I.J. Information Technology and Computer Science, **2015, 03, 1-11**
Published Online February 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.03.01

A Distributed Fault Tolerance Global Coordinator Election Algorithm in Unreliable High Traffic Distributed Systems

Danial Rahdari
Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Iran
E-mail: d.rahdary@srbiau.ac.ir

Amir Masoud Rahmani, Niusha Aboutaleby
Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Iran
E-mail: rahmani@sr.iau.ac.ir; n.aboutaleby@gmail.com

Ali Sheidaei Karambasti
No15, Ghafari Alley, Mashahir St., Ghaem Maghame Farahani Ave., Tehran, Iran
E-mail: a.sheidaei@etadbir.com

Abstract **—** Distributed systems consist of several management sites which have different resource sharing levels. Resources can be shared among inner site and outer site processes at first and second level respectively. Global coordinator should exist in order to coordinate access to multi site's shared resources. Moreover; some other coordinators should manage access to inner site's shared resources so that exerting appropriate coordinator election algorithms in each level is crucial to achieve most efficient system. In this paper a hierarchical distributed election algorithm is proposed which eliminates single point of failure of election launcher. Meanwhile traffic is applied to network at different times and the number of election messages is extremely decreased as well which applies more efficiency especially in high traffic networks. A standby system between coordinators and their first alternative is considered to induct less wait time to processes which want to communicate with coordinator
.

Index Terms— Distributed Algorithm, Coordinator Election, Fault Tolerance, Cloud Computing, Hot Standby, Unreliability, Global Coordinator

I. Introduction

Processes and systems collaboration are growing more and more by emerging new technologies and concepts in the world of distributed systems which give communication the most crucial role of these systems.

It is clear that Processes communicate with each other through operating system and Transition layer [1]. The functions of some processes during execution procedure are dependent on the others state because of the same shared resources.

As an example if a process wants to access an exclusive shared resource, it should check over to ensure to be the only one which uses the critical region to fulfill mutual exclusion property. There are lots of challenges, few of which are listed below, that should be considered during this process.

- Processes should ask all the others for permission before accessing the shared resources. Therefore a significant number of messages should be passed.
- During permission asking process, the processes response isn't guaranteed. It usually happens because of their failure or 100% usage of system's processor. Therefore they cannot access the resource because of lack of permission.

The existence of central controlling process is necessary to handle these situations. Single point of failure is clearly the so that if it crashes, new election should be launched to set new process as coordinator.

Leader election algorithms which are used for electing coordinator and its alternatives are useful in many various areas. They are used in distributed systems for load balancing and keeping resource replicas consistency [2]. When a coordinator crashes, some wait time is applied to processes supposed to communicate to coordinator. This time is affected by the power of an election algorithm which goes more by higher speed and less bandwidth usage. Hence, more powerful algorithms apply less wait time and appreciate system performance.

Meanwhile, there should be a trade-off between the safety and power of an election algorithm in the system, because if we need more powerful algorithms, we must attend to safety less since it needs to exchange more and more messages between processors which cause the weakness of the algorithm. This problem is also known as election problem [3].

Coordinators play a huge role in spread areas such as video conferencing, multiplayer games, recognizing processor or computer failure for data transferring, load balancing, and many others which show the importance of the coordinators once more and convince the researchers to give more attention to it.

II. Related Work

This area welcomed wide range of algorithms with the passing of time. Algorithms differ in specifications such as network topologies, the kind of communications between processes, and also setting the name of the processes. Bully [4] and Ring [5] are two classic ones that are referred to in many papers. Bully algorithm whose network topology is used in this paper launches election when processes find coordinator crashed. In the first step of election, these processes send Election messages to the processes with an upper process number than themselves. Then when processes receive Election labeled message, they will respond by an OK message. However, if no process responds, the sender would introduce itself as the new coordinator to the system by sending a Coordinator message to them. If process P_2 replies the sender, P_2 will send another Election message in the system by using the previous procedure. These steps continue until no other process with an upper number than the sender process exists or any other OK messages from the upper number processes didn't receive to informer.

Author in [6] proposed a uniform self–stabilizing distributed algorithm which is able to apply to any network with unique IDs. The algorithm elects the process of least ID as coordinator and constructs a breadth first search (BFS) tree rooted at coordinator within O (n) rounds which n is the network's process number. This algorithm's contribution is based on stabilization and is completely different with ours. An algorithm based on star graph is proposed in [7] which uses tournament scheme based on the recursive structure of the star graph. A star graph S_n of dimension n is decomposed into n substars S_{n-1} of dimension n-1. Election in S_1 is quite simple because it just contains a single node, then election in S_2 will be done by elected coordinators in S_1. This process will be continued to elect the coordinator in S_n by the elected coordinators in S_{n-1}. The message passing complexity in each step is from O ($\sqrt{n}$), but the whole algorithm is from O (n). The algorithm is applied on star networks and elects coordinators recursively. Moreover, it isn't considered multi management sites, which means any coordinator in lower layers is just elected in order to elect coordinators in the upper ones.

A probabilistic election algorithm with average message complexity O (n) for anonymous, unidirectional asynchronous bounded expected delay network is presented in [8]. Every node is in one of the following states: idle, active, passive or leader. The idle state is the default one at the beginning. The algorithm passes messages among the nodes and will change the idle ones to passive or active. Coordinator will be the active node that initially created and sent message in the network. This algorithm considered that network is anonymous and has restriction on response delay that is in contrast with ours which processes have IDs and delay is also accepted. Author in [9] studied stabilization and fault-tolerant elections in systems with static crash failures. They considered stabilization in the form of self-stabilization and pseudo-stabilization, so they tried to have election algorithms with these types of characteristics. Five systems are assumed in their paper. The base one has arbitrary slow or loosely communication links and then appropriate election algorithms are proposed for each of them. Stabilization and fault tolerance are the main ideas of the paper that cause more messages to pass in order to achieve these advantages.

Park [3] proposed an algorithm based on unreliable crash detecting procedure by processes assumption. New election will be run when local failure detectors of all processors commit coordinator failure. In this algorithm, processors send Coordinator messages to the other when they realize coordinator has crashed. Then receivers check message crash information and their local failure detectors information to reject or accept the failure. Commit message which is consisted of new coordinator number will be sent by informer if it receives accept from all processes. However, unreliable failure detector is not considered in our paper.

Effatparvar [10] and Shirali [1] are proposed algorithms by modifying the Bully. In the first paper processors which receive Election message, send their numbers to informer. After that it compares all processes numbers and then coordinator will be chosen. Processes are informed about the elected coordinator by receiving Coordination message from informer in the next step. These methods improve the Bully algorithms since it reduced the number of messages which should pass for coordinator election. Our algorithm consists of two algorithms. The first one is based on Bully algorithm and passes fewer messages than [10] which will be proved by simulation. Mirakhorli [11] proposed an algorithm by considering k candidate for coordinating in order to prevent new global election. When a typical process such as N finds a coordinator crashed, it'll send Crash-Leader message to candide1. If candide1 is not available, messages will be sent to the other ones respectively. If one candidate remains up, it will check the number of the last coordinator and message crashed coordinator, and then N will be introduced as new coordinator by it if they are the same. Alleviating the number of exchanged messages and lunching new election algorithm prevention are the algorithm advantages. By our algorithm, any time a process finds a coordinator crashed, it informs the others about it and then a new election will be postponed until the failure of all the coordinator alternatives.

Effatparvar [12] proposed an algorithm based on Ring election. Algorithm's Message format has one section which denies coordinator crash fault tolerance. Election's message number is reduced dramatically when more than one process realize coordinator crash at the same time. But if during crash time and new coordinator identification, other processes find the crash out; election will be launched again, which is the fundamental difference with one of our algorithm in this case. Xie [13] algorithm is based on bidirectional ring network.

Processors make election message and then send it to their successor and predecessor simultaneously when they are finding coordinator crashed. The most important advantage of the algorithm is raising the speed of election in Our algorithm is a hierarchical coordinator election that in the first level an algorithm based on Bully is applied, which elects the site's coordinators. Then global coordinator will be chosen among the elected sites coordinators in the next level.

The rest of paper is organized as follow. Section 3 is about system's assumptions that this algorithm is based on. Section 4 is considered to express the system's message format. In section 5 an algorithm based on Bully is proposed and section 6 describes the fault tolerance coordinator election algorithm in bidirectional ring (FCEABR) algorithm. Our proposed algorithm is described In Section 7 and section 8 is dedicated to mathematical analyzing of the algorithm. The algorithm is simulated in section 9 and the convergence of final results is approved by Section 10. At the end, section 11 is devoted to paper conclusion.

III. System Administraton

Our algorithm is based on distributed networks with multi management site in it. Sites are not forced to have similar number of processes. The algorithm works in systems such as cloud, Grid, cluster, and other regular distribute environments that have shared resources in each site and among themselves. A typical network's topology is shown in Fig. 1.

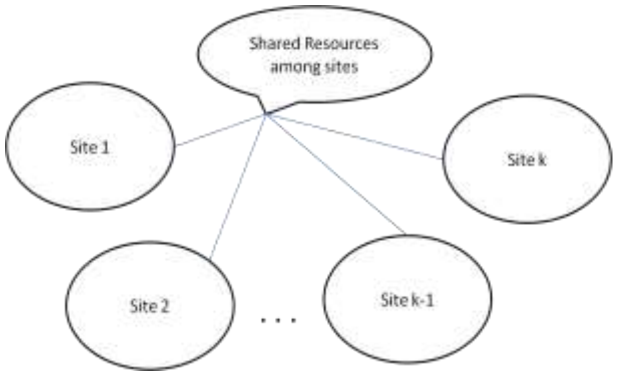


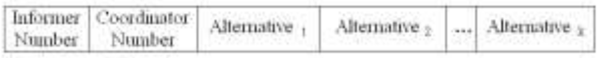
Fig. 1. Grid where there are k sites in the network

IV. Election Message's Format

The Message's format depends on the number of coordinator's alternative; therefore a different number in each site causes a different message's format in comparison to the others.

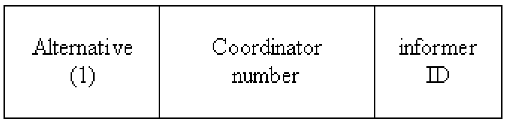
If K refers to the number of alternatives, the site's message format is determined in Fig. 2. By assigning 1 to K, the format will be the same as the format of FCEABR algorithm that is shown in Fig. 3. Many other formats are used by various types of algorithms in the area such as basic Ring algorithm's format. When processes receive Election message in this format, they'll add their number into it, and pass it to their successor. Hence, as the number of processes in a typical network increases, the size of message will be bigger and bigger. This format is shown in Fig. 4.

It should be noted that the other messages such as network controlling ones can pass easily because of message's label which causes difference among them. The first format is preferred in our algorithm because of advantages such as reducing the size of messages, inducting fault tolerance into system and being very practical in high traffic networks.



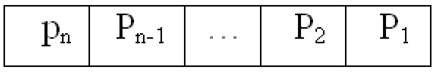
Informer Number	Coordinator Number	Alternative $_1$	Alternative $_2$	...	Alternative $_k$

Fig. 2. Sites Message format with K coordinator Alternative



Alternative (1)	Coordinator number	informer ID

Fig. 3. FCEABR message forma



p_n	P_{n-1}	. . .	P_2	P_1

Fig. 4. Basic Ring algorithm Format

V. FCEABOB Algorithm

A Fault tolerant coordinator election algorithm based on Bully algorithm (FCEABOB) is proposed in this section. According to inner site's topology, each process placed in the sites has full information about the other processes, so they can easily communicate (like the one Bully is based on). FCEABOB is applied to the inner site's election and has the following specifications.

- K coordinator alternatives $\{A_1, A_2, A_3, \dots, A_k\}$ are considered which replaced to coordinator respectively at any time the previous one crashed.
- T is denoted as the number of processes received Election messages and didn't reply back to it.
- The replying back Election message might not sent by processes or received to informer although processes are available. In this case the message will be sent to them once more to gain more powerful algorithm. The situations such as message loosing during network transition or 100% CPU usage in specified time and so on cause this case.

The algorithm elects coordinator and its k alternatives by these six steps following:

- First of all, the algorithm will be run when at least one process finds crash out.
- Then these processes send Election message to k (number of alternatives) processes with upper process number than themselves**.**
- After that, the available processes send their number to election informer processes.
- Next, the election message will be sent again to any processes which haven't replied (T ones totally). The messages also will be sent to the next T upper number

processes to place each of them respectively as alternative if any of those t processes do not response. It means that if $\{P_1, P_2, P_3, \ldots, P_t\}$ are formed processes which failed to response to the election procedure, the coordinator message will be sent to them and also to the next upper T number ones.

- The most upper number process is elected as coordinator with the next K upper ones as its alternatives.
- Finally, the informer propagates coordinator message in network to announce new coordinator and its k alternatives.

An example of algorithm and its pseudo code when there isn't any coordinator in network are shown in Fig. 5 and Fig. 6 respectively. The number of processes is eight and there are four coordinator alternatives. Process number 9 starts the election in this example. Four messages are sent to the four must upper number processes. In (Fig 5-b) all the processes replied back to 9 except process 12. Then in (Fig 5-c) process 9 sent messages to 5 and to 12 as well. In (Fig 5-d) process 5 replied back to 9, finally in (Fig 5-e) process 9 broadcast information in the network. It is clear by the pseudo code, if all processes that received election message respond to the informer, the election will be done; otherwise it will go to some other section and necessary functions will be done there.

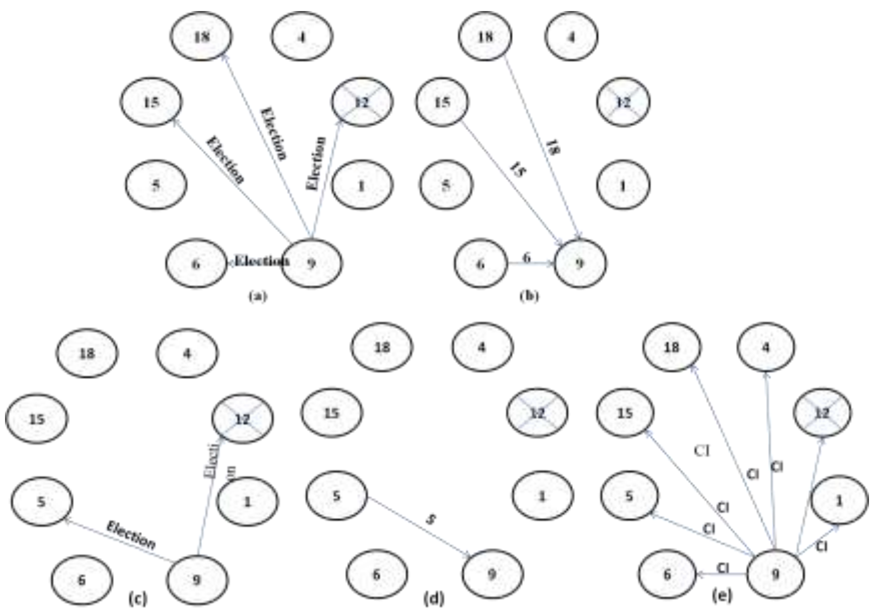
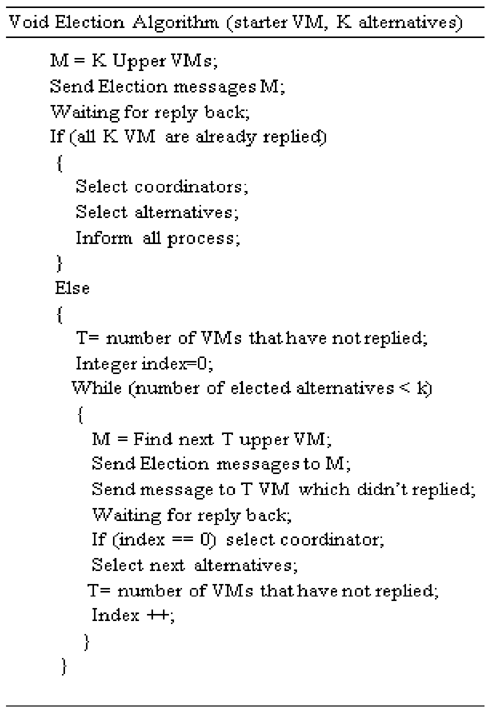


Fig. 5. FCBABOB algorithm example in a typical network



```
Void Election Algorithm (starter VM, K alternatives)
    M = K Upper VMs;
    Send Election messages M;
    Waiting for reply back;
    If (all K VM are already replied)
     {
        Select coordinators;
        Select alternatives;
        Inform all process;
     }
     Else
     {
        T= number of VMs that have not replied;
        Integer index=0;
       While (number of elected alternatives < k)
        {
          M = Find next T upper VM;
          Send Election messages to M;
          Send message to T VM which didn't replied;
          Waiting for reply back;
          If (index == 0) select coordinator;
          Select next alternatives;
         T= number of VMs that have not replied;
          Index ++;
        }
     }
```

Fig.6. FCBABOB pseudo code

VI. FCEABR ALGORITHM

Fault tolerance coordinator election algorithm in bidirectional Ring (FCEABR) is based on network's topologies in which each process just knows its successor and predecessor.

- In these networks, the processes are able to send messages just to their two sides (sides are really or logically existed). The algorithm was proposed by [14] and behaves according to following steps. First of all, Election message is created by a random selected process.
- Then selected process launches election by sending Election messages to its successor and predecessor simultaneously.
- Any time each process receives these messages, it checks their number to message's coordinator and its alternatives. If its number is upper than each of those items, the process will replace it with them and will send the message to their successor or predecessor according to its direction in the ring.
- At least one process such as P_c in the network will receive Election messages from its two sides. This

process, which has all the processes number in the network, will elect coordinator and its alternatives.

- Then P_c creates a coordinator message and sends it into the network in the same way which Election message was sent.
- Afterwards, anytime the coordinator crashes, the subrogate coordinator is replaced with it. Processes which realize crash create Selection message and propagate it to the network. Therefore, a new subrogate coordinator is identified.

Election's speed in the algorithm is more than simple Ring algorithm and number of messages that passed is fewer; especially when multi processes realize crash. However, a bit more processing time is needed.

This algorithm is modified and used in this paper by considering more than one coordinator alternative in the case, which causes change of message's format, the procedure of coordinator election and its alternative. We illustrate the algorithm by an example in Fig. 7. As we can see, the network has six processes and one alternative. In Fig. 7-a process 5 creates an Election message and sends it into the network, then process number 6 which received election messages with the same informer number from its two sides creates coordinator messages, and throws these messages into network from its sides as well.

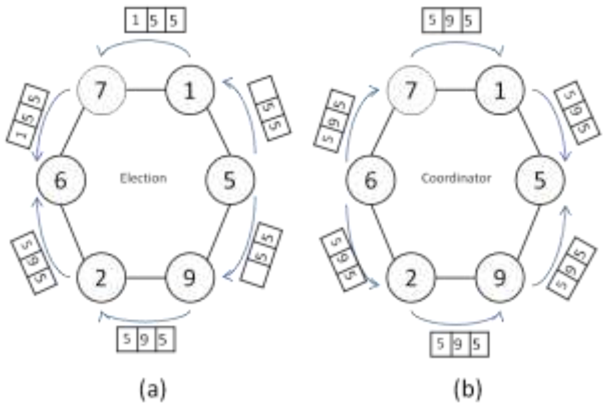


Fig.7. FCEABR algorithm in a typical network.

VII. Leader Election Algorithm

The algorithm is designed based on networks with multi management sites with no constraint on the number of processes in them. FCEABOB algorithm is applied to inner site election but the number of alternatives can be chosen by the site's administrator depending on specifications such as the site's traffic, reliability value, and required performance and so on. More alternative, that causes fewer election messages passing but on the other hand lower chance is applied to new processes that recently came into network to get the coordinator or its alternative role in no time, so there should be a tradeoff between these two characteristics. In each site there are two coordinators, internal and external ones, the most upper process numbers will be internal coordinator and the next one is considered as external coordinator.

- Internal Coordinator**.** This coordinator is used to coordinate accessing to the internal site's shared resources. In the rest of the paper we will refer to it by ICoordinator.
- External Coordinator**.** This type of coordinator is considered to coordinate accessing to shared resources among multi sites. It will be referred by ECoordinator

Typical site architecture is shown by Fig.8. Each Site's coordinator has a number of alternatives which are determined by the site's administrator. ICoordinator and ECoordiantor have connection with their first alternative which is considered as standby system to minimize its replacing time with the crashed coordinator. Hot, cold, and warm standbys form three types of standby systems and any of these can be chosen by the site's administrator but each of them has advantages and disadvantages and is used in appropriate situations

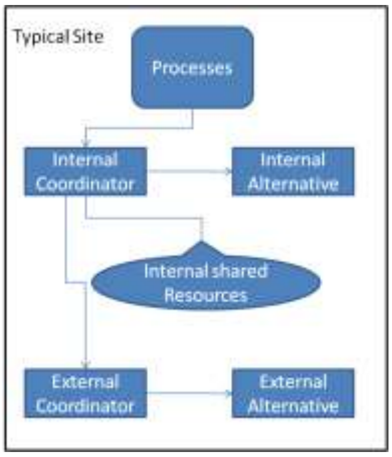


Fig.8. Typical Site Architecture

Hot standby systems are usually used in real time environments and increase the cost and probability of coordinator's alternative crash because it is always up and updated by each update of global coordinator but no wait time is applied to any process in the network to launch new global election. The advantages and disadvantages of cold standby are in contrast with hot standby.

When site's election is done, the ICoordinator, ECoordinator, and their alternatives are elected. Now it is the time to elect global coordinator in order to manage accessing to the multi site's shared resources. Virtual bidirectional ring network is set up among sites to elect Global Coordinator (GCoordinator) between them. For coordinator election in this virtual network we should apply one of the various election algorithms based on ring network topology. It is necessary to mention that GCoordinator is definitely elected from existing processes in sites and no special processes are considered for this role.

We elect coordinator and its alternatives in this network by FCEABR algorithm. There are n_1 processes (number of sites) and P alternatives for GCoordinator in the network ($1 \leq P \leq n_1$). Network architecture and the way that virtual bidirectional network is created are shown in Fig. 9.

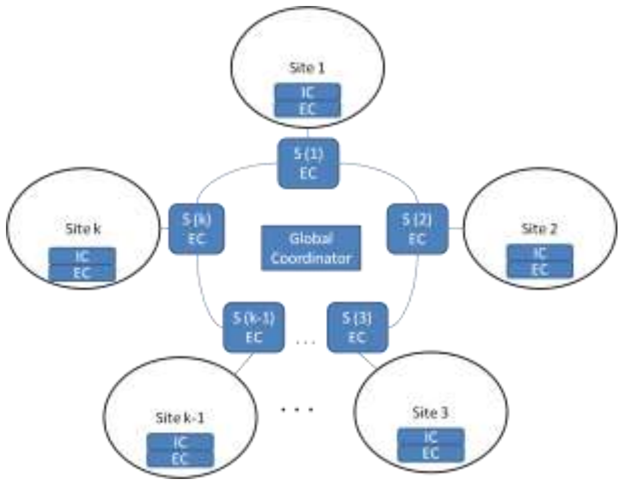


Fig.9. Network architecture. IC: internal coordinator, EC: External Coordinator, S(k) EC: External coordinator of site k. Each site has internal and external coordinator.

Any site's ECoordinator is placed in this network, so they just know about their successor and predecessor site's ECoordinator according to the bidirectional ring topology. We can set up this network by another topology which all processes in network are known by one another so if a new process wants to join a virtual network it should announce its entry to all other processes (such as Bully algorithm). If n_2 is denoted as the site's number of processes, n_2 messages should be exchanged when new a process wants to join a virtual network but in bidirectional ring network two message sending is enough. Hence, this topology is preferred in the virtual network because of the dynamic nature of heterogeneous distributed systems (resources can join or leave network by little management effort).

In this network all the time coordinator and alternatives exist but when all alternatives are crashed, a new election in the virtual network will be run. When the GCoordinator crashes, each process in the bidirectional ring network which finds it out will send a message to the site's ICoordinator of the crashed GCoordinator separately in order to inform it to broadcast Coordinator crash message to processes in its site. If the ICoordinator knows about coordinator crash, it will not pay attention to these messages in order to avoid the waste passing of messages in the network, but if it doesn't know, then it replaces the first external alternative to ECoordinator. With K alternatives for a typical site ICoordinator (K<n), when the first ICoordinator and K-2 internal alternatives crash, first of all new coordinator election will be run in the site because another crash causes processes which want to access some shared resource whether in the site or out of it to wait until new coordinators are elected. Then first external alternative is replaced to ECoordinator and in the similar way of inner site election, if no other alternative remains in the virtual network, a new coordinator election will be run by the processes which find crash out. When any site's ECoordinator crashes, its first external alternative will be replaced with it in the site and also in virtual network to send and manage accessing to the requests of outer site shared resources. The pseudo code of algorithm behavior to coordinator crash and how their alternatives replace with it is determined in Fig.10. Description of all used variables in equations is inserted in Table 1.

Table 1. Variable and Description

Symbol	Definition
VMN	Virtual Network Messages Number
FPN	Number of Processes which find coordinator Failed
NS	Number of Sites
SMN	Site's Message Number
$SCAN_i$	Site i Coordinator's Alternative number
SPN_j	Site j Process Number
EMN	Exchanged Message Number
CC_1	FCEABR Communication Cost
CC2	FCEABOB Communication Cost
PT_1	FCEABR Processing Time
PT_2	FCEABOB Processing Time
ECT_1	FCEABR Election Process Consuming Time
ECT_2	FCEABOB Election Process Consuming Time
HECT	Hierarchical algorithm Election Consuming Time
GCFCC	Global Coordinator Failure Communication Cost
ICFCC	Internal Coordinator Failure Communication Cost
α_1	Communication time between two processes in the virtual network
α_2	Communication time between two processes in the site.
α_3	Communication time between a process in the virtual network and a process one in a typical sites
B_1	Time consumed to compare two numbers by a process in the virtual network
B_2	Time consumed by a process to find out the processes which didn't reply back to it.

```
Void Crash_identify(process starter)

{
    If (Internal Coordinator has crashed)
        Internal Coordinator=next alternative;
    else
        External Coordinator = next alternative;
    If (next alternative exist)
        shift all the alternatives to left;
    else
        Election algorithm after crash (starter, k);
}End procedure
```

Fig. 10. Crash Identify procedure. It runs by a process which finds out coordinator crashed.

VIII. Mathematical Analyze

Proposed algorithm is consisted of two algorithms applied on sites and built virtual network. Therefore both should be analyzed in order to achieve complete analysis.

A. *Message Complexity Analyzing*

One of the most important features of an algorithm is the number of messages exchanged among the processes, which is vital in the high traffic networks.

1. FCEABR Message Complexity Analyzing

This algorithm is applied on ring network and virtual network message's number (VMN) is subjected to the number of sites (NS) in the network and number of those processes which find coordinator crash out (FPN). Therefore, VMN is calculated by (1).

$$VMN = (FPN - 1) * (NS/\ FPN) +\ 2 * NS \qquad (1)$$

2. FCEABOB Message Complexity Analyzing

This algorithm is applied to inner site election. Site's messages number (SMN) is subject to the site's process number (SPN) and site's coordinator alternatives number (SCAN). Hence, the total site's messages number during election at the best case achieves by (2).

$$SMN = 2 * SCAN + SPN - 1 \qquad (2)$$

However, when response disability by processes in the site is considered, SMN will be increased. Therefore, the worst case of the algorithm is calculated by (3). It should be mentioned this number is gained when all the process in the network are crashed except informer, so there is no need to inform the other about the elected coordinator, which is the informer itself.

$$\begin{aligned} SMN &= SCAN + 2 * (SCAN) * (\ SPN/SCAN) \\ &= SCAN + 2 * SPN \end{aligned} \qquad (3)$$

3. Hierarchical Message Complexity Analyzing

Exchanged message number (EMN) to elect coordinator in the network with initial configuration will be equaled to the below equations in the base which is from O (NS + SPN). For the sake of the problem Site coordinator alternatives number and site's process number is assumed to be the same. But the worst case number of exchanged message in the algorithm can be calculated by (5) which is from O (SPN + NS)

Hence the algorithm's message passing during coordinator election is from O (SPN + NS) and Ω (SPN + NS).

B. *Time Complexity Analyzing*

Low message complexity of an algorithm is considered as a great advantage. However, if these messages are exchanged during a long period of time, the algorithm is almost impractical and useless. Therefore, time complexity of our algorithm is discussed by analyzing FCEABR and FCABOB algorithms separately as same as the previous section.

1. FCEABR Time Complexity Analyzing

During election procedure by this algorithm, the Election messages are circulated among all the processes in the network, and then they should be informed about the elected coordinator and its alternatives. Moreover, any process compares its own number with Election message's coordinator and its alternatives. As discussed before, the number of messages passed by this algorithm is variable due to number of processes in the network. Communication time between each two processes is considered to be the same for simplicity, so Communication Cost (CC_1) of the algorithm is gained by (6).

Total Processing Time (PT) by processes in the network is calculated by the below (7). Therefore, (10) calculates Election process Consuming Time (ECT_1).

However, (7) and (10) will be changed to (11) and (14) respectively when all coordinator alternatives are already crashed but coordinator itself is still up.

SCAN, β_1 and α_1 are constant variables, so time complexity of the algorithm is from O (NS) and Ω (NS) which means the algorithm is from order number of processes in the network.

$$\mathrm{EMN} =\ \mathrm{NS} * (\ 2 * \mathrm{SCAN} + \mathrm{SPN}) + (\ \mathrm{FPN} - 1\ ) * (\mathrm{NS}/\mathrm{FPN}\ ) +\ 2 * \mathrm{NS} \qquad (4)$$

$$EMN = SCAN + 2 * SPN + (FPN - 1) * (NS/FPN\ )\ 2 * NS \qquad (5)$$

$$CC_1 = \begin{cases} 2 * (NS - 1) * \alpha_1 & NS\ is\ even \\ 2 * NS * \alpha_1 & Otherwise \end{cases} \qquad (6)$$

$$PT_1 = (SCAN + 1) * (NS - 1) * \beta_1 \qquad (7)$$

$$\lambda_1 = NS/2\ \ * ((SCAN + 1) * (NS - 1) * \beta_1 +\ 2 * (NS - 1) * \alpha_1) \qquad (8)$$

$$\lambda_2 = NS/2\ \ * ((SCAN + 1) * (NS - 1) * \beta_1 +\ 2 * NS * \alpha_1) \qquad (9)$$

$$ECT_1 = \begin{cases} \lambda_1 & NS\ is\ even \\ \lambda_2 & Otherwise \end{cases} \tag{10}$$

$$PT_1 = SCAN * (NS - 1) * \beta_1 \tag{11}$$

$$\lambda_3 = NS/2 \ * (SCAN * (NS - 1) * \beta_1 + \ 2 * (NS - 1) * \alpha_1) \tag{12}$$

$$\lambda_4 = NS/2 \ * (SCAN * (NS - 1) * \beta_1 + \ 2 * NS * \alpha_1) \tag{13}$$

$$ECT_1 = \begin{cases} \lambda_3 & NS\ is\ even \\ \lambda_4 & Otherwise \end{cases} \tag{14}$$

$$CC_2 = 2 * SCAN * \alpha_2 + \ SPN \tag{15}$$

$$CC_2 \ = (2 * SPN + SCAN) * \ \alpha_2 \tag{16}$$

$$PT_2 = (SPN/\ SCAN\ ) \ * \beta_2 \tag{17}$$

$$ECT_2 = (2 * SPN + SCAN) * \ \alpha_2 + \ (SPN/\ SCAN\ ) \ * \beta_2 \tag{18}$$

$$HECT = ECT_2 + \ ECT_1 \tag{19}$$

$$HECT = \begin{cases} 2 * SCAN * \alpha_2 + \ SPN + \ \lambda_1 & NS\ is\ even \\ 2 * SCAN * \alpha_2 + \ SPN + \ \lambda_2 & Otherwise \end{cases} \tag{20}$$

$$HECT = \begin{cases} (2 * SPN + SCAN) * \ \alpha_2 + \ (SPN/\ SCAN\ ) * \beta_2 + \lambda_1 & NS\ is\ even \\ (2 * SPN + SCAN) * \ \alpha_2 + \ (SPN/\ SCAN\ ) * \beta_2 + \lambda_2 & Otherwise \end{cases} \tag{21}$$

$$GCFCC = \alpha_3 + (SPN_i - 1) * \ \alpha_2 + (SN - 1) * \alpha_1 \tag{22}$$

$$ICFCC = \left(SPN_j - 1\right) * \ \alpha_2 \tag{23}$$

2. FCEABOB Time Complexity Analyzing

In the best case, Communication Cost of this algorithm (CC_2) is calculated by (15). However, processing time (PT_2) is equal to zero since all the alternatives are responding. Therefore, the election time is equal to communication time. Equation (15) will be changed to (16) in the worst case of the algorithm. The Processing Time of the algorithm (PT_2) and the whole Election Consuming Time (ECT_2) are also calculated by (17) and (18). As it is obvious this algorithm is from O (SPN) and Ω (SPN).

3. Hierarchical Time Complexity Analyzing

The Election Consuming Time (HECT) in the best and worst case is calculated by (20) and (21) respectively.

Both of the worst and the best cases of algorithm are from O (NS+SPN) and Ω (NS+SPN) since all the other parameters are constant and SCAN is lower equal to SPN. Global Coordinator Failure Communication Cost (GCFCC) is gained by (22) if GCoordinator belongs to site i.

Internal Coordinator Failure Communication Cost (ICFCC) with this assumption that ICoordinator belongs to site j equals the (23).

IX. Simulation

The simulation program is written by Microsoft Visual Studio 2010, C#.Net programming language. The processes numbers are assigned randomly in the entire network so the numbers are distributed in it.

We have run our simulation program in different situations to test the proposed algorithm's behavior in various conditions.

We assume that there are 8 sites in the network, GCoordinator has 3 alternatives, connection type between coordinator and its first alternative is hot standby, and process which received election message could respond to sender. The specification of each site is shown in Table 2. When network is started up, there is no coordinator in it so simulator is run to elect coordinators and it also counts the number of messages which are exchanging among the sites and the processes in each.

Table 2. Sites specifications. NOP: Number of Processes, NOA: Number of Alternatives

Site Number	1	2	3	4	5	6	7	8
NOP	252	341	412	44	802	153	265	183
NOA	5	6	11	2	8	5	8	9

Table 3. Number of Messages exchanged among processes in each site to elect coordinators. NOM: Number of Messages

Site Number	1	2	3	4	5	6	7	8
NOM	264	355	24	50	820	165	283	203

The number of messages that passed during coordinator election in each site is determined in Table 3 and the total number of it is 2164. 16 messages are also exchanged among the sites to elect the GCoordinator and its alternatives, so 2180 messages are totally being passed. As we can see the majority of message passing is dedicated to inform processes about the elected coordinators and their alternatives. This number was stable in 50 times of running the simulator by these different site's specifications because the number of message in FCEABOB algorithm depends on the number of alternatives in the site.

In the rest of the tests, the 4 following cases are considered to compare proposed algorithm with the other practical ones. In each case a different algorithm for the inner site election and virtual network election is applied.

(1) The virtual network election is simple Ring algorithm and the inner site election will be done by FCEABOB. This algorithm is referred by R.
(2) In this case, the virtual network is based on the topology which every process in the network has known one another; Modified bully algorithm [10] is applied to it and Inner site election is done by FCEABOB. This case is referred by MB.
(3) The inner site and virtual network coordinator election are done by simple Ring algorithm. We denote this case with TR.
(4) FCEABOB algorithm is considered for the inner site election and virtual network election is done by FCEABR. Algorithm referred by MA.

The four cases are compared from the total message number exchanging point of view. Sites quantity is changed in each test but the number of processes existed in them are 200, number of alternatives is 8 and GCoordinator has 2 alternatives. The result of our test is determined in Fig. 11. It is obvious that the number of exchanged messages by TR is almost more than others because of the nature of Ring algorithm which pass more messages in each site in order to elect coordinators. R, MB, and MA passed nearly the same number of messages according to their election algorithm procedure. Therefore, our algorithm behaves similar to the R and MB in this test.

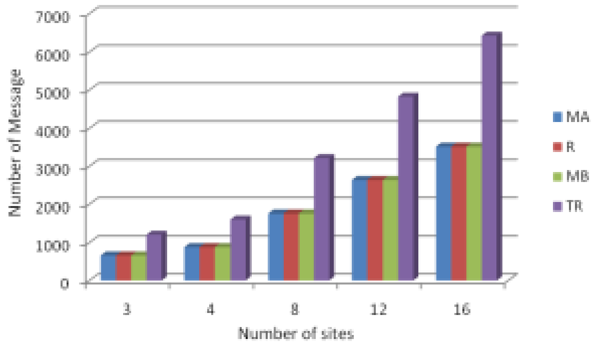


Fig.11. Number of exchanged messages in entire network to elect inner and globalcoordinator

The next test (Test three) is examined in the situation that coordinators in each site are already elected but the GCoordinator is crashed. We keep the last test network's specifications and compare these four cases by changing the numbers of processes that realize crash. The result is shown in Fig. 12. This figure shows that MN exchanged fewer messages than others since its mechanism is avoiding of waste messages when more than one process find crash out.

In other algorithms each process which realizes crash, launches election and exchanges messages separately, so if n_3 process finds out that the coordinator is crashed, n_3 separate election algorithm will be run simultaneously.

We consider network's specifications inserted in Table 4 to run the fourth test. In this test we obtain the number of messages that were exchanged among the processes during the 3 times crashing of the GCoordinator by the four cases which are already specified at the beginning of this section.

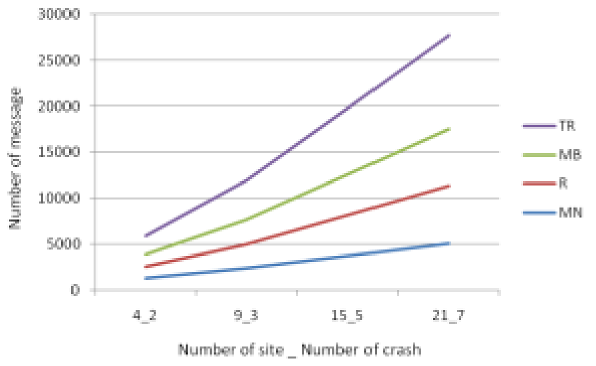


Fig.12. Comparison between the four cases when main coordinator is crashed

Table 4. Network specifications of test 4

Number of sites	9
Number of processes in each site	200
Virtual network alternative number	2
Each site alternative number	8

The result of the test is shown in Fig. 13. It is obvious that the number of messages passed by MN algorithm is fewer than the other cases. After 2 times of GCoordinator crash, since there is no other alternatives in the virtual network, the Last GCoordinator lunched another election algorithm in the virtual network in order to avoid the wait time induction to any process in the network.

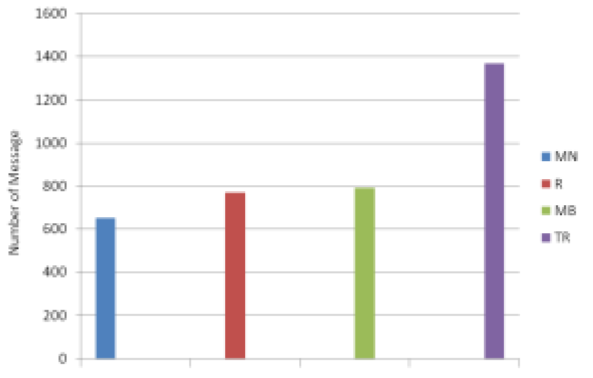


Fig. 13. Number of messages that passed after 3 times crash of main coordinator.

MB and R algorithms are similar since in both cases a new election algorithm should be lunched and when a new site's ECoordinator wants to join the virtual network in case MB, it should inform all the other processes about its own information. Hence, when election in the virtual network is done by the simple ring and modified bully, the number of messages is almost as same as each other.

It was clear in these tests that our algorithm passed fewer numbers of messages; especially when more than one process realizes crash, so this algorithm can work perfectly in the high traffic networks with low process reliability.

X. Convergance Approving

The processes which realize crash and process placement in the network are chosen randomly, so the number of messages in several times repetition of one test may differ from each other. We approve the final result (number of messages which are passed in network) convergence of our algorithm by calculating its standard deviation. The average number of messages $(\overline{X})$ that is exchanged after 200 times of test repetition is gained by (24) and due to unknown statistical community, sample variance (S_x^2), which is calculated by (25), should be used.

$$\overline{\mathrm{X}} = \left(\sum_{i=1}^{n} \mathrm{X}_i \right) / n \qquad (24)$$

$$\mathrm{S}_x^2 = \left(\sum_{i=1}^{n} \left(\mathrm{X}_i - \overline{\mathrm{X}}\right)^2 \right) / n - 1 \qquad (25)$$

Therefore, message's standard deviation ($S_{\overline{x}}$) is calculated by (26).

$$S_{\overline{x}} = S_x / \sqrt{n} \qquad (26)$$

We calculated the standard deviation for four different networks. The specification of networks and the average number of messages that passed after 200 times repeating the test is inserted in Table 5. It should be mentioned that the number of alternatives and processes in each site is the same.

Table 5. Total messages Standard deviation, network specification and number of messages which passed

Number of site	Number of Alternatives in Virtual Network	Number of Processes in each site	Total Number of Processes	Number of fault	Number of Alternatives in each site	Average number of send and received message	Standard deviation
4	2	120	480	3	4	772	0.0141
13	3	160	2080	4	6	2618	0.0059
24	5	200	4800	5	8	5698	0.0076
40	7	250	10000	7	10	11492	0.0080

XI. Conclusion

As we mentioned in the previous sections of the paper, our algorithm to elect the coordinator was based on the networks which have multi management sites without any restriction at number of processes. We proposed a new algorithm to elect sites' coordinators which was based on bully. Two coordinators in each site were elected which internal coordinators was for coordinating accessing to the internal shared resources and external coordinator had the duty to respond to the processes wanted to access to multi site's shared resources. For coordinating the requests of shared resources among the sites, we should have a global coordinator in the entire network which is elected by setting up a virtual network consisting of external coordinators of each site and applying FCEABR to it. We also considered that the number of alternatives in each site can be identified by its administrator depending on the characteristics it has. Simulation section approved that proposed algorithm exchanged fewer messages than the other algorithms

Refrences

[1] Shirali M, Hagighattoroghi A, Vojdani M, Leader Election Algorithms: History and Novel Schemes. 7th International Conference on Computer and Information Technology, 2008, 452-456.

[2] Obeidat A, Gubarev V. Leader Election in peer to peer systems, Siberian conference on control and communications,2009, 25-31.

[3] Park S. A Stable Election Protocol Based on Unreliable failure detector in Distributed Systems, Eighth International Conference on Information Technology New Generations, 2011, 979-984.

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

[5] Fredrickson N and Lynch N. Electing a Leader in Asynchronous Ring, Journal of ACM, vol.34, no.1, 2007, 98-115.

[6] Datta, A, Larmore L, Vemula P. An O(n)–time self–stabilizing leader election algorithm, Journal of Parallel Distributed Computing ,vol.71, no.11, 2011,1532-1544.

[7] Shi W, Srimani P K. Leader election in hierarchical star network, Journal Parallel Distributed Computing, vol.65, nom 11, 2005, 1435-1442.

[8] Bakhshi R, Endrullis J, Fokkink W, Pang J. Fast leader election in anonymous rings with bounded expected delay, Journal of Information Processing Letters, vol.111, no.17, 2011, 864-870.

[9] Delporte-Gallet C, Devismes B, Fauconnier H. Stabilizing leader election in partial synchronous system with crash failure, Journal of Parallel and Distributed Computing, vol.70, nom.1, 2008, 45-58.

[10] Effatparvar M, Effatparvar M R, Bemana M, Dehghan A. Determining a Central Controlling Processor with Fault Tolerant Method in Distributed System, 2007, 658 – 663.

[11] Mirakhorli M, Shariflloo A, Abaspour M. A Novel Method for Leader Election Algorithm, 7th IEEE International Conference on Computer and Information Technology, 2007, 452-456.

[12] Effatparvar M R, Yazdani N, Effatparvar M, Dadlani M, Khonsari A. Improved Algorithm for Leader Election in Distributed Systems, second International Conference on Computer Engineering and Technology, ,2007, 6 – 10.

[13] Villadangos J, Cordoba A, Farina F, Prieto M. Efficient leader election in complete networks, Pro of the 13th Euro micros conference on Parallel, Distributed and Network-Based Processing, 2005, 136-143.

[14] Rahdari D, Rahmani A.M, Arabshahi A. A Novel Message Efficient Fault Tolerant Coordinator Election Algorithm in Bidirectional Ring Networks, Journal of Information

Technology and Computer Science, Vol.5m no.1, 2012 , 15 - 25, 2012.

Authors' Profiles

Danial. Rahdari: received his B.S in computer engineering from Sistan and Balouchestan university, Zahedan in 20 in 2010, the M.S in computer engineering from IAU University in 2013. His research interests are in the areas of distributed computing, cloud services, quality of service, load balancing, fault tolerance, networking and Cisco engineering

Amir Masoud. Rahmani: received his B.S. in computer engineering from Amir Kabir University, Tehran, in 1996, the M.S. in computer engineering from Sharif University of technology, Tehran, in 1998 and the PhD degree in computer engineering from IAU University, Tehran, in 2005. He is assistant professor in the Department of Computer and Mechatronics Engineering at the IAU University. He is the author/co-author of more than 80 publications in technical journals and conferences. He served on the program committees of several national and international conferences. His research interests are in the areas of distributed systems, ad hoc and sensor wireless networks, scheduling algorithms and evolutionary computing.

Niusha. Aboutaleby: Received her B.S in computer engineering from Iran University of Science & Technology, in 2009 and she is studying computer engineering in M.S at IAU University. Her research interests are in the area of distributed systems, cloud services, load balancing and scheduling algorithms in cloud computing, ad hoc and sensor networks

Ali. Sheidaei Karambasti: received his B.S in computer engineering from Sistan and Balouchestan university, Zahedan in 2010. His research interests are in areas of social network analyzing, real time computing ecommerce systems, graph layout algorithms, similarity ranking algorithms, classification algorithms, enterprise application development.

How to cite this paper: Danial Rahdari, Amir Masoud Rahmani, Niusha Aboutaleby, Ali Sheidaei Karambasti,"A Distributed Fault Tolerance Global Coordinator Election Algorithm in Unreliable High Traffic Distributed Systems", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.3, pp.1-11, 2015. DOI: 10.5815/ijitcs.2015.03.01