A Survey and Taxonomy of Leader Election Algorithms in Distributed Systems

Article *in* Indian Journal of Science and Technology · June 2014

DOI: 10.17485/ijst/2014/v7i6.14

1 author:

Farhad Soleimanian Gharehchopogh
Islamic Azad University of Urmia
152 PUBLICATIONS **4,601** CITATIONS

Indian Journal of Science and Technology, *Vol 7(6), 815–830, June 2014*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Survey and Taxonomy of Leader Election Algorithms in Distributed Systems

Farhad Soleimanian Gharehchopogh* **and Hassan Arjang**

Department of Computer Engineering, Science and Research Branch, Islamic Azad University, West Azerbaijan, Iran; Bonab.farhad@gmail.com, Arjang.hasan1986@gmail.com

Abstract

Leader election is one of the important problems in distributed systems. This problem deals with selecting a node in distributed systems as a leader. In the election of nodes problems such as node failure and discrete parts of the network should also be examined. Investigation of this problem in wireless networks especially Adhoc, that do not have special structure and topology changes are frequently and inevitable, is much more complex than wired networks. This paper examines different algorithms for leader election. Some algorithms are designed for just one leader and some of them are designed for N leader. Some of them also create hierarchical structure in resulted graph of wireless networks. These algorithms are used in places that have been pre-clustered or clustering of them is possible. In this paper, in addition to various algorithms classification based on the used assumptions and methods, we deal with.

Keywords: Adhoc Networks, Leader Election, Self-stabilization

1. Introduction

Leader election is an important problem in wired and wireless networks and in general, is said to unique election of a leader or selected node in a distributed network. Many algorithms have been proposed for this problem. For example[1, 2] the authors analyze the statistical method of coin toss. In this method, each node has a coin toss and if the result be the coin back, this operation is repeated until a single node remains. Some papers[7–11] deal with single-hope radio networks various algorithms. In this network, each node that sends a packet, all the other nodes receive it. These algorithms are divided in multiple groups considering the information in leader election. For example some algorithms do not affect previous state of the channel in leader election[7, 10] and some other affect[9, 11]. In general, in this like coin toss algorithms, periods are repeated until eventually only one node remains. This algorithm in Ad hoc networks has different requirements and complexities. In these networks, the algorithm should be robust against simultaneously and sequentially topology changes. Algorithms[3–6,16] have been introduced for the election of a leader in a multi-hop wireless networks. The basis of algorithms[3,6] is TORA routing protocol in wireless networks. In this algorithm, the goal is to build a directional tree without loop that has a root. In this algorithm, the leader is randomly selected. In these algorithms[4,5] with introduction of a different method, leader election based on their values would be possible. Node value can be adjustable parameter like the remaining energy, computing power, etc. Of course that these algorithms are designed against Concurrent changes in robust channel structure but leader node replacing process in special states that leader node faced with problem or has been broken by changes, are not supported in these algorithms. Hierarchical leader election algorithms[14,15,17] have been introduced. These algorithms support many nodes in the network and plethora of nodes and their separation does not make a dent in these algorithms. Because in these algorithms, network is divided to clusters that contain several node and each of them has a head cluster. The goal is determination of one node or head cluster as

*Author for correspondence

the only leader in the network. These algorithms are more applicable in applications that require network clustering or networks that has been pre-clustered. Of course that many algorithms have been introduced for leader election, but just in araticle[12], pointed to k leader election. That is, instead of one leader, k leader with the maximum weight is selected. Even with losing one of these leaders, the algorithm tries to find replacement in the fastest time possible.

The remainder of this paper is as follows: second section deals with leader election algorithm in single-hop networks. Third section discuss about related algorithms in multi-hop networks. In the fourth section, hierarchical algorithms are introduced and investigated. In the fifth section, we have introduced k leader election algorithm. In the sixth section we deal with review of presented algorithms and finally in the seventh section conclusion of future tasks are presented.

2. Leader Election in Single-hop Network

The first research sets conducted about the election of the leader in wireless networks was a single-hop network. This means that all nodes communicate with each other directly and without intermediaries. In other words, each node sends a message, all nodes receive it. However, this assumption is not consistent with much of today's wireless networks. But a comparison of this algorithm and the idea of them are not unpleasant. These algorithms divide time into equal time slots (t). Each node sends a message with probability Pi at this time slot. This probability may change over time. During each time slot over nodes transmission, the following three states could be resulted for the channels:

Null: In this case, any node does not send a message.

Single: In this case, only one node sends the message.

Collision: In this case, more than one node sends the message.

Transmission procedure is repeated in time gaps, eventually channel comes out in single state, and in this case transmitter node is selected as leader. Depending to the type of algorithm, these algorithms may keep two kinds of information: one is history of channel state and the other transmission history (message sending or not sending in t time gap).based on these information's, these section algorithms are divided to three categories:

Oblivious: These groups of algorithms do not hold any history. Transmission probability may change over time but this amount is the same for all nodes.

Uniform: channel state history is saved in these algorithms. Transmission probability is a function of channel history and this probability is same for all nodes.

Non-uniform: in these algorithms, both of channel state history and transmission history are saved. Transmission probability depends to these two histories and is different for various nodes.

Also, the leader election problem may be examined in the following three scenarios7:

Scenario 1: the number of nodes (n) is specified.

Scenario 2: The number of nodes unknown, but its upper limit (u) is specified.

Scenario 3: neither of the number of nodes nor the upper limit is specified.

2.1 Leader Election with Forgetful Protocols

In this section, we deal with introduction of three algorithms for each of the three scenarios and show the estimated time of algorithms finishing. We did not present the time proof of most algorithms because of their long proof. You can refer to the reference point for prove.

2.1.1 Forgetful Protocol for First Scenario

If p be a series of probability like P=< P1,P2,P3,…> in Successive time gaps, in this way, each node with pt probability in t time, will send the message. This operation is repeated so many times until eventually channel state is single, and that way, message transmitter leader will be in single state.

We consider transmission probability in fixed time and equal to P_i=1/n .in this way, the probability that channel state is single would be:

$$P_i(1-p)^{n-1} = \left(1-\frac{1}{n}\right)^{n-1} > \frac{1}{e}$$

Then each round of running in each time slot with the probability of e/1 would be successful of leader election.

Then t time slot with the probability of $\left(1-\frac{1}{e}\right)^t < e^{-\frac{t}{e}}$ Would not be successful is equivalent to

$\frac{1}{f} = e^{-\frac{t}{e}}$ Then, t=e.ln f.

Theorem 1: If you select p=<1/n,1/n,...> algorithm with at least probability of 1-1/f in e. ln f time slot will be finished. More detail is presented[7].

2.1.2 Forgetful Protocol for Second Scenario

An algorithm[10] has been presented for leader election in second scenario. Suppose that upper limit of nodes number is u, transmission probability is defined as a series if extended series of transmission probability is $D_i^{\infty} = D_i$, D_i ... that is concurrent attachment of D_i, for example D_2^{∞}=<1/2,1/4,1/2,1/4,1/2,1/4,...> if we consider transmission probability a series like $D_{[\log u]}^{\infty}$ theorem 2 is proved.

Theorem 2: if probability series be $D_{[\log u]}^{\infty}$ with minimum probability of 1-1/f (for each f>=1) algorithm will be finished in time gap of log f [log u]. To prove the details of the algorithm refer article[10].

2.1.3 Forgetful Protocol for Third Scenario

Suppose that V_1=<1,2,3,…> then P(v) is defined as P(V)= D_1, D_2, D_3, ... =<1/2,1/2,1/4,1/2,1/4,1/8, ... >also suppose that V_c=< [c^0],[c^1],[c^2], ... > and c is an arbitrary number between 1 and 2, if we have P' =< P'_1, P'_2, ... >, P = < P_1,P_2, ... > then $\oplus$ operator is defined as P $\oplus$ P' =< P_1, P'_1, P_2, P'_2, ... >in this was theorem 3 is proved.

Theorem 3: if the probability series is P(V_1,V_c) with the probability of 1-1/f for each f>=1, algorithm is finished is time slot of O(min(($log\ n$)^2 + (log)^2, f^{ε} log n))

Detailed proof is presented[7].

2.2 Leader Election with Uniformed Protocols

In this section, we deal with introduction of one uniform protocol[11] that do not have any suppose for nodes number or upper limit of them (third scenario). At first broadcast function with p parameter is defined as follow:

Protocol Broadcast (p)

Every station broadcasts with probability $1/2^p$

If the status of the channel is SINGLE then the unique station that has transmitted becomes the leader.

This protocol has three consecutive phases. In the first phase the protocols are executed like Broadcast (2^{0^2}), Broadcast (2^{1^2}), Broadcast (2^{t^2}). Until channel state in t approach to null state. Then second phase is started in this phase, the algorithm conduct a binary search in [0, 2^{t^2}] as follow:

First Broadcast ($2^{t^2}/2$) is executed. If the channel state be single, algorithm is finished and leader is selected.

If the channel state be null, a binary search in [0, 2^{t^2}] that is the same as Broadcast ($2^{t^2}/4$) is executed.

If the channel state be null, a binary search in [$2^{t^2}/2$, 2^t] that is the same as Broadcast ($\frac{3}{4}$, 2^{t^2}) is executed.

Binary search phase in a loop can be repeated up to the point that there is not any possibility of yields division. Consider u as the last amount that has been executed in the second phase in Broadcost(u) in the third phase, function Broadcost(u) is called to the point that finally channel state, approaches to the single case. Of course in this phase, the u variable in each repetition of loop becomes one unit bigger or smaller. This amount change in each stage, may closer channel single probability to ideal state[11].

2.3 Leader Election with Non-uniform Protocols

In this section, we introduce a non-uniform protocol that designed like previous protocol in the third scenario. In algorithm running, we used Sieve(P) function that is defined as follow:

Sieve(1^2), Sieve(2^2),..., Sieve(t^2)

Protocol Broadcast(p)
every active station broadcast with probability $\frac{1}{2^{2^p}}$;
if the status of the channel is SINGLE then
the station that transmitted becomes the leader.
else if the status of the channel is GOLLISION
then the stations that have not transmitted become inactive.
end if

In the first phase of this protocol, the function is implemented as Sieve (1^2), Sieve(2^2),..., Sieve(t^2). This phase is executed until the channel state in Sieve(t^2) becomes null. In the second phase, the function is executed as Sieve (t^2-1), Sieve(t^2-2),..., Sieve(0). In the third phase Sieve(0) is called to the point that channel state becomes single and the leader is selected. The algorithm[9] is:

Protocol Non-Uniform-Election
Initially all the stations are active;
Phase 1: *for i←0 to 1 sieve (i^2); exit for loop if the status is NULL;*
Phase 2: *t←i^2-1; for i←t down to 0 sieve (i);*
Phase 3: *repeat sieve (0); forever*

Theorem 4: mentioned non-uniform algorithm with minimum probability of 1-1/f in loglogn+2.78logf + o(loglogn + logf) time slot for each f >=1 is finished. More details and how it is proved, is presented[9].

Table 1 shows leader election protocols along with their implementation time-order.

Table 1. Single-hop leader election protocols and their time-order

Protocol	Scenario	Timeslots average
Oblivious	1	E
Oblivious	2	O(log u)
Oblivious	3	O(log n)
Uniform	3	Loglogn+o(loglogn)
Non-uniform	3	Loglogn+o(loglogn)

3. Leader Election in Multi-step Network

In the previous section, we have investigated protocols based on single step. These protocols use Broadcast properties in wireless networks. With regard to this issue, it seems that they do not have much application in practice. Suppose of single hop for current wireless networks such as Adhoc networks or Wireless Sensor Networks is very unreasonable and impossible .then, special protocols of Adhoc networks and in general wireless has been in focus since 1990s. Also the problem of frequent topology change in Adhoc networks and its effect on leader election process were noted. The protocols that are investigated in this section contain the following assumptions[3–6]:

All nodes have a unique identifier. This is logical and possible with considering current wireless networks. All links are two-sided and FIFO (of course we can diagnose one way link and do not consider them). Nodes have sufficient buffer until buffer overhead do not happen in network arena. Nodes are replaced and are survived again and in general, network topology may change. There is not any pre-assumption for Number of nodes and also maximum number of nodes. Leader election protocols are classified in several aspect. In this section; we have divided leader election protocols to two categories that are based on used parameter in leader election:

Random Election: in these protocols, leader election is done based on their unique identifier.

Extreme-Finding: in these protocols, election is based on parameters like the remaining energy, computing power, etc. These protocols are in many cases more desirable. For instance, election of a node with maximum energy is very important in wireless sensor networks. We can also divide protocols based on self-stabilization. Self-stabilized protocols, even if changes occur continuously, again one leader is elected. For instance algorithm[6] is of this category. But algorithms like Malpani[3] are not self-stabilized. It means that, also the protocol is running and involved in leader election, these change are like that the protocol can be run again from scratch. So if a chain of changes is constantly happening, Malpani protocol will terminate never.

3.1 Random Election Algorithms

Form the initial algorithms that have been introduced for wireless networks and contained characteristics like Broadcast, we can mention Malpani. The main idea of this algorithm is derived from TORA routing protocol. The basis of these algorithms is building of directional trees without loops or DAG. In this tree, all nodes are guided towards a destination node or sink. In TORA this node, is destination node in routing but in Malpani, this node is leader node.

3.1.1 TORA Algorithm

In this algorithm, height is assigned to each of them. In nodes that are in the proximity of each other. Always there is drawn an edge from a node with higher height towards a node with lower height. Height in this protocol is a set of five values like(τ_i,oid_i,r_i,δ_i,i) comparison of two heights is conducted from left to right. It means that, firstly τ_i parameter is compared with each other. If they were equal, next parameter oid is compared. This procedure continues until, finally if all the four first parameters were equal, fifth parameter that is unique identifier of i node, is used for comparison. Three first parameters in height called reference level. When a node misses all its output links, it gives value to its reference level newly. τ_i Value is equal to the time that this event has occurred. Oid is equal to I or the node that shows the change. At first it is equal to zero, but if it becomes to one shows network separated parts. In this way each part should have its own tree. Fourth parameter is used for identification of edge direction among two neighbors. This parameter is conFigd in a situation that reference level is equal to desired protocol until elected by the appropriate edges.

When a new reference level is created in a node, this reference level is higher than other and neighbors reference level because its occurring time is more updated. As a result this node announces the changes to its neighbors. As it is proved in Malpani[3], this change is announced to all nodes that use this node for reaching to destination. These nodes should find new route or specify lack of access to destination (network fragmentation) most of it

and its decisions is similar to Malpani algorithm that will be described in 3.1.2. One of the weaknesses of TORA algorithm[9] is that there is not any prove for algorithm correct operation and it is not clear whether in Non-steady conditions, the algorithm operates well or not.

3.2.1 Malpani Algorithm

Malpani algorithm uses the idea of TORA protocol and with building of a DAG tree, that its destination is leader node, solves the leader election problem. In Malpani paper two algorithms are presented. One for a situation that only one topological change occurs in network and until the end of leader election, there is not any other change in the network. And the other for the situation that, multiple change can occur concurrently and continuously. For the first state, a proof is presented but unfortunately for the second algorithm there is not any proof.

Height in this algorithm contains six parameters. First parameter is leader node identifier from I node view and the other parameters are like TORA algorithm[9]. Reference level that is elected for the leader is equivalent to (-1,-1,-1) until ensure that it is sink node and all neighbors edges enter to it.

In TORA, when fragmentation is identified by a node, that node sends a message to other existing nodes until end the height change and Useless messages. In Malpani algorithm during fragmentation, that node elects itself as a leader and announces it to other nodes. If two parts connect to each other, the leader which has the lower height will be elected as the only leader.

Nodes in this algorithm just send messages containing their heights not anything else. Also, each node saves a data structure $height_i[j],N_i$ that holds identifier and height of their neighbors respectively. All the protocol is defined with five conditions as follow:

1-when I node, does not have any output edge because of link failure (Due to the node failure or its replacement):

If node i has no incoming links as well than
$lid_i := i$
$(\tau_i,oid_i,r_i) := (-1,-1,-1)$
$\delta_i := 0$
Else
$(\tau_i,oid_i,r_i) := (t,i,0)$//t is the current time
$\delta_i := 0$

II : when I node does not have any output edge due to edge direction change (due to update message receive) and neighbors reference level are not equal (τ_i,oid_i,r_i):

$(\tau_i,oid_i,r_i) := \max\{(\tau_j,oid_j,r_j) \mid j \in N_i\}$
$\delta_i := \min\{\delta_j \mid j \in N_i \text{ and } (\tau_j,oid_j,r_j) = (\tau_i,oid_i,r_i)\}-1$

III : when I node does not have any output edge due to edge direction change (due to update message receive) and neighbors reference level are equal (τ_i,oid_i,r_i) and for all the neighbors is : $r_j = 0,j$

$(\tau_i,oid_i,r_i) := (\tau_j,oid_j,1)$ for any $j \in N_i$
$\delta_i := 0$

IV : when I node does not have any output edge due to edge direction change (due to update message receive) and neighbors reference level are equal (τ_i,oid_i,r_i) and for all the neighbors is:

$oid_j = i,\ r_j = 1, j$
$lid_i := i$
$(\tau_i,oid_i,r_i) := (-1,-1,-1)$
$\delta_i := 0$

V : when I node receives a message from its j neighbor that $lid_i \neq lid_j$

If $lid_i > lid_j$ or $(oid_i = lid_j \text{ and } r_i = 1)$ then
$lid_i = lid_j$
$(\tau_i,oid_i,r_i) := (0,0,0)$
$\delta_i := \delta_i + 1$

Figure 1 shows algorithm operation during fragmentation detection and connection of two parts. In section a, A node detect IV state or fragmentation, hence elect itself as a leader and announce it. In section b of the Figure 1, B and D nodes update themselves. In section c to f, a situation that two sections are connected showed. And at last, A that has lower identifier rather than f is elected as the leader. The proof of its correctness is presented3 in a case that just one topological change occurs in each running of algorithm. For the case of occurring of several topological changes concurrently, another algorithm is introduced in Malpani that is extended of previous algorithm. To this reason a part is added to V and other state i.e. VI is introduced. Due to lack of second algorithm proof, it is unknown whether this algorithm terminates correctly in all situations or not.

3.1.3 Derhab Algorithm

Malpani algorithm, in addition to introduction of a method for leader election problem solvation in a situation that concurrent changes occur, not always responsive. For instance, motion of a node around a DAG tree causes disconnection and starts change announcement. But this change announcement is not conducted by one node

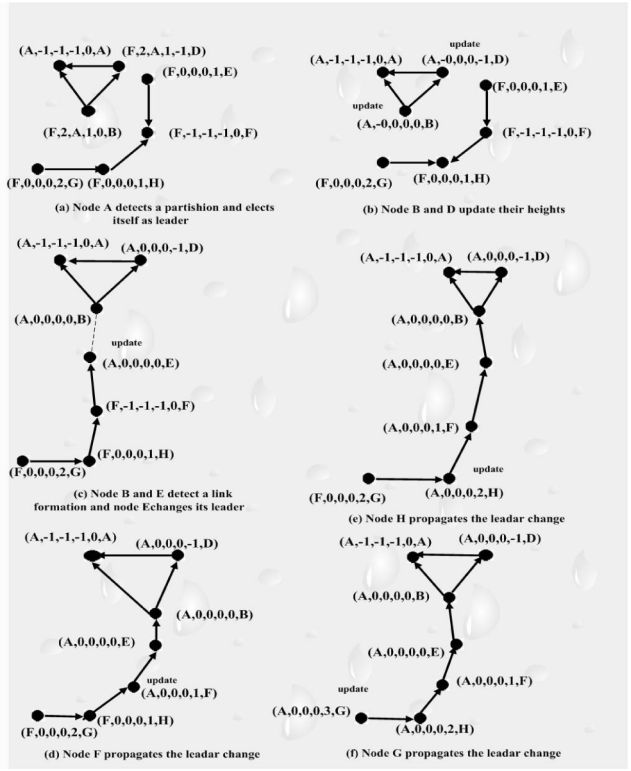

Figure 1. Performance of the detection algorithm and a connecting two pieces of location[3].

and at the same time. Maybe due to nodes replacement, various nodes report their change in different times. In this case, ach new change announcement is the result of previous process operation and seems that leader election process started again. In such circumstances due to permanent changes the algorithm may never fail. Derhab[6] paper tries to solve this problem .the main idea of Derhab algorithm is that, change the algorithm so that smaller reference level has higher priority than bigger reference level (contrast to Malpani).

Actually, reference level first parameter (t) that shows process start time is more acceptable if older or lower. The reason for this is that older processes done some calculations and are closer to leader election. Then it is better that new processes do not effect on it. But the problem is that we cannot separate concurrent and disjoint change. Then, may be after leader election and topology Consolidation, new changes neglected and the algorithm does not operate properly. Then we must consider a mechanism for two concurrent and separated processes. Derhab algorithm maintains additional information in each node, one is process start time and the other is the time that its node received special reference level .then we can refer to concurrent or separation of two processes by Overlapping or non-overlapping time.

In derhab algorithm, each node maintains two data structure: partition index (pi) and height. Partition index (pi) contains three parameters ($Certain_i$,Tc_i,lid_i). First parameter shows stability of the node. If the node misses all its output edges and is not proved yet, its value is zero. Second parameter shows the time that lid started DAG tree building. Last parameter is leader identifier. Height in this algorithm is shown like:($[Tb_i,Te_i]$, oid_i, r_i, δ_i, id_i). First parameter shows start and end of the process Interval. Other parameters are similar to malpani algorithm. In Derhab algorithm[6], Comparison of index partitions for acceptance or rejection of a new process is used. If I node receives a message from j node that $PI_j > PI_i$, accept it and Otherwise, it will prevent the continued expansion. The greater condition is defined as:

$$PI_j > PI_i \equiv ((Certain_j > Certain_i) \cup ((Certain_j = Certain_i) \cap ((Tc_j, lid_j) > (Tc_i, lid_i))))),$$

And

$$(Tc_j, lid_j > (Tc_i, lid_i)) \equiv (Tc_j < Tc_i) \cup (Tc_j = Tc_i) \cap (lid_j > lid_i))$$

Note that in the comparison of tc's, the comparison is done reversely. It means the one that has lower time, is considered greater. With this comparison, Derhab algorithm can neglect concurrent newer processes until initial

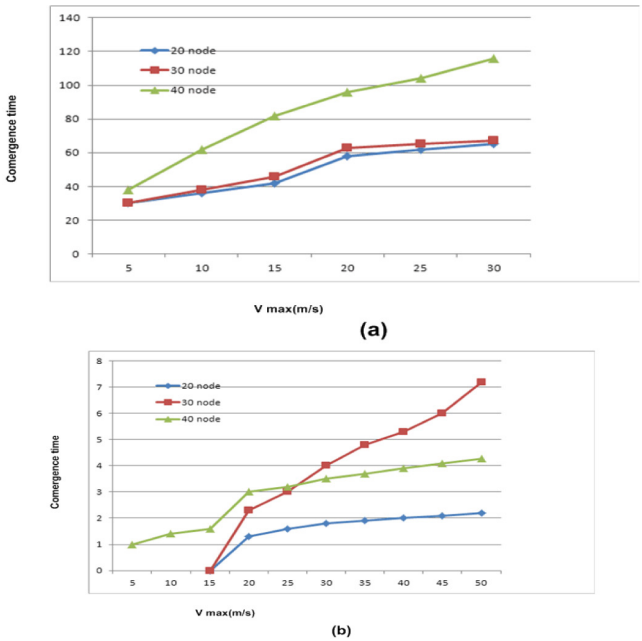

Figure 2. Convergence time (a) Malpani algorithm, (b) Derhab Algorithm.

processes terminates and the leader elected. More details and Derhab algorithm is presented6.also, proof of this algorithm's correct operation is presented6.

Figure 2 shows the comparison of Convergence time of Malpani and Derhab algorithm. Section a is for Malpani algorithm and section b is for Derhab algorithm. As you can see, Derhab algorithm with time difference of 10 to 100 times terminates faster than Malpani algorithm.

3.2 Extreme-Finding Algorithms

Of course algorithms like Derhab[6] elect the leader correctly and its self-stabilization is proved, but there is a fundamental weakness. These algorithms elects the leader randomly, i.e. a node that detects topology change, announces itself as the leader. Meanwhile in many occasions, leader election is based on node parameters like computational power or energy left. Referred algorithms in 3.1 do not have possibility of change and adaptation to this requirement. In the following we introduce two algorithms for selecting based on extreme-finding.

3.2.1 Leader Election Algorithm for Ad

Algorithm LEAA[4] present a different approach to solving the problem of extreme-finding in the election of leaders. In this algorithm, they used termination detection for diffusion computations that have been introduced by Dijkstra and Sholten. In this algorithm instead of node identifier for leader election, an adjustable parameter that can be defined arbitrarily, was used and the algorithm elect the leader through a node with highest value. Diffusion computations algorithm is very simple. Firstly, destination node, transmit E message for computations start. I node that has received this message for the first time, elects transmitter node as its parent and transmit this message to its neighbors. If receive that message again, ignores it .this phase continues until all nodes receive this message. Then, nodes that play the role of tree leaf and do not have any children, send their values to their parents. Then parents with receiving of all messages for their children elect the greatest value and send them to their parents. This process continues until the message finally reaches the start node. Then this node elects the highest value node and sends its value to all. Figure 3 shows this operation based on Dijkstra and Sholten algorithm.

In LEAA five messages are defined .election message that is sent for leader election process announcement from father to children. Ack message that is send from

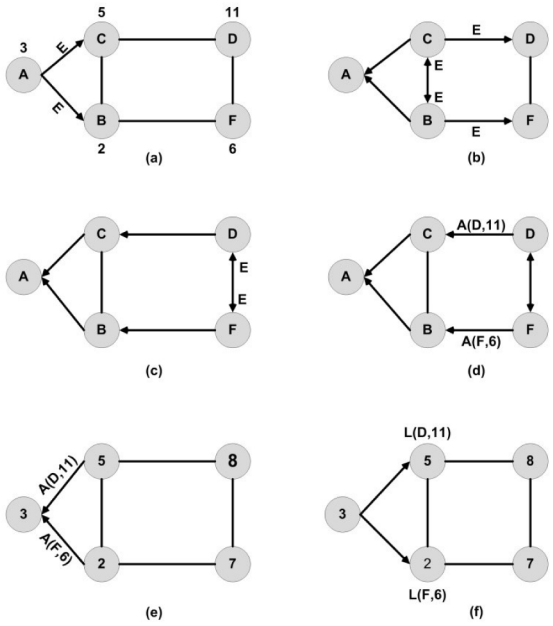

Figure 3. LEAA algorithm Based on Dijkstra and Sholten algorithm[4].

children to parents until the child announces the highest value node to parents. If a node had a father previously, receive a message from its neighbor, uses ack message until announce to that node, do not make a parent relationship. Election message is sent when leader elected until leader announces itself to all nodes. There are two other messages namely probe and reply .they assure of neighbors and leader by alternate receiving and sending of messages.

In this algorithm to avoid implementation of concurrent computation process an identifier is used that is showed by numi. Each node is only allowed to participate in a diffusion calculation at a time. Calculation index is defined as $<num_i,id_i>$ and contains calculation identifier and node's unique identifier. Each node that is going to start a calculation, elect a value for numi that is higher from previous values. When a node is computed receives a message greater than its calculation index, terminates its calculations and becomes a member of the calculations with the larger index. Comparison is done from left to right, i.e. first num amounts are compared and if they are equal, nodes identifier amounts compared. In Figure 4 for this purpose, control of some concurrent calculation in LEAA algorithms has been showed. When A node receives a message from B node with greater calculation index, Then publish it.

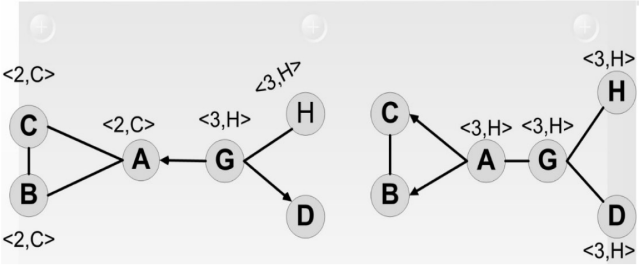

Figure 4. Control multiple concurrent computations in LEEA[4].

This algorithm described and proved for detail[4]. It reviews clearly partitioning states and joining to the network. In addition to that, multiple simulated samples for evaluation of message overloading and leader election average time is presented. These simulations show that in 97–99 percent of times, the nodes have leader and algorithms operates properly. Also, a proof for correctness of this algorithm is presented with temporal logic.

3.2.2 Candidate Base LEAA algorithm (CBLEEA)

CBLEAA algorithm with LEAA idea tries to improve performance and reduction of messages transmission overhead and as result reduction of energy consumption. The main idea of this algorithm is maintaining more than one node as candidate for being leader. In this case, if the leader is not available for any reason, the nodes elect next candidates as the leader. In this case, the overhead due to the re-election of the leader is removed.

In this algorithm, each node instead of maintaining the leader, maintain a five parameter of nodes in order of their priorities and values. When, in the process of leader election, a child want to announce its most high valued child to parents, sends a five set of five high valued nodes .correct leader election process is like LEAA algorithm except that nodes instead of one nodes data, sends information of five nodes.

In Figure 5, leader election process has been shown. As you can see, the process is like LEAA. In this Figure, value -1 indicates null or lack of existing candidates on that five places. First element leader, that is the most valuable node, is elected.

In this algorithm, the nodes after some time periods and due to lack of reply message receiving from the leader, sends a message for the second candidate until they be aware of its presence. If these candidates be present, they

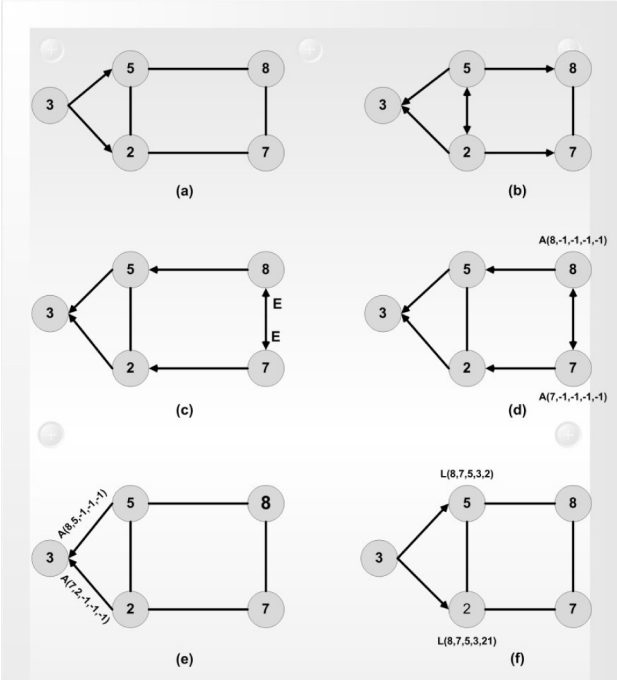

Figure 5. Leader election process in CBLEEA {5}.

will be elected as the leader. This procedure is repeated so that no candidates are presented; in this case leader election process will be started from initial stages. During the joining of two separate pieces together, the list of these five candidates exchanged between nodes and among these 10 candidates, 5 most valuable candidates would be elected. The rest of the algorithm is similar to LEAA and is very simple.

In this paper with carried simulations, leader election time is showed in scenarios that the nodes move in it and topology changes are so high. This algorithm in some cases is five times faster than LEAA algorithm. Therefore LEAA algorithm, with some change and modification of LEAA algorithm, increase its efficiency so much. This efficiency increase in ad hoc networks and especially wireless sensor networks, that low consumption of energy is considerable, is so useful.

3.2.3 Secure Extreme Finding Algorithm (SEFA)

SEFA algorithm is a hierarchical method of top down periods for leader election that is placed in Extreme Finding Algorithms. In this algorithm, it is assumed that there is a Synchronous Distributed Systems and all the candidate nodes return same amounts in various election groups. Synchronous means that along a period of implementation, nodes must be fixed. But replacing of nodes after or

before the algorithm running, do not have any barriers. In this algorithm security consideration is also regarded. It means that messages are encoded and have digital signatures. For observing security issues refer[18].

Method: in particular, a Common Election Algorithm (CEA) is defined as a function that maps a set of m parameter to a real number (f:Rm→*R*) and the one who has the highest CEA is elected as leader. Firstly, each node sends its weight evaluation parameter to all its neighbors and enables a Timer (TE). After finishing of this timer, the node investigates received packets and calculates the weight of each node. If node has the highest weight, elects it as the parent and sends approve message to it. If it has the greatest weight, wait t0 time and listens to messages from its children. Then detect its children and send a parent message and wait for some time.(if the node have a parent, will receive this message)

This process continues for several rounds and in each period, only the nodes without parents participate. The other difference is that in the l round, instead of sending it to neighbor nodes, send it to l steps farther. This process is repeated until a leader with the maximum weight remains. Expectation rate reminded nodes in each round correspond with l. In the last round, the leader node notes the lack of other node as the leader candidate and announces itself as the only leader. Election of this timer and its effect on special cases is investigated in (15).on that paper, for the number lower than 620ms; algorithm success rate in leader determination is zero and for greater numbers of 950ms, estimated 100 percent.

The working of the algorithm is shown in Figures 6 and 7. In Figure 6, initial algorithm in this algorithm is shown and Figure 7 shows the resulted tree after algorithm running. In some situations these algorithms can even be used for hierarchy election.

Algorithm accuracy prove is given[18]. Message complexity of this algorithm in the worst case is o(ln) and

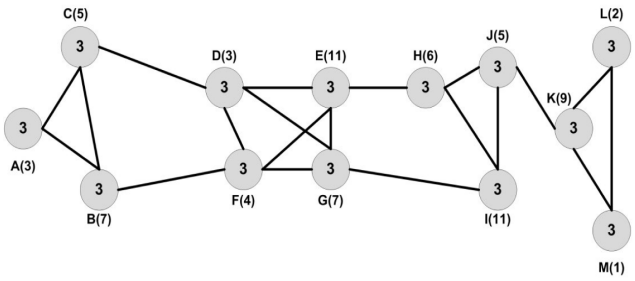

Figure 6. Initial tree in SEFA algorithm[17].

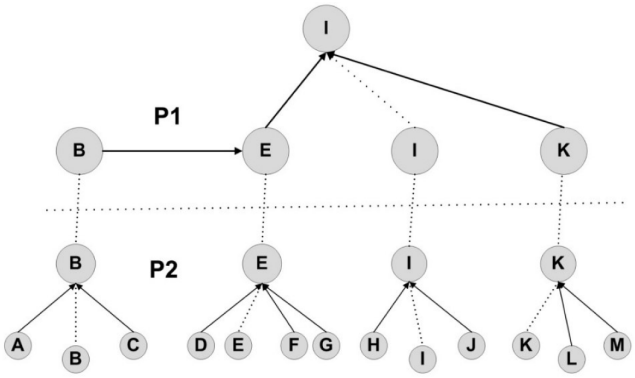

Figure 7. Resulted Tree from SEFA Algorithm[17].

in the best case is o(n) also this algorithm requires synchronous timer and on the other hand, network during the leader election process should be synchronous,means that topology change does not occur in the network. In general, security considerations in this algorithm caused more delay and more additional messages rather than other algorithms.

3.2.4 Secure Preference—BASED LEADER (SPLEA) Election Algorithm

In sefa algorithm, the nodes share a general CEA algorithm; therefore candidate groups were desirable in aspect of all groups but sefa algorithm allows definition of, I elected nodes, a utility function as a map from m related parameter to for I candidate node to a real number. In view of I node, most utility rate from candidate node sets is preferable for leadership. Now, describe this algorithm with placing of I node in various situations:

Initialization state: like sefa algorithm

Election state: like sefa algorithm, with this change that in splea each election message contains a list of I node children in k round .this list is hold by I node and is shown by di,k (Inclusion of list of children of I node in election message allows other nodes with regard to this information, evaluate themselves from I node)

Voting state: in this case, I node runs top function, that calculate utility of each node with consideration of I node and elects w node with highest utility and I sends a vote message meaning w suggestion as a parent to all nodes that received election message from them (meanwhile vote message includes a digital signature that I and w (i) election is added to them).after sending of vote message, I node starts a t timer until receive its own competitors

vote message. With expiration of tv timer, I node enter to announce votes state. Announce votes state: in this case, if I node do not receive vote message from its competitors, reports an exception message. Then, the groups that avoided from voting are obliged to broadcast their votes. I node announces the votes that received from its competitors (i.e. the ones that suggested I as the parent)and hence, an announce message is sent to all that gave election message to I based on obtained votes. Again at a timer starts until all announce messages are collected on behalf of competitors. (Lack of an announce message shows that the Competitor did not get any vote) with expiration of ta time, I node enter to parent nomination state.

Parent nomination state: like sefa, each group of I based on its competitors votes, elects the node with highest vote and sends an ok message to all its competitors. I node enable timer and collect all the messages of its competitors .when I node is not candidate and winner variable makes itself false, when to expires, I node enter to adoption state. Adoption state: like adoption state in sefa, with this difference that, when I node receives n ok message from j node, j and all children of j that are known by election message are added to the list of children. The rest of the algorithm is similar to sefa. This algorithm is the only introduced algorithm that difference of weight parameter in various nodes is considered in it. For instance, we can consider distance Weight parameter to receiver node. In this case each node has a different view towards leader weight but the leader of a node is elected that has more centrality and the sum of other node distances is the smallest towards them.

4. Hierarchical Algorithms

Recently, Adhoc networks have been used extensively due to speed simplicity and lack of structure but with increasing number of nodes in the network with regard to network operation, hop distance of communications would be harder. The solution of total Adhoc network division to clusters include one cluster leader group called cluster head and some other cluster node. Cluster general node relates to network of low level and all cluster head. Meanwhile internal nodes of cluster are related with each other. With regard to unpredictable situation of network structure particularly, missing cluster head causes loss of communication in cluster and even in the whole network. Then, when we miss a cluster head, require a new node election by a leader election algorithm. Some algorithms like14 divide network to clusters and then for each cluster, elect a cluster head. Finally, elect the best cluster head as the leader. Some algorithms with assumption of network clustering, previously deal with leader election problem.

4.1 Election Portion Leader (LPL) Algorithm

Introduced algorithm[14] is a hierarchical algorithm. This algorithm contains three phases. First phase is cluster creation by MCA algorithm .then in the second phase BFA algorithm is used until cluster head become like a loop. Finally the leader with the help of an algorithm like Chang-roberts algorithm is determined in this loop. In Figure 8 a sample of resulted network of this algorithm is showed.

Chang-robert algorithm is a Non-synchronous algorithm of leader election for one way looped network. It is assumed that each node can use two red colors, one as a potential candidate for leader election and the other as a back color means that the candidate has resigned. The algorithm operates this way: each red node can be starter of the algorithm (actually, at the outset of the algorithm, all starter nodes are red) at first each red node sends a token with its specification to Adjacent nodes. If a red node receives a token before algorithm start, then with conversion to a black node, resigns from being candidate. The ones who did not start, remains black and only play the role of a router .if a node receives a token with smaller Characteristic of itself, omit that token, otherwise, received token is sent to the next node and if it reaches to its producer, that node had a greater characteristic than the others and this node can announce itself as the leader. Complexity of this algorithm is $O(n^2)$. For algorithm optimization, each node maintains a list of characteristics of nodes that received tokens from them and instead of comparing the current token characteristic, itself, uses a list and only passes a token with greater characteristic from whole list members. Figure 8 shows resulted loop of LEP algorithm running.

Time-order of this algorithm is equal to sum of time-order of its phases that is as follows:

$$O(LEP)=O(MAC)+O(BFA)+O(Chang\text{-}Robert)$$
$$O(LEP)=O(n)+O(kn)+O(k^2)=O(kn)$$

4.2 Special-purpose Hierarchical Algorithm

Papers[15,18] deal with hierarchical algorithm for leader election. In these algorithms, it is assumed that the network

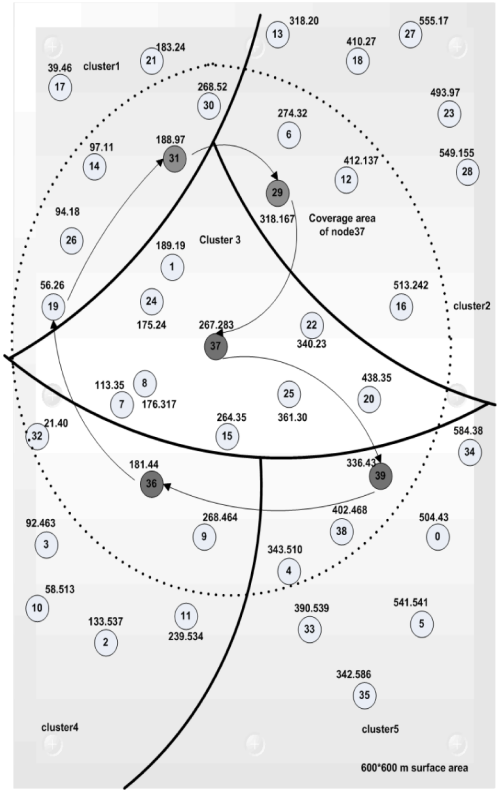

Figure 8. The Loop Algorithm LEP[14]

is divided to clusters. Figure 9 show network structure in the algorithm. This algorithm is applicable to a network with structure like Figure 9.

Each node has an ID that is consisted of two parts: node ID and cluster ID. It is assumed that nodes priorities increases with decrease of node ID and cluster ID. The node that both of these parameters are zero on it called root node, that has the highest priority. The network is made as follows:

Root node sends setup message to its neighbor node. Each node after receiving setup message compares its cluster ID with message cluster ID. If it was not equal, leaves the message and if it was equal, because, the node can be related cluster leader of a network of low level, repeats the second stage until there is not any other visited node in the network. Each node contains a characteristic called NWV or node priority or weight. For the nodes that are in the high level network, it is assumed that during algorithm running, leader election is fixed. (before and after the election round, the node can move) for low level networks nodes, it is not obligatory that assume the nodes are fixed or it is not obligatory that low level network topology be stabilized. Therefore nodes in the low level network can replace during algorithm

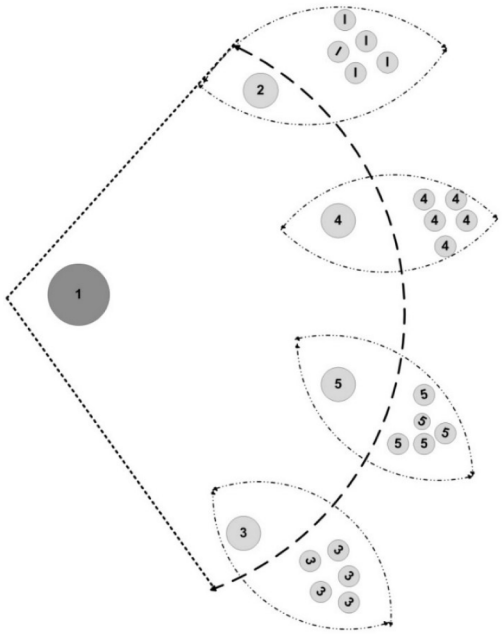

Figure 9. Network structure[15].

running and the network can divide or combined. High level algorithm is a synchronous algorithm then requires a synchronous timer. Each node periodically sends a help message to its neighbors in order to discern real existing nodes from others.

Leader election algorithm in high level network[17] made optimizations on sefa by some mechanisms and meanwhile by experiments output, fixes the period interval optimally. Like sefa algorithm, divide each round of election to five periods. Period vice is a node with the second highest nwv in the last round of election. In this round, each node checks whether is vice node or not? When a leader node loss, this node can be elected as a leader directly. If this occurs, the node that has become leader sends a leader message to the whole network. Otherwise, the node waits for receiving of leader message.

4.3 Leader Election Algorithm in Low-level Network (Election of Collision Avoidance)

This phenomenon that several nodes together find that leader node missed is ordinary; unlike synchronous algorithm this phenomenon can extensively increase time and message complexity. Hence, we design collision avoidance algorithm that assure us just one node is the starter of the new round. In this algorithm, internal

nodes of the network elect their leader by replacing of the message. Based on this, each node has the three following states:

UNVISITED: this node never been met by message election.

TRANSFER: this node is visited and there are nodes that have unvisited states and are connected to this node.

VISITED: this nose is visited and no unvisited node is connected to it. When several concurrent nodes did not find the leader, do not perform sending of election message on that time, hence, the node adjust an appropriate timer with NWV. Each node waits for at least some time before message sending. Then decides to lose the election opportunity or with considering receiving message, start an election process.

We describe the algorithm in two phases of leader announcement and election:

Election phase:

1. When a node wants to send a message, first checks it. If there was any node in its neighborhood in unvisited state, sends elect message to it otherwise sends it to a node with unvisited state and higher priority.
2. The node that does not have any unvisited node in its neighborhood, just have one transfer node. Goes to visited state and sends a message to transfer node. if number of nodes was more than one, sends the message to a node with higher priority.

Leader Notification phase:

After leader election, elect message exist in a node and this node start leader announcement with open elect message. In this phase leader message is sent to the whole network. With regard to more node replacement in Adhoc networks and the nodes are separated from each other, therefore we use this solution. Firstly, if a node in election phase leaves the network cannot return to the process again and at any time that returned sends a hello message to its neighbor and then, the node that received the message, based on comparison with leader decides to send a leader message to new node or broadcast new leader message. Secondly, there is an issue called network disconnection and the network is divided to some sub-networks. Each sub-network elect a leader and when discovers a node from another network behave like the first part. Actually, missing and returning of a node is only special state of the network disconnection.

5. Election of k leader

Paper[12] deals with election of the best k leader based on node's weight amount (that can be a sign of each parameter in relation with their specifications).the algorithms works as follows :for approaching to message efficiency, elect coordinator nodes locally and then consider this node as red node and the other nodes considered white and then, red nodes with some modifications in the application of diffusion calculation method that has been described above, collects information of other nodes. From then, just red nodes cooperate with each other and elect k leader. Meanwhile for saving of message cost, each white only participate in distributed calculation of a red node. This algorithm has three phases as follow:

- The first phase of the red node selection:

Each node sends an elect message with its weight to its adjacent nodes. Each node that had received elect message from all its neighbors, votes to the node with highest weight by a vote message. If a node receive a vote message from all of its adjacent nodes, changes its color to red. At the end of the first phase, there are some red nodes in the network that are not aware of each other.

- The second phase distributed computation by red nodes

Each red node starts a diffusion calculation separately and without other red node's awareness. Therefore each red node create a tree that its root is itself and with running diffusion calculation like above, extends its tree. One difference is that each red node during e message sending sends its weighted value also. Meanwhile each node with re-receiving of e message regardless of who is the sender of red node of this message just saves information of new e message. But do not participate in diffusion calculation of new e. the other difference is that each child node instead of returning node value with highest weight, return the information of all its children and also information of red node that did not participate in the calculation but just saved them. Therefore each node just participates in the diffusion calculation of one red node and this causes saving of message cost. At the end of the second phase, each red node knows about some red node and visited a subset of network nodes.

Third Phase: data integration and election of k leader

In this stage, red nodes exchange their information of network with each other until finally a node with greatest

weight has the information of all other nodes. Then, this node arranges them in order of weight and announces the first k nodes to all other nodes. The procedure is so that if each red node finds that there is another red node with lowest weight, sends its collected information from the second phase to another node that knows it. Then, receives the message of these nodes and omit this group of their red nodes list, then these node investigate whether the red nodes that they know have the lowest weight or not.

5.1 Failures of nodes in leader k algorithm

- Failure of red nodes: in the first phase with adoption of support method, dealt with this problem. When a node is elected as a red node, a node with the greatest weight would be elected as support node and is called green node. With consideration of low number of red nodes among all nodes and also probability of more certainty of them, this method is efficient also this method will not be helpful in network separation (fragmentation).
- Failure of white nodes: in the first phase, a node removes an adjacent node from its neighbors list until avoid the unlimited expect for vote message arrival of that node.
- Failure in the second phase : when the Corrupted node in the tree do not have a child, uses ack message until announce that node, it has not made parental relation,if I node that is corrupted has a child and send an e message to some of them, child nodes can detect failure of I node and in this case, send their data to the tree root(red node is the sender of e message)
- Failure in the third phase: Red nodes are only required to send and receive data. Then failure of white nodes does not have any affect in this phase on algorithm. Also, nodes failure can change network topology and influence routing protocol, this issue is not dealt with[12].

5.2 Performance evaluation of leader k election algorithm

For performance evaluation, the algorithm was simulated in two various versions. One is prop scenario and the other is prop-ft scenario. In addition, for showing of algorithm advantage, a naive algorithm for election of the best k leader was simulated. Each node runs calculation diffusion for information collection of all nodes and then specify k leader. Meanwhile for simulation along with nodes failures, nodes failure rate is considered 15%. with considering the above Figure 10, we conclude that when the number of nodes increases, number of messages that is exchanged increase. Meanwhile message complexity of leader k algorithm is $O(n*d+n*r)$ d is the average of adjacent neighbors of a node. And r is the number of red nodes in the network. Figure 10 shows the exchanged message in leader k algorithm. These Figure shows that the algorithms are scalable. Meanwhile with increase of number of nodes and with regard to 15% failure, still election accuracy for the leader is above 65%.

It also has objections despite a good performance. First, all states are not dealt with during network fragmentation and failure. Also, simulation is not performed during failure in order to evaluation of algorithm performance. On the other hand, this algorithm depends to routing protocol in the lower layer.

6. Review of the proposed algorithm.

For review of the algorithms some factors are important such as time-order of them, their dependence to routing protocols, their resistance against topological changes and etc. In the following we review them with classification of these algorithms:

6-1 investigation of leader algorithms in single-hop network

These algorithms are designed to communicate directly without intermediate nodes. Based on their kind of performance, it classified as: forgetful, uniform and non-uniform algorithms. For evaluation and performance of these algorithms, time-order parameter is

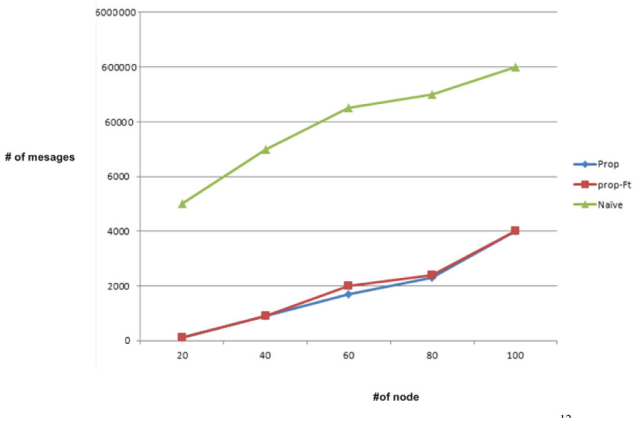

Figure 10. Number of messages exchanged in algorithms k Leader[12].

considered as a factor for comparison of them. In evaluation of forgetful algorithms, consider three states that are encountered by the algorithm and view their time-order.

First state: when we are aware of number of nodes. This algorithm has the time-order of e

Second state: when we do not have any information about number of nodes and only know the upper bound of nodes. In this case it is assumed that the upper bound of the nodes is u and series of sending probability is considered $D^{\infty}_{[\log u]}$. This algorithm has the time-order of O(log u).

Third state: when we do not have any information about number of nodes and their states. This algorithm has the time-order of O(log n).

As it is obvious from time-order of these algorithms, when we have complete information about channel state and number of nodes, algorithms deals with the problem of leader election in a more reliable manner with lower time. When we are speaking about probability, like second and third states, place in an unreliable situation in aspect of finishing time and proper election. In the next section, we deal with non-uniform algorithms and do not suppose of counting and statistical information like the third state of channel condition. As we have said in the description of these protocols, these protocols are constituted of three phases and with regard to performance of binary search in the second phase, time-order of this algorithm is O (loglogn+loglogn) and in the third section we deal with uniform algorithms that alike the previous protocols are made of three phases and the time-order of it is O(loglogn+loglogn). Now with regard to analyzing of the algorithms in single-hop networks and their performance, review them. In these networks all the nodes are aware of one node transmission in a channel and the method in these nodes are continuous repetition and time division by smaller cuts until the time is not dividable and the node remains alone until elected. The main weakness of these protocols is their lack of flexibility towards changes that occur during the protocol running in the network. Although these protocols present basic and general rules for leader election problem but due to broadcast and extension of wireless networks and lack of practical performance because of lack of flexibility, are released.

6.1 Investigation of Leader Algorithms in Multiple-hop Network

In multiple-hop networks, leader election protocols based on used parameter in the election classified as random and extreme-finding. First category includes algorithms like TORA, MALPANI and Derhab. TORA and MALPANI algorithm are similar and like single-hop networks use broadcast characteristic. The basis of these two algorithms is built of DAG tree. So that motion of all nodes is towards destination node. The main difference of them is in destination node, in TORA algorithm this node is the destination node in routing but Malpani has another approach to this problem and elect this node as the leader. Weakness of this algorithm is that topology changes occur in the network and due to lack of this important prediction in algorithm, its termination is not guaranteed. The idea of Derhab algorithm is considering Malpani weakness and approach for covering of promoted sample of Malpani algorithm. With this difference that lower refer level is considered with higher priority vice versa of Malpani idea and has robust reason for its idea i.e. the older is the time of process beginnings, some part of leader election calculation is conducted and consider interfering of newer processes destructive. With regard to self-stabilization specification and its accuracy proof, Derhab algorithm has two main weaknesses. Firstly cannot be promoted to extreme-finding state and secondly require synchronous timer.

In the comparison of random algorithms, we note to this notion that of course these algorithms solved leader election problem with acceptable solutions but there is a criticism on it and that is, they do not consider node-like parameters such as reminding energy and deal with this problem randomly. Extreme-finding algorithm is presented for the covering of this main weakness. From this kind of algorithms, firstly we review LEAA algorithm. These algorithms for covering the random algorithms weakness uses an adjustable parameter like reminding energy and also, for concurrent calculation problem apply a unique identifier for nodes and each node in a time is just obliged to participate in only one diffusion calculation. The strengths of this algorithm are acceptable average running time and running accuracy in the network connection cases.

Another algorithm that is reviewed here is cblea algorithm. The main idea of it is based on lea and tries to optimize energy consumption and is used when energy consumption is so vital. Its strength compared with lea algorithm is election of more than one node as the leader node candidate election. In this case if there was a problem in the leader election operation on a special node, instead of restarting the leader election process, another

candidate would be elected. Another algorithms that is introduced here is sefa algorithm. In this algorithm we can observe security problems and protective approaches in the message sending by various nodes. The strength of this algorithm is application of timer for synchronizing of leader election process. The weakness of this algorithm due to security considerations are delay and sending overload rather than other algorithms. At the end of this section we review based algorithms. The main difference of this algorithm with sefa is algorithm elects the leader that has the highest value of CEA but splea do not elect an optimized leader. The strength of this algorithm is its uniqueness in the application of parameter difference of nodes weights and in the practice with the election of the nodes that has the biggest centrality, referred as the leader.

6.2 Leader Algorithms Evaluation in Hierarchical Network

The main basis of hierarchical algorithms is clustering and resulted graph of wireless networks. And often are used where there is a need for clustering in the network. The first presented algorithm in this section is left. With careful note in the running stages of this algorithm find out that this algorithm apply mca algorithm for clustering in the first phase. In the second phase for making a loop in the cluster head, apply bfa algorithm and finally for the election of the leader uses elected cluster head in a loop of change-Roberts algorithm. Stages of this algorithm show that these algorithms with the help of combination of other algorithms have an optimized solution for leader election. We can also refer to special-purpose hierarchical algorithm in this case. The main difference of this algorithm with previous sample is that, in this algorithm, it is assumed that network clustering is designed previously. This algorithm with optimization of various algorithms and practical combination of them deals with the leader election problem. In technical analysis of these algorithms due to lack of complete investigation, we cannot comment on them carefully.

6.3 Leader k Algorithm Investigation

This algorithm for the beginning firstly separates nodes as coordinator from other nodes. Then, these nodes obtain information from other nodes that are not matched, by distributed calculation method. In the last stage; with coordination and assistance of coordinator nodes perform the election of k leader. Most vital strength of this algorithm is forecasting of conditions that a node failures and presenting a solution for its replacement. The weakness of this algorithm is intense dependence of it to routing protocols so that accuracy algorithm running depends to routing protocols of lower layers.

7. Conclusion and Future Works

Several algorithms studied in this paper. Leader election problem in single-hop networks due to lack of practical operation were not followed either because with increase of wireless networks application and lack of their support from structural common changes, it has been forgotten. In practice, being single-hop for modern extensive networks that their number of nodes are so various, is not assumed. On the other hand human studies are towards reliable methods of optimization both in time and cost. With regard to this principle and network expansion, application of broadcast that is the basis of these algorithms, in aspect of optimization is completely rejected. Some algorithms in multiple-hop networks with the aim of structural changes presented not only supported this aim but also due to use of broadcast mechanism is rejected in aspect of optimization such as Malpani algorithm that solves network changes with inspiration of TORA routing algorithm. Now what is the reaction of Malpani algorithm against concurrent changes? And is it working correctly? Needs more discussion.

Leader election problem in multiple-hop networks solved by acceptable algorithms. For this purpose, Derhab presents an algorithm that is indeed extended form of Malpani algorithm. And with presentation of a correct solution, solves Malpani problem. Derhab cleverly solves concurrent changes problem .but the main problem in that algorithm is that they behave randomly in leader election problem. And nodes consider topology changes detection as being leader, hence this cannot be a reliable method. Therefor studies went to more reliable method of extreme finding rather than random method. In these methods parameters like reminding energy in the nodes are used. The approach that this kind of algorithms applied, is a partially optimized approach because these algorithms believe the candidateship of more than one node for obtaining leadership. CBLEAA algorithm with inspiration from lea algorithm presents an efficient algorithm for leader election for extreme finding. But these algorithms that try to cover older problems, do not consider node failure in their predictions. Hence, hierarchical

algorithms are presented that try to lower the mistakes with presenting of predictions for node failures and their support.

Of course, this category of the algorithm wanted to present a modern method with study of previous algorithms. Leader k and hierarchical algorithms are not investigated precisely for example leader k algorithm operation during node failures or network fragmentation is not investigated properly. For more research and study of these algorithms, we can implement and discuss various combinations.

8. References

1. Fill JA, Mahmoud HM, Szpankowski W. On the distribution for the duration of a randomized leader election algorithm. Ann Appl Probab. 1996; 6(4):1260–83.
2. Janson S, kowski WS. Analysis of an asymmetric leader election algorithm electronic. Electron J Combinator. 1997; 4(1):1–16.
3. Malpani N. Weleh J, Vaidya N. Leader election modes of the service distribution protocol for ad hoc networks. Lecture Notes in Informatics. 2000; 11–24.
4. Vasudevan S, Kurose J, Towsley D. Design and analysis of a leader election algorithm for mobile ad hoc networks. Proceedings of the 12th International conference on network protocols; 1987. p. 350–60.
5. Rahman M, Abdullah_Al_Wadud M, Chae O. Performance analysis of leader election algorithms in mobile ad hoc networks. International Journal of Computer Science and Network Security. 2008; 8(2).
6. Derhab A, Badachm N. In Highly Dynamic AD Hoc Mobile Networks IEEE Transactions on parallel and Distributed Systems. 2008; 19(7):926–39.
7. Nakano K, Olariu S. A survey on leader election protocols for radio networks. Proceedings of the international on parallel Architecture. Algorithms and Network (ISPAN 02); 2002.
8. Nakano K, Olariu S. Randomized O(log logn)_ round leader election protocols in radio networks. Process of International Symposium on Algorithms and Computation. (lncs 1533); 1998. p. 209–18.
9. K.Nakano and S.Olariu(2000), "A Randomized leader election protocols for ad hoc Networks" SIROCCO, Carleton Scientific pp: 253-267.
10. Nakano K, Olariu S. Randomized Leader election protocols in radio networks with no collision detection. Lect Notes Comput Sci. 2000; 1969:362–73.
11. Nakano K, Olariu S. Uniform leader election protocols for radio network. International Conference on Parallel Processing. 2001. p. 240–49.
12. Raychoudhury V, Cao J, WUIOP W. K leader election in wireless ad network. Proceeding IEEE 17th international Conference on Computer Communications and Networks (ICCCN); 2008. p. 87–92.
13. Park VD, Corson MS. A highly adaptive distributed routing algorithm for mobile wireless networks. INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution; 1997.
14. Dagdeviren O, Erciyes K. A hierarchical - leader election protocol for mobile ad hoc networks. ICCSvol.5101-509-518; 2008.
15. Zhang G, Kuang X, Chen Z, Zang Y. Design and implementation of a leader election algorithm in hierarchy mobile Ad Hoc Network. ICCSE '09. 4th International Conference on Computer Science & Education; 2009.
16. Vasudevan S, DeCleene B, lmmerman N, Kurose J, Towsley D. Leader Election Algorithms for wireless Ad Hoc Networks. Proceedings DARPA Information Survivability Conference and Exposition; 2003. 261–72.
17. Zhang G, Chen J, Zhang Y, Liu C. Research of asynchronous leader election algorithm on hierarchy ad hoc network. Proceedings of the 5th international Conference on wireless Communications. Networking and Mobile Computing. 2009 Sep 24–26. p. 2929–32.
18. Zargarnataj M. New election algorithm based on assistant in distributed systems. AICCSA '07. IEEE/ACS International Conference on Computer Systems and Applications; 2007. p. 324–31.