An Election Algorithm to Ensure the High Availability of Leader in Large Mobile Ad Hoc Networks*

Shantanu Sharma[1] and Awadhesh Kumar Singh[2]

[1]Department of Computer Science, Ben-Gurion University of the Negev, Israel {sharmas@cs.bgu.ac.il} †

[2]Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana, India {aksingh@nitkkr.ac.in}

Abstract

A fundamental problem of distributed systems, leader election, is presented in the context of mobile ad hoc networks (MANETs). In many distributed systems, the presence of a leader is necessary in order to monitor underlying computations, guarantee quality functioning, take checkpoints, generate the lost token, detect quiescence conditions, etc. Hence, several leader election algorithms have been proposed in the literature. Although, most of the algorithms focus on reducing the control message (messages that have the highest priority to deliver) count, there have been almost no attention on ensuring high availability of a leader despite various types of failures, especially, in the scenarios like rescue and warfare, where the absence of the leader, even for a short duration, may lead to havoc. We focus on this issue, particularly, for large MANETs, where a large number of applications fails to perform in the absence of a leader.

We present a leader election algorithm for large mobile ad hoc networks. The algorithm is inspired by the concept of prevailing parliamentary democracy and elects three best-nodes — in terms of performance parameters like battery life, computing power, memory, hop distance, and mobility — as the president, leader, and vice leader. The president node works as the leader of the network, in case, the leader and the vice leader both become unavailable simultaneously. On the other hand, the leader node serves all the requests. Further, we create a house of elite nodes, which ensures the presence of an executive, *i.e.*, a leader during re-election to restrict the message overhead as well as the election latency while executing coordination related activities.

Keywords: Distributed computing, fault tolerance, high availability, leader election, mobile ad hoc networks.

*__This version is accepted in Taylor & Francis International Journal of Parallel, Emergent and Distributed Systems, IJPEDS, 2016.__ A preliminary version has been appeared in proceeding of the 8^{th} International Workshop on Wireless Ad hoc and Sensor Networks (WWASN), 2011 [34].

†This work was carried out when Shantanu Sharma was a student in the Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana, India.

1 Introduction

The mobile ad hoc network (MANET) is an on-demand infrastructureless collection of dynamic mobile processors (or nodes). The lifespan and mobility of the nodes in MANETs are constrained, unlike the cellular mobile systems where the mobile nodes have longer lifeline and higher mobility under some pre-designated leadership of nodes (or base stations).

Leader election (in mobile ad hoc networks). The leader election (LE) is a basic challenge of distributed systems and finds a distinguished node as a leader (or coordinator) of the network. A LE algorithm may be triggered by some nodes in the network, and after its successful termination, the network has exactly one node as leader, and others are normal. More details about LE problem may be found in Chapter 11 of [16], Chapter 3 of [3], and Chapter 13 of [15]. A leader election algorithm should satisfy the following properties:

- *Safety*. Eventually, all the nodes agree on a single node as a leader of the network (and no two nodes elect two different nodes as the leader). (This property is also known as an *eventual leader* [8, 7, 2].)
- *Liveness*. Every node elects a node as a leader of the network within a finite amount of time.

Several reasons, *e.g*., dynamic nature of nodes, variation in node's capabilities, an unsecured communication channel, and the network scalability [13], make the election process challenging in dynamic networks as compared to conventional (static) networks. Hereafter, throughout the article, the words "mobile ad hoc network," "system," and "network" have been used interchangeably.

Motivation. The leader election is a prerequisite for numerous applications. For example, the classical consensus algorithm, Paxos [23], requires a distinguished node as a leader that is responsible for progress of the algorithm. Several applications in MANETs such as intrusion detection [20, 28], group communication and data exchange [33], token generation in the token passing system [26], key distribution [25], consensus [4], virtual cellular backbone [10], and routing coordination [1, 19] require the presence of a leader. Hence, the leader election is a widely studied problem of distributed systems. A large MANET involves around a thousand or more mobile nodes (for a special military-rescue mission, emergency scenario, etc.). Thus, in a large MANET, whenever the leader fails, it is cumbersome to trigger an entirely new instance of the LE algorithm. This fact entails that the higher availability of a leader is a necessity of any large MANET, in order to avoid the application discontinuity between initiation and termination of an LE algorithm, in case the current leader fails. Furthermore, the higher availability of a leader avoids overheads due to the repeated invocation of LE algorithm. Thus, it ensures the application continuity and efficient utilization of node's resources.

Our contribution and outline of the paper. We propose a leader election algorithm, namely *DEmocratic Leader Finding Algorithm* (called *DELFA*, henceforth), for large MANETs (Sections 3 and 5). The *DELFA* handles successfully frequent leader failures, guarantees the high availability of a leader, and satisfies safety and liveness properties. Hence, the proposed algorithm provides an eventual leader in each component of the network.

The proposed algorithm elects three best-nodes (namely, President node, leader node, and vice-leader node) on the basis of their capabilities. In general, the *node capability* is a function of residual battery, computing power, memory, node degree, mobility, and hop distance from all the nodes. A comparison with other existing best-node-based algorithms [5, 24, 37, 38, 39] is given in Section 6. The message complexity and correctness proof of our algorithm are given in Appendix A and Appendix B, respectively.

2 Related Work

The problem of LE in distributed systems has been well-studied, and there are some novel contributions *e.g*. [9, 18, 14, 30, 8]. Broadly, we classify LE algorithms for MANETs into three classes: (*i*) routing protocol based [12, 21, 26, 29, 36, 35], (*ii*) best-node-based [39, 40, 5, 24, 37, 38, 31], and (*iii*) miscellaneous algorithms [32, 6, 7, 11, 17, 31, 27, 2]. Since our algorithm falls under the best-node-based category, in the following section, we present a brief summary of the best-node-based LE algorithms only, and the requirements for the design of a new LE algorithm.

2.1 Best-node-based leader election algorithms

The MANET is an environment consisting of less powerful and thus failure prone nodes. This fact indicates that a preferred class of LE algorithms is one that elects the "best" node as leader out of the available lots of nodes. Therefore, we develop a best-node-based LE algorithm for large MANETs. The best-node is elected on the basis of the node capabilities. The comparison of various best-node-based LE algorithms for asynchronous networks [5, 24, 39, 37, 38] is given in Table 1.

	AEFA [39, 40]	Lee et al. [24]	Boukerche et al. [5]	Raychoudhury et al. [31]	MELFA [37]	ELFA [38]	*DELFA*
Diffusion computation based	Yes	Yes	Yes	Yes	No	No	No
Data structure cost	1	2	2	2	3	4	5
Number of messages	5	5	5	4	3	3	1
Messages required at the time of initiation	$3NE$	$3NE$	$3NE$	$\mathcal{O}(NE)$	$3NE$	$3NE$	$2nE$
Best case message complexity	$\mathcal{O}(NE)$	$\mathcal{O}(NE)$	$\mathcal{O}(NE)$	$\mathcal{O}(N)$	$3(N-1)$	Only 4 messages	Only 2 messages
Average case message complexity	$\mathcal{O}(NE)$	$\mathcal{O}(NE)$	$\mathcal{O}(NE)$	$\mathcal{O}(NE)$	In between $\mathcal{O}(N)$ and $\mathcal{O}(NE)$, but not $\mathcal{O}(NE)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Worst case messages	$3NE$	$3NE$	$3NE$	$\mathcal{O}(NE)$	$3NE$	$3NE$	$2nE$
The high availability of a leader	No	No	No	No	No	No	Yes
Applicability to large MANETs	No	No	No	No	No	No	Yes
1, 2, 3, 4, and 5 represent an order of the cost, where 1 represents the lowest cost and 5 represents the highest cost. Notations: n: Number of participating nodes, N: Number of nodes in a MANET, E: communication links, and $Size_H$: Number of nodes in the house. These notations are defined in Section 4.							

Table 1: Comparison of different best-node-based leader election algorithms.

Vasudevan et al. [39, 40] presented an algorithm for asynchronous mobile ad hoc networks, namely Asynchronous Extrema Finding Algorithm (AEFA). AEFA is a weakly self-stabilizing LE algorithm. In AEFA, each node possesses some *node-weight* that represents the criteria to elect the best-node. The algorithm creates and maintains a spanning tree using the diffusion computation (Chapter 10 of [22]) to elect a leader.

Another LE algorithm, proposed by Lee et al. [24], provides limited fault tolerance to AEFA using a list of five leaders on each node. The five leaders are arranged in decreasing order of node-weights, where the first node is considered the active leader of the network. In case the first one fails, the second one becomes the active leader, and so on. It is noted that the algorithm [24] provides only a fault-tolerant version of AEFA without any attempt to overcome its disadvantages such as a huge message overhead.

Boukerche et al. [5] presented a LE algorithm that handles frequent topology change and node mobility. The authors claim that the algorithm succeeds to elect a unique leader in every connected cluster in a short period of time using fewer messages than AEFA. In [37], a message efficient algorithm, called MELFA, is proposed; however, for fault-free scenario. If the leader fails, then MELFA algorithm reelects a new leader, and hence, the network is without a leader during the re-election time. The algorithm *Elite Leader Finding Algorithm* (ELFA) [38] is based on MELFA [37] and provides a limited failure resiliency. However, it is not suitable for large MANETs, due to its dependency on MELFA that results in a large message overhead for electing a leader. In addition, in ELFA if the two best-nodes, *i.e.*, the leader and the vice leader crash simultaneously, the network reelects a leader from all the nodes.

2.2 Requirements of a new best-node-based leader election algorithm

All the best-node-based leader election algorithms [39, 40, 5, 24, 37, 38, 31] elect the best-node as the leader of the network. However, there are some reasons that make the above algorithms unsuitable for implementation in large MANETs, as follows:

1. **Hard to maintain a spanning tree.** Algorithms [39, 40, 5, 24, 31] create and maintain a logical structure, spanning tree, during the course of execution, where the initiator, the root of the tree, can announce the leader when it receives the desired information from its child nodes. Further, the child nodes of the root wait for their child nodes, and so on. Such a wait sequence increases the latency to announce the leader and requires at least $2E$ messages (E messages are from the root to all the child nodes and E messages are from the child nodes to the root node), where E is the total number of communication links. Further, the root node requires additional E messages to announce the leader node, hence, the total number of messages is $3E$. Therefore, due to a large message overhead, the algorithms [39, 40, 5, 24] are not suitable to large MANETs.
2. **Initiation and termination of the algorithm.** Any node with less node-weight (*i.e.*, less battery or computing power) may initiate the algorithms [39, 40, 5, 24] and may fail before the leader announcement. The algorithms [39, 40, 5, 24] try to collect node-weights of all the (N) nodes at the initiator before the announcement of a leader, which is not essential. Since the nodes with smaller node-weights cannot become leader, the collection of node-weights from all the (N) nodes enhances the message overhead and election latency.
3. **Re-initiation of the algorithm when the leader fails.** AEFA [39, 40], Boukerche et al. [5], MELFA [37], and Raychoudhury et al. [31] require an initiation of the algorithm over all the (N) nodes when the leader fails, which results in at most $3NE$ messages ($3NE$ messages are required if all the nodes executes the algorithm simultaneously) for each initiation. Further, the processing of a large number of messages is not supportive to battery constrained nodes. Lee's algorithm [24] also re-initiates the algorithm over all the (N) nodes when all the five leaders fail, which results in at most $3NE$ messages.
4. **Re-initiation of the algorithm when the network partitions.** AEFA [39, 40] and Lee's algorithm [24] require initiation of the algorithm over all the (N) nodes when the network partitions, which results in at most $3NE$ messages for each initiation. However, Boukerche et al. [5] handles the network partitioning efficiently and does not require a re-initiation of the algorithm.
5. **An existence of a FIFO channel.** Algorithms [39, 40, 5, 24] assume a first-in-first-out (FIFO) channel, which is hard to ensure in MANETs.
6. **Violation of the safety requirement.** Lee's algorithm [24] may violate the safety requirement due to the existence of multiple leaders simultaneously, in case, multiple nodes do not receive the heartbeat message[1] due to any reason.

In summary, all the above mentioned best-node-based LE algorithms [39, 40, 5, 24, 37, 38, 31] require a large amount of messages and provide very constrained failure resiliency. Thus, none of them is suitable for large MANETs, where the high availability of a leader is utmost necessary. In the next sections (Sections 3 and 5), we will present our LE algorithm that overcomes the above mentioned problems, and thus, it is suitable to large MANETs.

3 The Abstract Description of the *DELFA* Algorithm

The problem to guarantee high availability of a leader is analogous to a problem of the prevailing democratic system of polity, where the presence of an executive is always essential to make decisions regarding the affairs of the state during normal times or crisis hours. Hence, the approach of our algorithm is inspired by the system followed by various democratic countries, *e.g.*, India, Israel, Italy, etc.

Following the parliamentary system of the governance in democratic countries, we create a group of "best" nodes, namely the *house*; see Figure 1a. The house contains nodes having better capabilities like battery backup, computation power, etc. (we call the node capabilities as the *node-weight*). In our illustration, node-weight and node-capability have been used interchangeably. However, the member nodes of the house do not have any special characteristics, because it may turn the algorithm asymmetric.

For the purpose of simplicity, we consider the (residual) battery power as a parameter to elect the leader and scale it to 0-100. We consider that each node knows its battery power and scale it. In the proposed algorithm, initially the house holds nodes whose node-weights are between 70 and 100. Here, it is important to mention that in order to become a member of the house, there is no need to run any agreement algorithm. According to node-weights, the nodes become the house members. However, we restrict the size (the total number of nodes) of the house, Section 5.2.3).

We restrict initiation of the algorithm such that only a few nodes (the nodes having node-weight greater than

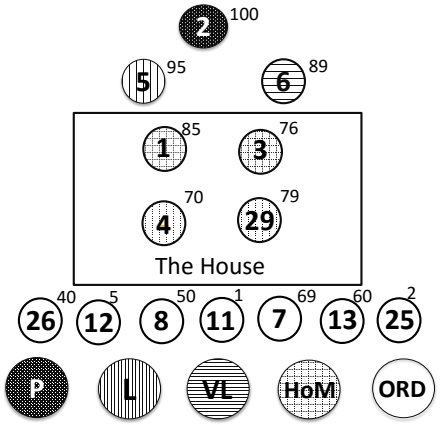
[1] A message sent by the leader node at a periodic time to show its presence in the network.

69) can initiate the algorithm. The reason behind such a restriction is that a node — that initiates the algorithm — with less battery power may crash during election or shortly after. After termination of the algorithm, we elect *three* best-nodes in terms of the maximum battery power, namely the first best-node as the *President* node (P), the second best-node as the *Leader* node (L), and the third best-node as the *Vice-Leader* node (VL); see Figure 1a. Hereafter, the terms president, leader, and vice-leader are used for the president node, the leader node, and the vice-leader node, respectively.

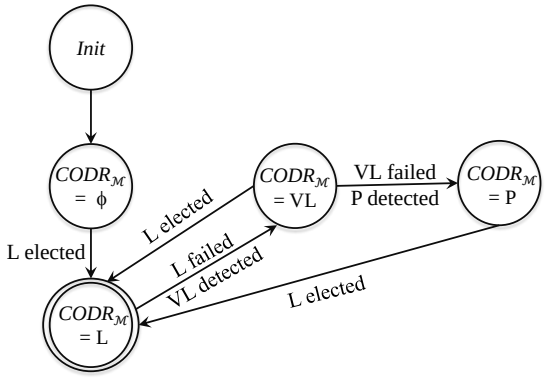
Among these three best-nodes, only the leader node, L, serves all the requirements of the network. Also, the special nodes, P and VL, do not provide any service to the network in the presence of the leader node. When the leader fails, the node VL (or P) takeovers the charge of the failed leader until a new leader is elected from the house. Note that, unlike other predecessor algorithms, we elect a new leader from the house instead from all the N nodes. (Later in Section 4, we assume that size of the house is much smaller than N. Hence, the election process over the house has reduced election latency and number of messages exchanged.)

Note that the presence of any special node, *i.e.*, P, L, or VL, signifies the existence of the chief executive in the network (as well as in the country). One may argue that why not there are 4, 5, or more special nodes? We suggest them to recall the democratic system for better understanding of the algorithm. For the sake of completeness, we provide the working of P, L, and VL in the context of the democratic systems (and our settings), as follows:

- The node P is like the President who acts as the head of the state except during crisis hours when it acts as the head of the government also.
- The leader node, L, is like the Prime Minister who acts as the leader of the house as well as the head of government, and thus, responsible for all the activities. In the context of our algorithm, the node L is the head of the house and acts as a leader of the network.
- The node VL works like the Deputy Prime Minister of the country. The node VL becomes the leader-in-charge of the system in the absence of the leader (and during the re-election of the leader).



(a) The abstract view of the *DELFA*.[2]



$CODR_{\mathcal{M}}$ shows the current coordinator of a mobile ad hoc network $\mathcal{M}$.
Note: The $CODR_{\mathcal{M}}$ = ϕ is unreachable from any state, except "*Init*" state.

(b) Failure of the special nodes, *i.e.*, the nodes L and VL.

Figure 1: High level description of the *DELFA* algorithm.

Fault handling of the nodes. All the existing best-node-based algorithms [39, 40, 5, 24, 37, 38, 31] elect the best node as a leader and the second best-node as a vice leader. The leader node serves all the requests in the network, and hence, losses its node-weight faster. However, in the same situation, the vice-leader may crash just before the crash of the leader; consequently, the system does not possess a leader. In order to avoid this situation, we elect three best-nodes in the system.

The President node, P, does not deliver any service to the network if the leader and the vice-leader nodes are available in the network. Hence, the node P is a node with the maximum node-weight in the system with a high

[2]The value inside a circle represents the identity of the node, and the value outside a circle represents the node-weight.

probability. However, the node P acts as a leader of the network if the nodes L and VL become unavailable simultaneously and elects a new leader from the house. The vice-leader provides services to the network when the node L fails; see Figure 1b, and elects a new leader from the house. Thus, the nodes P and VL work as the leader-in-charge until a new leader is elected from the house. Details about fault handling and node mobility are given in Section 5.2.4.

Keeping in view the inspiration and the operational semantics of the proposed algorithm, it is named as *DEmocratic Leader Finding Algorithm* (*DELFA*, for short).

3.1 Comparison with the existing best-node based algorithms and salient features of the algorithm

The performance of the *DELFA* is much better than the existing best-node-based algorithms [39, 40, 5, 24, 37, 38] in terms of latency to elect a leader and the total number of messages exchanged. Specifically, our algorithm has the following salient features:

1. **No logical structure.** We do not impose any logical structure on the network.
2. **Initiation of the algorithm.** All the (N) nodes do not execute the algorithm. Only $n \leq N$ nodes whose node-weights are between 70 and 100 are allowed to execute the algorithm, while the other existing algorithms [39, 40, 5, 24] do not disallow any node to initiate the algorithm. Moreover, in the *DELFA*, the initiator needs not to wait to collect node-weights of all the (N) nodes. These facts show a significant reduction in the total number of messages and election latency.
3. **No re-initiation of the algorithm when the leader fails or the network partitions.** When the leader fails, the node VL becomes a new leader (leader-in-charge) until the election of a new leader. Further, while the node VL acts as leader-in-charge, it sends messages to the members of the house only and receives responses accordingly, *i.e.*, the election of the new leader is carried out within the house (for details see Sections 5.2.4 and 5.2.5). We assume that the number of nodes in the house are significantly less as compared to N; hence, the time complexity and election latency are low.
4. Unlike other best-node-based algorithms [39, 40, 5, 24], the *DELFA* does not need FIFO channels' guarantee during message exchanges.
5. The election of three best-nodes as the President, the leader, and the vice-leader is not like Lee's algorithm [24]. In case of Lee's algorithm [24], they re-initiate the election over all the nodes when the five leaders fail, we never involve all the nodes in election, except initially.

4 The System Settings

This section outlines the preliminary assumptions about the environment, various types of messages, and data structures. All the notations used in our algorithm are given in Table 2. The system settings are explained below.

$\mathcal{M}$	Mobile ad hoc network	N	Number of nodes in $\mathcal{M}$	E	A set of edges between neighbor nodes
n	Number of participating nodes	NW	Node-weight	HoM	House Member node
$Size_H$	Number of nodes in the house	ORD	ORDinary Node	P	President node
L	Leader node	VL	Vice-leader node		

Table 2: Notations used in the *DELFA* algorithm.

Mobile nodes. We consider a mobile ad hoc network of N non-malicious and non-Byzantine mobile nodes $(M_1, M_2, \ldots, M_N)$, where each node has a unique identity that is used to distinguish a node and break the tie among the nodes, if needed. We give priority to the lowest identity node. Nodes are assumed to have sufficient node buffers to store messages until they are delivered.

Any two nodes that are in the transmission range of each other are called *neighbors*, and only neighbor nodes can communicate directly using a communication link between them. The node mobility may often lead to link breakage and link formation. Each node holds its neighbor information only, and no node has global information.

Node-weight. A node, M_i, has a priority to become a leader, called node-weight (NW_i). Node-weight may be computed based on the residual battery backup, computing power, memory, node degree, hop distance from all the

other nodes, etc. We assume only $n \leq N$ nodes whose $70 \leq NW \leq 100$ can initiate the algorithm.[3]
Node's status. The status of a node is defined on the basis of its NW, and it can be either of the following:

1. Ordinary node (ORD) – the node with $0 < NW < 70$.
2. House Member node (HoM) – the node with $70 \leq NW < 101$.

Note that there is no need to run election among nodes to attain their status. Since the nodes are not Byzantine, they themselves set the status correctly. In addition, there is no relation among ORD and HoM nodes.
Node's state. The node's state represents the current state of a node i. Initially, a node may be in either of the following states; however, after the termination of the algorithm, a node must be in the correct state. The states of nodes are as follows:

1. NORMAL – if the node is continuing assigned tasks in the presence of a leader.
2. CANDIDACY – if the node has lost its contact with the leader, the leader is failed, or the node receives an election message.
3. PRESIDENT (P) – an elected node with the highest NW is in the president state.
4. LEADER (L) – an elected node with the second highest NW is in the leader state.
5. VICE-LEADER (VL) – an elected node with the third highest NW is in the vice-leader state.

Communication links. The mobile nodes operate on a pre-decided wireless communication channel that is bidirectional, reliable, and non-FIFO. Any two neighbor nodes directly communicate over a wireless communication link by message passing. The channel does not generate and deliver any message to any node if the same message has not been sent by any other node previously.

Network Structure. We model a network in the form of a graph: $\mathcal{M} = [N, E]$, where $\mathcal{N}$ represents a set of vertices (or processors/mobile nodes in the network), $\mathcal{E}$ represents a set of edges where an edge between a pair of neighboring nodes shows a communication link.

Failure Model. We assume that a node can fail in two possible ways: (*i*) due to the swift movement of the node that may result in frequent topology change and transient non-interaction of the highly mobile node with other nodes in the network. We call such nodes the *failed nodes*, (*ii*) crash, *i.e.*, when a node does not possess necessary storage and processing resources, it results in permanent failure of the node, and such a node is called a *crashed node*. When a crashed node recovers by users' intervention, it does not possess the knowledge of updated data structures.

Size of the house. The maximum number of nodes in the house is denoted by $Size_H$, where $Size_H \ll N$.

Further, we assume the existence of a routing algorithm to deliver messages between any two nodes. Also, we do not assume the existence of any special node that is responsible for initiation of the algorithm.

Types of messages. We classify messages (used in the algorithm) into two categories: control messages (see Table 3) and non-control messages (see Table 4). Note that control messages have priority over all the other messages and require some communication costs. The non-control messages can be piggybacked on control messages, heartbeat messages, and application messages. The first three control messages (in Table 3) are used in all the existing best-node-based algorithms.

Message	Notation	Description
Election Message	EM	is used to initiate the election. At the initiation of the algorithm when the MANET does not possess a unique leader, any node whose $70 \leq NW \leq 100$ can initiate the algorithm by sending identical EMs to respective neighbors, such an initiator node is called an ori node.
Acknowledgement Message	AM	is sent by M_j to the originator node, ori, in response to an EM that includes NW_j.
Coordinator Message	CM	is used for a formal announcement of the leader in the network.
Update Message	UM	is sent by M_j to update the leader node about the identity and NW of a new joining node i.

Table 3: Control messages used in the *DELFA* algorithm.

[3]The constraint on node-weight depends on the criticality of the application selection and the number of the house nodes. For example, if there are no nodes having $NW > 69$, then we can change this node-weight restriction. On the other hand, if all the nodes have $NW > 69$, then the algorithm will be not be very useful. Hence, in this case, we are allowed to negotiate with node-weight.

Message	Notation	Description
Request Message	RM	send by a new joining node M_i to its neighbors to know the current leader.
Notify Message	NM	send by M_j to inform the current leader's identity to a new joining node M_i. It is also used to handle the network partitioning and merging.
Wait Message	WM	send by a waiting node M_j to another waiting node M_i to avoid unnecessary flow of RMs if M_j does not know the current leader.
Information Message	IM	send by node L to: (*i*) a new joining node M_i to change its status either HoM or ORD, and (*ii*) the highest node-weight and the third highest node-weight nodes to become the president and the vice-leader nodes, respectively.

Table 4: Non-control messages used in the *DELFA* algorithm.

Data Structure	Description	Default value
	At the ori node	
$receive\_ack[]$	$receive\_ack[j] = j$ represents that an AM has been received at the ori node from M_j	$receive\_ack[j] = -1$
$receive\_weight[]$	$receive\_weight[j] = NW_j$ represents node-weight of M_j	$receive\_weight[j] = -1$
t_e	Timer at the ori node	
	At all the nodes	
$neighbor\_name_i[]$	Identity of the neighbor nodes of M_i	
NW_i	Node-weight of M_i	
$STATUS_i$	Current status of M_i	ORD/HoM
$CODR_i$	The current leader of M_i	$\forall i, CODR_i = \perp$
$CODRelect_i$	Set to $TRUE$ if M_i has elected its leader	$\forall i, CODRelect_i = FALSE$
$State_i$	The current state of M_i	NORMAL/CANDIDACY/P/L/VP
$election\_send_i$	Set to $TRUE$ if M_i has been forwarded EMs to its neighbor(s)	$\forall i, election\_send_i = FALSE$
$send\_for_i$	Identity of the ori of the EM that is received at M_i	$\forall i, send\_for_i = \infty$
$coordinator\_send_i$	Set to $TRUE$ if M_i has forwarded CMs to its neighbor(s)	$\forall i, coordinator\_send_i = FALSE$
	Piggyback on an EM	
$receive\_election[]$	Ids of the nodes that have already received an EM. When M_i sends/forwards an EM to its neighbor, M_j, M_i updates $receive\_election[j] = 1$	$\forall j, receive\_election[j] = -1$
ori	Identity of the current initiator of that EM	
$forward\_flag$	Set to $TRUE$ to indicate to recipient nodes not to forward EMs	$forward\_flag = FALSE$
	Piggyback on a CM	
PID	Identity of the newly elected president node	
LID	Identity of the newly elected leader node	
$VLID$	Identity of the newly elected vice-leader node	
$HoMID[]$	Identity of all the house nodes	
	Common at P, L, VL, and all the nodes of the house	
Pid_i	Identity of the current president node of the network	$\forall i, Pid_i = \perp$
$VLid_i$	Identity of the current leader node of the network	$\forall i, VLid_i = \perp$
$HoMid_i[]$	Identities of all the house nodes	$\forall i, HoMid_i[] = \emptyset$

Table 5: Data structure used in the *DELFA* algorithm.

Types of data structures. The data structures are divided into five categories: (*i*) at the initiator of the algorithm, (*ii*) at all the mobile nodes, (*iii*) piggyback on election messages, (*iv*) piggyback on coordinator messages, and (*v*) at P, L, VL, and all the nodes of the house. Details of these data structures are given in Table 5.

5 The *DELFA* Algorithm

In this section, we specify the *DELFA* algorithm as guarded actions. A guarded action is written as: $\langle Guard \rangle \rightarrow \langle Action(s) \rangle$. A guard (or predicate) of actions (or rules) is a Boolean expression, and if a guard is true, then all the

The phases and the procedure	
Phase 1	Initiation.
Phase 2	Reception of concurrent EMs.
Procedure A	Election of the special nodes.
Phase 3	Leader's announcement.
The events	
Joining of a new incoming node	
EVENT 1	A new incoming node sends a RM message to join a pre-established coordinated MANET.
EVENT 2	Reception of a RM message.
EVENT 3	Reception of a UM message.
Impeachment of the President node and the Vice-Leader node, and handover the Leadership	
EVENT 4(I)	Reception of the heartbeat message.
EVENT 5(I)	Reception of acknowledgement messages with $ack\_flag = TRUE$ at the leader node and handover the leadership.
Maintaining size of the Lower House and the Upper House	
EVENT 5(II)	Reception of acknowledgement messages with $ack\_flag = TRUE$ at the leader node and handover the leadership.
EVENT 4(II)	Reception of the heartbeat message.
Absence of heartbeat messages	
EVENT 6	No heartbeat message from L and no network partition.
The network partitioning and merging	
EVENT 7	The network partitioning.
EVENT 8	The network merging.

Table 6: The phases, procedure, and events.

actions, corresponding to that guard, are executed in an atomic manner. At some point of time more than one guard may be true. The *DELFA* algorithm is given in Tables 7 and 8.

5.1 The phases and the procedure

The proposed algorithm, *DELFA*, is a 3-phase algorithm (see Tables 6 and 7) that allows topological changes during the execution. It is important to note that all the phases and the procedure of the *DELFA* are executed only once when the algorithm executes for the first time in an arbitrarily network. The description of the three phases and the procedure is given below:

- PHASE 1: *Initiation.* In the beginning, one or more nodes, whose $70 \leq NW \leq 100$, may initiate the election process due to the absence of a unique leader in the network, and thus, these nodes become the originator, ori, nodes. These ori nodes send election messages (EMs) to their neighbor nodes and set a timer t_e[4] for waiting acknowledgment messages, AMs; see P_1 in Table 7.

- PHASE 2: *Reception of concurrent EMs.* In this phase, node M_j receives at least one EM from the ori nodes or from another node, say M_i. On receiving EMs, M_j takes one of the following actions:
  1. If M_j receives at least one EM whose ori's identity is smaller than the identity of the most recently forwarded EMs, then M_j sends EMs to its k neighbors, which may yet to receive the same EM from other nodes. In addition, M_j switches to CANDIDACY state, sets its $CODR_j = \perp$, and sends an acknowledgement message (AM) to the ori node if $70 \leq NW_j \leq 100$; see the first guard of P_2 in Table 7. In this manner, eventually all the nodes receive election messages from a single ori node, and the network has only one ori node before the leader announcement.
  2. If M_j receives at least one EM whose identity of the ori is equal or greater than identity of the most recently forwarded EMs, then M_j discards the received EM; see the second guard of P_2 in Table 7.

- PROCEDURE A: *Election of the special nodes.* When PHASE 1 starts, the procedure also executes at the ori node. The ori node elects the special nodes (P, L, and VL) when the timer, t_e, expires. The node ori sends a coordinator message (CM) to the elected leader node — where the CM is piggyback with information of the

[4]The time may be based on the size of the network, message transmission time, and criticality of the computation.

Notations:
$\mathcal{M}_x$: x^{th} MANET component. $CODR_{\mathcal{M}_i}$: current coordinator of x^{th} MANET component.
$SEND_i(m, j)$: node M_i sends a message m to node M_j. $RECV_i(m, j)$: node M_i receives a message m from node M_j.
$m.x$: represents an appended data structure, x, with message, m.
//All the data structures have the usual meaning (see Table 5 for details of data structurs).
Functions:
`max_nw_id()`: selects a node with the maximum node-weight.
`(max-1)_nw_id()`: selects a node with the second maximum node-weight.
`(max-2)_nw_id()`: selects a node with the third maximum node-weight.
$\texttt{nw}_{\geq 70}\texttt{\_id}()$: selects $Size_H$ nodes whose node-weights are between 70 and 100.
`number()`: count the number of nodes and the occurrence heartbeat messages.

Pre $\{CODR_{\mathcal{M}_x} = \perp \vee CODR_{\mathcal{M}_y} = \perp \vee CODR_i = \perp \vee CODR_j = \perp\}$
do
P_1**: PHASE 1: Initiation.**
$[\!] CODR_i = \perp \wedge CODR_{\mathcal{M}_i} = \perp \wedge NW_i \geq 70 \rightarrow$
$ori := i, State_i :=$ CANDIDACY, $CODR_j := \perp, CODRelect_j := FALSE,$
$\forall j,\ j \in neighbor\_name_i[]$ **do**
$EM.receive\_election[j] := 1, EM.ori := i, EM.forward\_flag := FALSE, SEND_i(EM, j),$
$election\_send_i := TRUE, send\_for_i := i, coordinator\_send_i := FALSE,$
$\forall j,\ j \in neighbor\_name_i[] :: coordinator\_send[j] := -1,$
$\forall j,\ j \in receive\_ack_i[] :: receive\_ack[j] := -1,$
$\forall j,\ j \in receive\_weight_i[] :: receive\_weight[j] := -1,$

P_2**: PHASE 2: Reception of concurrent** EM**s,** $RECV_i(EM, j)$**.**
$\forall j,\ j \in neighbor\_name_i[]$ **do** $coordinator\_send[j] := -1,$
$[\!] EM.forward\_flag = FALSE \wedge send\_for_i > EM.ori \rightarrow$
if $70 \leq NW_j \leq 100$ **then** $SEND_i(AM, ori),$
$State_j :=$ CANDIDACY, $CODR_j := \perp, CODRelect_j := FALSE,$
$\exists k : k \in neighbor\_name_i[] :: receive\_election[k] := -1$ **do**
$EM.receive\_election[k] := 1, EM.ori := ori, EM.forward\_flag := FALSE, SEND_j(EM, k),$
$election\_send_i := TRUE, send\_for := EM.ori, coordinator\_send_i := FALSE,$
$[\!] EM.forward\_flag = FALSE \wedge send\_for_i \leq EM.ori \rightarrow$ Discard $EM,$

P_3**: PROCEDURE A: Election of the special nodes, where node** ori **receives an** AM **from node** j**.**
$[\!] t_e = FALSE \rightarrow$
$receive\_weight[] := receive\_weight[] \cup NW_j, receive\_ack[] := receive\_ack[j] \cup j,$
$[\!] t_e = TRUE \rightarrow$
$CM.PID :=$ `max_nw_id`$(receive\_weight[]),$
$CM.LID :=$ `(max-1)_nw_id`$(receive\_weight[]),$
$CM.VLID :=$ `(max-2)_nw_id`$(receive\_weight[]),$
$CM.HoMID[] :=$ $\texttt{nw}_{\geq 70}\texttt{\_id}(receive\_weight[]),$
if $(CM.PID = ori)$ **then** $State_{ori} :=$ P,
else if $(CM.LID = ori)$ **then** $State_{ori} :=$ L,
else if $(CM.VLID = ori)$ **then** $State_{ori} :=$ VL,
else if $NW_{ori} < 70$ **then** $State_{ori} :=$NORMAL,
$CODR_{ori} := CM.LID, CODRelect_{ori} := TRUE, SEND_{ori}(CM,$ L$),$
$election\_send_{ori} := FALSE, coordinator\_send_{ori} = TRUE,$
$\forall j,\ j \in neighbor\_name_{ori}[]$ **do** $receive\_election_{ori}[j] := -1, ori := \perp,$

P_4**: PHASE 3: Leader's announcement.**
$[\!]$ `number`$(heartbeat) = 1 \wedge State_i =$ CANDIDACY $\wedge RECV_i(heartbeat),$ L$) \wedge CODR_i = \perp \rightarrow$
if $(STATUS_i =$ HoM) **then**
$Pid_i := CM.PID, VLid_i := CM.VLID, HoMid_i[] := CM.HoMID[],$
if $(CM.PID = i)$ **then** $State_i :=$ P, $HoMid_P[] := CM.HoMID[], VLid_P = CM.VLID$
else if $(CM.VLID = i)$ **then** $State_i :=$ VL, $HoMid_{VL}[] := CM.HoMID[], Pid_{VL} = CM.PID$
else if $NW_i < 70$ **then** $State_i :=$ NORMAL,
$CODR_i := CM.LID, CODRelect_{ori} := TRUE,$
$ori_i := \perp, election\_send_i := FALSE, coordinator\_send_i := TRUE,$
$\forall j,\ j \in neighbor\_name_i[]$ **do** $receive\_election[j] := -1,$
od
Post $\{CODR_{\mathcal{M}_x} \neq \perp \wedge \forall i, CODR_i \neq \perp\}$

Table 7: The phases and the procedure of the *DELFA* algorithm as guarded actions.

highest NW node as the President node, the second highest NW node as the leader node, and the third highest NW node as the vice-leader node. In addition, the CM is piggyback with the identities of the nodes of the house. The node ori switches its state either P, L, VL, or NORMAL, and resets all the data structures used in PHASE 1; see the second guard of P_3 in Table 7.

Termination criteria. We define two criteria for the termination of the *DELFA*, as follows:

- *Strong termination*. The algorithm is known to be strongly terminated if and only if the ori node announces the nodes P, L, and VL after receiving $Size_H$ acknowledgement messages.
- *Weak termination*. The algorithm is known to be weakly terminated if the ori node announces the nodes P, L, and VL after a timeout where there is no guarantee of $Size_H$ acknowledgement messages at the ori node till timeout. The *DELFA* falls in this category. The main advantage of weak termination is an avoidance of endless waiting for strong termination.
  The significance of weak termination can be figured out by the following example: suppose, we start the LE algorithm over the total 100 nodes in the house. During the algorithm's execution, say 50 nodes crash or move to a different geographic location. Thus, it is impractical to wait for 100 nodes, because a leader may also be elected out of 50 nodes. However, the weak termination losses its significance, when it is unable to satisfy the safety requirements in the computation, *e.g.*, mutual exclusion and consensus.

- PHASE 3: *Leader's announcement.* This is the last phase of the *DELFA* that announces the newly elected leader in the network. There are two approaches to announce the leader in the network, the first is the distribution of CMs by the ori node, and the second is the first heartbeat message — piggyback with CMs and the identity of the ori node — by the leader, where the leader is informed by the ori node. We use the second approach.

  On receiving the first heartbeat message, every node knows the leader node. Moreover, all the house nodes, nodes P and VL, on receiving the first heartbeat message, hold information of the nodes P, L, VL, and all the other house nodes. Also, each node switches its state either P, L, VL, or NORMAL, and resets data structures used in PHASE 1 and PHASE 2; see P_4 in Table 7.

5.2 The events

After a successful termination of PHASE 3, the leader node continues to send heartbeat messages. In addition, the leader node provides a mechanism for joining a new incoming node (without a re-initiation of the algorithm); maintains the president, the vice-leader nodes, the lower house and the upper house; and handles the network partitioning-merging. All the events are given in Tables 6 and 8. Details of all the events are given below:

5.2.1 Joining of a new incoming node

In the presence of a leader node, any node may join the network. In this case, it is not necessary to execute an entirely new instance of the complete algorithm (from PHASE 1) in the network to join a new incoming node, unlike most of the contemporary algorithms [5, 24, 39, 40, 37] that execute a complete instance of the algorithm. In our approach, a new incoming node can join the network without executing the whole algorithm (starting from PHASE 1), as follows:

- EVENT 1: *A new incoming node sends a RM message to join a pre-established coordinated MANET.* EVENT 1 occurs when a node, M_i, arrives in the transmission range of a pre-established coordinated MANET (a MANET that holds a unique leader). In such a situation, M_i sends a request message (RM) to one of its neighbors; see E_1 in Table 8.
- EVENT 2: *Reception of a RM message.* When a node M_j receives at least one RM and M_j knows the current leader, M_j sends a notify message (NM) to the new incoming node M_i and an update message (UM) to the leader node; see the first statement of E_2 in Table 8. However, if M_j is also a newly arrived node, M_j sends a wait message (WM) to M_i and updates its $wait\_set_j[]$ ($wait\_set[]$ holds identities of new incoming nodes that do not know the leader); see the second statement of E_2 in Table 8. When M_j receives leader's information, it sends NMs to all the nodes of $wait\_set_j[]$.
- EVENT 3: *Reception of a UM message.* On the reception of a UM, the leader confirms the status of the

Notations:
$HoM_{remove}[]$: identity of nodes that relinquish their HoM status.
Other notations are borrowed from Table 7.
Function: `number()`, `max_nw_id()`, `(max-1)_nw_id()`, and `(max-2)_nw_id()`: defined in Table 7.

Pre $\{CODR_{\mathcal{M}_x} \neq \perp \vee CODR_{\mathcal{M}_x} \neq \perp \vee CODR_i = \perp \vee CODR_j = \perp\}$
do
E_1: **EVENT 1: A new incoming node sends a** RM **message to join a pre-established coordinated MANET.**
$[\!] State_i =$ NORMAL $\wedge$ $CODR_i = \perp \wedge \exists j,\ j \in \mathcal{M}_x \rightarrow SEND_i(RM, j)$,

E_2: **EVENT 2: Reception of a** RM **message,** $RECV_j(RM, i) \rightarrow$
if $(CODR_j \neq \perp)$ **then** $SEND_j(NM, i) \wedge SEND_i(UM, \text{L})$,
else $SEND_j(WM, i), wait\_set_j[] := wait\_set_j[] \cup i$,

E_3: **EVENT 3: Reception of a** UM **message.**
$[\!] State_i = \text{L} \wedge RECV_i(UM, j) \rightarrow$
$(STATUS_i :=$ HoM $\vee\ STATUS_i :=$ ORD$) \wedge SEND_{\text{L}}(IM, i)$,
$\forall i, i \in$ House $:: \vee SEND_{\text{L}}(IM, i)$

E_4: **EVENT 4 (Parts 1 and 2): Reception of the heartbeat message.**
$[\!]$ `number`$(heartbeat)\ modulo\ z \rightarrow$
$AM.ack\_flag = TRUE, \forall j, j \in$ House $\vee$ P $\vee$ VL **do** $SEND_j(AM, \text{L})$,
$[\!]\ heartbeat.HoM_{remove}[] \neq \emptyset \rightarrow$
$\forall i \in HoM_{remove}[]$ **do** $STATUS_i :=$ ORD,

E_5: **EVENT 5 (Parts 1 and 2): Reception of** AM **messages with** $ack\_flag = TRUE$ **at the leader node,** $RECV_{\text{L}}(AM, *)$, and handover the leadership.
$[\!] RECV_{\text{L}}(AM, *) \wedge AM.ack\_flag = TRUE \rightarrow$
$receive\_weight[] := receive\_weight[] \cup NW_j, receive\_ack[] := receive\_ack[] \cup j$,
$\forall j \in$ P, VL $:: (NW_j < 70) \vee (NW_P < NW_L) \vee (NW_L < 70)$
do P := `max_nw_id`$(receive\_weight[])$,
L := `(max-2)_nw_id`$(receive\_weight[])$,
VL := `(max-3)_nw_id`$(receive\_weight[])$,
$SEND_{\text{L}}(IM, P)$, $SEND_{\text{L}}(IM, VP)$, $STATUS_L :=$ ORD, and announce the new leader
$x \leftarrow$ `number`$(receive\_ack[])$,
if $x > Size_H$ **then for** $i \in (x \times Size_H)$ **do** $HoM_{remove}[x - Size_H] := HoM_{remove}[x], x \leftarrow x - 1$,
Append $HoM_{remove}[]$ with the heartbeat message,
else $x < Size_H$ **then** Allow $Size_H - x$ new nodes with $70 \leq NW \leq 100$ to be HoMs,
od
Post $\{CODR_{\mathcal{M}_x} \neq \perp \wedge \forall i \in \mathcal{M}_x, CODR_i \neq \perp\}$

Table 8: The events in the *DELFA* algorithm as guarded actions.

new joining node M_i to ORD or HoM based on the number of nodes in the house[5] and sends an information message (IM) to M_i to inform its current status. Also, the leader node sends IMs to all the house nodes to have an updated information of the new house node; see E_3 in Table 8.

5.2.2 Impeachment of the President and the Vice-Leader nodes, and handover the Leadership

The leader node inspects the president node and the vice-leader node at a regular interval.[6] Since the weight of every node reduces regularly due to its involvement in various tasks, it is required to maintain the best-node always as the president node. For the purpose of understanding, we assume that the leader node inspects the president and the vice-leader nodes at every $modulo\ z$ heartbeat message. In real applications, selection of the heartbeat message depends on the programmer's choice and the criticality of applications.

- EVENT 4 (*Part* 1): *Reception of the heartbeat message.* At every $modulo\ z$ heartbeat message, every node of the house, the president, and the vice-leader nodes send their node-weights to the leader node via *Acknowledgement Message*s (AM); see the first guard of E_4 in Table 8. Here, an *Acknowledgement Message* uses a Boolean flag,

[5]If the house is full (*i.e.*, the number of the nodes in the house is equal to the defined limit), then the house cannot accommodate more nodes whose $NW > 69$, and such new nodes become ORD nodes. This step prevents increasing the size the house by the defined limits.

[6]This event is similar to the prevailing government affairs where the President of the country may be impeached by the chief of the House (or Senate).

namely $ack\_flag = TRUE$ that indicates the *Acknowledgement Message* is sent by the nodes in response to the $modulo\ z$ heartbeat message, not in response to *Election Message*s, which are sent in PHASE 1. The default value of ack_flag is $FALSE$.

- EVENT 5 (*Part* 1): *Reception of acknowledgement messages with* $ack\_flag = TRUE$ *at the leader node and handover the leadership.* On the reception of AMs with $ack\_flag = TRUE$, the leader node checks node-weights of the president, the vice-leader nodes, and itself.
  If the leader node finds $NW_P < 70$, $NW_P < NW_L$, $NW_{VL} < 70$ or $NW_L < 70$, then the leader node selects three best-nodes from the house as the president, the new leader, and the vice-leader nodes. Afterwards, the current leader informs the previous president and the vice-leader nodes to become ordinary nodes (ORD) and sends information messages (IM) appended with the list of house nodes to the new nodes P and VL. In addition, the current leader announces the new leader (using the next heartbeat message) and becomes ORD; see the first seven lines of E_5 in Table 8.

5.2.3 The Maintenance of the house

It is required to maintain an upper bound on the size of the house in order to have message overheads under control. The following two situations trigger the need to maintain such bounds on the size of the house:

1. After a successful completion of PHASE 3, the number of nodes in the house may be *more* than $Size_H$,
2. During the execution of assigned tasks, node-weights decrease, hence, it is required for nodes to update their status.

Hence, the algorithm has to maintain a bound on the size of the house. The following EVENTs 4(part 2) and 5(part 2) limit the size of the house, as follows:

- EVENT 5 (*Part* 2): *Reception of acknowledgement messages with* $ack\_flag = TRUE$ *at the leader node and handover the leadership.* On the reception of AMs with $ack\_flag = TRUE$, the leader node counts the number of nodes in the house. If the nodes are more than the defined limits ($Size_H$), the leader node sends the next heartbeat message piggyback with a list of the nodes of the house that have to switch their status;[7] see the *if* statement of E_5 in Table 8.
  On the other hand, if the number of nodes is less than $Size_H$, the leader node keeps this information and allows the new joining nodes to stay in their status (HoM) until the limits are reached (see the *if* statement of E_5 in Table 8).
- EVENT 4 (*Part* 2): *Reception of the heartbeat message.* The nodes of the house on receiving every heartbeat message verify their ids in the piggybacked information. If the nodes find their ids in the piggybacked information, then they quit the HoM status and set their status to ORD (ordinary node). In addition, all the other house nodes update their information about other house nodes, resulting in an updated list of other house nodes at each house node; see the second guard of E_4 in Table 8.

5.2.4 Absence of heartbeat messages

The absence of heartbeat messages signifies either mobility, disconnection, or failure of the leader. We consider that the nodes P and VL wait for at least 5 consecutive heartbeat messages, and if they are unable to receive a heartbeat message, then they try to elect a new leader. In fact, the *DELFA* demonstrates its strength by handling this event and ensures the high availability of a leader. Note that here we consider that the network is not partitioned, which may be a consequence of the absence of heartbeat messages, and the next EVENT 7 and EVENT 8 show the network partition handling.

- EVENT 6: *No heartbeat message from L and no network partition.* In this scenario, the node VL becomes a new leader (leader-in-charge) of the network and try to elect a new leader *in its presence* from the house. Recall that the vice-leader node becomes the leader-in-charge when the node L fails, and the president node becomes the leader-in-charge when the nodes L and VL fail simultaneously, as shown in Figure 1b.
  How a new leader (or a new vice-leader) is elected? The nodes P and VL know each node of the house (by PHASE 1, EVENT 3, EVENT 4, and EVENT 5). Hence, the election of a new leader *in the presence of the leader-in-charge* is carried out in the house without invoking PHASE 1. The node VL (or P) sends EMs with a special flag, namely $forward\_flag$ to be $TRUE$, to all the nodes of the house. Such a flag variable with true value

[7] A node with higher identity is called to switch the status.

indicates that it is not an initiation of PHASE 1. On the reception of EMs (where, $forward\_flag = TRUE$), the respective nodes send an AM to the node VL (or P), and they *do not forward* EMs to any other node. Subsequently, the node VL (or P) elects a new L (and VL), and the new leader starts sending heartbeat messages.

EVENT 6 represents that once the current leader fails, a new leader is elected from a group of a few nodes (*i.e.*, the house) by the node VL (or P), where the node VL (and the node P) know each node of the house. Note that in this situation, the network holds a leader-in-charge that ensures the application continuity. Hence, an existence of the house shows that there is no need to run the LE algorithm starting from PHASE 1 in the network when the leader fails.

Aside. There may be situation when nodes P, L, and VL *crash* at a time. However, it is is not possible, because the node P is the best-node that does not provide any service to the network, resulting in best node-weight with a high probability. But, the *mobility* may lead to failure of nodes P, L, and VL at a time. In such a case, there is no need to execute PHASE 1. After an absence of 10 consecutive heartbeat messages, nodes of the house run an election among them with $EM.forward\_flag = TRUE$, and eventually, the lowest identity house node declares new P, L, and VL.

5.2.5 The network partitioning and merging

We show how the algorithm handles the network partitioning, in case, it detects the absence of 10 consecutive heartbeat messages due to disconnection or failure of the leader. Recall that in no case (the network partitioning), we require the initiation of the *DELFA* algorithm from PHASE 1 except for the first time. Also, the joining of a new node may combine two different MANET components. The following events describe the network partitioning and merging:

- EVENT 7: *The network partitioning.* When the network partitions, the following three cases can occur in our algorithm, as follows:
  1. The network partitions into three components such that the first one holds the node P, the second one holds the node L, and the third one holds the node VL. In this case, the nodes P, L, and VL become a leader of their components. The nodes P and VL elect a new leader in their components following the method given in EVENT 6. Also, the leader node designates nodes P and VL by following the method given in EVENTs 4(part 1) and 5(part 1).
  2. The network partitions into components such that each component holds at least one node of the house. In this case, since each component has at least one node of the house and all the nodes of the house know other house nodes, all the nodes whose $NW > 69$ elect nodes P, L, and VL among them like the method given in EVENT 6. Further, if P, L, and VL lie in different components, then the situation is similar to Case 1.
  3. The network partitions into components such that at least one of the components does not hold any node whose $NW > 69$. In this case, the node whose $NW > 69$ follows the first and the second cases. The nodes whose $NW < 69$ are not allowed to elect a leader in this case.
- EVENT 8: *The network merging.* Network merging occurs when two or more partitioned components merge due to a communication link formation. Note that the joining of a new node may also lead to the network merging. The leader node, say L_i, of any component that has the highest node-weight becomes the leader of the merged component and ids of the leaders are used for tie breaking, if needed.
  When a node receives heartbeat messages from two or more leader nodes, the node asks all the leader nodes about their node-weight. The node selects a node, say L_i, with the highest node-weight as a leader node and informs all the other leader nodes to have node L_i as the leader.
  In our approach, the other component's leader surrenders its leadership and data structures to L_i. Further, note that this scenario provides two or more president and vice-leaders nodes; however, eventually, the nodes P and VL of the components, whose leaders have surrendered their leadership, loss their states and accept L_i as their leader.

6 Simulation

We use a 500×500 meter^2 simulation area where 50 to 1000 nodes are randomly distributed. The message traversal time between any two neighboring nodes is 2 seconds, and the transmission range of every node is 250 meters. The leader node periodically, after every 90 seconds, sends a heartbeat message to all the other nodes in the given area. If

a node does not receive any heartbeat message for a fixed time (450 seconds because we consider that the absence of 5 consecutive heartbeat messages show the leader's absence), then the node executes the specified actions. We consider five algorithms [39, 5, 24, 37, 38] with our proposed algorithm for analysis purpose.

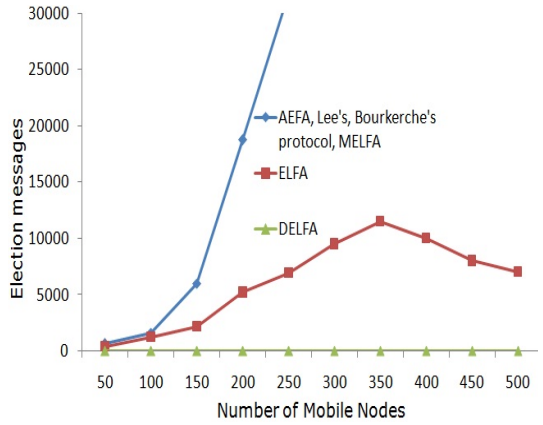


Figure 2: Re-election analysis of the *DELFA* with the algorithms [39, 5, 24, 37, 38].

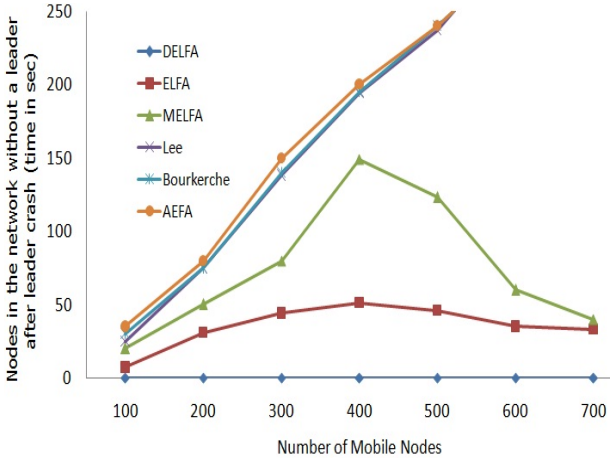


Figure 3: Leader availability analysis of the *DELFA* with the algorithms [39, 5, 24, 37, 38].

Re-election analysis. A new leader is elected when the current leader fails. In Figure 2, an analysis is shown to elect a new leader in the *DELFA* and all the algorithms [39, 5, 24, 37, 38] when the current leader fails. All the existing algorithms [39, 5, 24, 37, 38] require a re-initiation of the algorithm except the *DELFA*. We can observe a noticeable difference in the slope of the *DELFA*, which is close to the X-axis. It is due to the introduction of the house concept in our LE algorithm.

Leader availability analysis. The graph, in Figure 3, shows the leader's absence from the network due to failure or mobility of the leader node. The leader discontinuity is clearly visible in other algorithms. However, in case of *DELFA*, a leader exists at all the time, and thus, the curve of *DELFA* is very close to the X-axis. Further, all the other algorithms show the leader discontinuity for longer durations. Particularly, in case of AEFA, Bourkerche's algorithm, and Lee's algorithm, the leader is absent for a quite long duration.

The impact of node density and transmission range of the nodes. In order to examine the impact of node density, we use 250×250 meter^2 simulation area and 100-700 nodes. We do not consider a completely connected graph. Here, we consider the following two cases:

1. *Algorithm initiation.* Recall that an execution of all the algorithms including the *DELFA* is required at the time of initiation. In the graph, Figure 4, we observe that the election latency of AEFA is highest. However, the slopes of MELFA and ELFA have a critical point (at 600 nodes), after that point the slopes decrease due to increase in the node density and piggyback information with election messages. However, the curve of the *DELFA* has a less gradient than all the other curves, because in the *DELFA*, only the node of the house ($Size_H \ll N$) participate at the time of initiation; hence, fewer (election and acknowledgement) messages flow in the network.
2. *Failure of the current leader.* The algorithms [39, 5, 24, 37] are executed to elect a new leader when the current leader fails. In this case, see Figure 5, AEFA and Bourkerche's algorithm show the highest time to elect a new leader due to a complete execution of the algorithm as compared to the *DELFA*, which takes 2 seconds (almost 0). However, the election latency of MELFA increases up to 200 nodes, and then, decreases for the remaining nodes due to the node compactness.

In order to analyze the impact of transmission range, we consider a case, where each algorithm progresses to elect a new leader in the presence of the leader-in-charge; see Figure 6. However, we are not considering Lee's algorithm [24] because it does not provide any method to elect a new leader when a list of five leaders fails. In

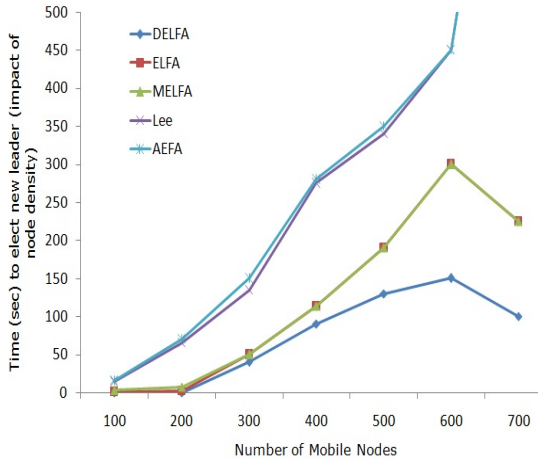


Figure 4: Impact of the node density when the algorithms initiate first time.

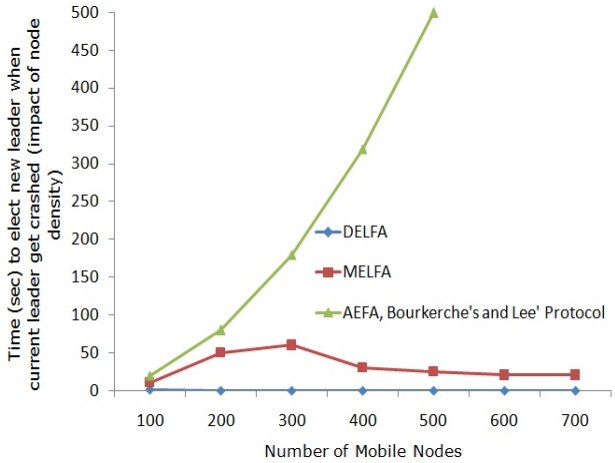


Figure 5: Impact of the node density when the algorithms elect a new leader in the presence of the leader-in-charge.

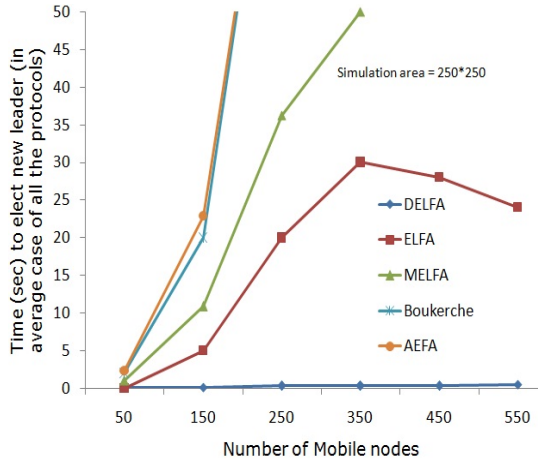


Figure 6: Impact of the transmission range on the nodes when they elect a new leader in the presence of the leader-in-charge.

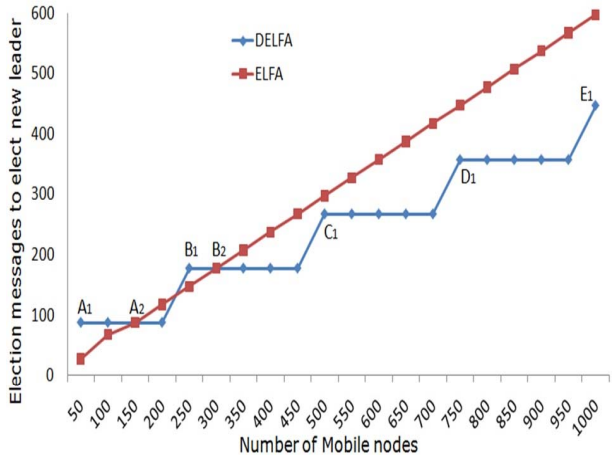


Figure 7: Election messages exchange to elect a new leader in the presence of the leader-in-charge.

addition, we use 250×250 meter^2 area and nodes from 50 to 550. Initially, curves of all the algorithms show almost the same time to elect a new leader. However, an increasing transmission range of the nodes brings more neighbors to each node that results in more messages in AEFA and Bourkerche's algorithm. On the other hand, MELFA and ELFA show a less time to elect a new leader due to the control messages piggyback on application messages. Further, the concept of the house in the *DELFA* signifies a very less time to elect a new leader and a new vice-leader.

Impact of the house concept in the leader election algorithm. In Figure 7, we compare ELFA [38] and the *DELFA* in terms of number of messages exchanged to elect a new leader in the presence of the leader-in-charge. However, we do not focus on Lee's algorithm [24] because it does not consider the election of a new vice-leader when the current leader fails. Note that the cabinet size in ELFA [38] increases linearly with increase in the total number of nodes. However, in case of the *DELFA*, only a significant increase in the total number of nodes results in the growth of the house size; see graph points B_1, C_1, D_1, and E_1 in Figure 7.

In Figure 7, a constant increase in the cabinet size results in a linear graph in ELFA. The graph comprises two critical points, namely A_2 and B_2. On these two points, the curves of ELFA and the *DELFA* overlap. The curve of

ELFA up to A_2 is falling down than the *DELFA*'s curve, and after the point B_2, ELFA curve is always above the curve of *DELFA*. The reason behind these inadequate curves is as follows: initially, at a point A_1, there are only 50 nodes in the network. According to the concept of ELFA, now the cabinet size should be lower, unlike *DEFLA*, where the house size should be greater even in the presence of a fewer number of nodes in the network. As the node count reaches 200, the *DELFA* operates better than ELFA because the cabinet size of ELFA increases faster. The same reason holds for the point B_2 and beyond.

In summary, our fault tolerant algorithm, the *DELFA*, performs better than the other existing best-node-based fault tolerant algorithms — Boukerche' algorithm [5], Lee's algorithm [24] and ELFA algorithm [38] — in terms of the control messages overheads and the election latency.

7 Conclusion

A fault tolerant leader election algorithm for large mobile ad hoc networks is presented. The algorithm elects the best-nodes on the basis of parameters like residual battery power, computing power, memory, node degree, mobility, and hop distance from all the nodes. The proposed algorithm uses the concept of a group (called house) of the better nodes, which is a common practice in the parliamentary system of many democratic countries. The introduction of the house is purely novel in the present context, and it makes the algorithm highly fault resilient. Due to the house, the algorithm requires very less number of exclusive control messages to elect a leader as compared to the other existing best-node-based leader election algorithms. More importantly, the algorithm ensures the high availability of a leader in the occurrence of frequent failures. Specifically, our algorithm elects three best-nodes as the President, the Leader and the Vice-Leader nodes, which serve all the requirements of the network. The President and the Vice-Leader nodes become the leader-in-charge in case of the leader mobility or failure.

The algorithm is suitable for implementation in large size practical networks. We see three future directions: (*i*) since a malicious node can become the leader, one can make the algorithm secure that prevents the network from being captured by a node, (*ii*) reduce the total number of messages when the algorithm initiates, (*iii*) make it self-stabilizing.

Acknowledgements

Authors thank anonymous referees for detailed comments that helped to improve the quality of the paper.

References

[1] A. D. Amis, R. Prakash, D. Huynh, and T. Vuong. Max-min d-cluster formation in wireless ad hoc networks. In *Proceedings IEEE INFOCOM 2000, The Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications, Tel Aviv, Israel, March 26-30, 2000*, pages 32–41, 2000.

[2] L. Arantes, F. Greve, P. Sens, and V. Simon. Eventual leader election in evolving mobile networks. In *Principles of Distributed Systems - 17th International Conference, OPODIS 2013, Nice, France, December 16-18, 2013. Proceedings*, pages 23–37, 2013.

[3] H. Attiya and J. Welch. *Distributed computing: fundamentals, simulations, and advanced topics*, volume 19. John Wiley & Sons, 2004.

[4] F. Borran, R. Prakash, and A. Schiper. Extending paxos/lastvoting with an adequate communication layer for wireless ad hoc networks. In *27th IEEE Symposium on Reliable Distributed Systems (SRDS 2008), Napoli, Italy, October 6-8, 2008*, pages 227–236, 2008.

[5] A. Boukerche and K. Abrougui. An efficient leader election protocol for mobile networks. In *Proceedings of the International Conference on Wireless Communications and Mobile Computing, IWCMC 2006, Vancouver, British Columbia, Canada, July 3-6, 2006*, pages 1129–1134, 2006.

[6] A. Boukerche and K. Abrougui. An efficient leader election protocol for wireless quasi-static mesh networks: Proof of correctness. In *Proceedings of IEEE International Conference on Communications, ICC 2007, Glasgow, Scotland, 24-28 June 2007*, pages 3491–3496, 2007.

[7] J. Cao, M. Raynal, C. Travers, and W. Wu. The eventual leadership in dynamic mobile networking environments.

In *13th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2007), 17-19 December, 2007, Melbourne, Victoria, Australia*, pages 123–130, 2007.

[8] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.

[9] E. Chang and R. Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes. *Communications of the ACM*, 22(5):281–283, 1979.

[10] I. Chlamtac and A. Faragó. A new approach to the design and analysis of peer-to-peer mobile networks. *Wireless Networks*, 5(3):149–156, 1999.

[11] O. Dagdeviren and K. Erciyes. A hierarchical leader election protocol for mobile ad hoc networks. In *Computational Science - ICCS 2008, 8th International Conference, Kraków, Poland, June 23-25, 2008, Proceedings, Part I*, pages 509–518, 2008.

[12] A. Derhab and N. Badache. A self-stabilizing leader election algorithm in highly dynamic ad hoc mobile networks. *IEEE Trans. Parallel Distrib. Syst.*, 19(7):926–939, 2008.

[13] G. H. Forman and J. Zahorjan. The challenges of mobile computing. *Computer*, 27(4):38–47, Apr. 1994.

[14] H. Garcia-Molina. Elections in a distributed computing system. *IEEE Trans. Computers*, 31(1):48–59, 1982.

[15] V. K. Garg. *Concurrent and distributed computing in Java*. Wiley, 2004.

[16] S. Ghosh. *Distributed Systems: An Algorithmic Approach*. Chapman & Hall/CRC Computer & Information Science Series. Taylor & Francis, 2010.

[17] M. A. Haddar, A. H. Kacem, Y. Métivier, M. Mosbah, and M. Jmaiel. Electing a leader in the local computation model using mobile agents. In *The 6th ACS/IEEE International Conference on Computer Systems and Applications, AICCSA 2008, Doha, Qatar, March 31 - April 4, 2008*, pages 473–480, 2008.

[18] D. S. Hirschberg and J. B. Sinclair. Decentralized extrema-finding in circular configurations of processors. *Commun. ACM*, 23(11):627–628, 1980.

[19] T. Hou and T. Tsai. A access-based clustering protocol for multihop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 19(7):1201–1210, 2001.

[20] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the 1st ACM Workshop on Security of ad hoc and Sensor Networks, SASN 2003, Fairfax, Virginia, USA, 2003*, pages 135–147, 2003.

[21] R. Ingram, P. Shields, J. E. Walter, and J. L. Welch. An asynchronous leader election algorithm for dynamic networks. In *23rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009, Rome, Italy, May 23-29, 2009*, pages 1–12, 2009.

[22] A. D. Kshemkalyani and M. Singhal. *Distributed Computing: Principles, Algorithms, and Systems*. Cambridge University Press, New York, NY, USA, 1 edition, 2008.

[23] L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, 1998.

[24] S. Lee, R. M. Muhammad, and C. Kim. A leader election algorithm within candidates on ad hoc mobile networks. In *Embedded Software and Systems, [Third] International Conference, ICESS 2007, Daegu, Korea, May 14-16, 2007, Proceedings*, pages 728–738, 2007.

[25] B. Lehane, L. Doyle, and D. O'Mahony. Ad hoc key management infrastructure. In *International Symposium on Information Technology: Coding and Computing (ITCC 2005), Volume 2, 4-6 April 2005, Las Vegas, Nevada, USA*, pages 540–545, 2005.

[26] N. Malpani, J. L. Welch, and N. H. Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M 2000), Boston, Massachusetts, USA, August 11, 2000*, pages 96–103, 2000.

[27] L. Melit and N. Badache. An energy efficient leader election algorithm for mobile ad hoc networks. In *Programming and systems (ISPS), 2011 10th International Symposium on*, pages 54–59. IEEE, 2011.

[28] N. Mohammed, H. Otrok, L. Wang, M. Debbabi, and P. Bhattacharya. Mechanism design-based secure leader election model for intrusion detection in MANET. *IEEE Trans. Dependable Sec. Comput.*, 8(1):89–103, 2011.

[29] P. Parvathipuram, V. Kumar, and G. Yang. An efficient leader election algorithm for mobile ad hoc networks. In *Distributed Computing and Internet Technology, First International Conference, ICDCIT 2004, Bhubaneswar, India, December 22-24, 2004, Proceedings*, pages 32–41, 2004.

[30] G. L. Peterson. An o(n log n) unidirectional algorithm for the circular extrema problem. *ACM Trans. Program.*

Lang. Syst., 4(4):758–762, 1982.
[31] V. Raychoudhury, J. Cao, R. Niyogi, W. Wu, and Y. Lai. Top k-leader election in mobile ad hoc networks. *Pervasive and Mobile Computing*, 13:181–202, 2014.
[32] D. Raz, Y. Shavitt, and L. Zhang. Distributed council election. *IEEE/ACM Trans. Netw.*, 12(3):483–492, 2004.
[33] G. Roman, Q. Huang, and A. Hazemi. Consistent group membership in ad hoc networks. In *Proceedings of the 23rd International Conference on Software Engineering, ICSE 2001, 12-19 May 2001, Toronto, Ontario, Canada*, pages 381–388, 2001.
[34] S. Sharma and A. K. Singh. Democratic leader finding algorithm for large mobile ad hoc networks. In *31st IEEE International Conference on Distributed Computing Systems Workshops (ICDCS 2011 Workshops), 20-24 June 2011, Minneapolis, Minnesota, USA*, pages 304–312, 2011.
[35] M. Shirmohammadi, M. Chhardoli, and K. Faez. CHEFC: Cluster head election with full coverage in wireless sensor networks. In *Communications (MICC), 2009 IEEE 9th Malaysia International Conference on*, pages 780–784, 2009.
[36] M. Shirmohammadi, K. Faez, and M. Chhardoli. LELE: Leader election with load balancing energy in wireless sensor network. In *Communications and Mobile Computing, 2009. CMC '09. WRI International Conference on*, volume 2, pages 106–110, 2009.
[37] A. Singh and S. Sharma. Message efficient leader finding algorithm for mobile ad hoc networks. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1–6, 2011.
[38] A. K. Singh and S. Sharma. Elite leader finding algorithm for manets. In *10th International Symposium on Parallel and Distributed Computing, ISPDC 2011, Cluj-Napoca, Romania, July 6-8, 2011*, pages 125–132, 2011.
[39] S. Vasudevan, B. DeCleene, N. Immerman, J. F. Kurose, and D. F. Towsley. Leader election algorithms for wireless ad hoc networks. In *3rd DARPA Information Survivability Conference and Exposition (DISCEX-III 2003), 22-24 April 2003, Washington, DC, USA*, pages 261–272, 2003.
[40] S. Vasudevan, J. F. Kurose, and D. F. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *12th IEEE International Conference on Network Protocols (ICNP 2004), 5-8 October 2004, Berlin, Germany*, pages 350–360, 2004.

A Message Complexity of the *DELFA* Algorithm

Most of the LE algorithms, given in [5, 6, 12, 21, 24, 26, 29, 31, 35, 36, 37, 38, 39] for MANETs, try to decrease the total number of messages by imposing some logical structures or by placing some static nodes among mobile nodes. Some of the algorithms [5, 12, 21, 31, 38, 36, 35] also respond the network partitioning effectively. The *DELFA* uses fewer control messages as compared to the existing algorithms. In this section, we provide message complexity of the *DELFA*. We consider three cases, namely the best, the average, and the worst cases based on the total number of messages flow in the network.

Best case message complexity. The best case message complexity of the *DELFA* refers to a case (of joining of a new node in the network) that results in the minimum number of messages flow in the network. In this case, the minimum number of messages (a request message, RM, a notify message, NM or an update message, UM) flow in the network.

Assume that a new joining node i becomes a neighbor of the leader node L, and the node i sends a RM to the node L. The leader node in response sends an IM to the new joining node i. Hence, only two messages (*i.e.*, 1 RM and 1 IM) are exchanged to attach the new joining node i in the network.

Note that there may be a case where in a one dimensional linear network (or in any topological network), a new joining i node becomes the neighbor of some nodes, say node j. Soon after that the node j receives a RM, the node j sends a NM and a UM. The reception of the UM at the leader node results in the propagation of an IM. Hence, this scenario results in four messages (*i.e.*, 1 RM, 1 NM, 1 UM and 1 IM) to join the newly arrived node i.

However, the algorithms [39, 40, 5, 24] require an initiation of the algorithm, which results in at most $\mathcal{O}(NE)$ messages exchange, to handle the same case. The algorithms [31], [37], and [38] require at most $\mathcal{O}(N)$, $3(N-1)$, and 4 messages exchange, to handle the same case.

Average case message complexity. The average case message complexity of the *DELFA* refers to a case (of failure of the leader node and re-election of a leader node) that results in more number of messages flow than the best case

message complexity. When the leader node fails, either node P or VL becomes the leader-in-charge and initiates the election process in the house, which results in at most $Size_H - 1 < N$ messages.

Note that under a special circumstance, the joining of a new node in the network may also result in at most N messages. Let a newly arrived node i, which is the neighbor of all the N nodes, sends a RM to its one neighbor and waits for a timeout. If the node i does not receive an IM before timeout, it again sends a RM to another neighbor. Now consider a special situation that the node i is unable to receive a single IM from all the $N-1$ nodes, and finally, the node i receives an IM from the N^{th} nodes. Thus, in this case at most $N+3$ messages (*i.e.*, N RM, 1 NM, 1 UM, and 1 IM) are used.

Thus, the *DELFA* requires at most N messages in the average case. However, the algorithms [39, 40, 5, 24, 31] require an initiation of the algorithm when the leader crashes or when a node joins, which results in at most $\mathcal{O}(NE)$ messages exchange, to handle the case of leader failure and the case of joining of a node. The algorithm [37] also requires at most $3NE$ messages in the average case; however, the algorithm [38] requires at most $\mathcal{O}(N)$ messages exchange in the average case.

Worst case message complexity. The worst case message complexity of the *DELFA* refers to a case (of initiation of the algorithm to elect a leader in an arbitrary network) that results in the maximum number of messages flow.

The *DELFA* requires at most $2nE$) messages in the worst case, when the algorithm finds a leader in an arbitrary network, nE election messages and nE acknowledgment messages are required, when $n \ll N$ nodes execute the election algorithm simultaneously. However, the algorithms [39, 40, 5, 24, 37, 38, 31] require at most $\mathcal{O}(NE)$ messages to handle the same case.

B Correctness Proof

In this section, we prove the safety and liveness properties of the *DELFA*, where we use Boolean operators, quantification operators ($\forall, \exists$), and temporal operators (always $\Box$, eventually $\Diamond$). Recall that the *DELFA* provides an eventual leader, and we prove this property in this section. The notations used in the correctness proof are given in Table 9.

n	Number of participating nodes	N	Number of nodes in the network
$CODR_i$	current leader at node i	$\mathcal{M}_x$	x^{th} MANET component
$CODR_{\mathcal{M}_i}$	the current leader of i^{th} MANET	$\mathrm{P}_{\mathcal{M}_x}$	the current president node of x^{th} MANET
$\mathrm{L}_{\mathcal{M}_x}$	the current leader node of x^{th} MANET	$\mathrm{VL}_{\mathcal{M}_x}$	the current vice-leader node of x^{th} MANET
$SEND_i(m,j)$	node i sends a message m to a node j	$RECV_i(m,j)$	node i receives a message m from a node j
$State_i$	the current state of node i	E	A set of edges between neighbor nodes

Table 9: Notations used in the correctness proof.

Theorem 1 (Safety) *An execution of the proposed algorithm eventually leads to a unique* P, L, *and* VL *nodes in each individual MANET component, and all the other nodes (of the components) agree on the same elected leader node* L. *Formally,*

$$\forall\, i,j : 1 \le i,j \le N, (State_i = \text{NORMAL} \wedge State_j = \text{NORMAL} \wedge NW_{\mathrm{L}} > 70) \Rightarrow (CODR_i = CODR_j) \quad (1)$$

Proof. We prove three claims to show the uniqueness of the leader node, as: (*i*) an execution of PHASEs 1-3 results in a single unique leader, (*ii*) when any two components merge (due to a communication link formation), there is a unique leader in the merged component, and (*iii*) when a component partitions into x components, then all the x components have their unique leaders. In each claim, we show that eventually each component has a unique leader node.

Claim 1 *An execution of* PHASE*s 1-3 results in a single unique leader.*

Proof. We prove the claim by contradiction. Assume that a single MANET component has more than one leader nodes. In other words, two nodes i and j that belong to a MANET have two different leader nodes. Formally,

$$\exists\, i,j \in \mathcal{M}_x, (State_i = \text{NORMAL} \wedge CODR_i = \mathrm{L}_1) \wedge (State_j = \text{NORMAL} \wedge CODR_j = \mathrm{L}_2) \quad (2)$$

When PHASEs 1-3 execute, multiple leaders in a MANET can exist if there are multiple originator (ori) nodes, which start PHASE 1 when a MANET does not hold a unique leader. In this situation, two ori nodes on timeout

may elect two different nodes as a leader of the network on the basis of the received NW of the nodes. However, such a condition exists until EMs with the minimum identity ori node receive at the nodes. Once a node that holds a leader and receives an EM whose identity of the ori node is smaller than the identity of the most recently forwarded EM, the node again switches to CANDIDACY state and sets its $CODR_j = \perp$; see Phase 2 in Table 7. This process will continue until all the node have the same value of ori node in their data-structure. This fact results in that eventually only a single ori node will elect a leader node, and a MANET component has a unique leader node, which contradicts the assumption given in equation 2.

Claim 2 *When any two components merge (due to a communication link formation), there is a unique leader in the merged component after at most* $\max(2D_1, 2D_2)$ *rounds, where* D_1 *and* D_2 *are the diameter of two components* C_1 *and* C_2*, respectively.*

Proof. Any two (or more) components can merge when a new node joins or one (or more) node of two different MANETs become neighbors. We prove the claim by contradiction. Assume that a node receives heartbeat messages from two different leader nodes, which belong to two different MANETs. A trivial case is when two leader nodes become neighbors. In this case, one of them with the lowest NW becomes a normal node and the other one becomes a leader of the merged component, and hence, the merged component has a unique leader.

Now consider a scenario where two components may merge when one or more nodes of different MANET components become neighbors, and these nodes are never aware of the merging of two components. In this case, the nodes that receive heartbeat messages from two leader nodes may delay the process to have a unique leader in the network. Such nodes may consider the network merging as a situation that PHASE 1 is triggered, and eventually they will receive an EM from the minimum identity ori node. However, they will not receive any EM. According to EVENT 8, the nodes that receive heartbeat messages from two leader nodes ask the leader nodes about their node-weights that takes at most $\max(D_1, D_2)$ rounds. Once the node receives leaders' node-weight, then the node informs the leader node whose node-weight is smaller to surrender its leadership that takes at most $\max(D_1, D_2)$ rounds. Therefore, when two or more components merge, the merged component has a unique leader node after at most $\max(2D_1, 2D_2)$ rounds.

Claim 3 *When a component partitions into* x *components, then all the* x *components have their unique leaders.*

Proof. This claim holds directly by following Claim 1, see above, and EVENT 7 for the three components, where each component has a special node (Case 1). Also, the claim holds for x components, when each component has at least a single node with $NW > 69$.

Therefore, the proposed algorithm ensures the existence of a unique leader in the network. Following the similar argument, we can show that the network holds unique P and VL nodes.

■

Theorem 2 (Liveness) *Every node* i *succeed to possess only a single leader within a finite amount of time. Formally,*

$$\nexists\, i, \Box\ CODR_i = \perp \tag{3}$$

Proof. We prove this theorem by contradiction. Assume a contrary that there is a node i that never possesses a leader. Formally,

$$\exists\, i, \Box\ CODR_i = \perp \tag{4}$$

Here, we prove five claims, which show equation 4 is impossible to be true forever, and hence, every node i eventually succeeds to have a leader.

Claim 1 *The algorithm is able to elect a leader after at most* $2D$ *rounds, where* D *is the network diameter. Formally,*

$$\forall i \in N : i \in \mathcal{M}_x \Rightarrow \Box[CODR_i \neq \perp] \tag{5}$$

Proof. Assume a contradiction that the algorithm is not able to elect a leader forever. Such a condition will exist if there is no node having $NW > 69$ or every node has a false leader; so that in both the cases, no node is able to execute PHASE 1.

However, in a network of no unique leader, at least a single node whose $NW > 69$ initiates the election process, *i.e.*, PHASE 1. The node i sends an election message, EM, to all its neighbor nodes, and the propagation of the

election message takes at most D rounds. When a node j whose $NW > 69$ receives an EM, the node j sends an acknowledgement message, AM, to the node i. If there are some nodes having $NW > 69$, then the node i receives AMs in at most $2D$ rounds. On the other hand, the node i elects itself as a leader of the network in the absence of AMs after at most $2D$ rounds. Hence, the algorithm is able to elect a leader after at most $2D$ rounds, where D is the network diameter.

Note that the network partitioning may leave a node without a leader forever. However, we proved in Theorem 1 that in such a condition, every node has a unique leader forever. Hence, in the network partitioning, the algorithm is able to elect a unique leader too.

Claim 2 *When a node i once receives an election message, EM, and after that is not moved to a different location, which results in the network partition, the node i receives a coordinator message, CM, within at most $3D$ rounds, where D is the network diameter. Formally,*

$$\begin{aligned} \forall\, \mathcal{M}_x,\, \exists\, i, j \in n,\, i \neq j, i, j \in \mathcal{M}_x \;\wedge\; RECV_i(EM, j) \\ \Rightarrow\; \Diamond[RECV_i(CM, j) \;\wedge\; (CODR_i = CODR_j)] \end{aligned} \tag{6}$$

Proof. After $2D$ rounds the ori node elects a leader node (see Claim 1 above), and sends a coordinator message, CM, to its neighbors. In this manner, one-hop neighbors of the ori node hold information about the leader node after at most $2D + 1$ rounds. Following that, in general, the nodes at distance d receives CMs after at most $2D + d$ rounds. Thus, the far most node j at distance D from the ori node receives CMs after at most $3D$ rounds, and hence, all the nodes receive CMs eventually, if they are not partitioned, within at most $3D$ rounds.

Claim 3 *If a new joining node i joins the network, then it possesses a leader eventually after at most $2x$ rounds, where x is the total number of new joining nodes. Formally,*

$$\begin{aligned} \forall\, \mathcal{M}_x, \exists\, j \in N, \exists\, i, i \neq j : i \in neighbor\_name_j[] \;\wedge\; SEND_i(RM, j) \\ \Rightarrow\; \Diamond[RECV_i(NM, j) \;\wedge\; RECV_i(IM, \mathrm{L})] \end{aligned} \tag{7}$$

Proof. Assume a contrary that a new joining node i does not receive a notify message, NM, and an information message, IM, forever. Formally,

$$\begin{aligned} \forall\, \mathcal{M}_x, \exists\, j \in N, \exists\, i, i \neq j : i \in neighbor\_name_j[] \;\wedge\; SEND_i(RM, j) \\ \Rightarrow\; \Box\neg[RECV_i(NM, j) \;\wedge\; RECV_i(IM, \mathrm{L})] \end{aligned} \tag{8}$$

Now we will see that there are only two scenarios that hold equation 8 to be true; however, we prove that in the *DELFA*, these scenarios are unable to hold forever, as follows:

Scenario 1: Assume that the node i is isolated after sending a request message, RM. In this trivial case, according to Theorem 1, the node i elects itself as a leader, if it has $NW > 69$.

Scenario 2: Assume that the node i has moved from its location after sending a RM, and becomes a member of another MANET, $\mathcal{M}_z$. We show that if the node i does not move out of the range of its neighbor $j \in \mathcal{M}_z$, then eventually, the node i, receives a NM. Let the node j is also a newly arrived node, and the node j does not hold the leader information. Thus, the node j sends a wait message, WM, to the node i. However, the node j has already sent a RM to its neighbor, say, p. The node p again sends a WM to the node j, if the node p is also a new joining node. This may form a waiting chain of a finite length. Any node z in the waiting chain either has the current leader's information or itself be a leader node. Hence, the node z would send the current leader's information to the node p. Once this occurs, there is a chain of NMs and IMs that propagates down to the node i. Therefore, any new joining node eventually becomes aware about the current leader. If x nodes join the network in the form of a linear network, then it takes at most $2x$ rounds to hold the leader information, where x rounds are used to propagate RMs and x rounds are used to propagate NMs.

Claim 4 *When two (or more) network components merge, each node of the merged component has a unique leader after at most $\max(2D_1, 2D_2)$ rounds, where D_1 and D_2 are the diameter of two components C_1 and C_2, respectively.*

We proved this claim in Theorem 1.

Claim 5 *In the event of the leader's crash, the proposed algorithm delivers an alternate leader within a finite time.*

Formally,

$$\forall\, i \in N, i \in \mathcal{M}_x : \neg \mathrm{L}_{\mathcal{M}_x} \;\Rightarrow\; \Diamond \mathrm{L}_{\mathcal{M}_x} \tag{9}$$

Proof. Assume a contrary that after the leader's crash, the network is unable to possess a new leader. Formally,

$$\forall\, i \in N, i \in \mathcal{M}_x : \neg \mathrm{L}_{\mathcal{M}_x} \;\Rightarrow\; \Diamond \neg \mathrm{L}_{\mathcal{M}_x} \tag{10}$$

Now we will see that there is only a single scenario that holds equation 10 to be true; however, we prove that this scenario is unable to hold in the *DELFA* forever, as follows:

Consider a scenario that the P, VL, and all the house nodes at time α, and the leader node crashes at time β, where $\alpha < \beta$. Hence, there is no special node that can handle this situation, and there is no node in the house that can initiate PHASE 1 to elect a new leader. However, according to our settings, the node P is the best NW node and remains stay after the failure of L, VL, and all the house nodes (EVENTs 4 and 5). Hence, the node P becomes a leader-in-charge and initiates the election process (EVENT 6) in the house, if they have some nodes, to elect tentative L and VL. Thus, it is impossible that the node P crashes before the leader's crash; so that the network holds an alternative leader within a finite time, and equation 10 holds to be false.

Therefore, every node i has a unique leader eventually, and equation 4 cannot hold to be true forever. ■

Theorem 3 *The DELFA is a fault-containing algorithm.*

Proof. An algorithm that assures that starting from a nice configuration and then subsequent transient faults, a nice configuration will be again reachable without a re-initialization of the network is known as a fault-containing algorithm. In the context of the *DELFA*, a nice configuration refers to a unique leader, president, and vice-leader nodes in the network, and it will be a fault-containing algorithm if it will elect a new leader when the current leader fails, without a re-initialization of the network (*i.e.*, execution of PHASE 1), and maintain a single president and vice-leader for the entire duration of the network. This property is directly followed by Theorems 1 and 2. ■