorithm should satisfy the following two properties [9].
1) Safety: Any process P, has coordinator = NULL if it is participating in the election, or its coordinator = P, where P is the highest identifier and it is alive at present.

2) Likeness: All the processes should agree on the chosen coordinator P after the election algorithm. That is, coordinator = PID Pi where i=1,2,...,N.

The Bully Election Algorithm, elects the coordinator system uniquely which satisfies the safety and likeness requirements. Depending on a network topology, many algorithms have been presented for electing coordinator in distributed environment. The numbers may be allocated in simply numerical order of the Ethernet address or some other numbers such as priority, the mere process id, etc. The Ring Election Algorithm is based on the ring topology with the processes ordered logically and each process knows its successor in a unidirectional way, either anti-clockwise or clockwise.

When any process notices that the coordinator system has crashed, it creates an election message is inserting its own identifier and sends the message to its successor. If the successor is also down, the message would skip that process and goes to the next process of the successor or to the next etc., till it reaches a process which is not dead, along the ring network. When the election message is received by any process, it adds its identifier to the list in the message. Like this, all the available processes in the ring would insert their respective identifier in the list. Finally, the election message comes back to the process which initiated the message and the process too would recognize that it only had initiated that message, by identifying its own identifier in the list.

The Election initiator process analyses and finds the highest identifier among the available processes, converts the election message into confirmation-message and removes all the identifiers from the list but the highest identifier. This confirmation-message message is circulated along the ring for one circulation to inform the running processes about who the new coordinator is. When this message is circulated once, it is discarded and the Election Algorithm ends here.
When the message comes back to the process that started it:

- The process sees its identifier in the list.
- It checks all the identifier and decides the coordinator (the process with the high test identifier).
- It changes the message type and enters the coordinator system in the message.
- Message is circulated again.
- When it comes back to the process that started it, and it gets discarded there.

Limitation of this Ring Election Algorithm: Multiple election messages may happen in parallel when more than one process detects the failure of the coordinator process. This creates a lot of overhead of creating and servicing each and every election message. This causes huge traffic and sometimes congestion in the network. In the best case, when a single system detects the crashed coordinator.

These messages are considered to determine its performance, definitely it will go down entire system performance when all the systems are sending a messages but destination systems are unable to handle those messages. To solve this type of problem and to improve effective usage of server resources and network utilization, we proposed a choice based election algorithm to elect a destination algorithm in a distributed environment.

3 Choice Based Selection Algorithm

In the previous sections, we have seen many elections algorithms and also we are discussed how and when we can use those algorithms. In the perspective of message passing, at the time of message receiving time if the destination system is unable to handle the sender’s messages, which affects those message may continually rotate in a network or discarded. If we see the performance the entire system, this situation is going down of performance of entire system and wastage of network resources. We proposed a choice based election algorithm for elect a destination system based on its past complete conversation among the peers. So that, through our algorithm it is very easy for choose very appropriate system for message passing.
System Assumptions
The assumptions of our algorithm are:
1. The complete system is under synchronization and it is reliable.
2. The all other nodes are knowing if anyone node/system is/are goes down or up.
3. Individual systems are identified by its unique identifier.
4. All systems are having information about all other systems unique identifier.
5. All the system are connected with each other via some network.
6. The coordinator knows complete history about individual system in their network.

In this proposed algorithm, we have n peer systems and the coordinator system has to identify suitable destination among its peer systems of actual destination system. We have totally n system and one system is acts as a coordinator in working. If any one of the system is receives a message from other system/network and actual destination system is unable to handle, the coordinator is coming under the picture. The coordinator applies an election algorithm to choose a best alternate destination system through its identifier and past communication history.

![Figure 1: System Architecture](image)

In this proposed architecture, if the sender identifies that the destination system is unable to handle sender’s message, immediately it is forwarded to its coordinator. The coordinator system is identifying right system among actual destination system peers and that information is passed to sender.
The sender is sends that message to suitable appropriate system. Later the intermediate system is forwarded the message to actual destination system on availability of system. The proposed architecture is supports more than one clusters also.

Here we are considered individual system identifier and history of communication. The algorithm identifies largest identifier and its history and construct multi priority queue with help of max heap. The max heap data structure will generate tree with root node is intermediate node to receive a sender’s message.

The max heap data structure is having heap structure property and heap order property. For our algorithm each individual system is having a history of communication with actual destination system. With this scenario we also included to heap data structure is “goodness” property.

Goodness: There exists a node \( \exists A \) such that \( \forall A \) nodes properly communicate with actual destination system.

Algorithm ChoicebasedElectionAlgorithm(S,D,C,m)
//S is a source System
//D is an actual destination system
//C is a coordinator system
//m is a message
// The array p[1..n] is having peer systems of destination system
{
    if (S is unavailable or S is unable to handle source message)
    {
        for i = 1 to n do
            //insert ith system maxheap
            insert-maxheap(i) 
            repeat
            {
                peer = delete-maxheap()
                if (max-history(peer))
                    Dispatch message to peer system
            return
    }until heap is empty
}  
else  
  Dispatch message to actual destination system  
}  

4 Conclusion  

Elections are very important in distributed environment. If any one of the destination system is unavailable or unable to handle source system messages, then the coordinator system applies proposed choice based election algorithm to find actual destination system of connected system. The connected system identifies through the proposed algorithm by coordinator system. This algorithm gives best results of effective utilization of network and server resources. In this algorithm, we are considered past history about individual peers. This data plays a key role to elect a trusted associate destination system to transfer actual message to a right system.

References  


