A Stable Covering Set-based Leader Election Algorithm (SCLEA) for Mobile Adhoc Distributed Systems

Sanaa A. Alwidian
Department of Computer Science and Applications
Faculty of Prince Al-Hussein Bin Abdallah II for IT
The Hashemite University
Zarqa 13115, Jordan

Alaa N. Alslaity
Department of Computer Science
School of Computer and Information Technology
Jordan university of Science and Technology
Irbid 22110, Jordan

ABSTRACT
Leader election process is known as the task of selecting a particular node or a process to be a unique leader of the distributed system. Many algorithms were proposed for leader election in classical, wired distributed systems. However, with the advent of wireless communication technology, the domain of distributed computing becomes much wide, and the concept of leader election in such environments has been changed due to the dynamic topology resulted from nodes' mobility. The existing classical leader election algorithms do not adapt well in mobile ad hoc environments. In this paper, we propose a new leader election algorithm that is conscious about nodes' mobility and the dynamic topology of ad hoc networks. The main idea of our algorithm is to select a subset of the nodes to participate in the election process, the selected nodes should ensure coverage of other nodes and that are of low mobility. We show through mathematical analysis that our proposed algorithm, the "*Stable CoveringSet-Based Leader Election Algorithm (SCLEA)*" outperforms any other algorithm that depends on the simple flooding to perform leader election. The enhancement of our algorithm is advent in terms of reducing the message overhead associated with leader election process and minimizing the number of redundant ELECTION messages as much as possible.

General Terms
Leader Election Algorithms, Distributed Systems. Mobile Ad hoc Networks

Keywords
Velocity, MANETs, Distributed System, Leader, Leader Election, ELECTION message, CoveringSet

1. INTRODUCTION

A distributed system is defined as a system that consists of a collection of independent computers (or processes) located at different geographical areas [1]. Due to the distributed nature of such systems, processes are expected to have equal responsibilities and interact with each other through message passing with a desire that there would be no centralized process that control and manage the system [2]. However, with this-time technology, people recognized that this desire could not be completely fulfilled, and there is a need to have an independent computer (or process) to be assigned an extra load and to be responsible about the coordination and management of the system. This particular process is referred to as the *leader*.

In distributed systems, a leader is responsible to act as an initiator and a coordinator and to handle a specific job such as directory server, token regenerator and central lock coordinator [3]

The process of selecting and assigning a unique leader in a distributed system is known as *leader election problem*. In the literature, several leader election algorithms have been proposed for classical distributed systems [4], [5], [6], [7].

However, with the advent of wireless communication technology, the domain of distributed computing becomes much wide, and the concept of leader election in such environments has been changed due to the dynamic topology resulted from nodes' mobility. A network that consists of mobile nodes that move arbitrarily causing the topology to be dynamic and unpredicted is known as a *Mobile Ad hoc Network (MANET)*[8] .Unlike the classical wired systems, MANETs have challenging characteristics including mobility, limited bandwidth and constrained resources (such as battery power). Such features make leader election in MANETs a non-trivial mission.

Many leader election algorithms have been proposed to work in ad hoc environments. However, most of them are described as extrema-finding algorithms in which nodes are assumed to have a unique ID numbers without considering any other criteria such as mobility or computational capability [9], and the elected leader is the node with the largest ID number. In order for a leader election algorithm to be deployable in MANETs, the algorithm should take into consideration the limited resources and the mobility of nodes (not only the node ID) as key factors to elect the best possible leader.

In this paper, we propose a new leader election algorithm that is conscious about nodes' mobility and the dynamic topology of ad hoc networks. We refer to our proposed algorithm as the: Stable CoveringSet-based Leader Election Algorithm for Mobile Ad Hoc Network (SCLEA).

The main idea of our algorithm is to make preferences among nodes such that only a subset of nodes that provide coverage of other nodes and that are of low mobility are chosen to participate in the election process to elect the best possible leader. The best leader node is the node with lower velocity than any other available nodes. Depending on the velocity of nodes as a major factor to elect leaders ensures high stability of the network.

Proposing this algorithm was motivated by two issues: first, the need to reduce the communication overhead in mobile ad hoc environments that is resulted from the redundant exchange (or broadcast) of messages (ELECTION and OK messages in the context of leader election). Such extra

transmission of messages will consume the limited resources of the ad hoc network such as battery power, bandwidth and buffers capacity. The second issue that motivate us is the need to accommodate with the dynamic topology of mobile ad hoc environment which resulted from the free and arbitrary movement of nodes. Such free movement, if not taken into consideration, will have negative impacts on the leader election process, and might result in either an incorrect election (i.e. the election of a high-mobility node), or an early crash of the leader (due to link breakages caused by the high mobility of nodes).

During the design of our proposed algorithm, we take the two previously mentioned issues into consideration, and we come up with an algorithm that is stable and that depends on a selected subset of nodes (called *CoveringSet*) to participate in the leader election process instead of allowing all nodes to do so.

The rest of this paper is organized as the following: Section 2 discusses the related work. The objectives and assumptions of the proposed algorithm are discussed in Sections 3 and 4, respectively. Section 5 illustrates the proposed algorithm, and a mathematical analysis of the proposed algorithm is provided in Section 6. Finally Section 7 concludes the paper and provides future directions.

2. RELATED WORK

Leader election is the process of electing a node (or a process) as the coordinator of a specific task that is distributed among several nodes (processes). Leader election is not a recent problem, and there are many algorithms designed for classic, static distributed systems. Bully algorithm designed by Garcia-Molina [10] is one of the most important traditional election algorithms. Bully algorithm works as follows: when a node *N* detects the crash of the current leader, it sends an *ELECTION* message to all other nodes with *ID*s higher than *N*'s *ID*. If no node answers, *N* wins the election and declares itself as a leader. However, if one of the nodes with higher *ID* responds, *N* takes over. The higher-up node that receives the election message sends an *OK* message back to the sender to announce that it is available and will take over the election. The receiver then holds an election unless it already holding one. This process is repeated until all nodes give up but only one, which is eventually the newly elected leader. In Bully algorithm, the best case happens when the node that detects the leader crash is the node with an *ID* just below the crashed leader. In this case, the algorithm requires *n-1* messages (where *n* is the total number of nodes in the system). In the worst case, that is, when the lowest-ID process detects the crash, the algorithm requires $O(n^2)$ messages. In addition, this algorithm is not deployable in mobile ad hoc networks

Mamun et al [11] have proposed a modified version of bully algorithm to enhance the performance of the algorithm by minimizing the number of redundant election messages that are used to discover and elect the leader. The algorithm proposed in [11] is more efficient than Bully algorithm in terms of message and time complexity, in the best case, it requires n-1 message, while in the worst case, it requires 2(n-1) messages. However, this algorithm also is not deployable in wireless ad hoc networks

Another modification of the bully algorithm had been proposed in [12]. The proposed algorithm is an overhead-aware leader election algorithm that is based on the basic assumptions of bully algorithm but outperforms the former in terms of reducing message overhead and time complexity. The main aim of [12] is to perform leader election by sending a minimum number of ELECTION messages as much as possible. This goal is fulfilled by sorting the nodes in a descending order based on their IDs. Once the current leader got crashed, the node with the next higher ID (after the crashed leader) is elected as the new leader. Applying this algorithm for leader election sufficiently reduces the number of election messages (O (N) in the worst case, where N is the total number of nodes in the system). Moreover, the number of steps required to perform leader election is also reduced, resulting in time saving, as if compared to Bully algorithm. However, since the algorithm proposed in [12] relies on the assumption that the environment is reliable and each node knows about the entire nodes in the system, then it is difficult to apply this algorithm on unreliable environments such as mobile ad hoc networks.

Gerard Le Lann [6] proposed the first leader election algorithm for unidirectional rings. The main idea of ring algorithm is that each node prepares an election message that contains its own ID and circulates this message around the ring clockwise. The process with the highest ID will have the priority and assigned as the leader. Le Lann's algorithm requires n^2 messages to perform the election, where n is the total number of nodes. In addition, this algorithm is a single point of failure and not efficient in mobile wirless environments.

Authors of [13] and [14] proposed leader election algorithms for wired networks. They take into consideration the probability of process crash and link failure. However, they assume strong and impractical assumptions. In [13], the network is assumed to be order-preserving. This means that if a particular message M1 sent by a node N1 at time T1, this message should be received by all nodes before receiving another message M2 that is sent by some other node, say N2, at time T2, where T2>T1. In [14], the authors assumed that the failure of processes occur before election start. Such impractical assumptions make these algorithms not deployable in mobile environments.

Tai Woo Kum [15] proposes a leader election approach to work in mobile ad hoc networks. In this proposed work, nodes tend to make another election of a provisional leader while the original leader is still available and executing. The purpose of this extra election is to replace the leader with the provisional leader when the current leader crashes. Although the approach of [15] can make the system operate fast and might reduce the performance degradation in distributed systems, but this is not guaranteed always and failures of nodes and links can occur

during or after election, making the approach impractical in mobile ad hoc distributed systems, where link failure is of a vital impact in the overall performance.

The authors of [16] proposed two algorithms for mobile ad hoc networks. The algorithms viewed the ad hoc network as Directed Acyclic Graph (DAC) and impose that each connected component of the graph should have a single leader. The proposed leader election algorithms where built based on the routing algorithm TORA [21]. The first algorithm designed to adapt to a single topology changes; this means that a new topology change occurs only after the algorithm has terminated its execution. While the second algorithm is designed for concurrent topology changes, which means that topology changes can occur at any time. Both algorithms require nodes in the network to communicate with their neighbours, thus, the algorithms are deployable in ad hoc environments, however, they have been provided with no proof of correctness.

Vasudevan et al [17] considered the issue of secure leader election in mobile ad hoc networks and proposes a leader election algorithm called the Secure Extrema Finding Algorithm (SEFA). In SEFA, as the name indicates, is an extrema finding algorithm which assumes that all nodes that participate in election process have the same evaluation function that result in the agreement upon the same candidate node. In addition, SEFA requires that the parameters used to select the leader in each round remain constant [17].

In [18], an extrema-finding leader election algorithm for mobile ad hoc networks had been proposed. Although designing an algorithm with an extrema-finding aspect is interesting for environments such as ad hoc networks (because it is important to select a leader with performance-related attributes as computational capabilities, battery power or nodes' velocity), the algorithms in [18] are considered to be unrealistic and not suitable for some applications since all nodes (without any exception) are required to exchange information in order to elect a leader, and this is impractical.

As a comparative summary, the algorithm proposed in [10] is costly and has a high message overhead (reaches to O(n2)), in addition, it is not deployable in mobile ad hoc systems. The approach in [11] does not take mobility of processes into consideration, therefore, it is impractical to be deployed in wireless ad hoc networks. The algorithm proposed in [12] relies on the assumption that the environment is reliable and each node knows about the entire nodes, thus, it is difficult to apply this algorithm on unreliable environments such as mobile ad hoc networks. The algorithms proposed in [6], [13] and [14] require large number of messages to perform leader election, have a single-point-of-failure problem, and rely on unrealistic assumptions that make it hard to be deployed in MANETs. The approach of [15] is exposed to frequent link breakages and failures of nodes that can occur during or after election, while the two protocols proposed in [16] have been provided with no proofs of correctness. The approaches provided in [17] and [18] are both impractical since the former assumes that all nodes that participate in election process have the same evaluation function, while the later assumes that all nodes (without any exception) are required to exchange information in order to elect a leader, and this is not realistic!

3. ALGORITHM OBJECTIVES

The main objective of our proposed SCLEA is to decrease the communication overhead resulted from the redundant exchange of ELECTION messages between all nodes in the network. This goal is achieved by selecting a *subset* of nodes (rather than all nodes) to participate in the leader election process. For any hop, H_i, the subset of nodes is chosen with a guarantee that the nodes belonging to this set will cover all the nodes in the next hop, H_{i+1}. Therefore, along all hops, the nodes will reached via a minimum number of nodes, thus, the message overhead will be reduced as much as possible. The second objective of our algorithm is to elect, as a leader, the most-valued-node among all nodes in the network, rather than electing the node with higher ordinary ID as in the case of the previous leader election algorithms. Since the preference attribute used in this work is the velocity of nodes, then electing the leader with lower velocity will ensure more stability and survivability of the leader. Thus, the probability of the frequent leader crash or the chance that the leader will move and leave the transmission range of other nodes will be lower.

4. ALGORITHM ASSUMPTIONS

For our proposed SCLEA algorithm, we model the mobile ad hoc network as a connected graph that consists of a set of nodes, such that each node is assigned a unique identifier, *ID* and a *VALUE*. The *ID* is used to identify nodes during the election process, while the *VALUE* represents the capacity of nodes which is used to make preferences among nodes during the process of leader election. This value could represent any performance attribute such as the level of nodes' velocity (speed), battery power or computational capabilities. For our algorithm, we depend on *velocity* as a major preference-based attribute, such that nodes with lower velocity (thus, more stability) will be preferred to be chosen to participate in the leader election process. There are cases where nodes have the same capacity, i.e. the same *VALUE*. In these cases node *ID*s are used to break ties among nodes which have the same *VALUE*.

In the graph model of the mobile ad hoc distributed system, each node is represented by a vertex, and the lines (or edges) between nodes represent communication links. Two nodes are said to be connected together if they are positioned within the transmission range of each other, hence, they can directly communicate with each other. It is worth to mention here that the graph is not always connected and nodes might lose the direct communication links toward other nodes. This is due to the fact that nodes are free to move arbitrarily and gets out of the transmission range of any particular node. Therefore, the graph becomes disconnected and changes over time as nodes move. In our proposed algorithm, we only consider those nodes that belong to the graph. That is, nodes that still exist

within the transmission radius of some other node. While nodes that leave all of the available transmission ranges and go out of the network scope will not be considered at all.

In addition, we assume that all nodes are periodically informed about and aware of other nodes. The periodic updates are achieved through the exchange of periodic *HELLO* messages between nodes every *HELLO_INTERVAL* time. In this *HELLO* messages, each node appends its unique *ID* and *VALUE* to inform the neighboring node about its status. Based on this information, each node maintains a neighbor table that contains <*nbr ID, VALUE*> entries. These entries are ordered ascendingly according to the node's *VALUE*. In case there is a tie (i.e. two or more nodes have the same *VALUE*), the order will be based on the node's *ID*. This way of ordering neighbor tables according to nodes' *VALUE* (which represents nodes' velocity) guarantees that the nodes with lower mobility will have a higher probability to participate in the leader election process, thus, the network will be more stable and less prone to link breakages and failures.

5. THE PROPOSED ALGORITHM

In this section, we describe our proposed leader election algorithm, SCLEA. But before we discuss its details, it is worthy to discuss the ordinary situation of leader election that happens in mobile ad hoc networks. In such types of networks, *flooding* is the major technique that is used for communication and message transmission between nodes. Flooding is the process wherein, each node, and upon receiving a particular message, sends the received message again to its neighboring nodes (i.e. the nodes that are positioned within its transmission range) [19]. This process continues until all nodes in the network receive the message (at least once). In the domain of leader election, if the current leader got crashed and a particular source node, *S*, detects the crash, it initiates a leader election process by sending an *ELECTION* message to all of its 1-hop neighbors. In turn, those 1-hop neighbors resend the *ELECTION* message to their entire 1-hop neighbors (that is, 2-hop neighbors of node *S*), and so on. If the network consists of *n* hops, the process of leader election terminates when the nodes of hop number (n-1) send the *ELECTION* message to all of their next hop neighbors, that is, the n^{th}-hop neighbors. It is obvious that the leader election process which is performed depending on the mentioned naïve flooding technique is of a big cost, and has a non negligible message overhead, especially when the number of nodes and hops becomes larger. It can be noticed that a lot of redundant messages are transmitted because all nodes (without any preferences) participate in the election of the new leader. This process is illustrated in Figure 1.

According to Figure 1, it is clear that the number of messages that should be exchanged between nodes using the simple flooding is high, imposing heavy traffic on the network. To overcome this problem, we present an enhanced CoveringSet-based and velocity-aware algorithm that significantly decreases the number of messages (*ELECTION* and *OK* messages) that should be exchanged between nodes to perform leader election task. Message reduction is achieved by reducing the number of nodes that perform leader election

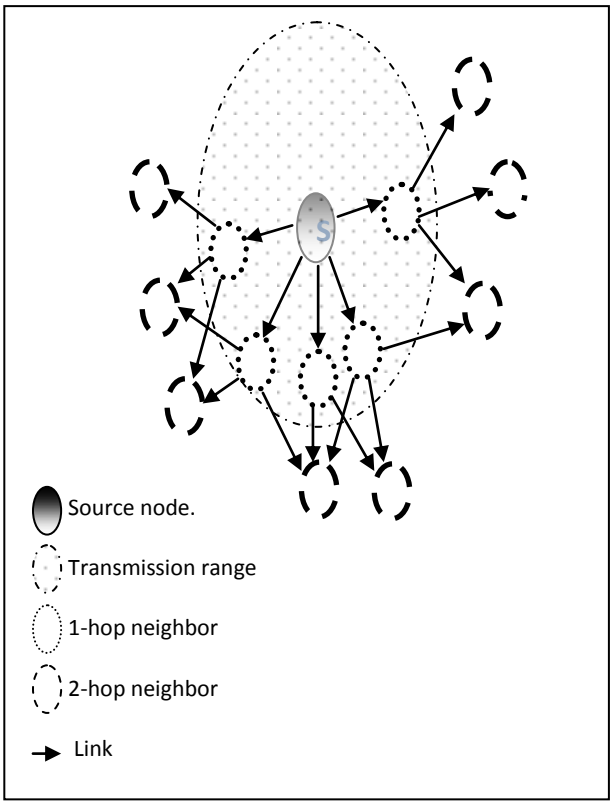


Fig 1: Flooding-based Leader Election Algorithms

process. This is done by depending on the *CoveringSet*, which is a subset of the 1-hop nodes that guarantee full coverage of the 2-hop nodes. The algorithm that we apply to select *CoveringSet*s is described in Section 5.1.

The idea of our proposed algorithm is illustrated in Figure 2. In this figure, we assumed that node *S* detects the crash of the current leader; therefore, it initiates the election process. Instead of broadcasting the *ELECTION* message to all its neighboring nodes (as in the case of the flooding-based algorithms), node *S* creates its CoveringSet, then it *multicast*s the *ELECTION* message to its *CoveringSet*. In turn, each one of the *CoveringSet* nodes (colored with blue in the figure) multicast the *ELECTION* message further to their *CoveringSet* nodes, and so on, until reaching the last hop.

The use of *CoveringSet* nodes introduces a major contribution in our work, since it reduces the message overhead associated with simple flooding-based leader election algorithms.

5.1.Creating CoveringSet

CoveringSet of node *X* is defined as the set of *X*'s 1-hop neighbors that ensures full coverage for the entire 2-hop neighbors of *X* [20]. It is worthy to mention here that the neighbors of any node are sorted in ascending order based on their *VALUE* (in our algorithm, the value is the velocity or speed of the node). Also it is important to mention that the building of *CoveringSet*s is performed in a distributed manner, that is, each node builds its own *CoveringSet* independently from any other node. Therefore, at any time *T*, the *CoveringSet* of a node *A* is different than the *CoveringSet* of a node *B* (unless both nodes have the same neighbors), Moreover, the *CoveringSet* of any particular node *N* at time *T1* is different than the *CoveringSet* of the same node at time *T2* (this is determined instantly based on the mobility status of nodes, and which node have joined the transmission range of *N* and which have departed).

In any hop, the members of *CoveringSet* are chosen carefully such that they provide full coverage of the nodes in the next hop. In addition, they are with lower mobility in comparison with other non-covering set nodes. These two features (i.e. the selection of a **subset** of nodes with **velocity-awareness**) are important strength points of our proposed algorithm, since using *CoveringSets* will reduce message overhead associated with flooding. In addition, allowing nodes with low mobility to participate in leader election will enforce stability in the network, since the lower the mobility of nodes, the more stable the links, thus, the less probable that the leader will got crashed.

When a node *X* wants to send an *ELECTION* message it firstly creates its *CoveringSet* using its neighbor table; starting from the first entry in the neighbor table (i.e. the neighbor with lower velocity) until reaching full coverage for the 2-hop neighbors. For each neighbor, *X* checks if this neighbor adds additional coverage (i.e., if it has path(s) to some of the 2-hop neighbors that are not covered previously by any of the selected nodes). If so, *X* adds the current neighbor to its *CoveringSet* and checks if there are more nodes that are not covered by any node yet; if so, *X* repeats the process for the next neighbor until all 2-hop neighbors are covered. The algorithm used to build the *CoveringSet* is shown in Figure 3.

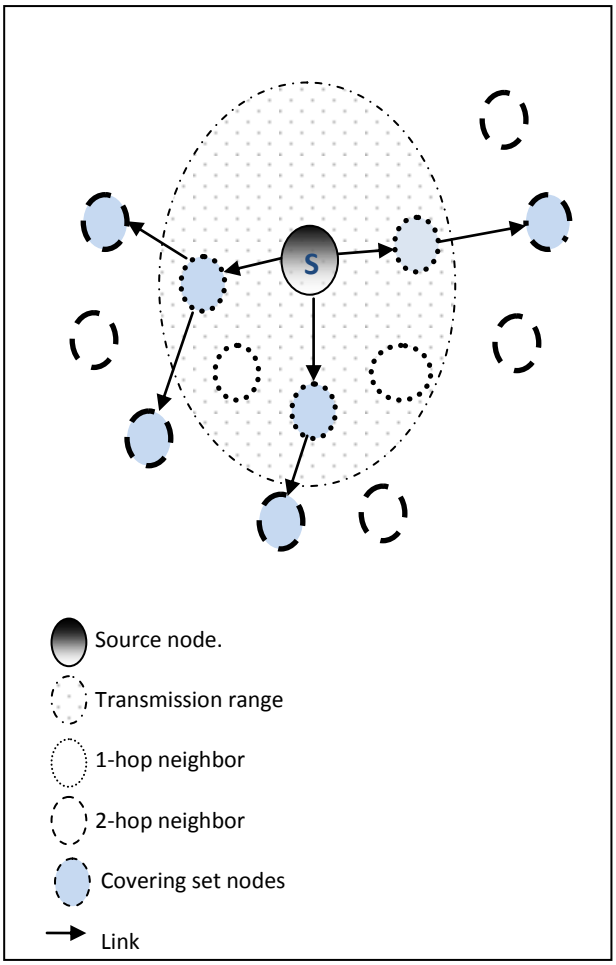


Fig 2: CoveringSet-based Leader Election Algorithm

1. START
2. *CoveringSet*(x) = NULL
3. For each node *m* in *nbrTable*(*x*);
4. If *m* gets additional coverage (i.e. it has a path for some 2-hop neighbors that are not covered previously by any other node)
5. add *m* to the *CoveringSet*(*x*);
6. If all the 2-hop neighbors are covered (i.e. reached) by the *CoveringSet*(*x*);
7. return *CoveringSet*(*x*);
8. END

Fig 3: Building a CoveringSet Algorithm

5.2. Reply Back Phase

As mentioned previously, *ELECTION* messages will be transmitted from one hop to another through the *CoveringSet* nodes until reaching the last hop. At this point, the *ELECTION* phase terminates and the time to reply back and send *OK* messages begins. Nodes are responsible to reply their parents informing them about their existence. The *OK* message includes entries of <*ID, VALUE*> for each node, to inform them about their capacity, precisely, their velocity. The "*VALUE"* piece of information is very useful for the parent to decide which one of its children will be considered as a candidate leader, such that the child with the lowest mobility will be passed to the parent, the parent of parent, and so on until reaching the source node that initiates the leader election. At the end, the node with lower mobility will be declared as the leader. Figure 4 illustrates how the reply back process done.

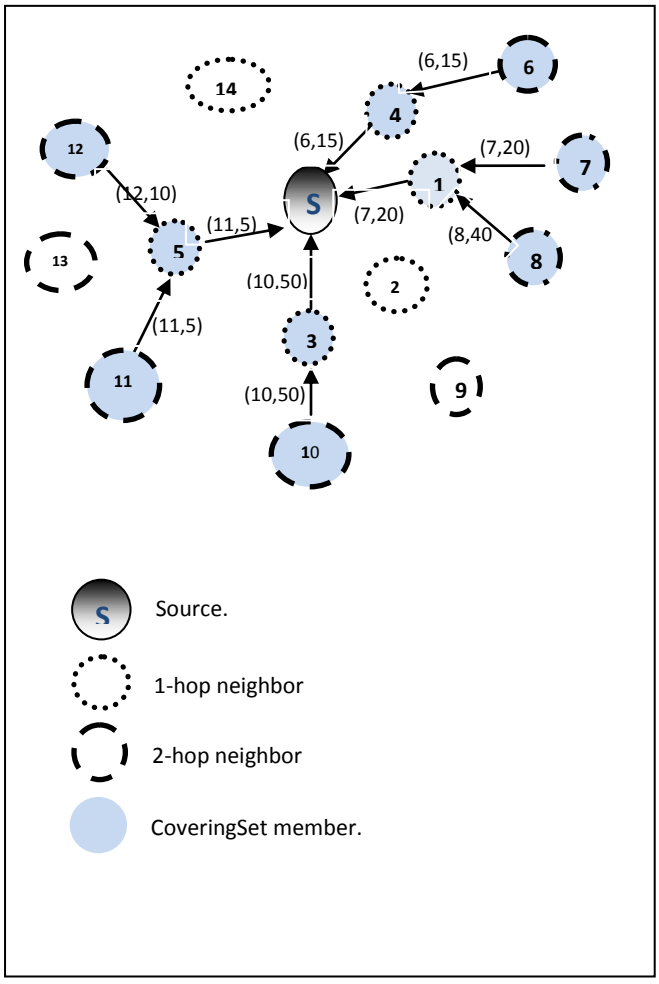


Fig 4: The Reply Back Process

6. MATHEMATICAL ANALYSIS

In the SCLEA algorithm, when a node *S* detects that the leader is crashed, *S* does not send an *ELECTION* message to all nodes in its 1-hop neighbor range (as in the case of flooding-based leader election algorithms) rather, it looks up its *CoveringSet* and sends the *ELECTION* message to the nodes that belong to this set. For the analysis of this algorithm, we will denote the number of nodes constituting the network as *N*. These nodes are distributed randomly in the network in the form of hops, where each hop consists of a particular number of nodes. Let us assume that the number of nodes in the 1-hop range of the source node *S* is equal to H_1, the number of nodes in the second hop is H_2, the number of nodes in the third hop is H_3, and so on. So the number of nodes in the last hop, say the i^{th} hop will be H_i. Further, let us assume that number of nodes that will not participate in the election process (that is, nodes that do not belong to the *CoveringSets* in any hop) is equal to *C*, so the covering set in any hop *i* will be H_i-*C*

The subsequent discussion illustrates message complexity in the worst case for both the ordinary leader election algorithms that depend on flooding to perform election and for our proposed SCLEA algorithm that depends on *CoveringSet* nodes to perform the election of leader.

6.1. Message Complexity in Flooding-based Leader Election Algorithms:

If node *S* has H_1 neighbors in its 1-hop transmission range and H_2 nodes in its 2-hop transmission range and H_i nodes in its i^{th} transmission range, then the number of *ELECTION* messages that will be sent by the source node *S* that detects the leader crash will be equal to H_1 messages, which is equivalent to the number of its 1-hop neighbors. Further, each node in the 1-hop transmission range will send H_2 messages for all of its 1-hop neighbors (that is, 2-hop neighbors of node *S*), therefore, the total number of messages that will be sent by nodes in the 1-hop transmission range will be $H_1 * H_2$. This process continues until reaching the last hop *i*, where the number of messages that are sent by H_{i-1} nodes will be $H_{i-1} * H_i$, where H_i is the number of nodes in the last hop. The message complexity associated with flooding-based leader election algorithms is illustrated in equation 1.

$$\boldsymbol{Msg\ Complexity = H1 + H1 * H2 + H2 * H3 + \cdots + Hi - 1 * Hi} \quad \ldots(1)$$

6.2. Message Complexity in Covering Set-Based Leader Election Algorithm:

If node *S* has H_1 neighbors in its 1-hop transmission range and H_2 nodes in its 2-hop transmission range and H_i nodes in its i^{th} transmission range, then the number of nodes in the covering set for these hops will be H_1-*C*, H_2-*C*, H_3-*C*, … and H_i-*C*, respectively. Where *C* is the number of nodes that are excluded from the covering sets and that will not participate in leader election. The number of *ELECTION* messages that will be sent by the source node *S* that detects the leader crash will be only H_1-*C*, where the number of messages sent by the covering set of the first hop will be H_2-*C*, which is equivalent to the number of nodes of the covering set in the second hop.

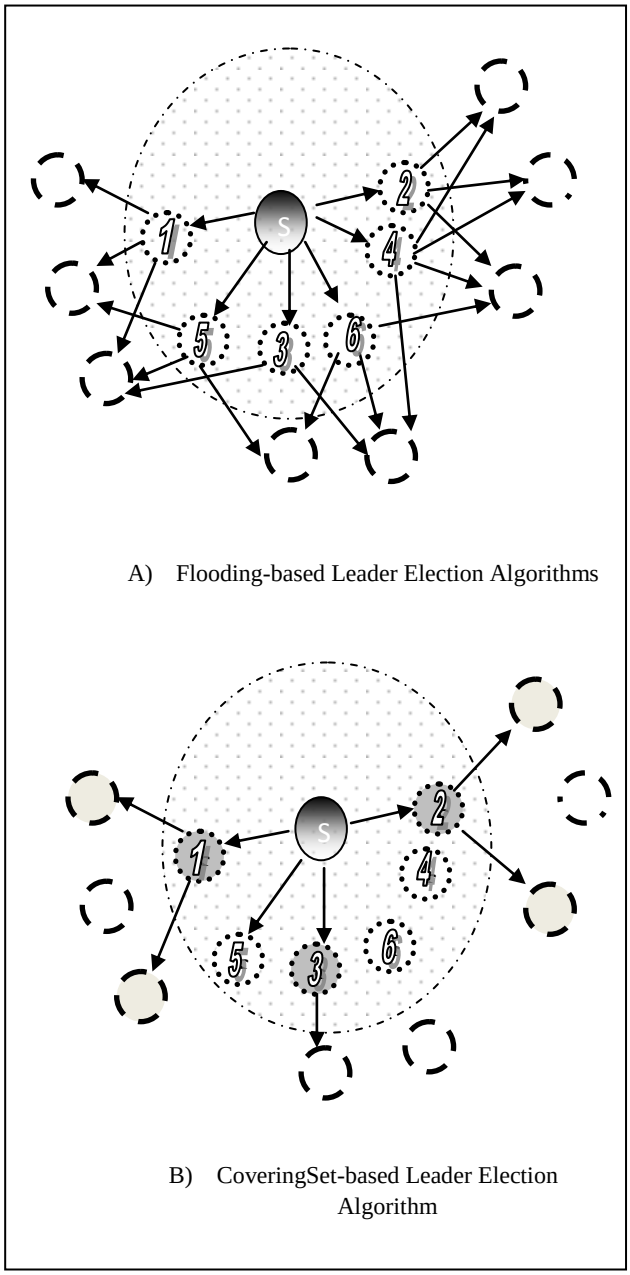
The nodes of the H_{i-1} hop will send messages to their next covering set nodes only. That is, H_i-C messages. Message complexity associated with CoveringSet-based leader election algorithms is illustrated in equation 2.

$$\boldsymbol{Msg\ Complexity = H1 - C + H2 - C + H3 - C + \cdots + Hi - C} \quad \ldots(\mathbf{2})$$

6.3. A Practical Example

In this section, we provided an illustrative example that shows clearly the message overhead (complexity) associated with both flooding-based leader election algorithms and our SCLEA algorithm. In Figure 5-A it is shown that if a node *S* detects the crash of the leader, it initiates a leader election process, and broadcasts an *ELECTION* message to all of its 1-hop neighbors, that is, it sends 6 messages (as H_1=6). In turn, each one of the H_1 nodes and upon receiving an *ELECTION* message broadcasts the message to its entire 1-hop neighbors. For example, node 4 broadcasts the election message to its 1-hop neighbors, in this example node 4 sends 4 messages. However, It is worthwhile to mention here that in the worst case, node 4 might have all of the nodes in the second hop (H_2) positioned within its transmission range, in other words, the nodes of H_2 might be neighbors of node 4, in this case, node 4 will send 8 messages. The same case might apply for each node in H_1, that is, nodes 1, 2, …, 6 may have all of the 8 nodes in the H_2 hop as their neighbors, therefore, each one will send 8 messages, with a total messages sent equals to 48 messages (6*8). In addition, it's clear from the figure that nodes 4 and 2 have three mutual neighbors, this means that there are useless redundant messages, and as the number of nodes increases the number of redundant messages increases. For simplicity, we didn't mention more than 2 hops in the figure. But it should be understood that the previously mentioned case applies for multiple hops, depending on the size of the network and the number of hops.

In the other side, Figure5–B shows the improvement achieved by our proposed algorithm in terms of message overhead reduction. In this example, node *S* detects that the leader is no longer alive, and initiates a leader election process by sending an *ELECTION* message to its *CoveringSet* (i.e. the covering, 1-hop neighbors), namely, nodes 1, 2 and 3, therefore, the message reduction is obvious such that node S sends only 3 messages instead of sending 6 messages (as in the case of Figure5-A). Furthermore, each one of the selected covering nodes forwards the message just to their 1-hop neighbors that provide coverage to their next hop nodes. Comparing to Figure5-A, node 4 and 2 will send 2 messages instead of 7. So the message reduction is evident here in that 5 nodes out of 8 will receive the *ELECTION* message and forward it further.



A) Flooding-based Leader Election Algorithms

B) CoveringSet-based Leader Election Algorithm

Fig 5: A Practical Comparative Example

7. CONCLUSION

In this paper, we proposed a velocity aware, and covering-set based leader election algorithm for mobile ad hoc networks. Our main contribution was in the selection of a subset of nodes (covering sets) to participate in the leader election process rather than depending on all of the nodes in the network (as the case of the traditional leader election algorithms). The selection of covering sets is performed in a distributed manner, that is, each node creates its own covering set independently from any other nodes. In any hop, the covering set nodes are chosen with a guarantee that they provide full coverage to the nodes of the next hop, and chosen to be with minimum velocity among all other nodes. These selection criteria ensure network stability (due to the low mobility of nodes) and reduces message overhead (due to

reducing the number of nodes that participate in leader election process.)

In this paper, although we show the improvement achieved by our algorithm through mathematical analysis, our next step will be to assess our algorithm by simulation using a well-known routing algorithms, such as AODV, to study the impact of covering set nodes with mobility awareness. In addition, we will use more preference-based attributes (i.e. value), for example, battery power or distance of nodes to examine their impact on the leader election process.

8. REFERENCES

[1] Sung-Hoon Park "A Stable Election Protocol based on an Unreliable Failure Detector in Distributed Systems". 2011 Eighth International Conference on Information Technology: New Generations. pp. 979 – 984, April 2011.

[2] Haddar, M.A., Hadj Kacem, A. , Metivier, Y. , Mosbah, M. and Jmaiel, M. "Electing a leader in the local computation model using mobile agents", IEEE/ACS International Conference on Computer Systems and Applications, 2008. AICCSA 2008. pp.: 473 – 480, April, 2008.

[3] Mehdi Mirakhorli, Amir A. Sharifloo, Maghsoud Abbaspour, "A Novel Method for Leader Election Algorithm". The 7th IEEE International Conference on Computer and Information Technology (CIT 2007), pp.452-456, October 2007.

[4] Mina Shirali, Abolfazl Haghighat Toroghi, and Mehdi Vojdani "Leader election algorithms: History and novel schemes". Third 2008 International Conference on Convergence and Hybrid Information Technology(ICCIT'08),pp. 1001-1006, November 2008.

[5] Scott D. Stoller. "Leader Election in Asynchronous Distributed Systems". IEEE Transaction on Computers Journal, volume 49, no. 3 pp.283- 284, March 2000.

[6] G.L. Lann, "Distributed Systems - Towards a Formal Approach", in Proc. IFIP Congress, pp.155-160, 1977.

[7] Randolph Franklin "An improved algorithm for decentralized extrema-finding in circular configurations of processes". Communications of the ACM Magazine. pp.336–337, May 1982.

[8] Nourddine E, Mohammed K, Amine B. "Enhancing AODV Performance based on Statistical Mobility Quantification". The IEEE International Conference on Information & Communication Technology (ICTTA06), pp. 2455-2460, 2006.

[9] Gerard Tel "Introduction to Distributed Algorithms. Second Edition", Cambridge University Press, 1995.

[10] Garcia-Molina H. "Election in a distributed Computing System". IEEE Transaction on Computers Journal, vol31, no. 1 pp. 48-59. 1982.

[11] Quazi Ehsanul Kabir Mamun, Salahuddin Mohammad Masum, and Mohammad Abdur Rahim Mustafa. "Modified bully algorithm for electing coordinator in distributed systems". WSEAS Transactions on Computers, Issue 4, Volum 3, pp. 948-953, October 2004.

[12] Muneer Bani Yassein, Ala'a N Alslaity and Sana'a A Alwidian "An Efficient Overhead-aware Leader Election Algorithm for Distributed Systems." International Journal of Computer Applications, volume 49 no. 6, pp: 10-15, July 2012.

[13] J. Brunekreef, J. Katoen, R. Koymans and S. Mauw. "Design and Analysis of Leader Election Protocols in Broadcast Networks". In Distributed Computing Journal, vol. 9 no. 4, pp. 157-171, February 1996.

[14] G. Taubenfeld. "Leader Election in presence of n-1 initial failures". In Information Processing Letters, Journal vol.33, no.1, pp. 25-28, October 1989.

[15] Tai Woo Kim, Eui Hong Kim, Joong Kwon Kim, and Tai Yun Kim. "A leader election algorithm in a distributed computing system". Proceedings of the 5th IEEE Workshop on Future Trends of Distributed Computing Systems, page 481, August 1995.

[16] N. Malpani, J. Welch and N. Vaidya. "Leader Election Algorithms for Mobile Ad Hoc Networks". DIALM '00 Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications, Boston, MA, pp. 96-103, August 2000.

[17] Sudarshan Vasudevan, Jim Kurose, and Don Towsley. "Design and analysis of a leader election algorithm for mobile ad hoc networks". Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. (ICNP04), pp:350–360, October 2004

[18] K. Hatzis, G. Pentaris, P. Spirakis, V. Tampakas and R. Tan. "Fundamental Control Algorithms in Mobil Mobile Networks". In Proc. of 11th ACM SPAA, pp: 251-260, March 1999.

[19] Abdalla MH, Aamir S, Irfan A, Mike W. "Dynamic Probabilistic Flooding Performance Evaluation of On-Demand Routing Protocols in MANETs CISIS '08 Proceedings of the 2008 International Conference on Complex, Intelligent and Software Intensive Systems, IEEE,pp. 200-204, 2008.

[20] Ala'a N. Alslaity "Stable Neighborhood-Based Route Discovery Protocol For Mobile Ad Hoc Networks", Dissertation, Jordan University of Science and Technology. May, 2012.

[21] V.D.Park and Scott.M.Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", INFOCOM '97 Proceedings of the INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution, pp. 1405, 1997.