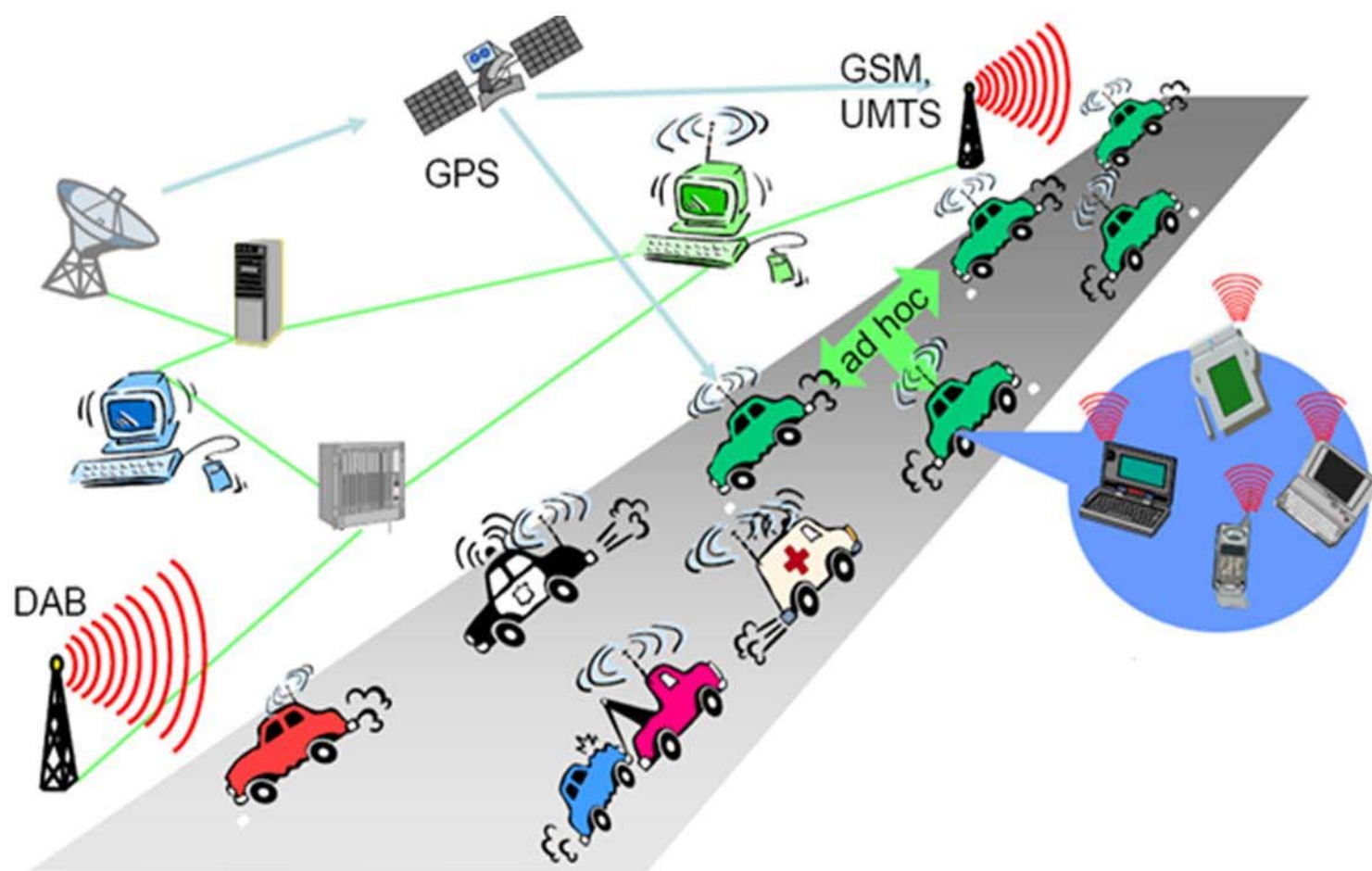


HIGHLY AVAILABLE COORDINATOR FOR MOBILE AD HOC NETWORKS

Dissertation for
Master of Technology in
Computer Engineering



SHANTANU
(Roll No. 209122)

Under the supervision of
DR. A. K. SINGH

National Institute of Technology, Kurukshetra
April-2011

Cover image borrowed from Lecture notes by Roger Wattenhofer, on Mobile Computing. Distributed Computing Group, Summer 2004. [42]

HIGHLY AVAILABLE COORDINATOR FOR MOBILE AD HOC NETWORKS

*Dissertation submitted in the partial fulfillment of the requirements
for the degree of*

Master of Technology
in
Computer Engineering
by

Shantanu
(Roll No. 209122)

Under the supervision of
Dr. Awadhesh Kumar Singh



**DEPARTMENT OF COMPUTER ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
KURUKSHETRA – 136119**

April - 2011

HIGHLY AVAILABLE COORDINATOR FOR MOBILE AD HOC NETWORK

A Dissertation

by

Shantanu

Submitted to

National Institute of Technology, Kurukshetra

in partial fulfillment of the requirements

for the

Master of Technology in Computer Engineering

Approved as to style and content by:

Dr. Awadhesh Kumar Singh, Supervisor

**Dr. Mayank Dave, Head of Department,
Computer Engineering**

CONTENTS

Contents	i
Abstract	iii
Acknowledgment	v
List of Acronyms	vi
List of Tables	vii
List of Figures	viii
1 Introduction	1
1.1 Mobile Ad hoc Network	2
1.2 Leader Election	6
1.3 Organization of Dissertation	9
2 Background and Objective	10
2.1 Static Scenario	11
2.2 Dynamic Scenario	13
2.3 Objective of Dissertation	18
3 Message Efficient Leader Election	20
3.1 System Model	21
3.2 Messages & Data Structures	22

3.3	The MELFA Protocol	24
3.4	Correctness Proof	29
3.5	Message Complexity	30
3.6	Discussion	32
4	Elite Leader Election	33
4.1	System Model	34
4.2	Messages & Data Structures	34
4.3	The ELFA Protocol	36
4.4	Correctness Proof	43
4.5	Message Complexity	47
4.6	Discussion	50
5	Democratic Leader Election	51
5.1	System Model	53
5.2	Messages & Data Structures	53
5.3	The DELFA Protocol	56
5.4	Correctness Proof	66
5.5	Message Complexity	70
5.6	Discussion	72
6	Simulated Results	73
7	Stopping Points	82
	Bibliography	84
	List of Papers Communicated	89

ABSTRACT

The distributed computing algorithms are well known for their complexity in design, verification and implementation. They behave more notoriously when they are exposed to more hostile environment like MANET. The mobile ad hoc networks have succeeded in drawing the vast attention of both, academia as well as industry, due to its diverse imputed apps that they provide for personal, commercial and emergency intentions. Another higher level research is also continuing on Self-Stabilization, since 1973, in distributed system. However, in context of MANETs, most of the research work is focused towards MAC and routing concerns. In this dissertation, we focus on leader election in mobile ad hoc networks with minimal help from MAC and routing layers.

In order to coordinate many mobile computing applications, in MANETs, it is essential to elect a leader. Hence, a large number of protocols have been proposed for coordinator election in MANETs. However, most of them use message broadcast technique and provide limited failure resiliency. Although, most of the protocols focus on reducing the number of control messages, there have been very little attention on ensuring the high availability of leader in the event of various types of failures, *e.g.*, leader crash and unreachability of leader, especially in the scenario like rescue and warfare, where the absence of leader, even for a short duration, may lead to havoc.

As title indicates, the dissertation is a collection of works oriented towards ensuring high availability of coordinator in MANETs. We have developed three new deterministic leader election protocols, namely MELFA, ELFA and DELFA. To the best of our knowledge, the concept used in the protocols has not been used earlier. Moreover, the protocols ELFA and DELFA are self-stabilizing ones.

Our first leader election protocol, MELFA, Message Efficient Leader Finding Algorithm, avoids the broadcast of messages and uses multicast and unicast for leader election. The protocol initiates election process with broadcast of

messages. Nevertheless, as election progresses, the protocol uses only multicast, and eventually unicast, with some information piggybacked. However, MELFA is not fault tolerant.

Our second leader election protocol, ELFA, Elite Leader Finding Algorithm, elects a vice-coordinator and a group of elite nodes, called cabinet, which provides a fault free leader election for average sized MANETs. The cabinet concept, inspired from prevailing parliamentary polity in most countries of the world, brings in failure resiliency to our protocol.

Finally, we propose DELFA, DEMocratic Leader Finding Algorithm, is especially designed for large MANETs, where the absence of coordinator may cause mayhem. DELFA is also stimulated by democratic civil order that always ensures the existence of an executive in order to coordinate and take decisions regarding affairs of the state. In the protocol, by adapting the terminology of parliamentary system, we create Lower House and Upper House of special nodes. In addition, there are some designated nodes, like President, Leader, Vice-Leader and Vice-President to ensure very high availability of leader in large MANETs.

ACKNOWLEDGEMENT

It is pleasure to acknowledge our debt to the many persons involved, directly or indirectly. No task is a single man's effort. The cooperation and coordination of various people at various places go into the successful implementation. It is impossible to thank individually. First of all, I would like to thank almighty God who gave me the inspiration to take up this task.

I acknowledge my supervisor Dr. Awadhesh Kumar Singh, without his guidance, advice and encouragement this thesis would not have been possible. He instructed me how to do quality research and was always available when I needed any technical advice. The high standards, he set through his professionalism and a rigorous approach towards research has contributed immensely towards my own professional growth. I hope to emulate these standards throughout my career. There is still a lot more I can learn from him, both professionally and personally. For this reason, I hope to continue interacting with him in the future.

LIST OF ACRONYMS

4G: Fourth Generation
AEFA: Asynchronous Extrema Finding Algorithm
AIM: Acknowledge Information Message
AM: Acknowledgement Message
APCM: Acknowledge President Check Message
BAN: Body Area Network
BS: Base Station
CAB: CABinet message
CAG: Coordinator Advisory Group member
CHOP: Configure, Heal, Optimize, Protect
CM: Coordinator Message
DELFA: DEMocratic Leader Finding Algorithm
ELFA: Elite Leader Finding Algorithm
EM: Election Message
HRM: HeaRtbeat acknowledge Message
IM: Information Message
LH: Lower House
LHM: Lower House Member node
MANET: Mobile Ad hoc NETwork
MELFA: Message Efficient Leader Finding Algorithm
MP3: Moving Pictures Expert Group Audio Layer 3
NM: Notify Message
NoM: Nomination Message
NW: Node Weight
ORD: ORDinary Node
ORI: ORIginator
PAN: Personal Area Network
PCM: President Check Message
PDA: Personal Digital Assistant
PM: Prime Minister
RH: Reserve House
RHM: Reserve House Member node
RM: Request Message
SEFA: Secure Extrema Finding Algorithm
SPLEA: Secure Preference-based Leader Election Algorithm
UAV: Unmanned Aerial Vehicles
UH: Upper House
UHM: Upper House Member node
UM: Update Message
VL: Vice-Leader
VP: Vice-President
WM: Wait Message

LIST OF TABLES

Table 1.1. General Applications of MANETs	5
Table 3.1. Initial Node's Neighbors for MELFA	27
Table 4.1. Initial Node's Neighbors for ELFA	41
Table 5.1. Initial Node's Neighbors for DELFA	65

LIST OF FIGURES

Figure 1.1: Real MANET Example	2
Figure 1.2: Characteristics of MANETs	3
Figure 1.3: Inter-node Communication and Data Exchange	7
Figure 1.4: Leader in Commit Operation	8
<hr/>	
Figure 2.1: Classification of Coordinator Election Protocol for Static System	11
Figure 2.2: Classification of Coordinator Election Protocol for Dynamic System	14
<hr/>	
Figure 3.1: MANET in form of Graph	21
Figure 3.2: System Model	22
Figure 3.3: Solution to Broadcast Storm Problem	26
Figure 3.4: MELFA in Execution	28
Figure 3.5: Two Isolated Nodes	29
Figure 3.6: Two Isolated MANETs	29
Figure 3.7: Best Case Scenario 1 of MELFA	30
Figure 3.8: Best Case Scenario 2 of MELFA	31
Figure 3.9: Average Case of MELFA	32
<hr/>	
Figure 4.1: System Model	35
Figure 4.2: ELFA Protocol State Diagram	40
Figure 4.3: ELFA in Execution	42
Figure 4.4: Network Partition and Merge Handling	44
Figure 4.5: Best Case Scenario 1 of ELFA	48
Figure 4.6: Best Case Scenario 2 of ELFA	48
Figure 4.7: Average Case of ELFA	49
Figure 4.8: Worst Case Scenario 1 of ELFA	49
Figure 4.9: Worst Case Scenario 2 of ELFA	50
<hr/>	
Figure 5.1: Nodes in DELFA	53
Figure 5.2: System Model	54
Figure 5.3: Modified MELFA in Execution	57
Figure 5.4: DELFA Protocol State Diagram	59
Figure 5.5: Event 8-12 of DELFA in Action	61
Figure 5.6: DELFA in Execution	64
Figure 5.7: Network Partition Handling and Merge Operation of DELFA	65
Figure 5.8: Best Case Scenario 1 of DELFA	70
Figure 5.9: Best Case Scenario 2 of DELFA	70

Figure 5.10: Average Case Scenario 1 of DELFA	71
Figure 5.11: Average Case Scenario 2 of DELFA	71
Figure 5.12: Worst Case of DELFA	72
<hr/>	
Figure 6.1: MELFA and AEFA – Number of Nodes v/s EMs Exchange, Case 1	75
Figure 6.2: MELFA and AEFA – Number of Nodes v/s EMs Exchange, Case 2	75
Figure 6.3: MELFA and AEFA – Number of Nodes v/s Election Latency, Case 1	76
Figure 6.4: MELFA and AEFA – Number of Nodes v/s Election Latency, Case 2	76
Figure 6.5: MELFA and AEFA – Node Density	77
Figure 6.6: Best case of MELFA, ELFA and DELFA	78
Figure 6.7: Average case of MELFA, ELFA and DELFA	78
Figure 6.8: Worst Case of MELFA, ELFA and DELFA	79
Figure 6.9: Election Message Exchange to Elect Tentative Leader in Average Case of ELFA and DELFA	80
Figure 6.10: Election Message Exchange to Elect Tentative Leader in Worst Case of ELFA and DELFA	80
Figure 6.11: Nodes in ELFA without Leader	81

CHAPTER 1

INTRODUCTION



1.1 MOBILE AD HOC NETWORK

- Characteristics
- Applications
- Major Issues

1.2 LEADER ELECTION

- Fundamental of Leader Election
- Application of Leader Election
- General Properties of Leader Election
- Challenges towards Coordinator Election in MANET

1.3 ORGANIZATION OF DISSERTATION

“When we had no computers, we had no programming problem either. When we had a few computers, we had a mild programming problem. Confronted with machines a million times as powerful, we are faced with a gigantic programming problem”

- Edsger Wybe Dijkstra

Wireless communication is not merely a buzz word, rather, it is an innovative paradigm that is successfully providing services to human being from wireless LANs [38] to mobile telephone communication, wireless PANs [39], and sensor networks. Especially, last decade has witnessed tremendous growth in wireless communication that has triggered and fueled the area of distributed computing. Now days, due to active research, the horizon of distributed computing has been further pushed to the field of Mobile Computing [40] with an objective to furnish efficient solutions in wireless domain with minimal cost and reduced power consumption by handheld devices.

1.1 MOBILE AD HOC NETWORK

The ad hoc networks have become focal interest of researchers in last few years due to its rapid deployability, self configuring nature. The mobile ad hoc networks (MANETs) [41] are collection of peer to peer mobile nodes which are dynamic and designed as per need. The life span and mobility of nodes in MANETs are very constrained, unlike cellular systems where nodes have larger life line as well as greater mobility under some pre-designated coordinator (BSS). Any node, in such a network, can communicate directly with its neighbors that are in its transmission radii. A mobile ad hoc network can be defined as following:

A mobile ad hoc network is an infrastructureless collection of mobile nodes without any pre-designated coordinator. In such a network, each participating node acts as host as well as router. Any two nodes are called neighbors if they are in the transmission range of each other and can communicate. The node mobility may often lead to link breakages and link formations.

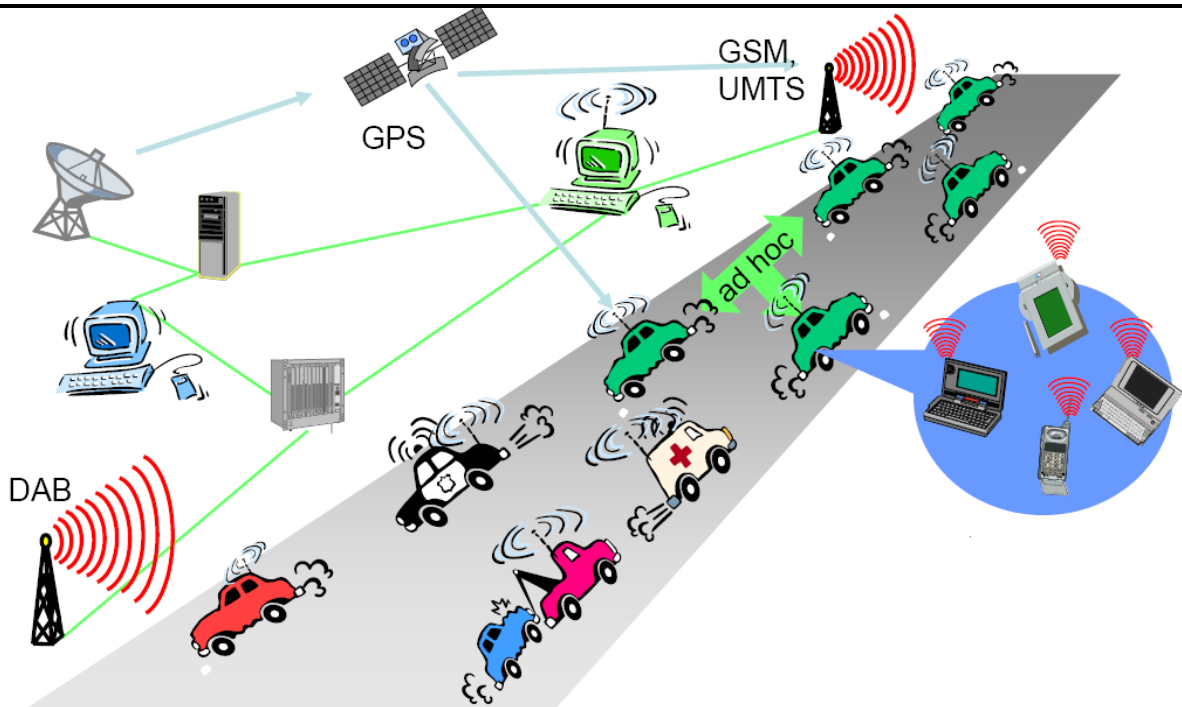


Figure 1.1: Real MANET Example. [42]

We can classify ad hoc networks, depending on their coverage area, into several classes [43] like Body Area Network (BAN), Personal Area Network (PAN), etc.

1.1.1 Characteristics of MANETs:

As, ad hoc networks are autonomous network, IBM [37, 44] defined four major and four minor characteristics of ad hoc network. Based upon human biological system, they have termed self-CHOP to these four major characteristics. All these characteristics are as follow:

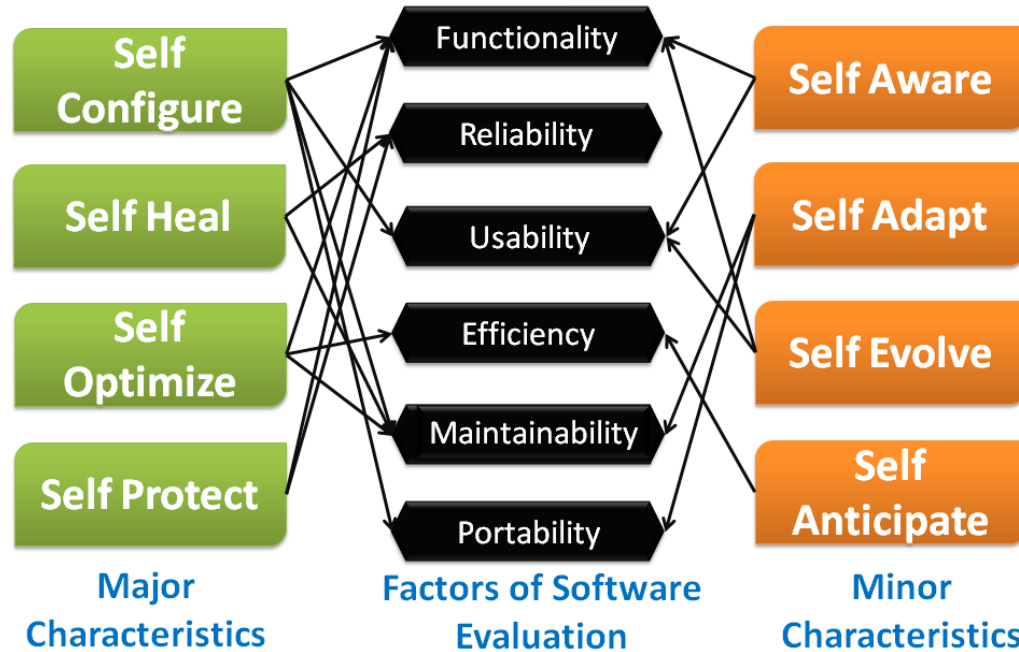


Figure 1.2: Characteristics of MANETs. Four major and minor characteristics of MANETs. [37]

- **Self Configure:** Self-configure is the property to implement specific strategies to change the relations among the components to guarantee either survivability in changing environments or a higher performance.
- **Self Heal:** Self heal is the property to detect or predict faults and automatically correct faults (events that cause the entire system or parts of it to malfunction).
- **Self Optimize:** Self optimize is the property of monitoring its components and fine-tune the resources automatically to optimize the performance.
- **Self Protect:** Self Protect is the property of anticipating, detecting, identifying and protecting itself from attacks in order to maintain overall integrity.

Additionally, the four minor characteristics are:

- **Self Aware:** Self aware means knowing itself (its components, resources, the relations among them, and the limits) in a detailed manner.
- **Self Adapt:** Self adapt means automatically identifying the environment, generating strategies on how to interact with neighboring systems, and adapt its behavior to a changing environment.

- **Self Evolve:** Self evolve means generating new strategies and implementing open standards.
- **Self Anticipate:** Self anticipate means anticipating the requests for resources from the users without involving them in the complexity of its functionality.

In addition, MANETs also have the following characteristics [29]:

- **Ad hoc Deployment:** The nodes may be positioned according to need of application, which is not necessarily in a regular pattern like grid and hypercube, etc. Due to mobility characteristic, all nodes get their arbitrary location in the network. In addition, high mobility results in frequent topology changes and unpredictably of position, in which many mobile nodes move in and out of wireless network without any fixed access point. Moreover, the portability of nodes and the communication medium being wireless, MANETs provide very high speed of deployment.
- **Limited Resources and Energy:** Unlike, static hosts, mobile hosts have limited resources, *e.g.*, battery power, memory size, computational power, bandwidth, transmission range, etc. Thus, it is required to execute minimum number of actions by all nodes in order to overcome the resource constraints.
- **Multiple Components:** The communication links among mobile nodes may go up and down frequently, due to node movement and limited resources at nodes. Thus, the MANETs may consist of multiple connected or disconnected components.
- **Multi-hop Communication:** In ad hoc network, limited transmission range of nodes result in multi-hop communication. Suppose, for example, mobile node *A* wants to send data to some other node *B* that is out of range of *A* and hence, the data transmission needs to relay through several in-between nodes.
- **Error-prone Wireless Medium:** The wireless medium, being an open channel, is more error-prone than the wired medium and collisions could occur more frequently.
- **Limited Security:** Security is a prevention measure against malicious attacks at the cost of high computation; thus, due to limited resources at host, it cannot be implemented easily in MANETs.

1.1.2 Application of MANETs:

Ad hoc networks have been gaining more research interest in recent few years. They can be used for various personal, commercial and emergency purposes. Some of the major applications of MANETs are as following:

- **Personal Purposes:** MANETs have been gaining more popularity in personal usage field, *e.g.*, mobile phones, laptops, palmtops, tablets, PDA, MP3 player, camcorders, etc, that assist the home applications, like conferences and meetings, to create instant network for the purpose of sharing file, data, video and audio, without the presence of any assigned leader.
- **Commercial Purpose:** Ad hoc network can also be connected to a fixed backbone network through a dedicated gateway enabling IP networking services in the areas where Internet services are not available due to lack of preinstalled infrastructure [43]. In addition, ad hoc networks have also become the base for 4G networks which provides pervasive computing environments that can

seamlessly and ubiquitously support users in accomplishing their tasks, in accessing information or communicating with other users at anytime, anywhere, and from any device [49].

- **Emergency Scenarios:** Due to unplanned nature of ad hoc networks, they are easily implemented in emergency conditions, *e.g.*, war, rescue, terrorist attacks, disaster management and search [49], where rapid deployment of mobile node is utmost necessary. In war fare scenario, ad hoc networks are used for military purposes where soldiers can track enemies and their activities when they move through the geographic area covered by the pre-established network.

Table 1.1. General Applications of MANETs. [45]

S. No.	Application Domain	Description
1.	Tactical networks	Military communication and operations Automated Battlefields
2.	Emergency services	Search and rescue operations Disaster recovery, <i>e.g.</i> , early retrieval and transmission of patient data (record, status, diagnosis) from/to the hospital Replacement of a fixed infrastructure in case of earthquakes, hurricanes, fire, etc.
3.	Commercial environments	E-Commerce, <i>e.g.</i> , electronic payments from anywhere Business: – dynamic access to customer files stored in a central location on the fly provide consistent databases for all agents – mobile office Vehicular Services: – transmission of news, road conditions, weather, music – local ad hoc network with nearby vehicles for road/accident guidance – Unmanned Aerial Vehicles (UAV) — remotely piloted or self-piloted aircrafts that can carry cameras, sensors, communications equipment or other payloads
4.	Home and enterprise networking	Home/office wireless networking (WLAN), <i>e.g.</i> , shared whiteboard application, use PDA to print anywhere, trade shows Personal Area Network Spontaneous Networking
5.	Educational applications	Set up virtual classrooms or conference rooms Set up ad hoc communication during conferences, meetings or lectures
6.	Entertainment	Multiuser games Robotic pets Outdoor Internet access
7.	Sensor networks	Home security and tracing Indoor/outdoor environmental monitoring Disaster prevention Health and wellness monitoring Power monitoring Location awareness Factory and process automation Military applications
8.	Mesh networks	Residential zones (broadband Internet) Highway communication facilities for moving vehicles business zones, important civilian regions, university campuses
9.	Hybrid networks	Enhancements of cell coverage and connectivity of holes

1.1.3 Issues in MANETs:

There are number of issues in MANETs and can be classified as: (i) medium access related (*e.g.*, distributed operations, hidden terminal problem, exposed node terminal, access delay, throughput, real time traffic support and resource reservation), (ii) routing algorithm related (*e.g.*, mobility, bandwidth, error prone, shared medium and resource constraint), (iii) multicasting related (*e.g.*, robustness, efficiency, scalability, security and efficient group management), (iv) QoS related (*e.g.*, reliability, delay, jitter, bandwidth and QoS aware routing), (v) security related (*e.g.*, shared medium, lack of central authority, low power at nodes, insecure operational environment and physical vulnerability), (vi) energy management related (*e.g.*, transmission power management, battery power management, process power management and device power management), and (vii) development related (*e.g.*, low cost, choice of protocol, area coverage and reconfiguration).

The above mentioned issues affect the performance of MANET. However, the performance and efficiency of distributed services running on MANET must be unaffected by these factors as far as possible. Hence, the distributed protocols, which handle various computing challenges, *e.g.*, mutual exclusion, leader election, termination detection, etc., should be developed keeping these issues aside, though these issues do affect them.

1.2 LEADER ELECTION

The coordinator election is a classical challenge in static as well as dynamic networks. Therefore, it is an extensively studied problem in distributed systems to coordinate and monitor various applications running in the network. In fact, our contemporary literature includes a large number of leader election protocols for static environment. However, leader election in MANETs is comparatively a less explored area of research. In order to monitor ad hoc applications, it is necessary for MANET to possess a coordinator. In the following section, we will describe the basic of leader election protocol and then conclude the chapter with the specification of objective of this dissertation work.

1.2.1 Fundamentals of Leader Election:

Electing a coordinator is a basic challenge of distributed computing environment in order to coordinate many applications among nodes. Say, there are N nodes in the network, each with a unique identity. Initially, all processes are in the same state, called the candidacy state and any one may initiate the election protocol. After the termination of protocol, exactly one node is chosen as a coordinator by all N processes based on some features like *id*, battery power, computing power, etc, and rest $N-1$ remain ordinary node. The word 'leader' and 'coordinator' have been used interchangeably throughout this dissertation. In general, the protocols for electing a coordinator in MANET have the following assumptions [18]:

- Each node has a unique identifier.
- All nodes are connected by a bidirectional link.
- The communication channel is not necessarily FIFO.
- A node only knows the IDs of its neighboring nodes.

- The nodes remain static during election process.
- All nodes must agree and use the same election protocol.
- Election message has higher priority than any other messages.
- No central controller is considered for initiation of algorithm.

1.2.2 Application of Leader Election:

Coordinator election protocol is used in various scenarios in MANET as following:

- **Inter-node Communication and Data Exchange:** Monitoring communication is the basic responsibility of leader. The leader provides privilege to some mobile node to communicate with each other or to exchange data with other host or with coordinator by allocating channel among them, refer Figure 1.3.

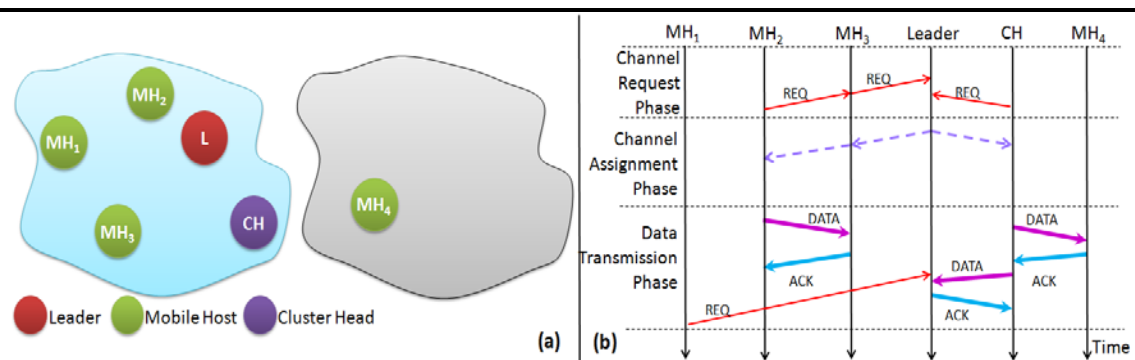


Figure 1.3: *Inter-node Communication and Data Exchange.* In figure (a), two MANETs are shown with mobile host, cluster head and leader. In the figure (b), channel request phase, mobile host MH_2 and MH_3 and cluster head CH request the leader, L , for channel assignment. In response, leader assigns channel to cluster head CH (for inter-cluster communication) and mobile host. Subsequently, mobile node, *i.e.*, MH_2 , MH_3 , and CH exchanges the data. It is worth noting that channel request is queued at leader after channel assignment phase till the next channel allocation.

- **Key Distribution:** In secure communications, the coordinator is responsible for generating secret keys and distributing them among all mobile hosts which enable them to encode and decode information.
- **Serve the Incoming Request Messages:** Leader is also responsible to serve all types of requests generated by various mobile nodes, *e.g.*, the service could generate the valid sequence number and send it by multicasting to the requesting process.
- **Grant Privilege:** In ad hoc network, each node is not free to perform the tasks, *e.g.*, operation on data, accessing shared resources like broadcast on particular radio channels, without receiving privilege from the leader. Coordinator is only creditworthy to provide privilege to host in order to perform these operations.
- **Routing Coordination:** In mobile ad hoc network, each node acts as router as well as host, thus, the coordination between the routers is an important issue. A mobile node holds the routing information in the form of routing table, which stores entry for each destination, *id* of next hop neighbor, number of hops (metric), sequence number for the destination, active neighbors for this route and expiration time for the route table entry.

- **Agreement Problem:** Coordinator, in this scenario, distributes client's decision among replicas and then decides which operation is to be committed or aborted, preferably, without blocking any node. Moreover, it is also responsible to detect faulty replicas and faulty communication channel, refer Figure 1.4.

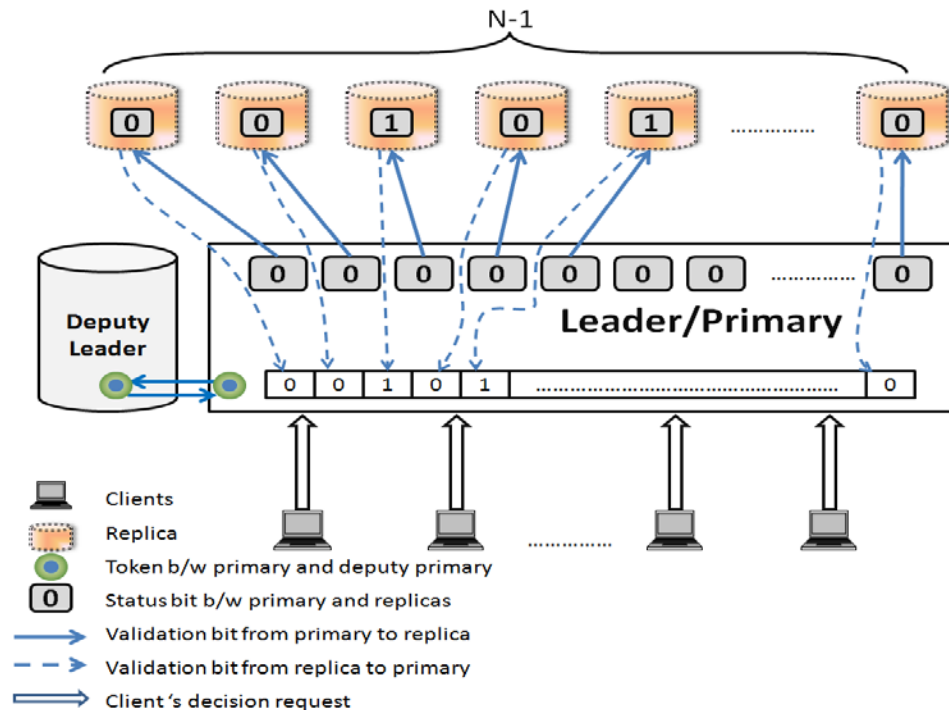


Figure 1.4: Agreement Problem. Leader replicates the decision to all its replicas. Deputy Leader behaves like vice-leader and token is used to make agreement proactive. Interested reader may refer [50] for more details.

1.2.3 General Properties of Leader Election Protocols:

In general, all nodes in the system may stay in one of the three states as normal, candidacy or leader. In addition, all nodes hold the ID of current leader in a variable $CODR_i$ and a boolean variable $CODRelect_i$ is used to show that node has decided its coordinator, when it is true. Coordinator election protocols for MANET choose a correct coordinator eventually. These kinds of protocols are known as *stable coordinator election* protocols. The stable coordinator election protocols must satisfy the following two properties:

- **Safety:** After the termination of coordinator election protocol, all N nodes in the system agree on the same chosen coordinator.

$$\forall i, j : 1 \leq i, j \leq n : (STATE_i = NORMAL \wedge STATE_j = NORMAL) \Rightarrow (CODR_i = CODR_j)$$

- **Liveness:** Liveness property represents that all mobile nodes start election process in candidacy state and eventually all node, but one, progress towards normal state in which all nodes connected to the system agree to the only coordinator.

$$\diamond (\forall i : CODRelect = TRUE)$$

1.2.4 Challenges towards Coordinator Election in MANET:

MANET inherits all the challenges of wireless domain with enhanced complications; thus, coordinator election in ad hoc environment is more sinister than in its infrastructured counterpart. The following factors add to difficulties in developing a leader election protocol in such environment: [41, 46, 47]

- **Dynamic System:** Election protocols developed for conventional distributed system, usually, depend upon the network topology; while in MANET, nodes are highly mobile, hence, the topology changes at a very fast rate. Intuitively, node mobility enforces frequent network reconfiguration which makes the network more susceptible to attacks.
- **Variation in Node Capabilities:** Each node has different battery power, computational power and memory stacks. Moreover, they may have varying software configuration, traffic load distribution, congestion control mechanism, transmission/receiving capabilities and bandwidths. The heterogeneity in node radio capabilities can result in possibly asymmetric links. Thus, designing network protocols for such heterogeneous network can be complex, requiring dynamic adaptation to the changing conditions.
- **Failure of Mobile Nodes:** Mobile nodes are relatively high error prone unlike static wired network nodes due to limited resources at nodes.
- **No Secure Medium:** The wireless network is more infringed by a wide range of passive and active attacks. Also, the security mechanisms, like secret keys, trusted authorities, are difficult to implement. Thus, the working of MANETs is often obstructed by the attacks, like man-in-middle, denial-of-services, which may lead to elect a false coordinator.
- **Limited Network Scalability:** At present, most of the static environment protocols have significant network scalability due to relatively small wireless components. However, MANET applications involve large wireless components with thousands of nodes. Hence, scalability is critical to the successful deployment of these networks.

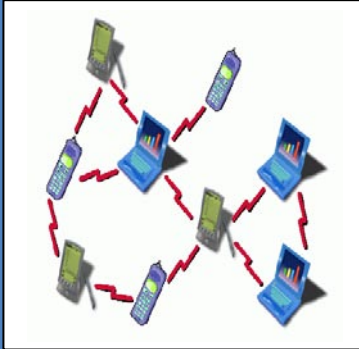
Although, these challenges do exist in cellular mobile systems, the coordinator election in MANET is difficult due to absence of some static node like base station (BS) and mobile switching center (MSC) in cellular systems.

1.3 ORGANIZATION OF DISSERTATION

The rest of this dissertation is structured as follows. *Chapter 2* provides a brief background of leader election in static as well as in dynamic environment and, finally, discusses motivation behind this novel leader election protocol development. In *Chapter 3*, we describe our first leader election protocol, MELFA and its fault tolerant version, ELFA, is described in *Chapter 4*. *Chapter 5* presents our novel highly available leader election protocol, DELFA. Simulated results are given in *Chapter 6*. Finally, we summarize our dissertation and propose future directions in *Chapter 7*.

CHAPTER 2

BACKGROUND AND OBJECTIVE



2.1 STATIC SCENARIO

- Ring Based Protocol
- Non-Ring Based Protocol
- Other Protocols

2.2 DYNAMIC SCENARIO

- Routing Protocol Based Protocols
- Miscellaneous Protocols
- Extrema Finding Protocols

2.3 OBJECTIVE OF DISSERTATION

“Most of what we call management consists of making it difficult for people to get their jobs done”

- Peter Drucker

In contemporary literature, a rich collection of leader election protocols are already existing for static as well as for dynamic environment. In this chapter, firstly, we provide a critical survey of some of these protocols in both static and dynamic domain. Finally, we conclude the chapter by highlighting the need for further research in leader election for MANETs.

2.1 STATIC SCENARIO

In this section, we consider some ascent protocols of static distributed environment. Here, I put them into three major categories as ring based [1, 2, 3, 4, 5], non-ring based [6, 7, 8] and other [9, 10, 11, 12, 13, 14] protocols. We will discuss some of notable protocol of these families in subsequent part.

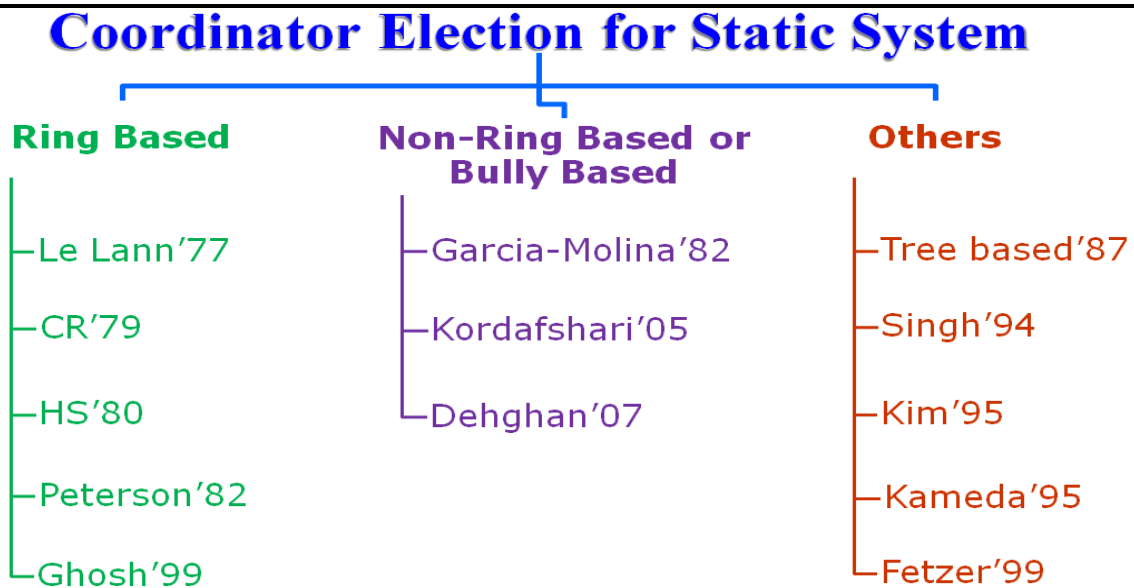


Figure 2.1: Classification of Coordinator Election Protocol for Static System.

2.1.1 Ring Based Protocol:

The first ever leader election protocol elect a coordinator in a unidirectional logical ring of nodes. Later on, several protocols are developed for ring network. Here, we will discuss some most popular protocols among them.

LeLann Protocol: The first leader election protocol for unidirectional rings was proposed by G. LeLann [1]. In the protocol, all nodes are arranged in a clockwise ring and know the size of ring with FIFO channel between two neighbor nodes. Any node, not central coordinator, can initiate the election by placing its identifier in an election message and forwarding it to its neighbor clockwise. Node priority is obtained by organizing nodes into a (logical) ring. The basic idea of the Le Lann algorithm is that each node sends a message around the ring bearing its identity. Node with the highest priority should be elected as coordinator. The algorithm requires a total $O(n^2)$ message, where n is the node of nodes in the ring.

Chang Robert Protocol: In 1979, Chang [2] provided the solution for unidirectional ring to reduce the large message complexity, *i.e.*, $O(n^2)$, to $O(n \log n)$. It is also noting point in CR protocol that all nodes know the

size of ring and it elects maximum ID node as leader. In this protocol, each node sends election messages piggybacked with its ID to its successor node in ring. On the reception of election message, node compares piggybacked ID with its own ID. If its own ID is lower, the node passes the election message to its successor; and if its own number is higher, the node neglect the incoming election message and send new election message piggybacked with its own ID; and if equal both IDs are equal then also node follows the previous case.

Hirschberg-Sinclair Protocol: HS [3] protocol is also comparison-based algorithm in fixed size of the ring where node with the maximum ID is elected. It is slimly different from CR protocol in the sense ring size is unknown to all nodes and implementation of bi-directional ring. In the protocol, each node operates in the phases. In each phase k , node i sends election message piggybacked with its ID in both directions to travel distance 2^k and return back to it. If both messages return then node i continues in phase $k + 1$. When a node receives an outgoing ID, it compares this one with its own and takes decision as – if the received ID is smaller, then it discards it; if the received UID is greater then it passes it to the next node, if it is not the end of its path; it returns it back to the previous one, otherwise; and if it is equal, then the node declares itself the leader.

Peterson Protocol: Peterson protocol [4] is designed over the HS protocol and it is simpler than HS protocol, also requires $O(n \log n)$ messages in clockwise unidirectional ring. In this protocol, all nodes are divided into two categories: active and relay. All nodes are initially active. Active nodes operate in phases. Relay nodes pass any messages only that they receive. By the setting of active and relay nodes, protocol diminishes total number of active nodes in each phase. The protocol initiates with its own value as its temporary identifier (t_{id}). During a phase, each active node receives the t_{id} of its nearest two active neighbors to the left. If the first node sees that its left active neighbor has the largest of the three t_{id} , then it sets its t_{id} to that value and starts the next phase; it becomes a relay node, otherwise.

2.1.2 Non-Ring Based Protocol:

Ring network provide a leader in efficient manner, however, accent Bully algorithm is keeping attraction of researcher till date. Now, we present the Bully algorithm and then some recent protocols which are based on Bully protocols.

Bully Algorithm: Bully algorithm [6] was presented by G. Molina in 1982. Suppose that the node N_i realizes the coordinator got crashed, it initiates an election algorithm and sends an election message to all higher ID nodes N_j , with respect to its ID. If no one responses, then N_i wins the election and becomes a coordinator. When a node N_j receives an election message from one of the nodes with lower ID, say N_k , it sends an OK message back to the sender nodes N_k to indicate that it is alive and will take over the election node. Finally, all nodes give up except one that is the new coordinator. The new coordinator sends coordinator message to all other nodes to notify them about its presence as new coordinator. Key advantage of Bully protocol is that only the node with higher ID will involved in election, eventually, instead of all the nodes. However, it exchanges high number of message, *i.e.*, $O(n^2)$.

Kordafshari Protocol: Modified Bully protocol [7] drop-offs the message complexity of Bully protocol to $O(n)$. When a node N_i detects that the coordinator has crashed, it initiates an election algorithm by sending election message to all higher ID nodes. Each node, that receives election messages, sends OK message with its ID to node N_i . If no node sends OK message to responses to node N_i , it will broadcast coordinator message to all nodes, declaring itself as a coordinator. On the other hand, if some node N_j sends OK message to node N_i , then node N_i will select the node with the highest ID N_k as coordinator and then sends to it the GRANT message. Reception of GRANT message at highest ID node N_k , initiates broadcast of coordinator message to all other nodes and informs itself as a coordinator.

Dehghan Protocol: In the protocol [8], when node N_i detects crash of leader, it sends an election message to all the nodes with higher ID. Each node sends its ID, in response of election message, to node N_i . If no node sends its ID to node N_i , it will broadcast coordinator message to all the nodes piggybacked with its ID. If some node response to node N_i , it will select the node with the highest ID as coordinator and broadcast the coordinator message to all the nodes. In this manner, all nodes know the new leader.

Now, the response message by highest ID may be lost destined to node N_i . In such scenario, node N_i select the second highest ID node N_j as leader number and broadcast the coordinator message to all the nodes, now the new leader sends election message to nodes with greater ID to sure that there is no node with greater ID. If a message is received from nodes with greater ID, it causes the greatest one as leader. Otherwise, it remains the leader again.

2.1.3 Other Protocols:

Singh Protocol: Singh protocol [10] is entirely different from all other protocol. In the protocol, leader is elected on the basis of some preference, not on the basis of node's ID. Authors suggested the need of this extrema finding approach as: (i) when a leader is elected on the basis of an ID number, it is chosen without regard to the preferences of the individual nodes or equivalency and performance level; (ii) all existing algorithms relies on the robustness and reliability of node.

In addition, several other protocols are described for leader election [11, 12, 13, 14], however, all of these cannot be implemented directly in MANETs.

2.2 DYNAMIC SCENARIO

The protocols for static environment cannot be useful in case of dynamic mobile scenario due to several existing challenges of MANET like dynamic system, variation in node capabilities, failure of mobile nodes, no secure medium and limited network scalability (Section 1.2.4). These are the some of the common reasons for development of new election protocols in mobile system. All these dynamic environs protocol can be classify in three broad categories as: routing protocol based [15, 16, 17, 18, 19, 20], best node or extrema finding based [21, 22, 23, 24, 25], and miscellaneous protocols [31, 32, 33, 34, 35].

3.2.1 Routing Protocol Based Protocols:

TORA (Temporary Ordered Routing Algorithm) [26], ZRP (Zone Routing Protocol) [27] and LEACH (Low Energy Adaptive Clustering Hierarchy) [28] are main routing protocol in aforementioned class. In this section, we first consider TORA, ZRP based leader election protocol and then LEACH based leader election protocol.

Coordinator Election for Dynamic System

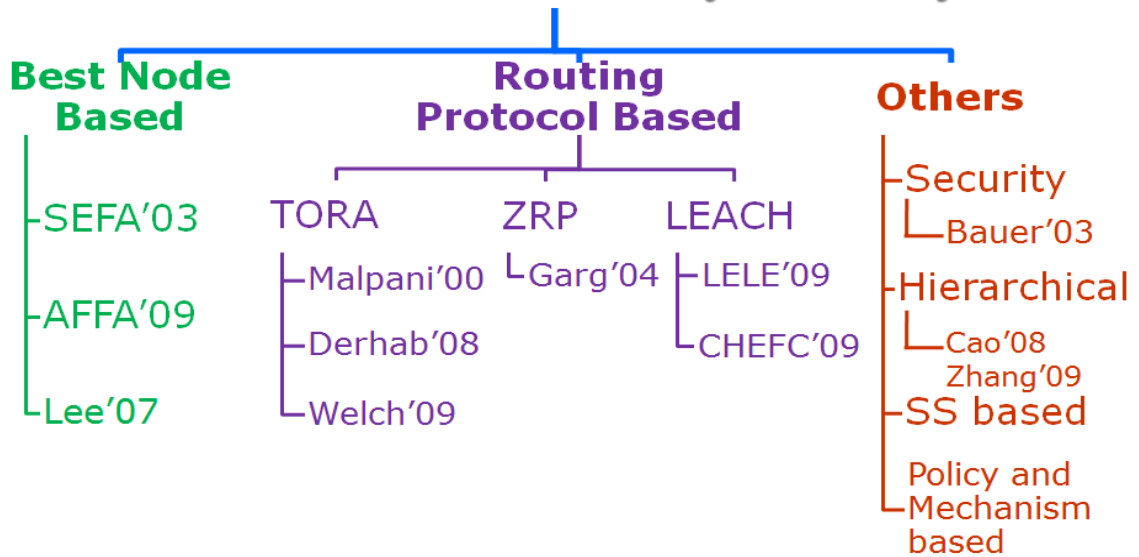


Figure 2.2: Classification of Coordinator Election Protocol for Dynamic System.

Malpani Protocol: The protocol, in [15], unbends the constraints in [29] and allows to move all mobile nodes in any space location instead bounding them in physical location. Malpani protocol is based on TORA [26] which in turn is based on partial link reversal [30]. In [30], height of any node is maintained in a triple of form (α_i, β_i, i) which is used to maintain the directed acyclic graph (DAG). These triple is used to maintain the DAG in any circumstances in the network. These triples are compared lexicographically to maintain the height. This triple calculation is also used in TORA with an extension of triple as 5-tuples $(\tau_i, oid_i, r_i, \delta_i, ID)$ to handle network partitioning as opposing [30]. Thus, in TORA, all nodes maintain height in form of 5-tuples. In 5-tuple, first three components are form a reference level and last two components are known as offset. A new reference level is generated by node i , if it losses it all outgoing link due to link failure and it assigns τ_i as the time when became the sink. oid_i is the originator identity, *i.e.*, i , in this case. r_i is the reflected reference level, initialized to 0, which is used to detect a partition in a network when having value 1. In addition, δ_i is an integer used to order nodes with respect to a common reference level and ID represent the identity of node i . Malpani also used the same 5-tuple with an addition of lid_i filed to height vector. Therefore, in this protocol, height of each node is maintained by following 6-tuple:

$$(lid_i, \tau_i, oid_i, r_i, \delta_i, i)$$

Initially, reference level has the value $(-1, -1, -1)$. In this protocol, when a node i becomes the sink due to link failure, it transmits this information to all its neighbors in the form of updated 6-tuples, who in turn propagate

this information to their neighbors and so on. Eventually, all the nodes in the new component will become aware of the new reference level of node i . If node i in turn also receives the updated reference level, then it detects the network partition and declares itself as a new leader. When two or more components meet due to the formation of new links, the leader of the component whose ID is the smallest will eventually become the sole leader of the entire new component.

This protocol handle network partitioning and merging efficiently, however, it is unable to elect the best node as a leader. In addition, when multiple topology changes occur, the algorithm can fail.

Derhab Protocol: In article [16], authors proposed a self-stabilization based leader election protocol, which provides the leader for highly dynamic MANET, not for large MANETs. They used the concept of Malpani's protocol with an extension in a height of a node. According to authors, algorithm, in [15], would never terminate if reference level generations do not stop. Therefore, the execution of the partition detection algorithm is not bounded and nodes can be without a leader for undetermined time. Thus, they stop newer computations in favor of the oldest one by introducing time factor in the height tuple as following:

$$(Certain_i, Tc_i, lid_i, Tb_i, Te_i, \tau_i, oid_i, r_i, \delta_i, i)$$

However, the protocol is complex and uses large message types. In addition, the algorithm is not correct when communication is asynchronous and multiple link changes and network partitions occur during the leader election process [17].

Welch Protocol: In paper [17], Welch again provided a clever mechanism for leader election protocol based on TORA and his previous article [15]. In current protocol, they have also extended the height vector by introduction of time factor as following:

$$(\tau_i, oid_i, r_i, \delta_i, nlt_s_i, lid_i, i)$$

Where, nlt_s_i represents non-positive timestamp whose absolute value is the time when the current leader was elected and δ_i represents distance of node i from leader. In addition, they have used wave algorithm, initiated by newly elected leader when different leaders collide at a node, then leader with most recent timestamp is chosen leader and this new height is further propagated. When multiple link changes and network partitions occur during the leader election process, protocol works well, however, in the paper, author have not introduced any new concept, they just extend their previous work. In addition, they have simplified the Derhab protocol, nothing else.

Garg Protocol: Garg, in [18] uses Zone Routing Protocol (ZRP) [27] to elect a new leader with minimizing the number of message and time to elect a coordinator. It is based upon diameter of the network. In addition, authors impose a heavy assumption to accomplish the protocol as network definitely gets stabilized after a single change occurs during leader election process; thus, they allow only a single topological change, which is quiet impossible in MANETs. The algorithm executes in rounds proportional to diameter if network is stable,

while it executes more than diameter times if nodes are mobile. Correctness proof is also given in the paper; however, it is only for the synchronous case assuming only one topology change.

LELE Protocol: In [19, 20], LELE is designed over the LEACH [28] protocol for wireless sensor network. LEACH protocol divides all sensors into cluster under one cluster-head, and cluster-head is chosen based upon some probability function. It describes LELE protocol for coordinator election in sensor network based upon node's residual energy, hence increasing network lifetime. In LELE, coordinator is chosen by comparing the energy of, initially, one hop away nodes, then two hop and so on. Then compare all energy level with respect to hop-distance, and choose *GOOD* or *AVERAGE* coordinator.

2.2.2 Miscellaneous Protocols:

Distributed Council Election Protocol: The article, in [31] focus on electing a small number of representatives (council) out of a (possible large) group of anonymous candidates. The protocol elects a group of any size in a predefined range. In addition, tradeoff between expected number of rounds for election and the expected number of unicast messages needed is shown. However, paper is oriented towards random protocols.

Election in Quasi-Static Mesh Networks: In article [32], leader election algorithm for wireless quasi-static mesh network is provided, where some nodes designated as mesh routers are static and other nodes designated as mesh clients are mobile. It is based on the construction of a spanning tree that includes all the static wireless mesh routers. Furthermore, impressive message complexity analysis and time complexity is also given.

Raychoudhury Protocol: The protocol, in [33], discovers the coordinator out of the top K special nodes called 'valued' nodes. This protocol executes in three phases *election*, *diffusion computation* and *coordinator*. Initially, all nodes are *WHITE*. In election phase, every node select most valued node as *RED* among its one hop neighbors. After that, RED nodes initiate diffusion computation and also include WHITE nodes in the tree. After termination of this phase, all RED nodes are merged and eventually the highest valued RED node will get the complete weight information about the network and it sends coordinator message to top K RED valued node, which again propagate to all WHITE nodes.

Dagdeviren Protocol: In [34], MANET system is divided into 4 level hierarchy of nodes. At the lowest level (L_0), nodes interact with each other using AODV or any routing protocol; at the next level (L_1), the Merging Clustering Algorithm (MCA) is used to clusterize the nodes in the network with one cluster-head for each; at L_2 , Backbone Formation Algorithm (BFA) is used to create the logical structure of all cluster-heads (ring); and finally at the top layer (L_4), modified CR algorithm is executed to find the supreme coordinator of network with lowest identity.

Haddar Protocol [35]: Mobile Agent (MA) based environment is developed for MANET, and coordinator election protocol is simulated using VISIDIA. Although, the security has to be compromised up to certain extent, the mobile agent based protocols are beneficial, than message passing protocols, in terms of their energy efficiency. Being agent-based, the protocol is free from synchronization and does not need continuous

use of all computational resources. The computational resources on a site are engaged only when the mobile agent reaches there.

2.2.3 Extrema Finding Protocols:

SEFA and AEFA Protocol: Routing based and some other protocols do not consider best node as leader. Vasudevan [21] provided two algorithms for MANETs based on best node as leader, eventually. In the article, three protocols are suggested named SEFA, SPLEA which assume a synchronous distributed system and proceeds in the various rounds of election in a lock-step fashion and AEFA which work for asynchronous distributed system.

Secure Extrema Finding Algorithm (SEFA) assumes that all candidate nodes share a single, Common Evaluation Function (CEF) that returns the same value at any candidate node when applied to any node. It works in hop-by-hop manner *i.e.*, it chooses the leader firstly at one-hop, and then at two-hop and so on, and, finally, chooses the global leader of the system with the assist of seven phases of election. A candidate-node's identifier (ID), battery life, computational power or certified level-of-trust within the system may be criteria to elect a best node. In addition, this protocol provides clock synchronization to prevent the replay attack and cryptographic secure communication between nodes through private-public key.

Secure Preference-based Leader Election Algorithm (SPLEA): SPLEA handles different preferences of candidate nodes to elect a leader. For example, if topological distance is taken as a metric for determining a node's preference among candidate nodes, then each candidate node have a different value to elect a leader. Thus, individual utility functions at each elector-node determine the candidate node's preference for a given tentative leader node.

Asynchronous Extrema Finding Algorithm (AEFA): AEFA is designed for asynchronous distributed system which does not provide any security, however, it allows the topology to change during the process of election. In AEFA, each node has a value known as node weight which represent the criteria for electing best node. The protocol uses diffusion computation to elect leader and provides the facility to avoid multiple election at same time by every node using computation index. Authors also suggested that AEFA belongs to *weak* self-stabilization [36]. In addition, AEFA handles network partitioning and merging are handles efficiently.

Lee Protocol: The protocol, in [24], is based on AEFA and imposes a list of leaders on each node in order to reduce election phases. However, it provides limited fault tolerance. In the protocol, Lee implemented AEFA without any major enhancement except the introduction of a list of five leaders on each node, in decreasing order of node weight value, where the first node is considered as the active leader of the network. If the first one is crashed, the second one becomes the active leader and so on.

Boukerche Protocol: The article, in [25], presents a leader election protocol that works well under frequent network changes and node mobility. Authors also suggested that protocol succeeds in electing a unique leader in a connected cluster in a short period of time with few messages unlike AEFA. Also, authors repentant message complexity. However, dynamic results are not presented.

Zhang Protocol: Zhang [23] also provided the fault resistant version of leader election protocol by imposing a hierarchy on the nodes using SEFA with the help of assistant-leader *i.e.*, once the leader crashes, the assistant will take responsibility of leader as soon as possible. The protocol imposes two level hierarchy of nodes as high level and low level nodes in order to use SEFA. Therefore, it also suffers from large message complexity. Furthermore, all the nodes hold the node ID and cluster ID. High level nodes use clock synchronization which is itself a challenge in the MANETs and assume that having very less mobility. Only, lower level nodes are assumed to move during election process. Thus, authors have implemented SEFA in hierarchy; they do not provide any solution for frequent leader failure in the network. Moreover, the protocol must be initiated again if the vice-coordinator-in-charge (*i.e.*, current leader) crashes.

2.3 OBJECTIVE OF DISSERTATION

A handful of protocols exist for leader election in various environments. The popular dynamic scenarios are wireless network, sensor network and mobile ad hoc network. Despite of flurry of protocols, is there any room for new protocol in coordinator election?

Yes, it is. The MANET is an environment of less powerful and therefore failure prone nodes. Hence, the preferred class of protocols is one that elect comparatively better node as coordinator out of the lot of less powerful and failure prone nodes. Therefore, in this volume, we consider only the best node or extrema based leader election. In the previous section, we have discussed four protocols. Now, we outline some limitations of these protocols. In AEFA [21], the large number of control message exchanges, overburden the communication link due to broadcast nature of the protocol. Also, AEFA is not fault tolerant. Furthermore, in context of SEFA [21], author used clock synchronization, which is also not a preferred approach for MANETs. The protocol, in [24], may violate safety requirement due to the existence of multiple leaders simultaneously if multiple nodes do not receive heartbeat message due to some reason. Moreover, they implemented AEFA directly without addressing the cons of AEFA. The protocol also assumes the FIFO channel that is extremely difficult to warrant. In [23], authors have implemented only SEFA in a logical hierarchy without providing any solution for frequent leader failure in the network. Nevertheless, protocols, in [23, 24] are fault tolerant. The protocol, in [25], is different from AEFA is term of network partition handling; however, the contribution is marginal. Essentially, the authors have re-presented AEFA in some different words without any enhancement except network merge handling.

In summary, all these four protocols broadcast the control messages and provide constrained failure resiliency. Moreover, none of these protocols are suitable for large MANETs where higher availability of leader is utmost necessary.

Broadcast Storm Problem: Broadcast is the process of sending a message from one node to many nodes. It is a primal operation for ad hoc node communication like update of network information, route discovery, etc. Broadcast storm, also known as redundant rebroadcast [37], occurs when a node decides to rebroadcast a

message to its neighbors; however, majority of neighbors might have already received the same message. Thus, a significant part of transmission is redundant and therefore must be avoided as far as possible.

Therefore, the main objective of the dissertation is to design an election protocol which uses broadcast of algorithm message in worst case only; otherwise, it uses multicast and eventually unicast. Another objective is to have the protocols that ensure higher level of fault tolerance and can be implemented in large MANETs too. Primarily, we focus on the following problems of leader election in MANETs:

1. How to forward the algorithm messages in whole network without executing explicit neighbor discovery? In addition, the solutions to this problem should also be able to address the broadcast storm problem, which may require some assistance from routing layer.
2. How to handle the dynamism introduced due to frequent topology changes that may result in either loss of leader or may trigger the concurrent existence of multiple leaders? The solution to this problem adhere the fault tolerance in leader election and leads to a more eminent degree of failure resiliency.
3. How to design a scalable protocol that can be applied to large MANETs, where even the transient absence of a leader may not be tolerable? The solution to this problem contributes another higher level of fault tolerance.

CHAPTER 3

MESSAGE EFFICIENT LEADER ELECTION



3.1 SYSTEM MODEL

3.2 MESSAGES & DATA STRUCTURES

- Types of Messages
- Data Structures

3.3 THE MELFA PROTOCOL

- Solution to Broadcast Storm
- MELFA Explanation

3.4 CORRECTNESS PROOF

3.5 MESSAGE COMPLEXITY

- Best Case
- Worst Case
- Average Case

3.6 DISCUSSION

“If we knew what it was we were doing, it would not be called research, would it?”

- Albert Einstein

In Chapter 2, we have discussed the motivation behind development of new leader election protocols. In this chapter, we present our first novel leader election protocol, Message Efficient Leader Finding Algorithm (MELFA), which uses piggybacking technique to avoid broadcast storm problem. Chapter starts with system model and continues with message types and data structures for election problem. Then, we present MELFA and discuss the prevention of broadcast storm problem, in detail, without neighbor discovery. Finally, correctness proof and message complexity analysis of protocol extend the chapter.

3.1 SYSTEM MODEL

We represent a MANET comprising N independent mobile nodes in the form of undirected graph $G = (V, E)$, where set of vertices V correspond to the set of nodes and edges E between any two vertices represent that corresponding nodes are within transmission radii of each other, refer Figure 3.1.

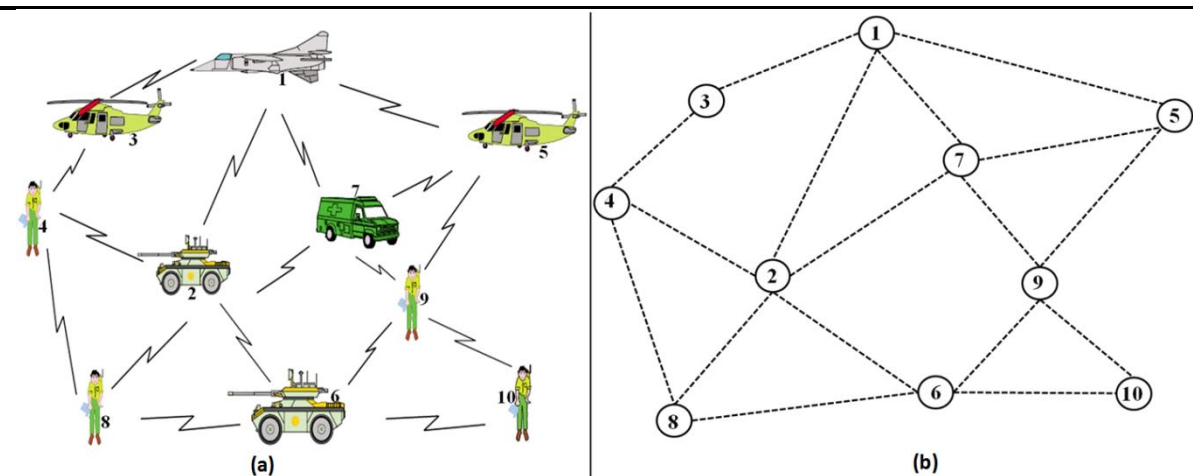


Figure 3.1: MANET in form of Graph. In Figure 3.1 (a) [48], a real mobile ad hoc network for rescue or war situation is established where each host has unique ID. In Figure 3.1 (b), corresponding MANET is shown in form of graph.

In addition, we have the following primary assumptions for election protocol:

- **Node Identity (ID):** The nodes have unique IDs. It is used to identify participants during the election process and to break the tie between nodes during election.
- **Communication Link:** Communication links are bidirectional, reliable not necessarily FIFO.
- **Neighbor Information:** Each node holds its neighbors information only. No node has information about whole network.
- **Node Communication:** Nodes communicate by passing messages over the wireless link. The neighbor nodes can directly communicate with each other.
- **Constrained Node Mobility:** High node mobility may result in arbitrary topology changes including network partitioning and merging. Hence, we allow limited node mobility during election.
- **Message Delivery:** A message delivery is guaranteed only when the sender and the receiver remain connected for the entire duration of message transfer.

- **Election Message (EM) and Coordinator Message (CM) Transmission:** Both messages are assumed to have priority over application messages.
- **Node Weight (NW):** Each node has a priority associated with it to become coordinator. The priority of a node may be based on its id, battery power, computing power, etc.
- **Unit cost of Acknowledgement Message (AM):** We consider that application messages are flowing regularly in the network, so piggybacking of AM with application message has unit cost.
- **No two isolated MANETs:** For successful execution of MELFA, we assume that there is only one MANET.

The following are secondary assumptions about MELFA:

- **Existence of Routing Protocol:** We assume that an underlying routing protocol exists to deliver messages between two nodes.
- **No Fixed Initiator:** Our algorithm does not assume the existence of some special node, called initiator.
- **Node Buffer:** Every node has sufficiently large buffer to store the EM and any further application messages until consumed.
- Higher NW (node weight) node with lower ID has highest priority.

3.2 MESSAGES & DATA STRUCTURES

In this section, different types of messages and data structures are presented which are base for election process, as following:

3.2.1 Types of Messages:

We use three type messages as – (i) Election Message (EM) for election process, (ii) Acknowledgement Message (AM) used in response of EM that includes node weight (NW) of sender and always destined to the originator *ori*, and (iii) Coordinator Message (CM) for announcement of coordinator.

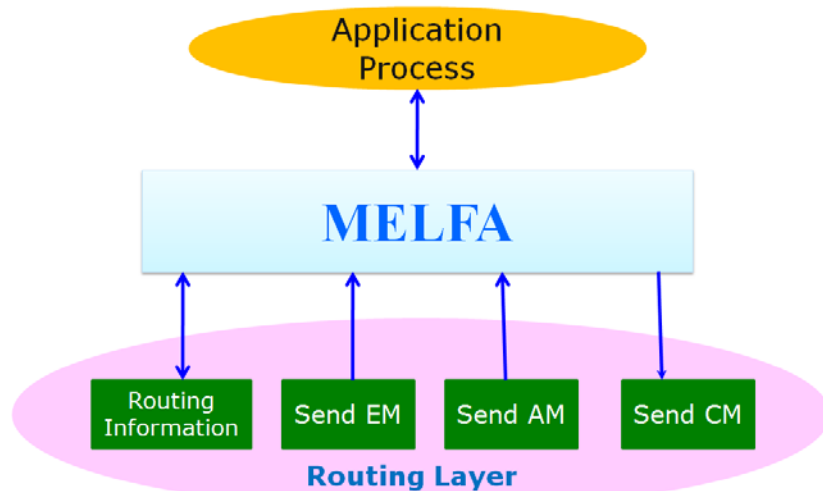


Figure 3.2: System Model. In this figure, different types of messages and their relation with MELFA is shown.

3.2.2 Data Structures:

There are four categories of data structures. We define the data structures maintained at each node. Further, we associate the data structures with *EM* and *CM*. The data structures are as follows:

- i. **At *ori* Node:** A node, called *ori* node, which initiates the election protocol either due to loss of its contact with the coordinator or the coordinator itself crashes. *ori* node has following data structures:
 - a. *receive_ack[n]*: an array of tuples representing AM received from node *j* and it is updated, by *ori* when *ori* receive AM from *j*, as *receive_ack[j] = j*. Initially, $\forall j, receive_ack[j] = -1$.
 - b. *receive_weight[n]*: an array of tuples representing NW of the node acknowledged by node *j* and it is updated, by *ori* when *ori* receives an AM from *j*, as *receive_weight[j] = NW_j*. Initially, $\forall j, receive_weight[j] = -1$.
 - c. **Time Out Value (*t_e*):** an interval timer, *ori* waits to collect AMs from all possible nodes.
- ii. **At All Nodes:** All nodes including *ori* in the system maintain following data structures:
 - a. **ID:** identity of the node.
 - b. *neighbor_name_i[j]*: an array of tuples representing neighbors of node *i*.
 - c. *NW_i*: represents the priority of node *i* to become coordinator.
 - d. *CODR_i*: represent the current coordinator of node *i*. Initially, *CODR_i = ϕ* , $\forall i$.
 - e. *CODRelect_i*: a boolean variable and it is set to TRUE if node *i* has elected its coordinator. Initially, *CODRelect_i = FALSE*, $\forall i$.
 - f. *STATE_i*: represent the current state of node *i*. The mobile nodes may be in one of the three states: NORMAL – node is in normal state if it is continuing normal computation in presence of the leader; CANDIDACY – node is in candidacy state either it lost its contact with the coordinator or the coordinator got crashed; LEADER – a node is in leader state if rest all nodes accept it as leader. Initially, *STATE_i = NORMAL*.
 - g. *election_send_i*: a boolean variable and it is set to TRUE if node *i* forwards EM to its neighbors. Initially, *election_send_i = FALSE*.
 - h. *send_for_i*: a variable that contains ID of originator of the EM received at node *i*. Initially, *send_for_i = ϕ* , $\forall i$.
 - i. *coordinator_send_i*: a boolean variable and it is set to TRUE if node *i* forwards CM to its neighbors. Initially, *coordinator_send_i = FALSE*.
- iii. **Piggybacked with EM:** EM piggybacks the following data structures that are used to forward EM.
 - a. *receive_election[n]*: an array of tuples representing the IDs of nodes that have already received EM. When node *i* sends/forwards EM to all its neighbors *k*, it updates *receive_election[k] = I*. Initially, $\forall j, receive_election[j] = -1$.
 - b. *ori*: represents the identity of the current initiator of protocol.
 - c. *sender_election*: represents the ID of the node that forwarded the most recent EM.
- iv. **Piggybacked with CM:** Like EM, CM also piggybacks the following data structures that are used to forward CM.

- a. $receive_coordinator[n]$: an array of tuples representing the IDs of nodes that have already received CM. When node i sends CM to all its neighbors k , it update $receive_coordinator[k] = 1$. Initially, $\forall j, receive_coordinator[j] = -1$.
- b. CID_{new} : represents ID of the newly elected coordinator.

3.3 THE MELFA PROTOCOL

Our three phase protocol, MELFA, has been designed for MANET; therefore, the nodes may move and/or crash during the election. The description of various phases and corresponding pseudo codes are given below.

Phase 1 – Initiation: The coordinator periodically sends *heartbeat* message to all other nodes. A NORMAL node initiates the election if it does not receive the heartbeat message or it moves to a location that is beyond reach of coordinator and it becomes the candidate and, hence, termed as *ori* node (CANDIDACY). Now, the *ori* node broadcasts the EM to its neighbors and set the timer t_e . Before broadcasting the EM, *ori* updates $receive_election[]$, $sender_election$, and sets $election_send$ as TRUE.

Phase 2 – Concurrent EM Receive: In this phase, node j would receive the EM from *ori* or from some other node i . On receiving EM, node j switches to CANDIDACY state. Afterwards, node j takes either of the following actions depending on situation.

Case 1: If node j has already forwarded an EM (i.e., $sender_election_j = TRUE$), then it discards all subsequent EMs that are received from the *ori* nodes that have ID higher than the most recently forwarded EM's originator ID by comparing it from $send_for_j$; otherwise, node j broadcasts EM to all its neighbors and sends AM.

Case 2: If $sender_election_j = FALSE$ and node j receives more than one EMs concurrently, then it forwards the EM, which have lowest *ori*, to the nodes that have yet not received EM. In addition, it discards other EMs.

Procedure A – Coordinator Election at *ori*: This procedure executes at *ori* with the start of *Phase 1* and outputs coordinator after timer t_e expires. If timer t_e has expired and for all j , $receive_election[j]$ is still -1, *Phase 1* is initiated again. Now, node *ori* declares highest NW node as coordinator and switches to the state NORMAL or COORDINATOR, as the case may be, and broadcasts CM, like EM broadcasting, and updates relevant data structures.

Phase 3 – CM Distribution: This is the last phase of MELFA. If node j has yet not succeeded in electing coordinator and receives CM, it accepts the newly elected node as leader and becomes NORMAL or COORDINATOR as the case may be.

Pseudo Code: Phase 1 – Initiation

```

IF (STATEi = NORMAL AND CODRi = ∅) THEN
  ori ← i;
  STATEi ← CANDIDACY;
  sender_election ← ori;
  election_sendori ← TRUE;
  coordinator_sendori ← FALSE;
  codr_electori ← FALSE;
  ∀ j: j ∈ neighbor_nameori;
    receive_election[j] ← 1;
  Set timer te;
  ∀ j: j ∈ neighbor_nameori;
    Broadcast EM to j;

```

Pseudo Code: Phase 2 – Concurrent EM receive

```

Node j receive EM form i and x
STATEj ← CANDIDACY;
coordinator_sendj ← FALSE;
CODRelectori ← FALSE;
IF (election_sendj == TRUE AND send_forj > EM.orii AND EM.orii < EM.orix) THEN
  Discard EM.orix // EM.orix represents ori of the EM //
  Send AM
  ∀ k: k ∈ neighbor_namej;
    Broadcast EM.orii to all k;
ELSE IF (election_sendj == FALSE AND EM.orii < EM.orix) THEN
  Discard EM.orix
  Send AM;
  IF (∀ k: k ∈ neighbor_namej, receive_election[k] == 1) THEN
    Do nothing
  ELSE IF (∃ k: k ∈ neighbor_namej, receive_election[k] == -1) THEN
    Send AM;
    election_sendj ← TRUE;
    send_forj ← ori;
    receive_election[k] ← 1;
    sender_election ← j;
    ∃ k: k ∈ neighbor_namej, receive_election[k] == -1
    Forward EM to k;

```

Pseudo Code: Procedure A – Coordinator Election at ori

```

IF (receive_ack[j] == -1 AND te == TRUE) THEN
  reinitiate Phase 1 (4 times)
ELSE
  ori receive AM from j;
  receive_weight[] = receive_weight[] ∪ NWj;
  receive_ack[] = receive_ack[j] ∪ j;
  IF (te == TRUE) THEN
    CID ← max_weight_id (receive_weight[]);
    CODRori ← CID;
    CODRelectori ← TRUE;
    IF (CID ≠ oriid) THEN
      STATEori ← NORMAL;
    ELSE
      STATEori ← COORDINATOR;
      ∀ k: k ∈ neighbor_nameori;
        receive_coordinator[k] ← 1;
      coordinator_sendori ← TRUE;

```

```

election_send_ori ← FALSE;
∀ k: k ∈ neighbor_name_ori;
Broadcast CM to k;
    
```

Pseudo Code: Phase 3 – CM Distribution

```

Node j receive CM from i
IF (CODRelect_j == FALSE) THEN
    CODR_j ← CID_new;
    CODRelect_j ← TRUE;
    election_send_j ← FALSE;
    IF (CID_new ≠ j) THEN
        STATE_j ← NORMAL;
    ELSE
        STATE_j ← COORDINATOR;
    IF (∀ k: k ∈ neighbor_name_j, receive_coordinator[k] == 1) THEN
        coordinator_send_j ← TRUE;
    ELSE IF (∃ k: k ∈ neighbor_name_j, receive_coordinator[k] == -1)
        coordinator_send_j ← TRUE;
        receive_coordinator[k] ← 1;
        ∃ k: k ∈ neighbor_name_j, receive_coordinator[k] == -1
        Forward CM to k;
    
```

3.3.1 Solution to Broadcast Storm:

As stated in Section 2.3, broadcast storm generate lots of messages in the network. In MELFA protocol, we adapt an entirely new concept, piggybacking some information, which avoids the broadcast storm without prior knowledge of all nodes, *i.e.*, neighbor discovery. According to *Phase 1, ori* node, say node 1, broadcast election message (EM) to all its neighbors piggybacked with *receive_election[n]*. Whenever, any neighbor node of *ori*, say node *j*, receives EM, it checks *receive_election[n]* and its neighbors; then, it sends EM to only neighbors which have not received EM already, indicated by -1 value in *receive_election[n]*.

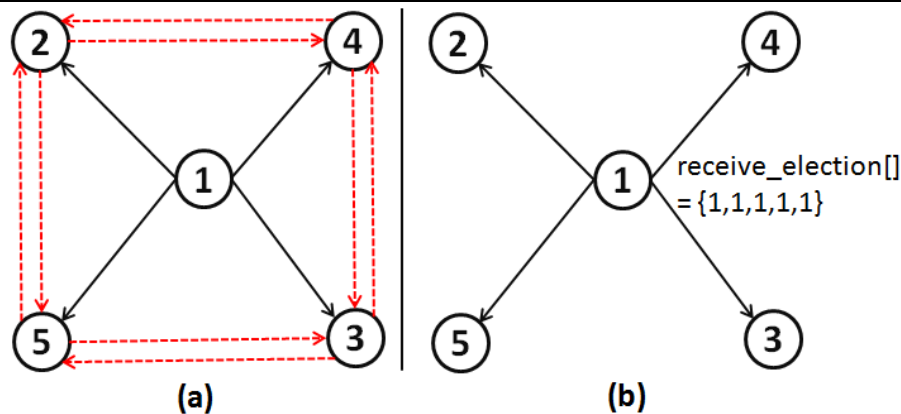


Figure 3.3: Solution to Broadcast Storm Problem. In Figure 3.3 (a), broadcast storm problem is shown where node1 sends some application message to all its neighbor node 2, 3, 4 and 5, eventually, node 2 broadcast it to its neighbor nodes 4 and 5 and similar execution by node 3, 4 and 5. The overall process generates 8 redundant messages, shown by red color. On the other hand, in Figure 3.3 (b), node 1 sends EM piggybacked with *receive_election[n]* that assists in avoidance of eight redundant messages.

In this way, our protocol avoids the broadcast storm problem with taking help from routing layer. As we know that, routing layer causes more dependency of election (or application) protocol on underline routing

algorithm, hence, avoidance of this dependency makes our protocol less dependent on routing layer, unlike most contemporary election protocols.

3.3.2 MELFA Explanation:

The example in Figure 3.4 shows a sample execution of MELFA. The neighbors assumed for respective nodes are given in Table 3.1. Consider the MANET in Figure 3.4 (a). We initiate MELFA at node 0. Hence, node 0 becomes *ori*. Subsequently, *ori* sends EM, piggybacked with *receive_election[]* = {1, -1, 1, -1, 1, -1, 1, -1, -1, -1, -1}, to its neighbors 4, 6 and 2. On receiving EM, nodes send AM. On receiving AM, *ori* updates *receive_ack[]* and *receive_weight[]*, as shown in Figure 3.4 (a). In Figure 3.4 (b), node 4, 6, 2 forward the EM to their neighbors, and sets *election_send* and unsets *coordinator_send*. Node 4 checks the entries, corresponding to its neighbors, in *receive_election[]* and forwards the EM to neighbor(s) that have -1 in *receive_election[]* and updates *receive_election[]*. In this example, node 4 sends EM only to 1, 5, 9, 8 and excludes node 0 and node 6. Similarly, node 6 forwards EM only to node 1, 8 excluding node 0, 2, 4 and node 2 sends EM only to node 3. Also, node 1, 5, 3, 8, 9 send AM and *ori* updates *receive_ack[]* and *receive_weight[]*.

In Figure 3.4 (c), node 1, 5, 3, 8, 9 begin forwarding of the EM after checking *receive_election[]*. Hence, node 1 does not forward EM; node 5 and node 9 forward EM to node 7; node 8 forwards EM to node 7 and node 3; node 3 forwards EM to node 8 only. In the mean time, node 7 sends AM, and *ori* updates *receive_ack[]* and *receive_weight[]*. After timer t_e expires, *ori* declares highest NW node as coordinator of acknowledged nodes. Subsequently, *ori* broadcast the CM, piggybacked with updated *receive_coordinator[]*, to its neighbor and unsets *election_send* and sets *coordinator_send*, as shown in Figure 3.4 (d).

In Figure 3.4 (e), neighbors of *ori* i.e., node 2, 4, 6 forward CM, as similar to Figure 3.4 (b). Finally, node 5, 8, 3, 9 forwards CM, refer Figure 3.4 (f).

Table 3.1. Initial Node's Neighbors for MELFA

Nodes	Neighbor Nodes							
0	2	4	6					
1	4	6	8					
2	0	3	6					
3	2	8						
4	0	1	5	6	8	9		
5	4	7	8	9				
6	0	1	2	4	8			
7	5	8	9					
8	1	3	4	5	6	7	9	
9	4	5	7	8				

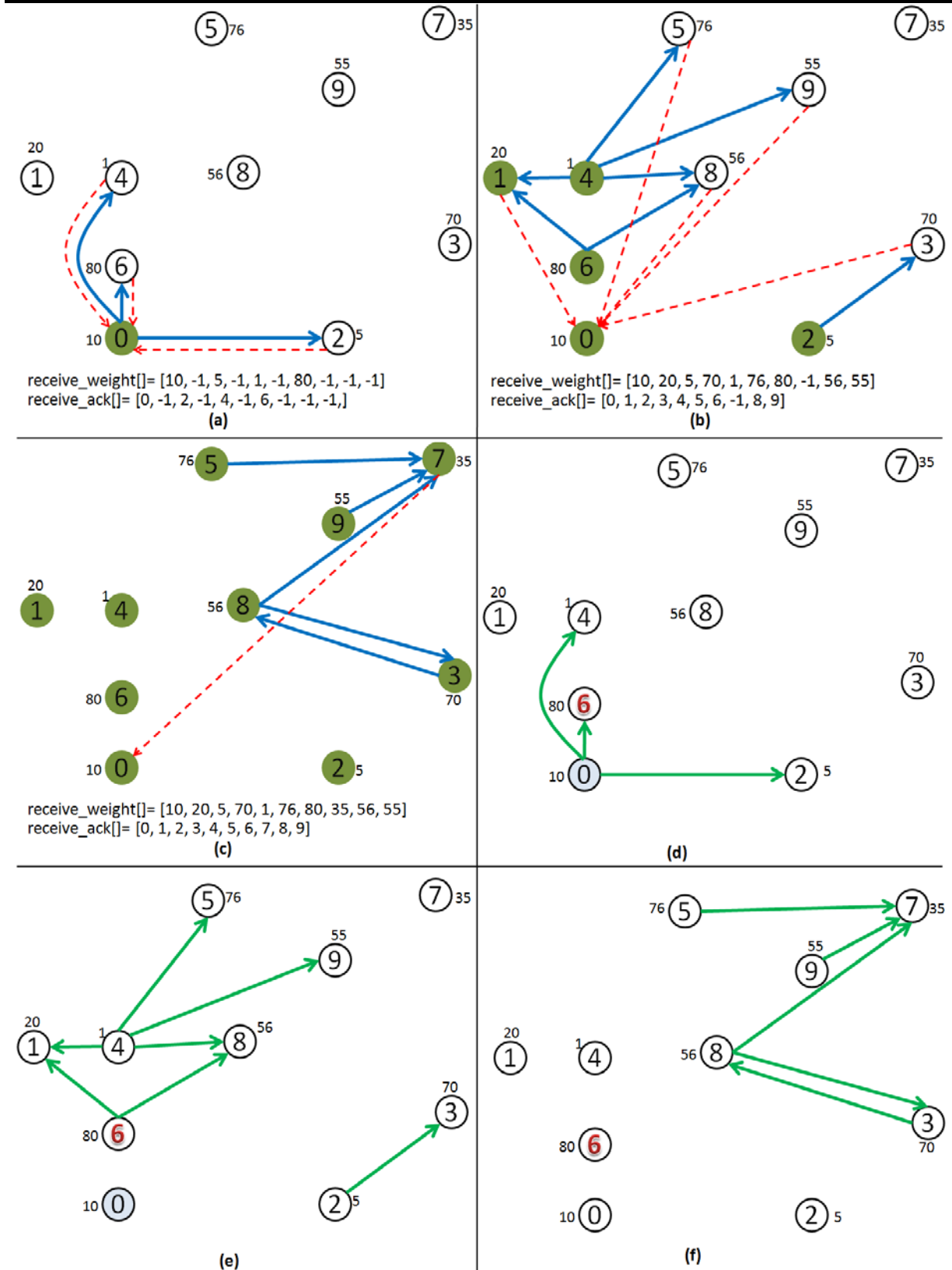


Figure 3.4: MELFA in Execution. Blue lines shows, red dashed lines shows AM and green lines represents CM.

3.4 CORRECTNESS PROOF

As stated in Section 1.2.3, the coordinator election protocols are required to choose a correct coordinator within finite time. MELFA also falls in this category and satisfies safety and liveness. In this section, we proof these two prosperities as following:

- **Safety:** After the termination of coordinator election protocol, all $N-1$ nodes in the system agree on the same chosen coordinator.

$$\forall i, j : 1 \leq i, j \leq n : (\text{STATE}_i = \text{NORMAL} \wedge \text{STATE}_j = \text{NORMAL}) \Rightarrow (\text{CODR}_i = \text{CODR}_j)$$

Proof: Assume the contrary. There are two nodes i and j whose state is NORMAL with highest node weight and there CODR value is different.

$$(\text{STATE}_i = \text{NORMAL} \wedge \text{STATE}_j = \text{NORMAL}) \wedge (\text{CODR}_i = i \wedge \text{CODR}_j = j) \wedge (i \neq j)$$

This statement holds in either of the following two cases:

Case 1: Node i and j are isolated from the network and no other node exist in there transmission range, refer Figure 3.5. In this case, both node i and j choose them as highest priority (NW) node and become coordinator of itself, not for the whole network. It contradicts our assumption that there exists single MANET.



Figure 3.5: Two Isolated Nodes.

Case 2: Node i and j lie in two MANETs and these MANETs have no common node, so node i and j become the coordinator of their respective MANET, refer Figure 3.6. However, it is in contradiction to our assumption that there exists single MANET.

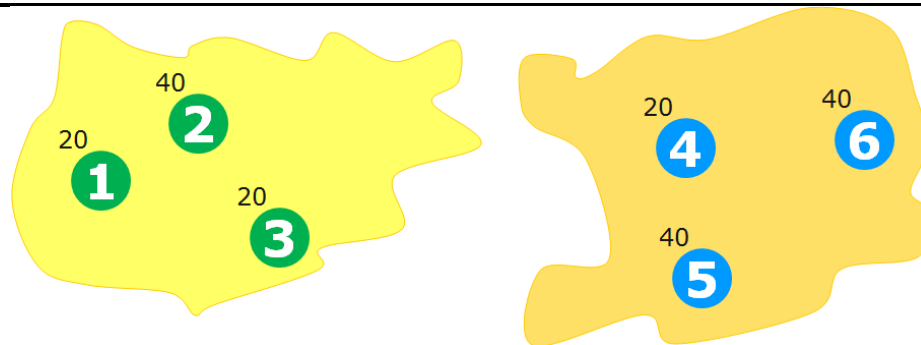


Figure 3.6: Two Isolated MANETs.

Hence, nodes i and j must belong to single MANET and the procedure, *Coordinator Election at ori* of *Phase 1*, chooses the highest priority (NW) node as coordinator. Thus, either node i or j become coordinator of MANET. This is a contradiction.

- **Liveness:** Liveness property represents that all nodes start election process in candidacy state and eventually all node progress towards normal state in which all nodes connected to the system agree to the only coordinator.

$$\diamond (\forall i : \text{CODRelect} = \text{TRUE})$$

In order that this statement is to be true, either of following conditions must hold: (i) Node i losses its coordinator and require a new coordinator, (ii) The coordinator crashed.

In the first case, node i initiates the protocol and in second case, any node looking for coordinator initiates MELFA. Due to the time out mechanism, single MANET, and the number of nodes being finite, *ori* would receive the AM from correct nodes. Afterwards, it would find maximum NW node as coordinator. Therefore, eventually all nodes become aware of the coordinator ID.

Thus, MELFA satisfies safety and liveness properties.

3.5 MESSAGE COMPLEXITY

A number of protocols exist for leader election in MANET that ignores the wireless bandwidth constraints and always use message broadcast; hence, they overload the wireless network. The proposed protocol, MELFA, also uses EM broadcast in the worst case; nevertheless, the message complexity of MELFA is much better in considerable number of cases. Now, we consider them one by one.

3.5.1 Best Case:

In MELFA, in the best case, the minimum ID node always will become the *ori*, resulting in minimum number of messages (EM, AM and CM). Now, we explain it with the help of two example scenarios, as follows:

Case 1: Let us consider that, unique single node lies in transmission radii of each node. Hence, network look like a one-dimensional chain of mobile nodes connected by wireless links, refer Figure 3.7, and thus, all nodes forward EM or CM to its only neighbor.

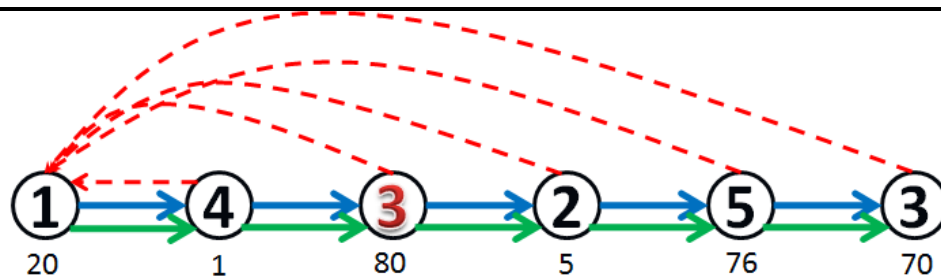


Figure 3.7: Best Case Scenario 1 of MELFA. One dimensional line of nodes.

Case 2: Let us assume that, all remaining $n-1$ nodes lay at 1-hop distance from *ori*, refer Figure 3.8. Therefore, *ori* sends EM and CM directly to all nodes in the network.

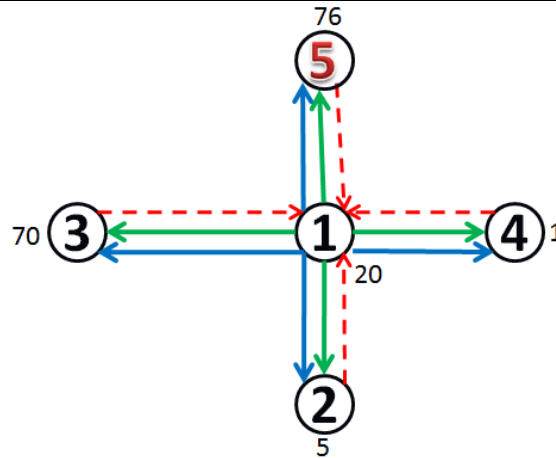


Figure 3.8: Best Case Scenario 2 of MELFA. All nodes are neighbors of each others.

In both the above cases, total $n-1$ EMs and equal number of AMs and CMs flow, leading to minimum number of messages flow in the network. Therefore, MELFA uses total $3(n-1)$ messages, in the best case.

3.5.2 Worst Case:

In worst case, each nodes broadcast EM to all its neighbors (*Phase 2, Case 1*). Nevertheless, it is worth noting that, in no case, broadcast of CM take place in MELFA. For better understanding, let us consider that, there are total ten nodes in the system and node 10 initiates the EM broadcasting. In the mean time, node 9 also broadcasts the EM, and so on. This case will execute recursively until *node 1* broadcasts the EM. Hence, worst case leads to $(n-1)(n-1)(n-1)\dots(n-1)$, i.e., approximately n^n EM broadcasts.

3.5.3 Average Case:

In the average case, MELFA executes like the way it is given in Figure 3.4. The execution can be interpreted as Diffusion like Computation (*DC*). Hence, the average case execution of MELFA generates lesser number of EM and CM messages due to multicasting and unicasting of EM and CM. Consider, there are total ten nodes in system, and, say, node 6 becomes the *ori*. In the mean time node 2 also becomes aware of leader crash, hence, it also acts as *ori*, refer Figure 3.9.

This type of concurrent initiation leads to DC tree formation for EM in the network. However, the message complexity would be more than $n \log_2 n$ messages as we do not assume formation of binary-DC tree and we have allowed concurrent initiation too. Nevertheless, CM message forwarding always use *Phase 3* and AMs are piggybacked with the application messages; thus, our protocol produces lesser number of control messages. It is obvious from above discussion that, the average case message complexity of MELFA would lie between two extremities, i.e., n^n and $n \log_2 n$. More implicitly, the reduction in message complexity would become even more explicit and significant when n become very large.

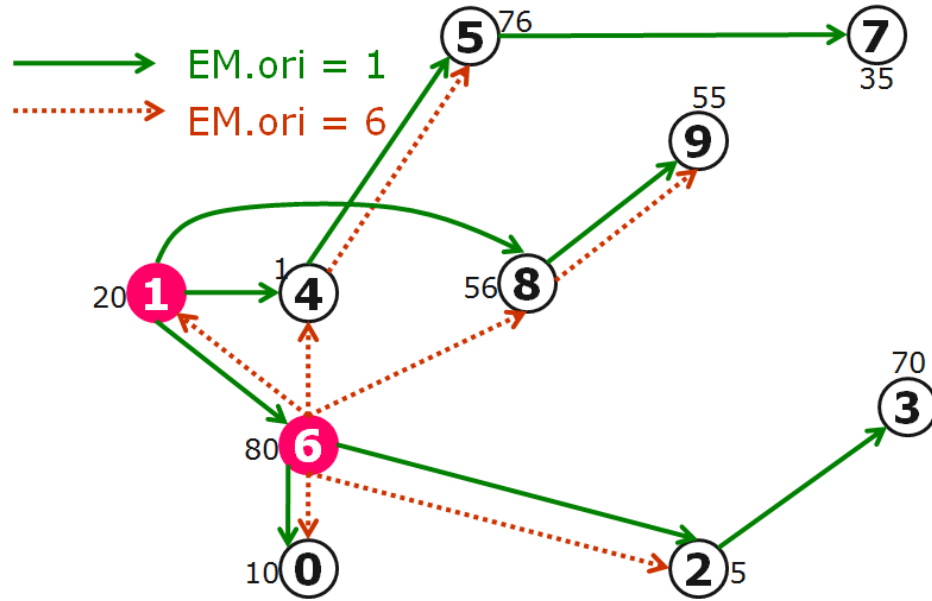


Figure 3.9: Average Case of MELFA.

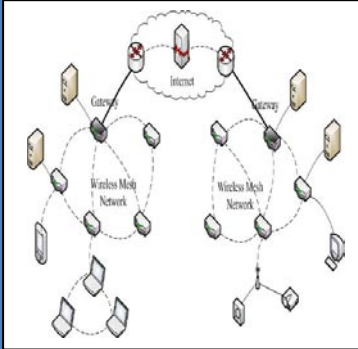
3.6 DISCUSSION

In this chapter, we have presented first ever a message efficient coordinator election approach which uses piggybacked data structure to avoid broadcast storm problem without considering prior neighbor discovery. The protocol uses multicast and unicast, unlike most of other contemporary protocols that always use message broadcast. Therefore, the proposed approach, having better message efficiency, reduces the overhead involved in electing leader for mobile ad hoc network.

CHAPTER 4

ELITE LEADER

ELECTION



4.1 SYSTEM MODEL

4.2 MESSAGES & DATA STRUCTURES

- Types of Messages
- Data Structures

4.3 THE ELFA PROTOCOL

- Revised Procedure and Augmented Phase of MELFA
- Special Events of ELFA
- Protocol Explanation

4.4 CORRECTNESS PROOF

4.5 MESSAGE COMPLEXITY

- Best Case
- Worst Case
- Average Case

4.6 DISCUSSION

“A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable”

- Leslie Lamport



ELFA communicates minimum number of control messages for election process; however, it is not fault-tolerant. In this chapter, we present our second leader election protocol, based on MELFA, for small ad hoc networks, which elects a vice-coordinator and a group of elite nodes as cabinet that provides a fault free leader election. Our protocol declares the new coordinator for the system as soon as the current leader crashes by maintaining the vice-coordinator and a cabinet of elite nodes, called CAGs (Coordinator Advisory Group members).

The nodes having better capabilities like battery backup, computation power, etc, have been called 'elite' nodes. However, elite nodes have not been assumed having any special characteristic that may turn the algorithm asymmetric, except they are supposed to maintain an additional data structure. On the occurrence of failure, new vice-leader is elected from the elite nodes. Therefore, the protocol has been named as Elite Leader Finding Algorithm (called ELFA, henceforth). Unlike other protocols, ELFA has three novel properties:

- i. Due to the dependence on MELFA, it does not use broadcast at every steps. Therefore, it uses less number of messages.
- ii. The CAGs concept, inspired from prevailing parliamentary polity in the world, brings in failure resiliency to our protocol, to the best of our knowledge this concept has never been used early in the present literature.
- iii. The possibility of network partitioning can never be fully avoided in MANETs. The ELFA can also handle network partitioning and merging efficiently.

Next, the chapter contains system model, message types, data structures, concept, correctness proof, and finally, message complexity of ELFA protocol.

4.1 SYSTEM MODEL

ELFA protocol assumes MANET in the form of undirected graph G . Furthermore, it also has the primary and secondary assumptions for election protocol like MELFA, Section 3.1. In addition, ELFA has one more primary assumption as following:

- **Unit cost of AM , IM , AIM , NM , WM , UM , LM , CAB :** We consider that application messages are flowing regularly in the network, so piggybacking of AM with application message has unit cost.

4.2 MESSAGES & DATA STRUCTURES

Successive section detects various messages and data structures for election process underline by ELFA, as following:

4.2.1 Types of Messages:

ELFA, likewise MELFA (Section 3.2.1), uses three basic control messages as – (i) Election Message (EM), (ii) Acknowledgement Message (AM), and (iii) Coordinator Message (CM). In addition, following six types of messages are used in ELFA:

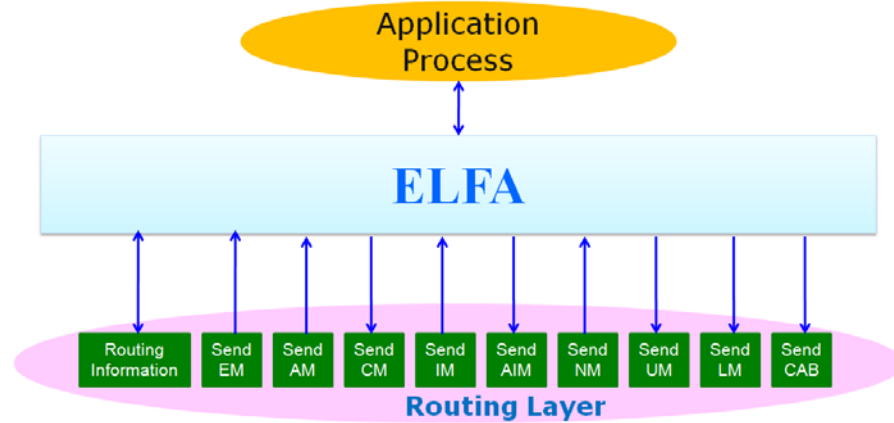


Figure 4.1: System Model. In this figure, different types of messages and their relation with ELFA is shown.

1. **Information Message (IM):** send by CAG i to the current coordinator.
2. **Acknowledge Information Message (AIM):** send by coordinator in response to IM to inform about its presence in the system.
3. **Notify Message (NM):** send by node i to inform about the ID of current coordinator to new joining node j in pre-established coordinated MANET. It is also used for handling network partition and merging (Figure4.4).
4. **Update Message (UM):** send by node i to provide the ID of new joining node j to the current coordinator in pre-established coordinated MANET.
5. **Leader Message (LM):** used to transfer the charge and responsibility of crashed leader to vice-coordinator by any CAG.
6. **CABinet message (CAB):** is send by vice-coordinator-in-charge to lowest identity CAG to initiate the MELFA within CAGs to elect next vice-coordinator.
7. **Wait Message (WM):** is send by waiting node i to another waiting node j to avoid unnecessary EM flow if node j does not have current leader ID.

4.2.2 Date Structures:

In ELFA, there are five categories of data structures. Likewise MELFA (Section 3.2.2), we define the data structures at each node with some more data structures and associate with EM and CM . In addition, we associate some data structure at CAGs as follows:

- i. **At ori Node:** A *ori* node has $receive_ack[n]$, $receive_weight[n]$ and Time Out Value (t_c) like MELFA. Details are given in Section 3.2.2.
- ii. **At All Nodes:** In ELFA, all nodes including *ori* and CAGs have ID, $neighbor_nam_{e_i}[j]$, NW_i , $CODR_i$, $CODRelect_i$, $election_send_i$, $send_for_i$ and $coordinator_send_i$. Further, all nodes have following additional data structures:
 - a. **STATE_i:** represent the current state of node i . The mobile nodes may be in one of the five states: NORMAL – node is in normal state if it is continuing normal computation in presence of the leader; CANDIDACY – node is in candidacy state either it lost its contact with the coordinator or

the coordinator got crashed; LEADER – a node is in leader state if rest all nodes accept it as leader; VC – the node is vice-coordinator (VC); ELITE – node is a CAG. Initially, $STATE_i = NORMAL$.

- b. CAG_status_i : a boolean variable and it is set to TRUE if node i is CAG. Initially, $CAG_status_i = FALSE$.
- c. $wait_set_i[w]$: an array of tuples which stores the ID of node to whom wait message has sent. Initially, $\forall w, wait_set_i[w] = \emptyset$.
- iii. **At CAGs**: Each Cabinet Advisory Group members keep the information about all the other CAGs using following data structure:
 - a. $CAG_set_i[n]$: an array of tuples representing the IDs all CAGs. Initially, $\forall n, CAG_set_i[n] = \emptyset$.
 - b. VID_i : represent the current vice-coordinator of node i . Initially, $VID_i = \phi, \forall i$.
- iv. **Piggybacked with EM**: Election Message has $receive_election[n]$, ori and $sender_election$ data structures.
- v. **Piggybacked with CM**: Coordinator Message has earlier $receive_coordinator[n]$ and CID_{new} with following extra data structures:
 - a. $VCID$: represents the ID of newly elected vice-coordinator.
 - b. $CAG[K]$: an array of tuples representing the IDs of coordinator advisory member nodes.

4.3 THE MELFA PROTOCOL

Although, ELFA uses, pre-established coordinated MANET, entirely new concept of cabinet, it runs MELFA, refer Section 3.3, within CAGs to decide new vice-coordinator among CAGs. It uses *Phase 1* and *Phase 2* of MELFA for distribution of EM. However, ELFA differs in the working of *Phase 3* and *Procedure* of MELFA, in order to create the cabinet and elect the vice-coordinator. Before stating the ELFA in detail, we introduce the concept of cabinet as the following:

- **CAG**: CAGs (K) is the set of higher NW node except leader. At the time of leader election by ori , it also elects the CAGs. Whenever leader crashes, CAG transfers the charge and responsibilities of crashed leader to vice-coordinator, and then, vice-coordinator-in-charge initiate the MELFA within CAGs to decide new vice-coordinator of the network.
- **Cabinet**: The set of CAGs form the cabinet like prevailing parliamentary system of governance.

Next, in this section, we first present modified procedure and augmented phase of MELFA and then some special event with complete protocol explanation.

4.3.1 Revised Procedure and Augmented Phase of MELFA:

Procedure B – Coordinator Election and Cabinet Formation at ori : This procedure is almost similar to Procedure A (Section 3.3), and executes at ori with the start of *Phase 1* (Section 3.3) and announces highest NW node as coordinator and K higher NW nodes as CAGs, after timer t_e expires, and node ori switches to the state NORMAL or COORDINATOR or ELITE or VC, as the case may be, and broadcasts CM. Soon after

that, the *ori* switches to any state except NORMAL and it also maintains the complete information about other CAGs received from CM.

Pseudo Code: Procedure B – Coordinator Election and Cabinet Formation at *ori*

```

IF (receive_ack[j] == -1 AND te == TRUE) THEN
    reinitiate Phase 1 (4 times)
ELSE
    ori receive AM from j;
    receive_weight[] = receive_weight[] ∪ NW;
    receive_ack[] = receive_ack[j] ∪ j;
    IF (te == TRUE) THEN
        CID ← max_weight_id (receive_weight[]);
        VCID ← (max-1)_weight_id (receive_weight[]);
        CAG[] ← max_K_weight_id (receive_weight[]);
        CODRori ← CID;
        CODRelectori ← TRUE;
        IF (CID ≠ oriid) THEN
            STATEori ← NORMAL;
        ELSE IF (VCID == oriid) THEN
            STATEori ← VC;
            CAG_statusori ← TRUE;
            CAG_setori[] ← CAG[];
        ELSE IF (oriid ∈ CAG[]) THEN
            STATEori ← ELITE;
            VIDori ← VCID;
            CAG_statusori ← TRUE;
            CAG_setori [] ← CAG[];
        ELSE
            STATEori ← COORDINATOR;
            CAG_setori[] ← CAG[];
    receive_election[n] ← -1;
    ∀ k: k ∈ neighbor_nameori;
        receive_coordinator[k] ← 1;
    coordinator_sendori ← TRUE;
    election_sendori ← FALSE;
    ∀ k: k ∈ neighbor_nameori;
        Broadcast CM to k;

```

Augmented Phase 3 – CM Distribution: *Augmented Phase 3* distributes the CM in the whole network. In addition, it also constructs the cabinet within the system in the presence of vice-coordinator and coordinator. Moreover, node *j* becomes NORMAL, COORDINATOR or ELITE, as the case may be, if node *j* has yet not succeeded in electing coordinator and receives CM in the mean time.

Pseudo Code: Augmented Phase 3 – CM Distribution

```

Node j receive CM from i
IF (CODRelectj == FALSE) THEN
    CODRj ← CIDnew;
    CODRelectj ← TRUE;
    election_sendj ← FALSE;
    IF (CIDnew ≠ j) THEN
        STATEj ← NORMAL;
    ELSE IF (VCID == orij) THEN
        STATEj ← VC;

```

```

        CAG_statusj ← TRUE;
        CAG_setj[] ← CAG[];
    ELSE IF (j ∈ CAG[]) THEN
        STATEj ← ELITE;
        VIDj ← VCID;
        CAG_statusj ← TRUE;
        CAG_setj[] ← CAG[];
    ELSE
        STATEj ← COORDINATOR;
        CAG_setj[] ← CAG[];
    IF (∀ k: k ∈ neighbor_namej, receive_coordinator[k] == 1) THEN
        coordinator_sendj ← TRUE;
    ELSE IF (∃ k: k ∈ neighbor_namej, receive_coordinator[k] == -1)
        coordinator_sendj ← TRUE;
        receive_coordinator [k] ← 1;
        ∃ k: k ∈ neighbor_namej, receive_coordinator[k] == -1
        Forward CM to k;

```

Afterwards, the termination of *Augmented Phase 3* marks the completion of both the tasks, *i.e.*, cabinet construction and election of coordinator as well as vice-coordinator. At this stage, each node, *CAG* and vice-leader becomes aware of the coordinator and each *CAG* also knows about other *CAGs*.

4.3.2 Special Events of ELFA:

As stated earliest, ELFA brings fault resiliency to leader election. *Procedure B* and *Augmented Phase 3* designates some nodes as leader, vice-leader and *CAGs*. Now, we explain events of ELFA which contributes fault tolerance, as following:

Event 1: New node sends EM to join the pre-established coordinated MANET. *Event 1* will occur when one or more nodes arrive within the transmission range of pre-established coordinated MANET, where every node know about the current coordinator and new joining node want to participate in that MANET. In this state, new nodes broadcast EM to their neighbors only (some modification in *Phase 1* of MELFA) and wait for the time out value t_e , reinitiate *Event 1* up-to four times similar to IEEE 802.11, otherwise.

Pseudo Code: Event 1

```

IF (STATEi = NORMAL AND CODRi = ∅) THEN
    Set timer  $t_e$ ;
    ∀ j: j ∈ neighbor_namei;
        Broadcast EM without piggybacked data structure to  $j$ ;

```

Event 2: Reception of EM in the presence of leader. *Event 2* occur when any node of pre-established coordinated MANET, (node or *CAGs* or leader), in the presence of existing coordinator, receives EM from any new joining node to join the system. Afterwards, the recipient, which may be node, *CAG* or leader, will forward NM; however, it sends UM also if recipient is not leader. Nevertheless, if recipient does not have CODR, it sends WM.

Pseudo Code: Event 2

```

Node  $j$  receive EM form  $i$  and  $x$ ;
IF (CODRj == ∅) THEN

```

```

Send WM to node  $i$  and node  $x$ ;
ELSE
  Sends NM to node  $i$  and node  $x$ ;
  Send UM;

```

Event 3: Reception of EM in the absence of leader. EM will be forwarded with some special mark within CAGs on the occurrence of *Event 3* that makes our protocol fault tolerant. Due to reasons like the loss of current leader or arrival of new node in the system, if CAGs or vice-leader receive the EM from any node via routing layer and CAG or vice-leader does not receive heart beat message from the leader, then they forward IM only twice. If the CAG receives the AIM within $t_{\max_propagate}$, then forward the NM; otherwise, transfer the charge and responsibility of crashed leader to vice-coordinator using LM. In addition, vice-coordinator-in-charge sends heartbeat message to all nodes and also sends CAB message to lowest identity CAG, which terminates by electing new vice-coordinator within CAGs.

Pseudo Code: Event 3

```

CAG  $i$  receives EM from node  $j$ ;
CAG  $i$  sends IM to coordinator;
Set  $t_{\max\_propagate}$ ;
IF ( $t_{\max\_propagate} = \text{TRUE}$  AND CAG  $i$  do not receive AIM) THEN
  Set  $t_{\max\_propagate}$ ;
  Send IM only one time;
  IF ( $t_{\max\_propagate} = \text{TRUE}$  AND CAG  $i$  still do not receive AIM) THEN
    CAG  $i$  sends LM to vice-coordinator;
    new leader sends heartbeat message to the CAGs;
    IF ( $\forall i: i \in N$ , receive the heartbeat message) THEN
       $\exists y: y \leq K :: \text{wait\_set}_k[p] \neq \emptyset$ ;
      Send NM to all  $p$ ;
    ELSE
      wait for heartbeat message;
       $\min \leftarrow \text{minimum}(\text{CAG\_set}_i[])$ ;
      new leader send CAB to  $\min$ ;
       $\min$  initiate Phase 1, Phase 2, Find Vice-Coordinator using Procedure B, and
      Distribute CM using Augmented Phase 3;
  ELSE
    Send NM to node  $j$ ;

```

Event 4: No Heartbeat Message from Vice-Coordinator-in-Charge. In response to LM, new leader sends heartbeat message. Thus, the absence of heartbeat message represents that either vice-coordinator-in-charge has crashed or there may be network partition (Figure 4.3). In order to overcome this situation, CAGs initiate the MELFA similar to explained in *Event 3*. Now, it may be possible that minimum ID CAG also got partitioned. After a reasonable timeout, if no new leader could be chosen from the CAGs then CAGs initiate MELFA again, recursively, within individual components to find new leader.

Pseudo Code: Event 4

```

CAG  $i$  do not receive heartbeat message from new leader;
CAG  $i$  detect network partition;
CAGs initiate MELFA using Event 3;

```

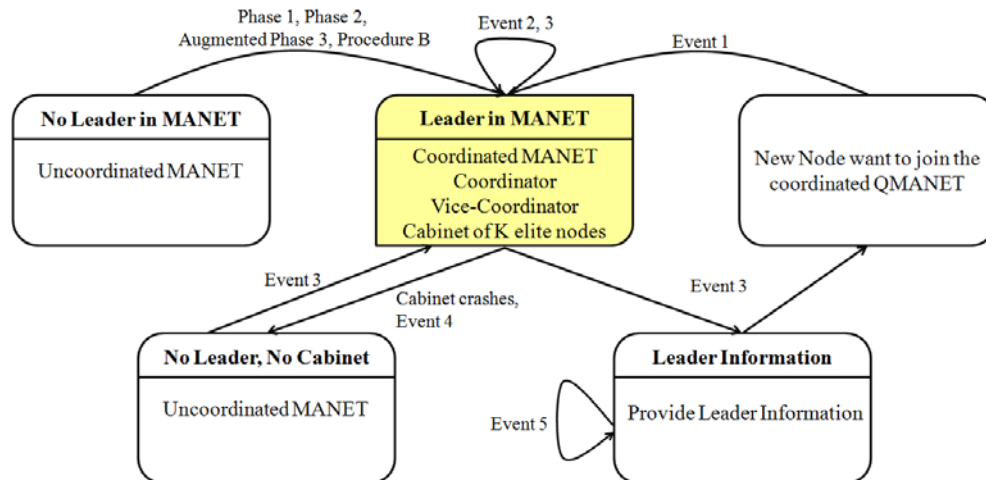


Figure 4.2: *ELFA Protocol State Diagram.* In the figure, relationship between events and phases of ELFA are shown.

Event 5: Network Merging. This problem is handled like [15]. *Event 5* will execute when two partitioned components meet due to the formation of a new link. The coordinator of one of the components which has the higher NW becomes the coordinator of whole component and lower ID is used for tie breaking, if needed. However, unlike [15], in our approach another component's leader surrenders its leadership and becomes the CAG (Figure 4.3). It may be further noted that *Event 5* elects two vice-coordinators. These vice-leaders become the new leader one by one whenever current leader crashes.

Pseudo Code: Event 5

```

Node  $i$  sends EM to its neighbors;
IF (node  $i$  joins two components) THEN
    Neighbor of component 1 and 2 send NM to  $i$ ;
    Node  $i$  select higher NW and lower ID node as its leader;
    Node  $i$  forward NM to lower NW and lower ID leader's component;
    Lower NW node accept higher NW node as leader and become CAG;
ELSE
    Event 2 or Event 3 will fire;

```

4.3.2 Protocol Explanation:

The example, in Figure 4.3, shows a sample execution of ELFA. The neighbors assumed for respective nodes are given in Table 4.1. In Figure 4.3 (a), the *Augmented Phase 3* outputs node 6 as the coordinator of system with node 5 as vice-coordinator and node 8, node 9 as CAGs. Node 10, which is new joining node in Figure 4.3 (b), sends EM (election message) to its neighbors in the presence of coordinator node 6 (*Event 1*). Figure 4.3 (c) represents the execution of *Event 2*, where node 3, 9, 7 send the identity of their leader node *i.e.*, node 6, using NM (notify message) to new joining node 10. In addition, node 3, 7, 9 also send UM (update message), *i.e.*, the information about node 10, to leader.

In Figure 4.3 (d), the leader of system got crashed, and a new node, say with ID 11, wants to join the system (*Event 2*). Now, node 11 will send EM to its neighbor node 7, 9, 10 (*Event 1*). However, the neighbors of node 11 *i.e.*, node 7, 9, 10 do not have the ID of current leader. Therefore, they send WM (wait message) to node 11

(*Event2*). Also, they send EM to their nearest CAGs or vice-leader with the help of routing layer (*Event 3*). In addition, node 5 which is vice-leader and node 3, 8 which are CAGs send IM (information message).

In Figure 4.3 (e), neither vice-leader nor CAGs receive the AIM (acknowledge information message) from leader. Thus, CAGs send LM (leader message) to vice-leader to become new leader of the system. Subsequently, new leader *i.e.*, vice-coordinator-in-charge node 5, sends CAB (cabinet message) to lowest ID CAG to elect next vice-leader within CAGs.

Node 3 sends EM to remaining CAGs with some special flag to distinguish it from MELFA *Phase 1*' EM and execute *Phase 2, Procedure A* and *Phase 3* CAGs, refer Figure 4.3 (f). Afterward, node 3 elects node 8 as new vice-coordinator of the system and in Figure 4.3 (g), it distributes CM within CAGs, and nodes having $wait_set[p] \neq \emptyset$, send NM. Here, in this example, node 3, 8, 7, 9, 10 send NM.

Table 4.1. Initial Node's Neighbors for ELFA

Nodes	Neighbor Nodes						
0	2	4	6				
1	4	6	8				
2	0	3	6				
3	2	8					
4	0	1	5	6	8	9	
5	4	7	8	9			
6	0	1	2	4	8		
7	5	8	9				
8	1	3	4	5	6	7	9
9	4	5	7	8			
10	3	7	9				
11	7	9	10				

4.4 CORRECTNESS PROOF

ELFA is *stable coordinator election* and satisfies safety, liveness, deadlock freedom. We prove all these properties with the help of predicate calculus which include boolean operator (\wedge , \Rightarrow , \neg), quantification operators (\forall , \exists), and temporal operator (always \square , eventually \diamond). To verify all these prove, we use following notations:

- N : Total number of nodes;
- $CODR_i$: ID of current coordinator at node i ;
- N_x : MANET component x ;
- $CODR(N_x)$: Current coordinator of MANET N_x ;
- $VC(N_x)$: Current vice-leader of MANET N_x ;
- $SEND_i(M, j)$: Node i sends message M to node j ;
- $RECV_i(M, j)$: Node i receives message M from node j .

LEMMA 1: *ELFA ensures that each component of the system has unique leader. Formally, if N_x and N_y are two components of MANET then,*

$$\forall N_x, N_y : (N_x \neq N_y) \Rightarrow (CODR(N_x) \neq CODR(N_y)) \quad (1)$$

Argument: Assume that the MANET has a leader, say node g (see Figure 4a). Now, say node g , crashed and network got partitioned. In this situation, CAGs c, e, i, k, m or vice-leader f send IM to investigate the presence of leader.

However, no one succeeded in receiving AIM and vice-leader f becomes the new leader of one component with CAG c as vice-leader and node e as CAG. Although, CAGs j, k, m send LM, they will not receive the heartbeat message in appropriate time due to network partition. Thus, they initiate MELFA with CAG which terminate with CAG m as leader and CAG k as vice-leader of another component, refer Figure 4.4 (b).

Soon after that, say node p comes in the range of both *i.e.*, leader f and node h that results in merging of two components, refer Figure 4.4 (c). Subsequently, node p sends EM to its neighbor node f and h , and both send NM as offer to join the system, refer Figure 4.4 (d). However, node p will always consider higher NW node as its leader; thus, node p accept node f as its leader, refer Figure 4.4 (d). Afterwards, node p forward NM to another component to accept higher NW node as leader; hence, node h accepts node f as it leader, refer Figure 4.4 (e) and forwards to all remaining nodes. Therefore, node i, j, k, l accept node f as its leader (see Figure 4.4f). In Figure 4.4 (g), node m also accepts node f as leader and becomes CAG and forwards NM to node n . ■

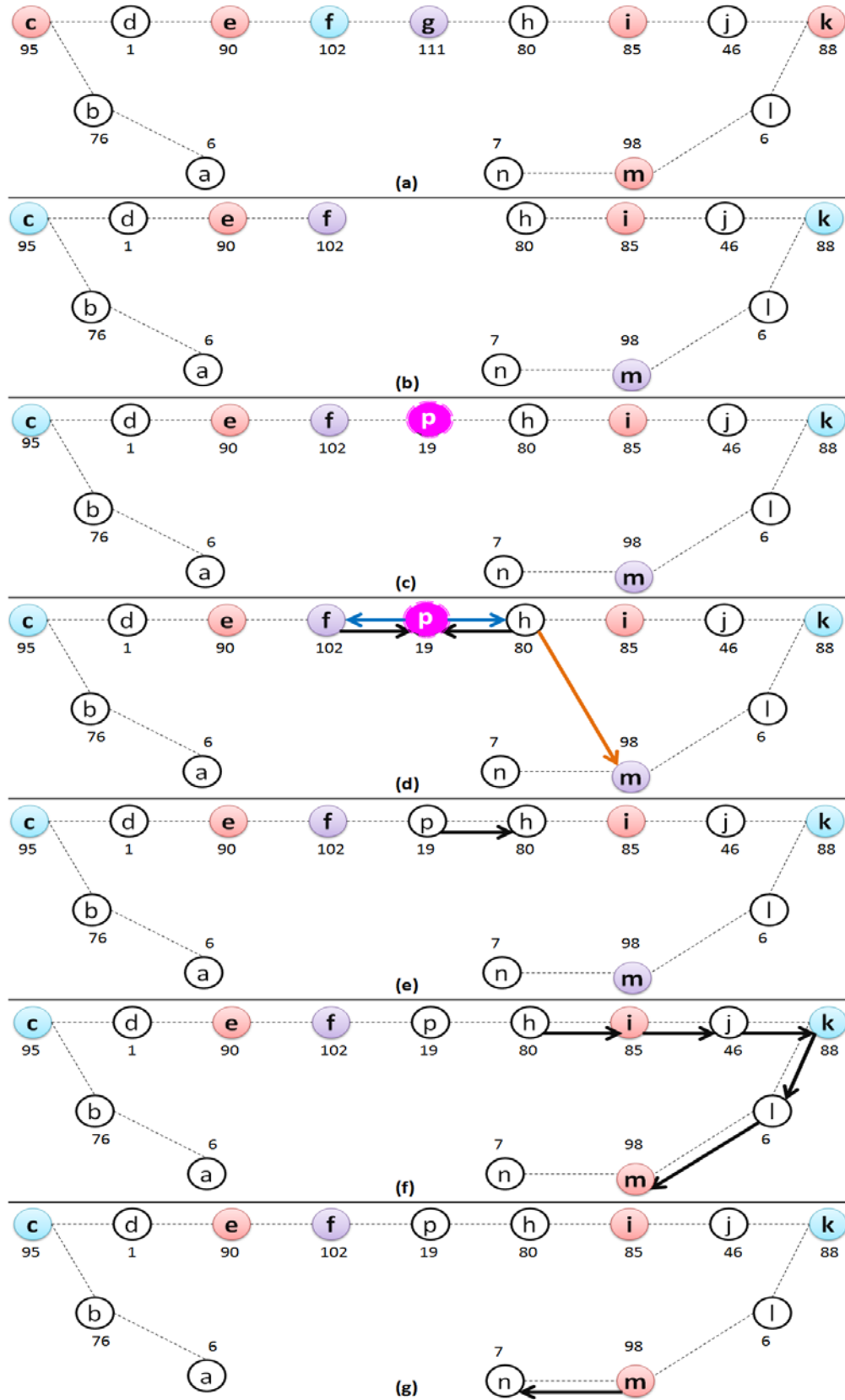


Figure 4.4: Network Partition and Merge Handling.

THEOREM 1: *ELFA guarantees safety. In other words, after the termination of coordinator election protocol, all $N-1$ nodes agree on the same chosen coordinator and there exist K CAGs in the system. Formally,*

$$\forall i, j, 1 \leq i, j \leq N : (\text{STATE}_i = \text{NORMAL} \wedge \text{STATE}_j = \text{NORMAL}) \Rightarrow (\text{CODR}_i = \text{CODR}_j) \quad (2)$$

Proof. Assume the contrary. There are two nodes i and j whose state is NORMAL with highest node weight and there CODR value is different. Formally,

$$(\text{STATE}_i = \text{NORMAL} \wedge \text{STATE}_j = \text{NORMAL}) \wedge (\text{CODR}_i = i \wedge \text{CODR}_j = j) \wedge (i \neq j) \quad (3)$$

This statement can hold in either of the following two cases:

Case 1: Node i and j are isolated from the network and no other node exist in their transmission range. Formally,

$$\forall N_x, \exists i, j \in N, i \neq j : [(i \notin N_x) \wedge (j \notin N_x)] \Rightarrow [(\text{CODR}_i = i) \wedge (\text{CODR}_j = j) \wedge (\text{CODRelect}_i = \text{TRUE}) \wedge (\text{CODRelect}_j = \text{TRUE})] \quad (4)$$

In this case, both node i and j choose them as highest priority (NW) node and become coordinator of itself only and not for the whole network. However, the formation of a link between node i and node j will force either node i or node j to lose its leadership and accept other as its leader. Formally,

$$\forall N_x, \exists i, j \in N : (i \wedge j \in N_x) \Rightarrow [\text{CODR}_i = \text{CODR}_j = \text{CODR}(N_x)] \quad (5)$$

It contradicts the assumption given in equation (3) and (4).

Case 2: Node i and j lie in two MANETs and these MANETs have no common node, so node i and j become the coordinator of their respective MANET. Formally,

$$\forall N_x, N_y, \exists i, j \in N, i \neq j : (i \in N_x) \wedge (j \in N_y) \wedge (N_x \neq N_y) \Rightarrow [(\text{CODR}_i = \text{CODR}(N_x)) \wedge (\text{CODR}_j = \text{CODR}(N_y)) \wedge (\text{CODRelect}_i = \text{CODRelect}_j = \text{TRUE}) \wedge (\text{CODR}_i \neq \text{CODR}_j)] \quad (6)$$

However, it can continue to hold till no link formation takes place between them. Formally,

$$\forall N_x, N_y, \exists i, j \in N, i \neq j : (i \in N_x) \wedge (j \in N_y) \wedge (N_x = N_y) \Rightarrow [\text{CODR}_i = \text{CODR}_j = \text{CODR}(N_x) = \text{CODR}(N_y)] \quad (7)$$

It contradicts the assumption (3) and (6). In addition, as stated previously, elite nodes constitute the cabinet, and because of single leader in a component, a cabinet belongs to the only leader and according to our assumption $K \ll N$, all N node cannot become CAG. ■

LEMMA 2: *If node i losses its coordinator and requires a new coordinator, eventually it will get the leader. Formally,*

$$\nexists i : \square \text{CODRelect}_i = \text{FALSE} \quad (8)$$

Argument: Assume the contrary.

$$\exists i: \square \text{CODRelect}_i = \text{FALSE} \quad (9)$$

Equation 9 can hold only in any of the following three situations:

Case 1: Node i does not receive the CM forever. Formally,

$$\forall \mathcal{N}_x, \exists i, j \in \mathcal{N}, i, j \in \mathcal{N}_x, i \neq j: j \in \text{neighbor_name}_i \wedge \text{RECV}_i(\text{EM}, j) \Rightarrow \square \neg \text{RECV}_i(\text{CM}, j) \quad (10)$$

However, if some node is not partitioned, it is assumed that it will eventually be able to receive CM in any of the four attempts. Formally,

$$\forall \mathcal{N}_x, \exists i, j \in \mathcal{N}, i \neq j: (i \in \mathcal{N}_x) \wedge \text{RECV}_i(\text{EM}, j) \Rightarrow \diamond [\text{RECV}_i(\text{CM}, j) \wedge (\text{CODR}_i = \text{CODR}_j)] \quad (11)$$

Therefore, equation (9) and (10) cannot hold for ever.

Case 2: Node i does not receive any message from any node. Formally,

$$\begin{aligned} & \exists i, j \in \mathcal{N}, i \neq j: j \in \text{neighbor_name}_i \wedge \\ & \neg(\text{RECV}_i(\text{AM}, j) \vee \text{RECV}_i(\text{CM}, j) \vee \text{RECV}_i(\text{AIM}, j) \\ & \vee \text{RECV}_i(\text{NM}, j) \vee \text{RECV}_i(\text{heartbeat}, j)) \end{aligned} \quad (12)$$

This situation can be interpreted as node i is isolated due to network partition. Formally,

$$i, j \in \mathcal{N}, i \neq j: \forall j \notin \text{neighbor_name}_i \quad (13)$$

However, this situation cannot remain true for infinitely long time due to Theorem 1. Therefore, equation (9) and (12) cannot hold for ever.

Case 3: The protocol could not elect leader forever. Formally,

$$\forall i \in \mathcal{N}: (i \in \mathcal{N}_x) \wedge \square (\text{CODR}_i = \phi) \quad (14)$$

However, this situation is infeasible due to Theorem 1. Hence, equation (9) and (14) cannot hold for ever. Thus, equation (1) will always hold.

Therefore, all these three case contradicts our assumption. ■

LEMMA 3: *If leader and vice-leader of system crash together, eventually new leader will take the charge. Formally,*

$$\forall i \in \mathcal{N}, i \in \mathcal{N}_x: \neg \text{CODR}(\mathcal{N}_x) \wedge \neg \text{VC}(\mathcal{N}_x) \Rightarrow \diamond \text{CODR}(\mathcal{N}_x) \quad (15)$$

Argument: Assume the contrary.

$$\forall i \in \mathcal{N}, i \in \mathcal{N}_x: \neg \text{CODR}(\mathcal{N}_x) \wedge \neg \text{VC}(\mathcal{N}_x) \Rightarrow \square \neg \text{CODR}(\mathcal{N}_x) \quad (16)$$

Equation (16) can be true in either of following two cases:

Case 1: In the first case assume that crash of leader and vice-leader leads to network partition. Formally,

$$\exists \mathcal{N}_x, \mathcal{N}_y, \exists i, j \in \mathcal{N}, i, j \in \mathcal{N}_x, i \neq j: \neg \text{CODR}(\mathcal{N}_x) \Rightarrow (i \in \mathcal{N}_x) \wedge (j \in \mathcal{N}_y) \quad (17)$$

However, Lemma 1 states that every component has unique leader and according to Theorem 1, every MANET will eventually get a leader. Thus equation (16) and (17) cannot be true.

Case 2: Consider that the crash of leader and vice-leader does not lead to network partition, thus all the nodes of pre-established coordinated MANET lose their coordinator. Formally,

$$\forall N_x, \forall i \in N, i \in N_x: \neg \text{CODR}(N_x) \Rightarrow \square (\text{CODR}_i = \phi) \quad (18)$$

According to Lemma 2, whenever any node loses its leader, it will get the leader eventually. Formally,

$$\forall N_x, \forall i \in N, i \in N_x: \neg \text{CODR}(N_x) \Rightarrow \diamond \text{CODR}(N_x) \quad (19)$$

Therefore, equation (16) and (18) are false. ■

LEMMA 4: *If any new node y joins the system, it will get the leader eventually.*

Argument: Whenever any new node y wants to join the system, it sends EM to its neighbor z . Node z might also be new joining node which may also not have the leader information; thus, it sends WM to node y . Now, node x wants to join the system and sends EM to its neighbor y . Node y again sends WM to node x . If another node v wants to join the system, it also sends the EM to node x . This may form a waiting chain; however, it would have the finite length. Since the node z is the earliest new joining node in the system, it would have already sent its request to pre-established coordinated MANET's node or CAG or vice-leader or leader. Thus, node z will receive NM, eventually. Once this occurs, there is a chain of NM and UM that propagates down until node v receives the NM. Therefore, the new joining node eventually knows the ID of current leader. ■

THEOREM 2: *ELFA elects a leader eventually.*

Proof. Liveness property represents that all nodes start election process in candidacy state and eventually all node progress towards normal state in which all nodes connected to the system agree to the only coordinator in the presence of K CAGs. Thus, from Lemma 2, 3 and 4, it is oblivious that system always holds a leader. Formally,

$$\diamond (\forall i : \text{CODR}_{\text{elect}_i} = \text{TRUE}) \wedge (\exists K \ll N : \text{CAG_status}_K = \text{TRUE}) \quad \blacksquare$$

THEOREM 3: *ELEA is deadlock free.*

Proof. ELFA is deadlock free because EM can never get blocked at any CAG and the minimum ID CAG executes MELFA in the best case within CAGs. ■

4.5 MESSAGE COMPLEXITY

ELFA is based on MELFA, however, it does not use message broadcast till the entire cabinet vanishes. The message complexity of ELFA is much better in considerable number of cases. Now, we consider them one by one.

4.5.1 Best Case:

In ELFA, in the best case, the joining of new node in the pre-established coordinated MANET, results in minimum number of message (EM, NM and UM) exchanges. It can be understood with the following two scenarios.

Case 1: Assume that, new joining node becomes the neighbor of some CAG. Afterwards, CAG receives EM and sends NM and UM. Hence, only three messages (1 EM, 1 NM and 1UM) are exchanged to allow new joining node to access the services of current leader in the system, refer Figure 4.5.

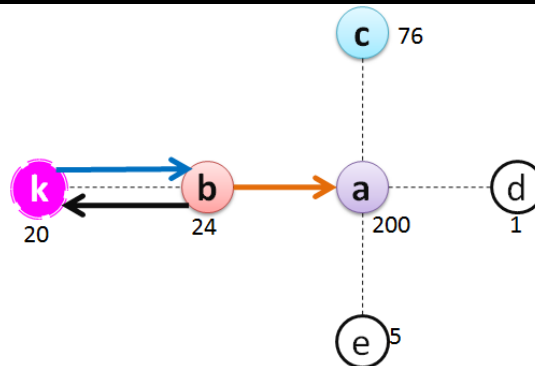


Figure 4.5: Best Case Scenario 1 of ELFA. New joining node, say node k , is neighbor of some CAG, say node b .

Case 2: Again, assume that, new joining node becomes the neighbor of some ordinary node i . Soon after that, node i receives the EM and sends NM and UM. Now, it is possible that node i is not neighbor to its coordinator, however, we have already assumed that the cost of UM is one. Hence, this scenario also leads to exchange of only three messages (1 EM, 1 NM and 1UM) to join a newly arrived node, refer Figure 4.6.

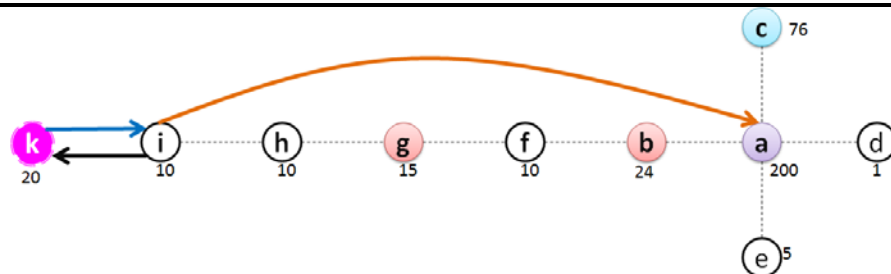


Figure 4.6: Best Case Scenario 2 of ELFA. New joining node, say node k , is neighbor of some ordinary node, say node i .

Hence, ELFA uses total 3 messages, in the best case.

4.5.2 Average Case:

We consider a case when a new joining node wants to join the pre-established uncoordinated MANET whose leader is crashed, refer Figure 4.7. In this scenario, say node k , wants to join the system, hence, it will send EM to its neighbor(s), say node n . However, due to leader crash, node n also does not have the identity of its leader. Thus, with the help of routing layer, either CAGs or vice-leader receive the EM and execute *Event 3*. The

execution of *Event 3* leads to alike best case execution of MELFA within CAGs. Therefore, it uses total $3(K-1)$ messages (i.e., $K-1$ EMs, $K-1$ AMs and $K-1$ CM) to elect a new vice-leader. However, $K \ll N$ and, thus, total number of EMs and NMs exchanged due to new joining node cannot be greater than $2N$.

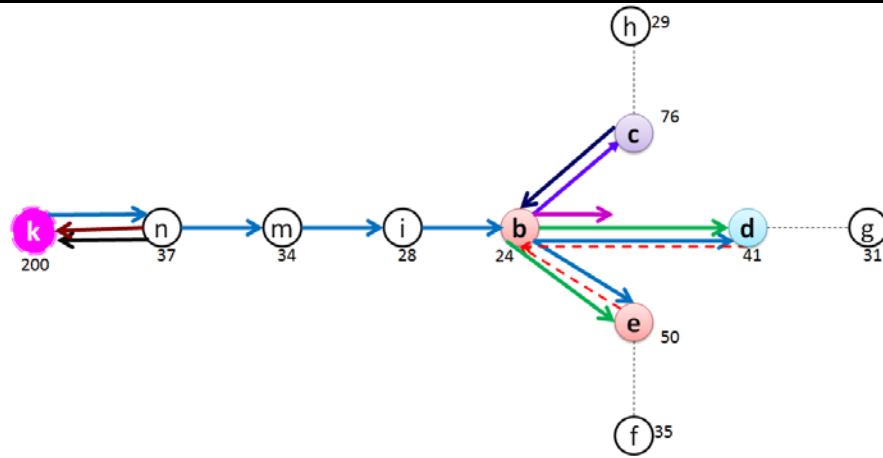


Figure 4.7: Average Case of ELFA.

4.5.3 Worst Case:

In the worst case, complete execution of MELFA will take places, resulting in a new leader, vice-leader and cabinet. Worst case of ELFA can be explained in two scenarios as following:

Case 1: Assume that, when a new joining node enters in the pre-established coordinated MANET, whole cabinet with leader and vice-leader crashes. In this case, different components will initiate MELFA individually and chose its leader, vice-leader and cabinet, refer Figure 4.8.

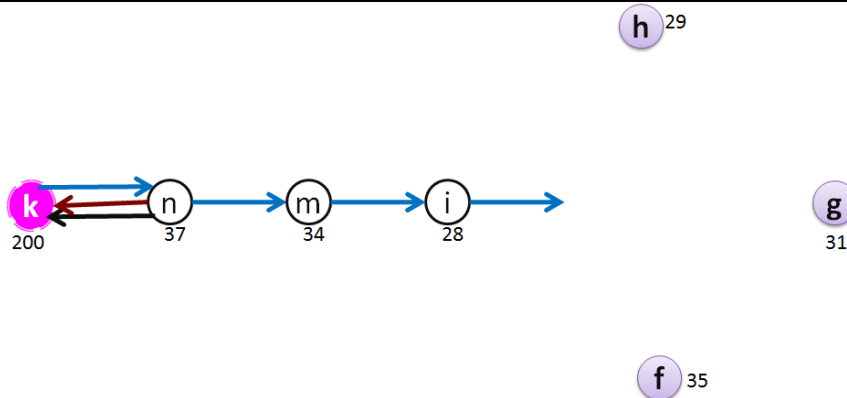


Figure 4.8: Worst Case Scenario 1 of ELFA. Crash of leader, CAG (results in network partition) and entrance of new joining node.

Case 2: This case is simpler than worst case 1. In this scenario, without joining the new node, all pre-established uncoordinated MANET nodes detect failure of leader, vice-leader and cabinet as well. Again, they will lead to complete execution of MELFA, refer Figure 4.9.

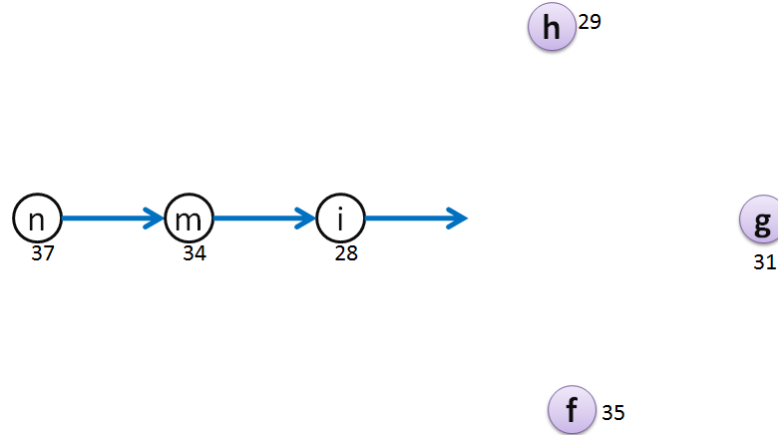


Figure 4.9: Worst Case Scenario 2 of ELFA. Crash of leader and CAG results in network partition.

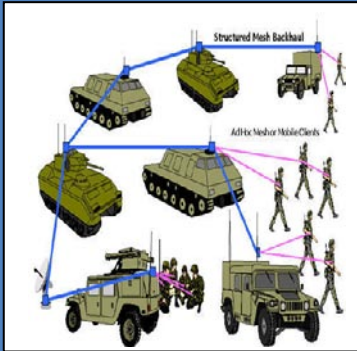
Hence, in both the above cases, MELFA may be executed in its worst case and that may amount to message complexity $O(n^n)$.

4.6 DISCUSSION

In this chapter, we have developed a fault tolerant coordinator election protocol, ELFA, which uses the concept of elite nodes, that makes the protocol more failure resilient. Also, the dependence on MELFA to elect a leader as well as cabinet leads to minimum number of control messages exchange. In addition, there is no fixed criterion to decide the size of cabinet. Nevertheless, the size of cabinet must grow with increase in size of total mobile nodes; furthermore, the susceptibility of nodes to failure may be considered the critical parameters to decide the size of cabinet. Therefore, there can not be a deterministic method to find the optimal size of the cabinet.

CHAPTER 5

DEMOCRATIC LEADER ELECTION



5.1 SYSTEM MODEL

5.2 MESSAGES & DATA STRUCTURES

- Types of Messages
- Data Structures

5.3 THE ELFA PROTOCOL

- Revised Procedure and Augmented Phase of MELFA
- Special Events of ELFA
- Protocol Explanation

5.4 CORRECTNESS PROOF

5.5 MESSAGE COMPLEXITY

- Best Case
- Worst Case
- Average Case

5.6 DISCUSSION

“Doing nothing is very hard to do . . . you never know when you’re finished”

- Leslie Nielsen

Although ELFA is fault tolerant, it cannot work for large MANETs. But, why? The answer is ELFA's *Event 3* where minimum ID *CAG* initiates election process within *CAGs*. Similar concept cannot apply to large MANETs due to increasing number of *CAGs* in large mobile ad hoc network. In addition, ELFA does not provide any method to accommodate new joining node with better potential as *CAG* that's why ELFA is not highly available protocol for leader election.

As stated in *Chapter 2*, a large number of protocols have been proposed for coordinator election in MANETs. In the event of failure, most of the protocols consume considerable amount of time in providing the alternate leader. Sometimes, the entire applications, refer Section 1.2.2, has to be aborted due to absence of alternative leader for long time. Therefore, for most of the applications, it is highly desirable to ensure the availability of alternative leader quickly, in the event of leader crash and/or unreachability of leader. This challenge is similar to the problem faced by prevailing democratic system of polity where the existence of some executive is always required to take decisions regarding the affairs of the state during the normal situation as well as during the crisis. Hence, the approach used in our protocol is inspired by the system followed by various democratic countries. By adapting the parliamentary system, we create Lower House (*LH*), Upper House (*UH*) of special nodes, having better capabilities like battery backup, computation power, etc. However, members of parliament do have any special characteristics because it may turn the algorithm asymmetric.

In addition, we consider the existence of best node called President (*P*), the second best node called Leader (*L*), the third best node called The Vice-Leader (*VL*) and the fourth best node called Vice-President (*VP*). Leader *L* is like Prime Minister, who act as head of government and *P* is like president who act as head of state and act as head of government only during crisis in democratic system. Formally, *L* is the head of *LH* and acts as coordinator of the system. *P* does not deliver any service to the network in the presence of *L*; however, *P* acts as coordinator of the system if *L*, *VL* and *VP* crashed together. *VL* is also member of *LH* and provides services to the network when *L* crashes. *VP*, the head of *UH*, acts as coordinator of the system, if *L* and *VL* crash simultaneously (see Figure 5.1). *L* and *VL* are like Prime Minister (PM) and Deputy PM in democratic system. Keeping in view the operational semantics of the proposed protocol, it has been named as DEMocratic Leader Finding Algorithm (called DELFA, henceforth). Like ELFA protocols, DELFA has four novel properties:

- i. Due to the dependence on MELFA, it does not broadcast control messages at every step. Therefore, it uses less number of messages.
- ii. The parliament concept brings in failure resiliency to our protocol and ensures greater availability of leader, especially on occurrence of failure. To the best of our knowledge, this concept has not been used earlier in the literature.
- iii. The possibility of network partitioning can never be avoided fully in MANETs. The proposed protocol, DELFA, can also handle network partitioning and merging efficiently using very small number of algorithm messages.

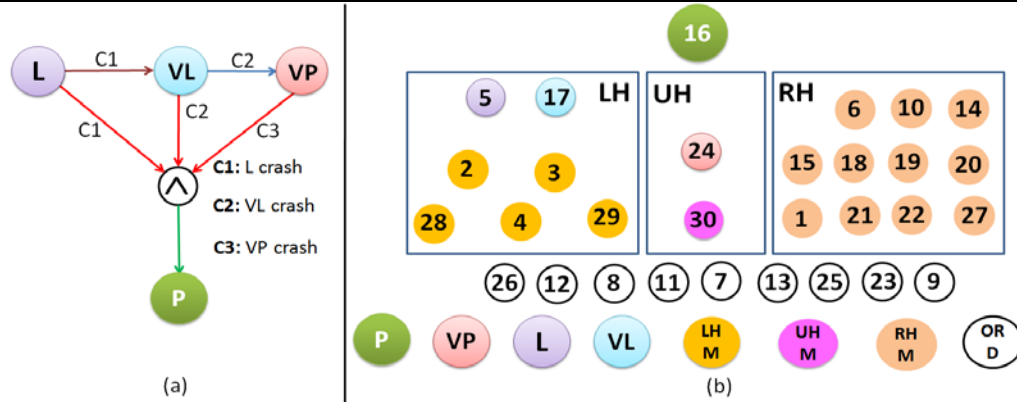


Figure 5.1: Nodes in DELFA. In Figure (a), crash of special nodes, *i.e.*, L, VL and VP is shown. In Figure (b), abstract view of all nodes in system with LH, UH and RH is shown.

- iv. DELFA is a self stabilizing protocol [36]. An algorithm is known as self-stabilizing when regardless of its initial state, it is guaranteed to arrive at a legitimate state in a finite number of steps. DELFA also has this property as it converges to a stable state, *i.e.*, leader UH for the large MANETs, in finite time.

Next, in this chapter, we will present system model, message types, data structures, concept, correctness proof, and message complexity of DELFA protocol.

5.1 SYSTEM MODEL

DELFA protocol assumes MANET in the form of undirected graph G . Furthermore, it also has the primary and secondary assumptions for election protocol like MELFA, Section 3.1. In addition, DELFA has one more primary assumption regarding the size of LH and UH , as following:

- **Size of LH and UH :** The number of Lower House Member nodes ($LHMs$), *i.e.*, $K \ll N$, the total number of nodes in the system and the number of Upper House Member nodes ($UHMs$), *i.e.*, $M = K/2$.

5.2 MESSAGES & DATA STRUCTURES

In this section, we discuss different types of messages and data structures used by DELFA, as following:

5.2.1 Types of Messages:

In the beginning, DELFA initiates MELFA by using three basic algorithm messages, *i.e.*, Election Message (EM), Acknowledgement Message (AM) and Coordinator Message (CM), like MELFA and ELFA. In addition, DELFA uses following types of messages:

1. **Request Message (RM):** send by new joining node i to its neighbor to get the ID of current leader.
2. **Notify Message (NM):** send by node j to inform about the ID of current leader to new joining node i in pre-established coordinated MANET. It is also used for handling network partition and merging (Figure 5.7).
3. **Wait Message (WM):** send by waiting node j to another waiting node i to avoid unnecessary RM flow if node j does not have current leader ID.
4. **Update Message (UM):** send by node j , which have the ID of current leader L , in order to update L about the ID of new joining node i with its NW.

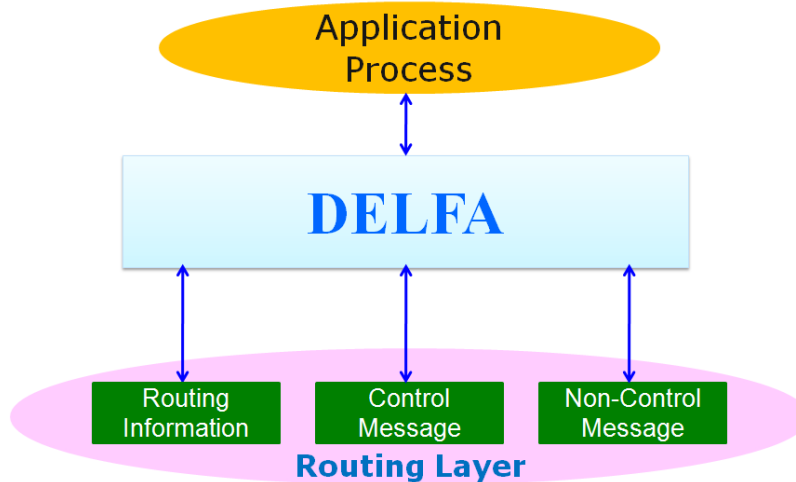


Figure 5.2: System Model. In this figure, different types of messages and their relation with DELFA is shown.

5. **Information Message (IM):** send by node a to direct: (i) the new joining node i to change its STATUS either Ordinary Node (ORD) or LHM or UHM or RHM , (ii) higher NW nodes of UH to become LHM , and (iii) highest NW UHM to become P . Here, node a may be L , VP or P .
6. **Leave Message (LM):** send by newly elected leader to leader-in-charge to quit the charge.
7. **Heartbeat acknowledge Message (HRM):** send by all $LHMs$ including VL , in response to every tenth heartbeat message, to L with their NWs.
8. **CABinet message (CAB):** send by leader-in-charge to lowest ID node i to elect new leader and next vice-leader within $LHMs$ or $UHMs$, as the case may be.
9. **Nomination Message (NoM):** send by L to VP in order to maintain the size of LH upper bounded by K , if some $LHMs$ got crashed. NoM includes expected total number of nodes to maintain K .
10. **President Check Message (PCM):** send by VP to P after every 20th heartbeat message.
11. **Acknowledge President Check Message (APCM):** On the reception of PCM, P sends its NW to VP via APCM.

The types of messages are large in DELFA. However, only five types of messages (*i.e.*, EM, AM, CM, UM and LM) are control messages and rest (*i.e.*, RM, NM, WM, IM, HRM, CAB, NoM, PCM and APCM) are non-control messages. Thus, we assume that all non-control messages have unit cost as they can be piggybacked on application message. It is worth noting that control messages have priority over non-control message.

5.2.2 Data Structures:

In DELFA, there are six categories of data structures. First, we define the data structures maintained at each node and then, describe associated data structures with EM and CM . The data structures used are as follows:

- i. **At *ori* Node:** In DELFA, definition of *ori* is slightly changed. In DELFA, a node having node weight, NW, greater than 70, called *ori* node, which, initiates the election protocol when the MANET does not have leader in the beginning or the leader, L , got crashed. It may be noted that a node having NW equal to 70 is interpreted as the node has NW which is the 70% of NW of the tentative best node which have been assumed to have NW equal to 100. However, *ori* node has similar data structure like

MELFA as $receive_ack[n]$, $receive_weight[n]$ and Time Out Value (t_e). Details of these data structures are given in Section 3.2.2.

- ii. **At All Nodes:** Likewise ELFA, all nodes in DELFA have ID, $neighbor_name_i[j]$, NW_i , $CODR_i$, $CODRelect_i$, $election_send_i$, $send_for_i$, $coordinator_send_i$ and $wait_set_i[w]$. Moreover, in DELFA, all nodes have one more data structure with slight variation in $STATE$ data structure as following:
 - a. $STATE_i$: Again $STATE_i$ represent the current state of node i . In DELFA, all the nodes may be in one of the six states: NORMAL – node is in normal state if it is continuing normal computation in presence of the leader; CANDIDACY – node is in candidacy state either it lost its contact with the coordinator or the coordinator got crashed; PRESIDENT (P) – an elected node having highest NW by ori ; LEADER (L) – an elected node, having second highest NW, is in leader state and provide services to whole network; VICE-LEADER (VL) – an elected node having third highest NW; VICE-PRESIDENT (VP) – an elected node having fourth highest NW. Initially, $STATE_i = NORMAL$.
 - b. $STATUS_i$: represent the current status of node i based on its NW. In DELFA, each node sets its status NW according to its NW. All the nodes may be in one of the three statuses: ORDINARY Node (ORD) – nodes having NW between 0-30; RESERVE HOUSE Member Node (RHM) – nodes having NW between 31-70; LHM – nodes having NW between 71-100.
- iii. **Common at Special Nodes:** In addition to node data structures, special node like P , VP , L and VL has the following data structure, to provide higher availability of leader.
 - a. Pid_i : represent the identity of current President of system. Initially, $Pid_i = \phi, \forall i$.
 - b. $VLid_i$: is the identity of current Vice-Leader of system. Initially, $VLid_i = \phi, \forall i$.
 - c. $VPid_i$: is the identity of current Vice-President of system. Initially, $VPid_i = \phi, \forall i$.
 - d. $LHMid_i[K]$: is an array of all $LHMs$. Initially, $LHMid_i[K] = \phi, \forall k$.
- iv. **Only at P and VP:** Further, P and VP have the following additional data structures:
 - a. $UHM_i[M]$: is the array of all the Upper House Member nodes ($UHMs$).
 - b. $UHMNW_i[M]$: is the array of NW of all Upper House Member nodes ($UHMs$).
- v. **Piggybacked with EM:** Election Message has $receive_election[n]$, ori and $sender_election$ data structures with one additional data structure, as follows:
 - a. $forward_flag$: is initially set to 0 and represent that EM should be forward by node i when it receives it. Value 1 indicates that EM should not be forwarded by any receiving node i and ori node with ID i sets $forward_flag$ to 1 on reception of CAB.
- vi. **Piggybacked with CM:** Like MELFA, Coordinator Message has earlier $receive_coordinator[n]$. In addition, CM has following data structures too:
 - a. $forward_flag$: is initially set to 0 and consider like $forward_flag$ of EM.
 - b. PID : represents ID of the newly elected President.
 - c. LID : represents ID of the newly elected Leader.
 - d. $VLID$: represents ID of the newly elected Vice-Leader.

- e. *VPID*: represents ID of the newly elected Vice-President.
- f. *LHMID[K]*: represents IDs of all Lower House nodes.

5.3 THE DELFA PROTOCOL

Although, DELFA uses altogether new concept of parliament, it runs MELFA (Section 3.3) in the beginning. However, it modifies all phases as well as procedure of MELFA to ensure higher availability of leader. In this section, we present all these modified phase with execution of DELFA in detail.

5.3.1 Revised Procedure and Augmented Phase of MELFA:

Modified Phase 1: This phase is almost similar to *Phase 1* of original MELFA protocol. In this phase, only nodes having NW in-between 71-100 are eligible to initiate MELFA. Because the nodes having NW less than 71 are considered susceptible to crash while executing MELFA. The constraint is required to preserve the NW of other nodes except *LHMs*. This fact is in contrast with original MELFA where all nodes of the system could initiate election. Rest of the phase is similar to *Phase 1*.

Modified Phase 2: In this phase, only *LHMs* send AM to *ori* and rest of the phase is similar to *Phase 2* of MELFA.

Procedure C – Election of Special Nodes, i.e., *P*, *VP*, *L* and *VL*: This procedure is almost like Procedure A and executes at *ori* simultaneously with the start of *Modified Phase 1*. It elects and broadcasts CM piggybacked with highest NW node as President, second highest NW node as Leader, third highest NW node as Leader and forth highest NW node as Vice-President to its neighbors.

Augmented Phase 3 – CM Distribution: *Augmented Phase 3* distributes the CM in the whole network like *Phase 3*, as discussed above. The special nodes *P*, *L*, *VL* and *VP* also contain the information about each other and *LHMs*. In addition, each *LHM* maintains the complete list of *LHMs*.

Pseudo Code: Modified Phase 1

```

IF (STATEi = NORMAL AND CODRi = ∅ AND NWi > 70) THEN
  ori ← i;
  STATEi ← CANDIDACY;
  sender_election ← ori;
  election_sendori ← TRUE;
  coordinator_sendori ← FALSE;
  codr_electori ← FALSE;
  ∀ j: j ∈ neighbor_nameori;
    receive_election[j] ← 1;
  Set timer te;
  ∀ j: j ∈ neighbor_nameori;
    Broadcast EM to j;

```

Pseudo Code: Phase 2 – Concurrent EM receive

```

Node j receive EM form i and x
STATEj ← CANDIDACY;
coordinator_sendj ← FALSE;
CODRelectori ← FALSE;

```

```

IF (election_sendj == TRUE AND send_forj > EM.orii AND EM.orii < EM.orix) THEN
  Discard EM.orix // EM.orix represents ori of the EM //
  IF (NWj > 70) THEN Send AM;
  ∀ k: k ∈ neighbor_namej;
    Broadcast EM.orii to all k;
ELSE IF (election_sendj == FALSE AND EM.orii < EM.orix) THEN
  Discard EM.orix;
  IF (NWj > 70) THEN Send AM;
  IF (∀ k: k ∈ neighbor_namej, receive_election[k] == 1) THEN
    Do nothing
  ELSE IF (∃ k: k ∈ neighbor_namej, receive_election[k] == -1) THEN
    IF (NWj > 70) THEN Send AM;
    election_sendj ← TRUE;
    send_forj ← ori;
    receive_election[k] ← 1;
    sender_election ← j;
    ∃ k: k ∈ neighbor_namej, receive_election[k] == -1
    Forward EM to k;
  
```

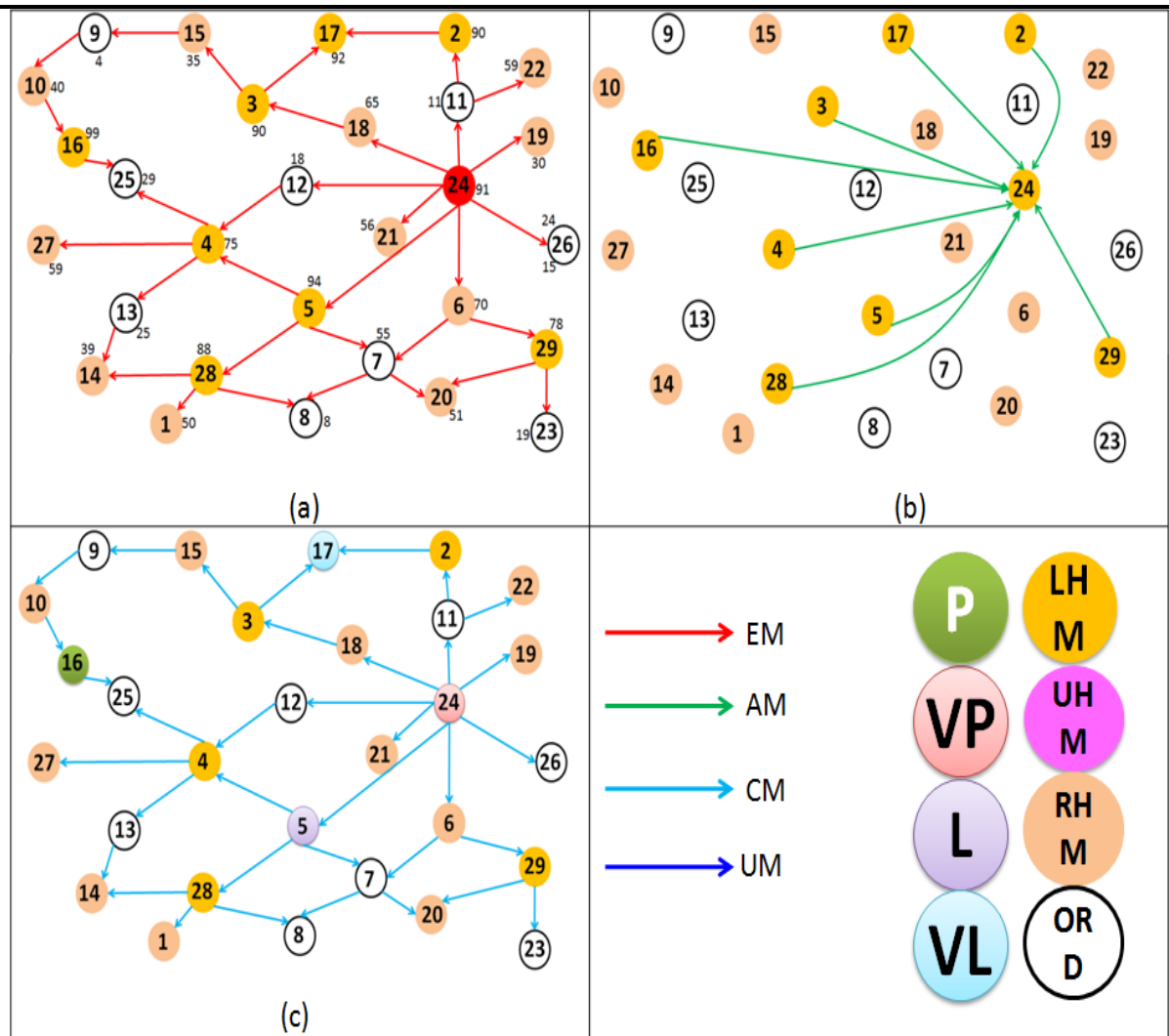


Figure 5.3: Modified MELFA in Execution. In this figure, MELFA with modified phase and procedure is shown.

Pseudo Code: Procedure C – Election of Special Nodes, i.e., P, VP, L and VL

```

IF (receive_ack[j] == -1 AND te == TRUE) THEN
    reinitiate Phase 1 (4 times)
ELSE
    ori receive AM from j;
    receive_weight[] = receive_weight [] ∪ NWj; receive_ack[] = receive_ack [j] ∪ j;
    IF (te == TRUE) THEN
        PID ← max_weight_id (receive_weight []);
        LID ← (max-1)_weight_id (receive_weight []);
        VLID ← (max-2)_weight_id (receive_weight []);
        VPID ← (max-3)_weight_id (receive_weight []);
        LHMID[] ← receive_ack[];
        CODRori ← LID; CODRelectori ← TRUE;
        Pidori ← PID; CODRidori ← LID; VLidori ← VLID;
        VPidori ← VPID; LHMidori[] ← LHMID [];
        IF (PID == oriid) THEN STATEori ← P;
        ELSE IF (LID == oriid) THEN STATEori ← L;
        ELSE IF (VLID == oriid) THEN STATEori ← VL;
        ELSE IF (VPID == oriid) THEN STATEori ← VP;
        ∀ k: k ∈ neighbor_nameori;
            receive_coordinator [k] ← 1;
        coordinator_sendori ← TRUE; election_sendori ← FALSE;
        ∀ k: k ∈ neighbor_nameori;
            Broadcast CM to k;

```

Pseudo Code: Augmented Phase 3 – CM Distribution

```

Node j receive CM from I
IF (CODRelectj == FALSE) THEN
    CODRj ← PID; CODRelectj ← TRUE; election_sendj ← FALSE;
    IF (STATUSj == LHM) THEN
        Pidori ← PID; VLidori ← VLID; VPidori ← VPID;
        LHMidori[] ← LHMID [];
    IF (PID == oriid) THEN STATEori ← P;
    ELSE IF (LID == oriid) THEN STATEori ← L;
    ELSE IF (VLID == oriid) THEN STATEori ← VL;
    ELSE IF (VPID == oriid) THEN STATEori ← VP;
    ELSE STATEori ← NORMAL;
    IF (∀ k: k ∈ neighbor_namej, receive_coordinator[k] == 1) THEN
        coordinator_sendj ← TRUE;
    ELSE IF (∃ k: k ∈ neighbor_namej, receive_coordinator[k] == -1)
        coordinator_sendj ← TRUE; receive_coordinator[k] ← 1;
        ∃ k: k ∈ neighbor_namej, receive_coordinator[k] == -1
    Forward CM to k;

```

After termination of *Augmented Phase 3*, *L* continues to send heartbeat message.

5.3.2 Special Events of DELFA:

DELFA handles following events to ensure higher availability of leader:

Event 1: Limiting the Size of LH to K. All nodes know their STATUS; however, the number of *LHMs* may be greater than *K*, initially. In this situation, leaving *K* lower ID nodes, others are directed to quit as *LHM*. This is achieved by piggybacking *LHM_remove*[] on second heartbeat message.

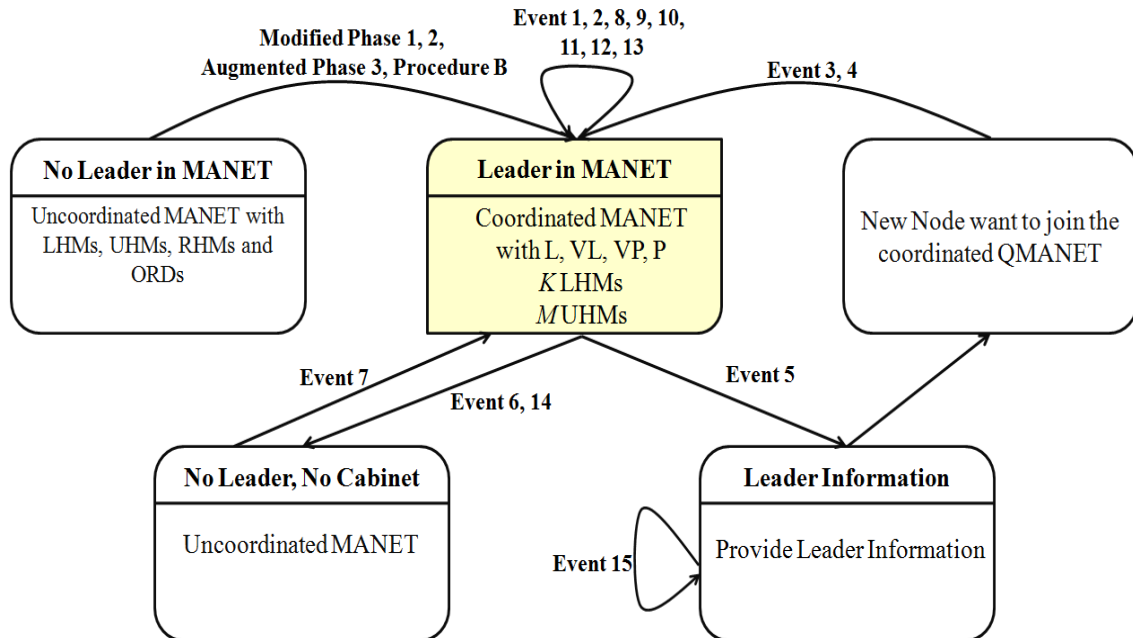


Figure 5.4: DELFA Protocol State Diagram. In the figure, relationship between events and phases of DELFA are shown.

Event 2: Reception of ‘Second Heartbeat’ at LHMs. The LHMs, on receiving ‘second heartbeat’ message from leader, verify their ID in *LHM_remove[]* and quit the STATUS as LHM, if need be, and become RHM or ORD based upon their current NW, as the case may be.

Pseudo Code: Event 2

```

Node i receive second heart beat message form L
IF ( $i \in \text{LHM\_remove}[]$  AND  $\text{NW}_i > 30$ ) THEN
    STATUSi ← RHM;
ELSE IF ( $i \in \text{LHM\_remove}[]$  AND  $\text{NW}_i < 30$ ) THEN
    STATUSi ← ORD;
    
```

Event 3: New incoming node sends RM to join the pre-established coordinated MANET. Event 3 will occur when one or more nodes arrive within the transmission range of pre-established coordinated MANET, where every node knows about the current coordinator and new joining node that wants to participate in the MANET. In this state, new node sends RM to one of its neighbors, selected arbitrary, and wait for the time out value t_e , reinitiate this event with some neighbor other than previously selected, otherwise.

Pseudo Code: Event 3

```

IF ( $\text{STATE}_i = \text{NORMAL}$  AND  $\text{CODR}_i = \emptyset$ ) THEN
    DO
        Set timer  $t_e$ ;
        Select j randomly :  $j \in \text{neighbor\_name}$ ;
        IF (Not send RM earlier to j) THEN
            Send RM to j;
        WHILE (NM is not received from j)
    
```

Event 4: Reception of RM. When some node *j* receives RM and it has the ID of current coordinator, it forwards NM and UM; sends WM and update *wait_set_j[],* otherwise.

Pseudo Code: Event 4

```

Node  $j$  receive RM from  $i$  and  $x$ ;
IF (CODR $j$  ==  $\emptyset$ ) THEN
    Send WM to node  $i$ ;
    wait_set $j$ [] = wait_set $j$ []  $\cup$   $i$ ;
ELSE Send UM to node  $i$ ;

```

Event 5: Reception of UM. On the occurrence of this event, leader L sets the STATUS of new joining node i to ORD , LHM , UHM or RHM based on NW of node i and sends IM to node i in order to inform node i about its new STATUS. The nodes having NW greater than NW_P are allowed to become UHM on the occurrence of Event 5 till the size of UH reaches M , where M is upper bounded by $K/2$.

Pseudo Code: Event 5

```

L receives UM from node  $j$  about new joining node  $i$ 
IF (NW $i$  > NW $P$  AND UH  $\neq$  FULL) THEN
    STATUS $i$   $\leftarrow$  UHM;
ELSE IF (NW $i$  > NW $P$  AND UH == FULL AND LH  $\neq$  FULL) THEN
    STATUS $i$   $\leftarrow$  LHM;
ELSE IF (NW $i$  > NW $P$  AND UH == FULL AND LH == FULL) THEN
    STATUS $i$   $\leftarrow$  RHM;
ELSE IF (NW $i$  < NW $L$  AND NW $i$  > 30) THEN
    STATUS $i$   $\leftarrow$  RHM;
ELSE
    STATUS $i$   $\leftarrow$  ORD;
Send IM to node  $i$ ;

```

Event 6: No heartbeat message from L and no network partition. No heartbeat message from the leader signifies that either the leader crashed or became unreachable. In fact, DELFA demonstrates its strength by handling this event. It ensures the higher availability of leader and comprises various cases to provide a very high degree of fault tolerance. We consider them one by one as following:

Case 1: VL exists. In this scenario, VL becomes the leader of system and sends CAB (CABinet message) to lowest ID LHM .

Case 2: VL crashed and VP exists. Like Case 1, in this scenario, VP becomes the leader of system and sends CAB to lowest ID LHM .

Case 3: VL and VP both crashed, however, P exists. Like Case 1 and Case 2, P becomes the leader of system and sends CAB to lowest ID LHM .

Case 4: VL , VP and LH crashed, however, P exists. P becomes the leader of system and sends CAB piggybacked with $UHM_P[M]$ to lowest ID UHM . Also, P changes the STATUS of $UHMs$ to $LHMs$ and sends IM to all $UHMs$. Hence, DELFA ensures existence of leader even on the occurrence of an event when most of special nodes (i.e., L , VL and VP) as well as *Lower House* crashed together.

Case 5: VL and LH both crashed, however, VP exists. This case is handled similar to Case 4 above except here VP becomes the leader.

In addition, as the new L is harbingered, it becomes the coordinator of the system and sends LM (Leave Message) in all the above five cases.

Event 7: Reception of CAB. Leadership is transferred to some new leader, depending on situation, in the event of L crash. The new leader-in-charge (i.e., VL , VP or P) sends CAB to lowest ID LHM i if LH exists; otherwise, to lowest ID UHM j . Therefore, we consider these two cases one by one as following:

Case 1: *Reception of CAB at LHM.* In this case, LHM i sends EM, without any data structure piggybacked, to all K $LHMs$. In addition, node i sets *forward_flag* to 1. On reception of EM, node j sends AM to node i and subsequently, node i elects the new L , VL (Event 6, Case 2) and VP (Event 6, Case 3) after timeout.

Case 2: *Reception of CAB at UHM.* In this case, lowest ID UHM i receives CAB from P (Event 6, Case 4) or from VL (Event 6, Case 5). Node i broadcasts the EM, without any data structure piggybacked, to all M $UHMs$. In addition, node i sets *forward_flag* to 1 and node i elects L , VL and VP after timeout, as the case may be.

Event 8: Reception of every 10th heartbeat message. L sends heartbeat message to all nodes in the network. Although, NW of every node reduces regularly due to various activities performed by node, every 10th heartbeat message is special to $LHMs$ and VP . After receiving every 10th heartbeat message, all $LHMs$ and VP send the HRM to L that includes their NW.

Pseudo Code: Event 8

Node j receive 10th heartbeat message;
Node i send HRM where $i \in LH$;

Event 9: Reception of every 20th heartbeat message. Like Event 8, every 20th heartbeat is special for VP . It sends PCM to P . P is the best node, however, it participates in application, hence, it also losses its NW. In order to check the NW of P , VP sends PCM.

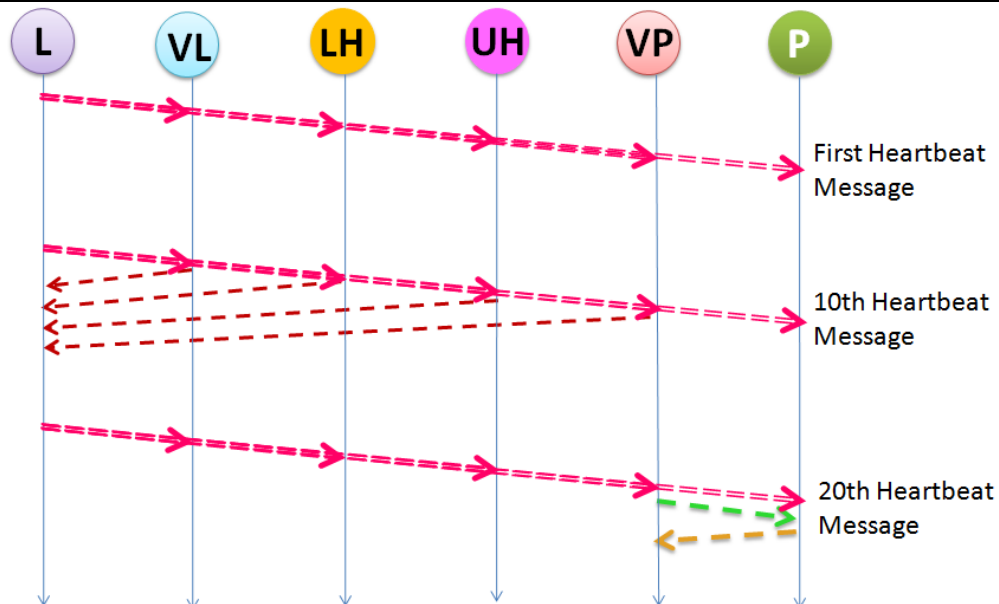


Figure 5.5: Event 8-12 of DELFA in Action.

Event 10: Reception of HRM. On receiving HRM, L detects the LHM s which have NW equal or less than 70 and sends IM to all such nodes to change their STATUS.

Pseudo Code: Event 10

```

 $L$  receives HRM form LHM  $i$ ;
IF ( $NW_i < 71$ ) THEN
    Send IM to node  $i$ ;

```

Event 11: Reception of PCM. On the reception of PCM, P sends its NW piggybacked with APCM.

Event 12: Reception of APCM. On receiving APCM, VP removes the current P and designates highest NW UHM as new P , if NW of current P is less than 71; does nothing, otherwise.

Pseudo Code: Event 12

```

 $VP$  receives APCM form  $P$ ;
IF ( $NW_P < 71$ ) THEN
     $m \leftarrow \max\_weight\_id(UHM_{NW_{VP}}[ ])$ ;
    Send IM to node  $m$ ;

```

Event 13: Reception of NoM. NoM is send by L to VP with specified value k to maintain the specified size of LH . VP selects highest k NW nodes from $UHM_{VP}[M]$ and sends IM to change their STATUS from UHM to LHM .

Pseudo Code: Event 13

```

 $VP$  receives NoM including value  $k$  from  $L$ ;
 $VP$  select  $k$  highest NW  $UHMs$ ;
Send IM to  $k$   $UHMs$ ;

```

Event 14: Network Partitioning. No heart beat message from L may lead to network partitioning. Network partitioning divides the network in such a way that some LHM s and $UHMs$ including VL , VP and P may belong to either components, in average case, see Figure 5.7 (b). In this situation, either special node (*i.e.*, VL , VP or P) sends CAB and becomes leader-in-charge; in addition, lowest ID house member initiates election by sending EM with *forward_flag* value 1 to remaining house members and elects new L , VL , VP or P , as the case may be, see Figure 5.7(c) and Figure 5.7 (d). The worst case partitioning can divide the bridge network in such a way that L , VL , VP , P , LHM s and $UHMs$ resides in one side of partitioned network and other side holds only RHM s and ORD s, see Figure 5.7 (h). In this scenario, RHM s and ORD s cannot initiate the election protocol according to *Modified Phase 1*; however, L will continue to be leader of its component.

Event 15: Network Merging. This problem is handled like [15]. This event occurs when two partitioned components meet due link formation. The coordinator node i of one of the components which has the higher NW becomes the coordinator of whole component and ID is used for tie breaking, if needed. However, unlike [15], in our approach, another component's leader j becomes the LHM and surrenders its leadership (see Figure 5.7e, 5.7f). It may be further noted that *Event 15* elects two VL s, VP s, and P s. The VL , VP and P of the component whose leader j has surrendered its leadership, becomes the LHM s and accept node i as its leader.

5.3.2 Protocol Explanation:

The example, in Figure 5.6¹, shows complete execution of DELFA. In Figure 5.6 (a), new joining node with ID 30 sends RM to its one neighbor and waits for timeout to receive the NM, *Event 3*. Mean time node with ID 3 sends NM to node 30 which contains the ID of current coordinator, *i.e.*, 5, *Event 4*. In addition, node 3 sends UM to leader node 5.

In Figure 5.6 (b), on the reception on NM, node 5 sends IM to node 30, *Event 5*. Next, in Figure 5.6 (c), leader, node 5, got crashed and subsequently, node 17, vice-leader, takes the charge of crashed leader and sends CAB to lowest ID LHM with ID 2, *Event 6: Case 1*. Reception of CAB initiates EM flow among LHMs shown in Figure 5.7 (d), *Event 7: Case 1*.

In Figure 5.6 (d), *Event 6: Case 2* will fire in the event of leader, ID 5, and VL, ID 17, crash, VP, ID 24, sends CAB to lowest ID LHM with 2. Moreover, node 24 becomes the leader. Again, reception of CAB initiates EM flow among LHMs shown in Figure 5.7 (d), *Event 7: Case 1*.

In Figure 5.6 (e), president node, with ID 16, after taking the charge of crashed leader, sends CAB to node 2 in the event of leader, ID 5, VL, ID 17, and VP, ID 24, crash, *Event 6: Case 3*. In addition, node 2 sends EM to all LHMs.

In Figure 5.6 (f), L, VL, VP and all LHMs with ID 5, 17, 24, 2, 3, 4, 28, 29 respectively, got crashed. Subsequently, node 16, president, takes the charge of leader and sends CAB to lowest ID UHM with ID 30, *Event 6: Case 4*.

In Figure 5.6 (g), L, VL and LH both crashed, *Event 6: Case 5*, and VP, ID 24, takes the charge of crashed leader and sends CAB to lowest ID UHM, ID 30. Next, in Figure 5.6 (h), all LHMs and VL sends HearTbeat acknowledge Message (HRM) on the occurrence of every 10th heartbeat message, *Event 8* and *Event 10*.

In Figure 5.6 (h), on the reception of every 20th heartbeat message VP, with ID 24, sends PCM to P, with ID 16, *Event 9*. In response, node 16 sends APCM to node 24, *Event 11*.

¹ Neighbors of all nodes are given Table 5.1.

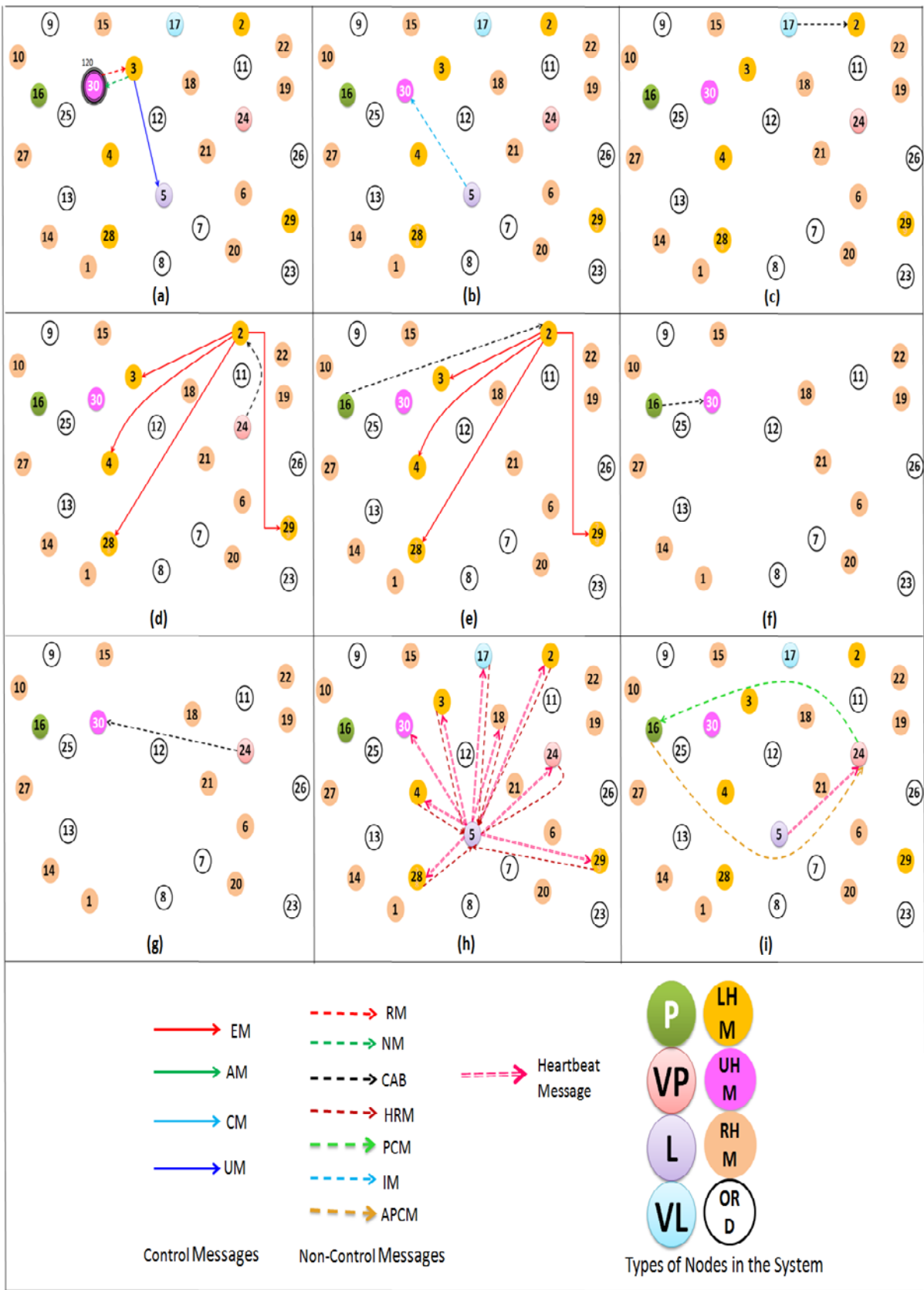


Figure 5.6: DELFA in Execution.

Table 5.1. Initial Node's Neighbors for DELFA

Nodes	Neighbor Nodes						
1	28						
2	11	17	22				
3	15	17	18				
4	5	12	13	25	27		
5	4	6	7	21	24	28	
6	5	7	24	26	29		
7	5	6	8	20	28	29	
8	7	28					
9	10	15					
10	9	16					
11	2	18	19	22	24		
12	4	18	21	24			
13	4	14	27				
14	13	28					
15	3	9	17				
16	10	25					
17	2	3	15				
19	11	24					
20	7	23	29				
21	5	12	18	24			
22	2	11					
23	20	29					
24	5	6	11	12	18	19	21
25	4	16	27				
26	6	24					
27	4	13	25				
28	1	5	8	14			
29	6	7	20	23			

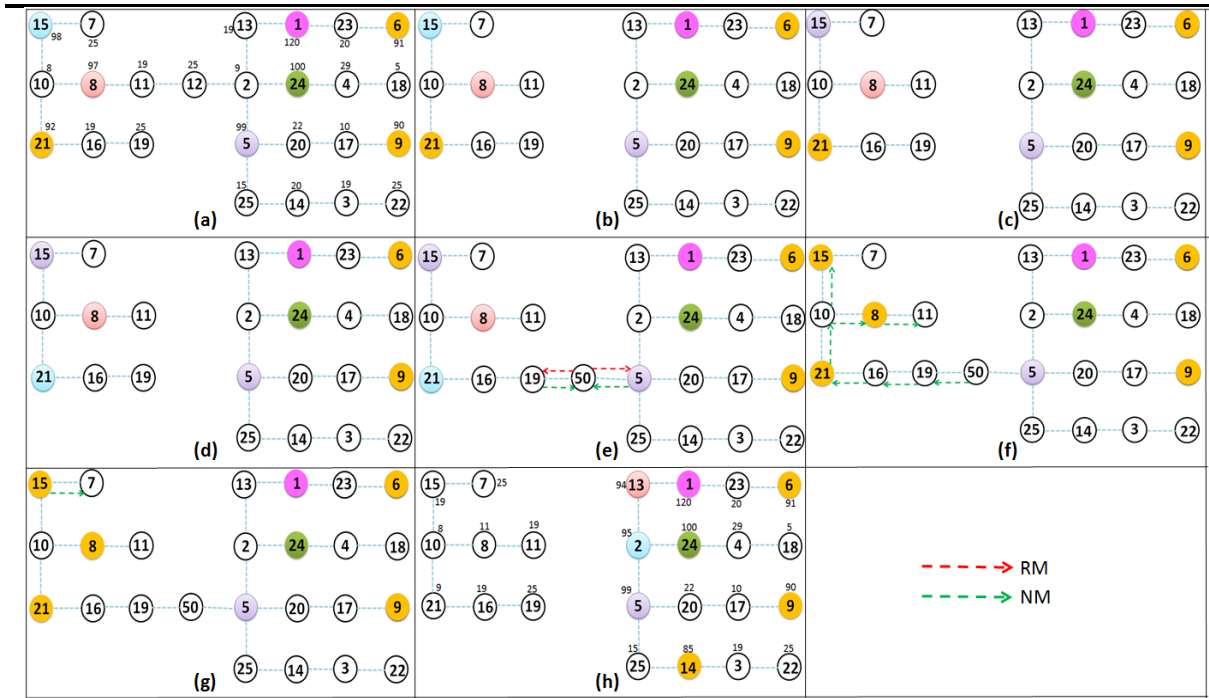


Figure 5.7: Network Partition Handling and Merge Operation of DELFA.

5.4 CORRECTNESS PROOF

DELFA is *stable coordinator election* protocol and meets safety and liveness standards. We again prove these properties with the help of predicate calculus and some lemmas of ELFA. Firstly, we proof lemmas for self-stabilization (SS, Lemma 1, 2, 3), fault tolerant (Lemma 4, 5) and termination (Lemma 6) properties, and then proof safety and liveness. We use same notation like ELFA with some additional as following:

- N : Total number of nodes;
- $CODR_i$: ID of current coordinator at node i ;
- N_x : MANET component x ;
- $P(N_x)$: Current President of MANET N_x ;
- $L(N_x)$: Current leader of MANET N_x ;
- $VL(N_x)$: Current vice-leader of MANET N_x ;
- $VP(N_x)$: Current Vice-President of MANET N_x ;
- $LH(N_x)$: Current Lower House (*LH*) of MANET N_x ;
- $SEND_i(M, j)$: Node i sends message M to node j ;
- $RECV_i(M, j)$: Node i receives message M from node j .

LEMMA 1: *MELFA execution with modified and augmented phases leads to a unique L , VL , VP and P and all $N-1$ nodes agree on the same chosen L . Formally,*

$$\forall i, j, 1 \leq i, j \leq N : (\text{STATE}_i = \text{NORMAL} \wedge \text{STATE}_j = \text{NORMAL} \wedge \text{NW}_L > 70) \Rightarrow (\text{CODR}_i = \text{CODR}_j) \quad (1)$$

Proof. Assume the contrary. There are two nodes i and j whose state is **NORMAL** with **CODR** value is different. Formally,

$$(\text{STATE}_i = \text{NORMAL} \wedge \text{STATE}_j = \text{NORMAL}) \wedge (\text{CODR}_i = i \wedge \text{CODR}_j = j) \wedge (i \neq j) \quad (2)$$

This statement can hold in either of the following two cases:

Case 1: Node i and j are isolated from the network and no other node exist in their transmission range and this case is like Theorem 1: Case 1 of ELFA (Section 4.4), thus, it is worthlessness to explain it once more here.

Case 2: Node i and j lie in two MANETs and these MANETs have no common node, so node i and j become the coordinator of their respective MANET. Again, this case is totally similar to Theorem 1: Case 2 of ELFA (Section 4.4).

Therefore, the Lemma 1 holds for above two cases. It is trivial to show similar proof for the uniqueness of for VL , VP and P also. ■

LEMMA 2: *DELFA ensures that each component of the system has unique leader. Formally, if N_x and N_y are two components of MANET then,*

$$\forall N_x, N_y : (N_x \neq N_y) \Rightarrow (L(N_x) \neq L(N_y)) \quad (3)$$

Argument: Concurrent initiation of *Modified Phase 1* does not affect on unique leader election, however, it significantly increases EMs flow in the network. In concurrent execution phase, higher priority is given to lowest ID LHMs. Thus, eventually, it harbingers the second best node as L either in single MANET \mathcal{N}_x or in both MANETs \mathcal{N}_x and \mathcal{N}_y . In addition, network partitioning may further create new components, say, \mathcal{N}_a and \mathcal{N}_b . Moreover, LHMs or UHMs of \mathcal{N}_a and \mathcal{N}_b initiate EM forwarding to elect tentative L , VL , P and VP , see Figure 5.7. Thus, in DELFA, each components has a unique leader. ■

LEMMA 3: Node i get a leader eventually. Formally,

$$\nexists i : \square \text{CODRelect}_i = \text{FALSE} \quad (4)$$

Argument: Assume the contrary.

$$\exists i : \square \text{CODRelect}_i = \text{FALSE} \quad (5)$$

Equation (5) can hold only in any of the following four situations:

Case 1: Node i never receives CM, although, it had received EM. This case is handled like Lemma 1: Case 1 of ELFA (Section4.4).

Case 2: Node i does not receive any message from any node. Formally,

$$\begin{aligned} \exists i, j \in \mathcal{N}, i \neq j : j \in \text{neighbor_name}_i \wedge \\ \neg(\text{RECV}_i(\text{CM}, j) \vee \text{RECV}_i(\text{NM}, j) \\ \vee \text{RECV}_i(\text{IM}, j) \vee \text{RECV}_i(\text{heartbeat}, j)) \end{aligned} \quad (6)$$

This situation can be interpreted as node i is isolated due to network partition. Formally,

$$i, j \in \mathcal{N}, i \neq j : \forall j \notin \text{neighbor_name}_i \quad (7)$$

However, this situation cannot remain true for infinitely long time due to Lemma 1. Therefore, equation (5) and (7) cannot hold for ever.

Case 3: The protocol could not elect leader forever. Once again, this case can be proved like Lemma 1: Case 3 of ELFA (Section4.4).

Case 4: If any new joining node i joins the system, it will get the leader eventually. Formally,

$$\forall \mathcal{N}_x, \exists j \in \mathcal{N}, \exists i, i \neq j : i \in \text{neighbor_name}_j \wedge \text{SEND}_i(\text{RM}, j) \Rightarrow \diamond (\text{RECV}_i(\text{NM}, j) \wedge \text{RECV}_i(\text{IM}, L)) \quad (8)$$

Argument: Assume the contrary, that node i has not received NM forever. Formally,

$$\forall \mathcal{N}_x, \exists j \in \mathcal{N}, \exists i, i \neq j : i \in \text{neighbor_name}_j \wedge \text{SEND}_i(\text{RM}, j) \Rightarrow \square \neg (\text{RECV}_i(\text{NM}, j) \wedge \text{RECV}_i(\text{IM}, L)) \quad (9)$$

Equation (9) can hold only in any of the following two scenarios:

Scenario 1: Assume that node i got isolated after sending RM. Therefore, according to Lemma 1, node i elects itself as leader, iff having NW greater than 70.

Scenario 2: Again, assume that node i has moved from its location after sending RM and becomes member of another MANET \mathcal{N}_z . Now, node i again sends RM to its neighbor. However, if node i does not go out of range of its neighbor j then eventually, node j , receives the RM. Say, node j is also newly arrived node, it may also not have the leader information; thus, it sends WM to node i . However, node j already sends the RM to its neighbor, say, p . Node p again sends WM to node j , if node p is also new joining node. This may form a waiting chain; however, it would have the finite length. Now, any node z in the waiting chain may either has ID of L or may itself be L , hence, node z will send the ID of L to node p . Once this occurs, there is a chain of NM and IM that propagates down to node i . Therefore, the new joining node eventually becomes aware about the ID of current leader.

Therefore, all these four cases contradict our assumption in equation (5), hence, Lemma 3 holds. ■

LEMMA 4: *In the event of leader crash, DELFA delivers alternate leader within finite time. Formally,*

$$\forall i \in \mathcal{N}, i \in \mathcal{N}_x: \neg L(\mathcal{N}_x) \Rightarrow \diamond L(\mathcal{N}_x) \quad (10)$$

Argument: Assume the contrary.

$$\forall i \in \mathcal{N}, i \in \mathcal{N}_x: \neg L(\mathcal{N}_x) \Rightarrow \diamond \neg L(\mathcal{N}_x) \quad (11)$$

Equation (11) can be true in either of following two cases:

Case 1: In the first case assume that VL , VP and P have already crashed before crash of leader, and there is no such special node to handle this situation. However, according to our assumption, P is the best NW node and remains stay after single fault, i.e., L crash. When P does not receive subsequent heartbeat from L , it becomes leader-in-charge and initiates new leader election in UH to elect tentative L , VL and VP . Hence, this case contradicts our assumption in Equation (11).

Case 2: Assume that, crash of L leads to worst case network partitioning, refer Figure 5.7 (h). In this scenario, VL takes the charge of crashed leader like previous case within finite time. In addition, another component has no right to initiate the election due to limited NW. On the other hand, both components will get the leader-in-charge within finite time, in average case. Thus, assumption, in equation (11), is also false.

Therefore, new leader-in-charge becomes leader within finite time and system leads to fault-free scenario. ■

LEMMA 5: *Starting from multiple faults, i.e., crash of L , VL , VP and LH together, DELFA delivers alternate leader within finite time. Formally,*

$$\forall i \in \mathcal{N}, i \in \mathcal{N}_x: \neg(L(\mathcal{N}_x) \vee VL(\mathcal{N}_x) \vee LH(\mathcal{N}_x) \vee VP(\mathcal{N}_x)) \Rightarrow \diamond L(\mathcal{N}_x) \quad (12)$$

Argument: Assume the contrary.

$$\forall i \in \mathcal{N}, i \in \mathcal{N}_x: \neg(L(\mathcal{N}_x) \vee VL(\mathcal{N}_x) \vee LH(\mathcal{N}_x) \vee VP(\mathcal{N}_x)) \Rightarrow \square \neg L(\mathcal{N}_x) \quad (13)$$

Equation (13) can be true in only when President P of the system also get crashed with L , VL , VP and LH or P is already crashed. However, P , being the best node, never crashes as per assumption in our protocol. Hence, P

is always available to coordinate in uncoordinated MANET. Moreover, the crash of LH also does not affect P and P initiates election in UH . Therefore, equation (13) cannot hold forever. ■

LEMMA 6: *DELFA terminates after exchanging finite number of messages.*

Argument: As stated in Section 6, DELFA is based on MELFA and it employs some modified and augmented phases of MELFA to elect special nodes. Since, MELFA uses timeout mechanism, its termination is guaranteed. Nevertheless, DELFA also terminates eventually. It is obvious from the following operational semantics.

The joining of new node i leads to N RMs, N UMs and N NMs, in addition to single CAB message (*Event 3, 4*). Furthermore, *Event 6* leads to $3(K-1)$ or $3(M-1)$ EMs and equal number of AMs, CMs messages with single CAB. *Event 8* requires K HRMs in worst case. Moreover, *Event 11, 12* and *13* requires only single PCM, APCM, IM, NM and at most M IMs. The network partitioning and merging may also lead to at most $M+K$ initiations of election. Therefore, DELFA requires maximum two CABs, $(N+2+M)$ IMs, single NoM, PCM, APCM, N RM and equal number of UM, NM, in addition, $9(K-1)$ EMs with equal number of AMs and CMs are required. The above message count is for the worst case, nevertheless, it is finite. Therefore, termination of DELFA is also guaranteed. ■

THEOREM 1: *DELFA achieves self-stabilization.*

Proof. Self-stabilize algorithms follows closure (*i.e.*, system once in legal configuration, continues in legal configuration unless failure) and convergence (*i.e.*, regardless of initial state, the system eventually returns to legal configuration) properties. DELFA protocol also follows closure (once a leader is elected, it continues to be leader till it fails) and convergence (starting from any initial condition, DELFA provides a leader to the system, eventually) properties. This is direct implication of Lemma 1, Lemma 2 and Lemma 3. ■

THEOREM 2: *DELFA guarantees safety.*

Proof. It also follows directly from Lemma 1 and Lemma 2. ■

THEOREM 3: *DELFA guarantees liveness.*

Proof. It is ensured due to Lemma 3. ■

THEOREM 4: *DELFA has fault-containment.*

Proof. Fault-containment property masks the non-faulty nodes from becoming faulty. On the crash of existing leader, DELFA provides alternate leader without bringing the system down to initial state. The fault-containment property of DELFA is direct implication of Lemma 4 and Lemma 5. ■

5.5 MESSAGE COMPLEXITY

In following section, we consider message complexity of DELFA.

5.5.1 Best Case:

In the best case, the joining of new node in the pre-established coordinated MANET, results in minimum number of message (RM, NM and UM) exchanges. It can be understood with the following two scenarios:

Case 1: Assume that, in a one dimensional line network, a new joining node i becomes the neighbor of L . This scenario leads to reception of RM at L and IM at new joining node i . Hence, only two messages (*i.e.*, 1 RM and 1 IM) are exchanged to allow new joining node i to access the services of current leader L in the system, see Figure 5.8.



Figure 5.8: Best Case Scenario 1 of DELFA. New joining node is neighbor of leader.

Case 2: Again, assume that, in a one dimensional line network, a new joining i node becomes the neighbor of some ordinary node j . Soon after that, node j receives RM and it sends NM and UM. Now, it is possible that node j is not neighbor of L , however, we have already assumed that the cost of UM flow is one. Reception of UM at L leads to propagation of an IM which also has unit cost. Hence, this scenario leads to exchange of only four messages (*i.e.*, 1 RM, 1 NM, 1UM and 1 IM) to join a newly arrived node i , see Figure 5.9.

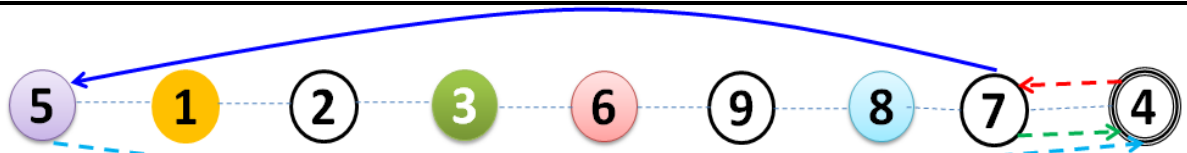


Figure 5.9: Best Case Scenario 2 of DELFA. New joining node is neighbor of some ordinary node.

Hence, DELFA uses either 2 or 4 messages, in the best case.

5.5.2 Average Case:

In average case, joining of new node i and leader crash have been considered two different cases in DELFA as follows:

Case 1: Assume that, new joining node i is the neighbor of all nodes in pre-existing MANET, see Figure 5.10. In this scenario, newly arrived node i will send RM to its one neighbor and wait for timeout. If node i does not receives IM before timer expires, it again sends RM to another neighbor. Consider that, node i has not received IM from all $N-1$ nodes and finally, receives it from N^{th} nodes. Thus, in this case total $N+1$ message (*i.e.*, N RMs, 1 NM or 1 UM) will be generated.

Case 2: In MANET, L may crash at any instant. This event will initiate election in LH (or UH) using CAB, see Figure 5.11. Subsequently, the lower ID LHM i sends EM to all $K-1$ nodes. The reception of EM at all $LHMs$ causes forwarding of AM intended to node i . Afterwards, node i will send CM to all $K-1$ nodes. Thus, in this case, the number of messages would be total $3(K-1) + 1$ (i.e., $k-1$ EMs, $k-1$ AM, $k-1$ CM and 1 CAB).

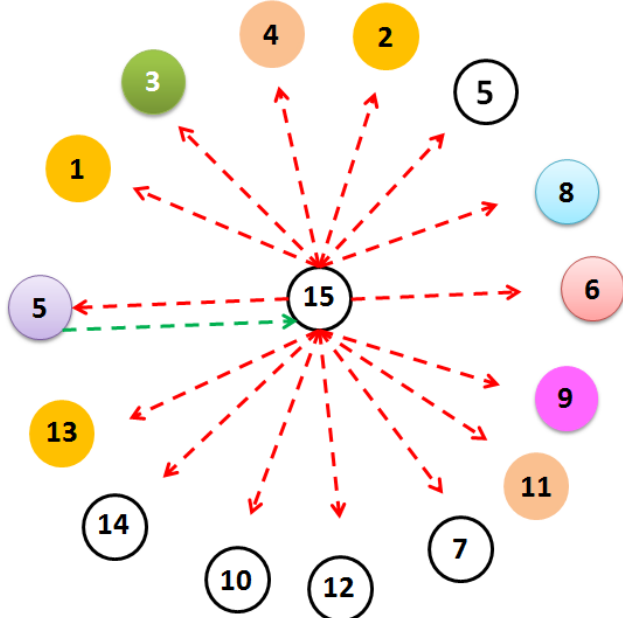


Figure 5.10: Average Case Scenario 1 of DELFA. New joining node is neighbor of leader.

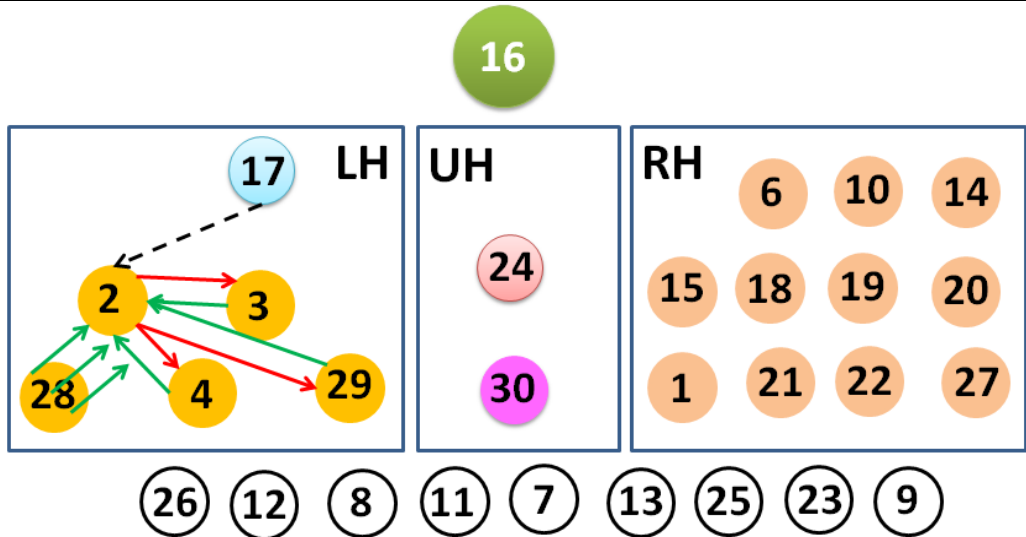


Figure 5.11: Average Case Scenario 2 of DELFA. Leader got crashed.

5.5.3 Worst Case:

In the worst case, L , VL , LH and VP crashes together, however, the best node of system, i.e., P , sends CAB to lowest ID UHM , see Figure 5.12. In addition, P also sends IM to all M $UHMs$. The reception of CAB initiates EM forwarding to all $M-1$ $UHMs$, similar to above case 2 in average case. Moreover, EM causes $M-1$ AMs and

$M-I$ CMs messages. Therefore, worst case leads to total $4(M-I) + 1$ (i.e., $M-I$ IM, $M-I$ EM, $M-I$ AM, $M-I$ CM and 1 CAB) messages.

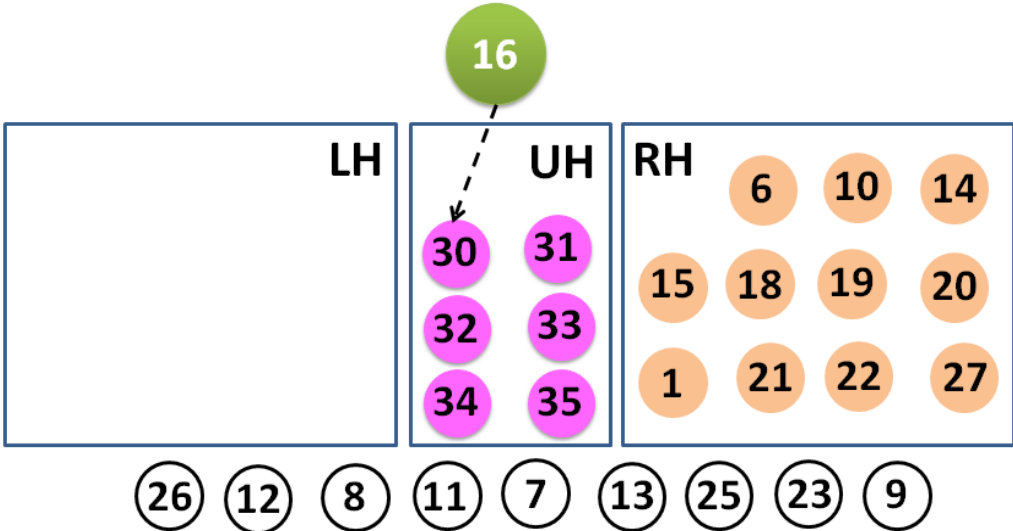


Figure 5.12: Worst Case of DELFA. Crash of leader, lower house and vice-leader.

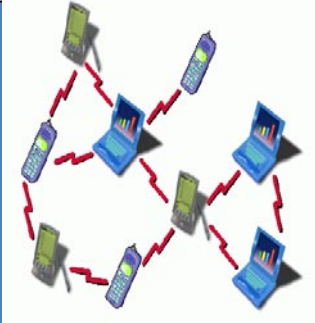
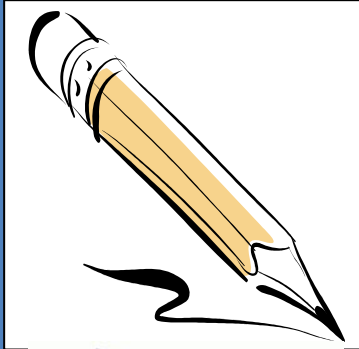
5.6 DISCUSSION

In this chapter, we have developed a fault tolerant and highly available leader election protocol. The concept of Lower House and Upper House, thought used in parliamentary system that makes the protocol more failure resilient. Furthermore, nodes which have facility to enhance their NW, they have brighter chance to become L , VL , VP and P which enhances stability of the system. Also, DELFA uses MELFA, to elect L , VL , P and VP in minimum number of control message exchange. There is no fixed criterion to decide the size of LH , i.e., K . Nevertheless, size of LH must increase on very significant increase in total number of mobile nodes; in addition, the total number of $LHMs$ in the MANET and their susceptibility to failure may be considered one of the critical parameters to decide the size of LH . Therefore, there is no method to find the optimal size of Lower House (LH). The appropriate LH size estimation is an optimization challenges in this protocol.

CHAPTER 6

SIMULATED

RESULTS



“Elegance is not a dispensable luxury but a factor that decides between success and failure”

- Edsger Wybe Dijkstra

In this chapter, we present our simulated results based on NS-2 network simulator which compares the performance of four protocols, AEFA, MELFA, ELFA and DELFA, on the various parameters, *e.g.*, election latency, control message flow in election, node density, failure rate of leader, etc.

We have assumed the nodes randomly distributed in a 500×500 meter² simulation area and we assume the number of nodes varying from 50 to 250. In addition, we suppose maximum message propagation time between any two nodes is 0.002 second and transmission range of each node is 250 meters. Leader node periodically, after every 90 seconds, sends heartbeat message to other nodes in the area. Furthermore, the non-receipt of heartbeat message, for 190 seconds, at some node triggers it to start election protocol. It is also a noting point here interval for heartbeat message and absence of heartbeat message can be set according to application requirements. Finally, each point on the graph has been plotted by taking the average values of 10 simulations runs.

1. Impact of Number of Nodes (Number of Nodes v/s EMs Exchange):

In order to curve these parameters, we place 50-250 nodes randomly in 500×500 meter². In addition, we consider only two protocols namely AEFA and MELFA under this circumstance. The resultant graph is shown Figure 6.1 and Figure 6.2. Figure 6.1 represents average case plot of AEFA and MELFA for number of EMs, where election process held within all or subset of nodes. Initially, graph points A_1 and A_2 are close to each other which show that the performance of both the protocols is comparable, when there are less number of nodes in the network. Afterwards, both curve rise; however, the gradient of the plot for MELFA is significantly less as compared to AEFA. The reason behind the difference in slop of these curves is the piggybacking and multicast techniques used in MELFA whereas AEFA uses broadcast. In Figure 6.2, we consider an optimistic scenario where all nodes are neighbor of each other. This situation is equivalent to best case of MELFA (Case 2, refer Section 3.5.1); however, the plot of AEFA still shows high gradient as the protocol is not able to derive advantage from this favorable situation.

2. Impact of Number of Nodes (Number of Nodes v/s Election Latency):

Similar to previous graph (Figure 6.2), we place 50-250 nodes randomly in 500×500 meter² and consider only AEFA and MELFA. The graph in Figure 6.3 and Figure 6.4 shows the number of node versus election latency which is, in fact, relative execution time of MELFA and AEFA, under two conditions: (i) all nodes are neighbors of each other; (ii) protocol execution in all or subset of nodes, respectively. Here also, the performance of MELFA is considerable better than AEFA.

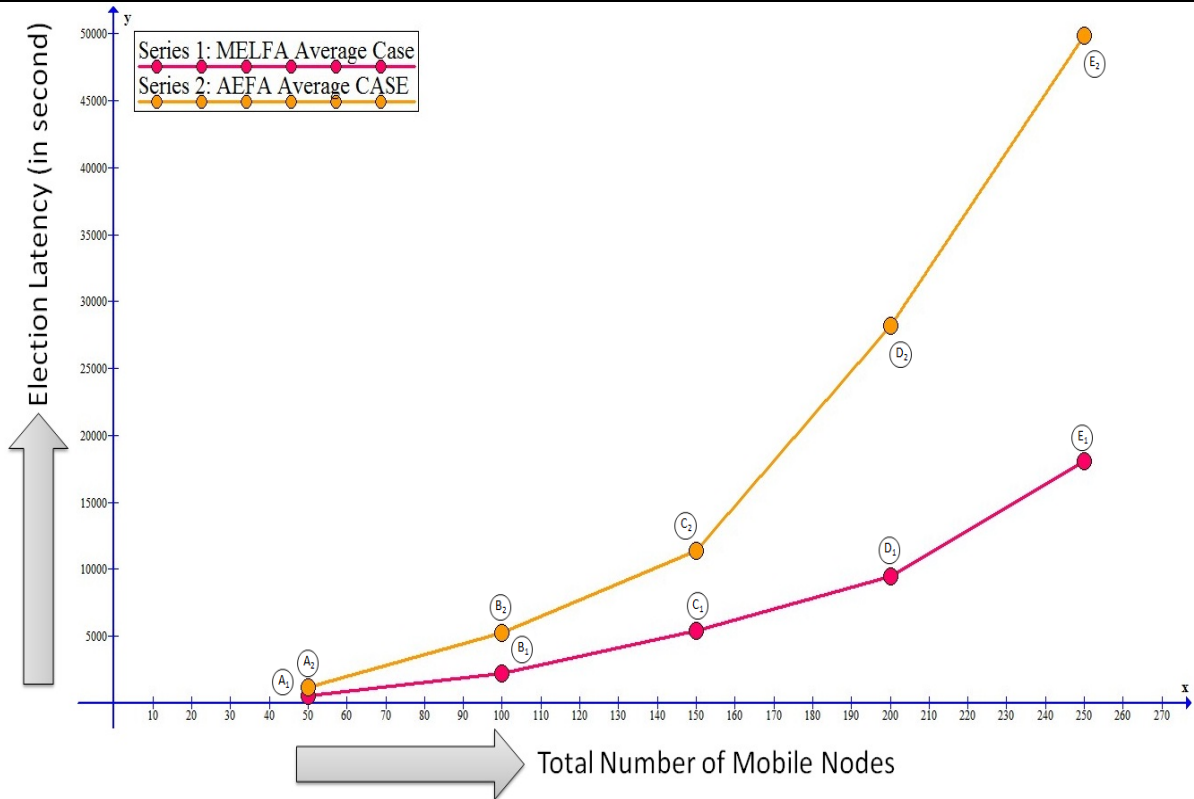


Figure 6.1: MELFA and AEFA - Number of Nodes v/s EMs Exchange, Case 1. All nodes are neighbors of each other.

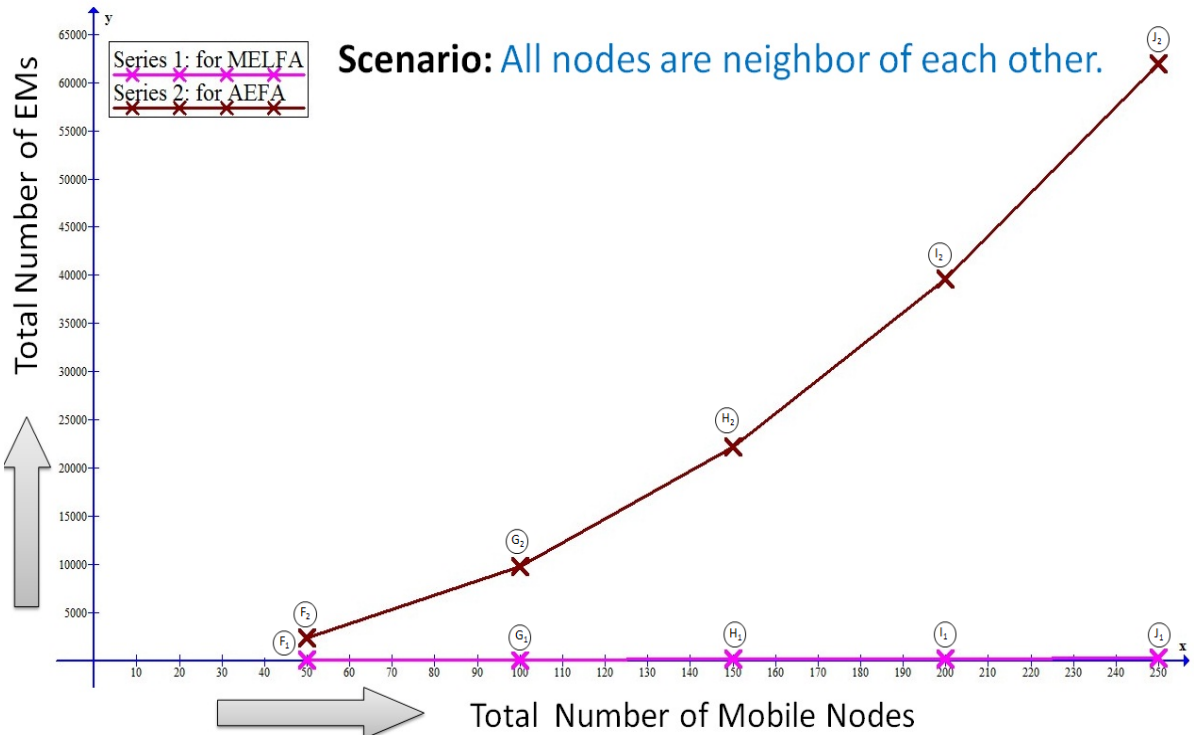


Figure 6.2: MELFA and AEFA - Number of Nodes v/s EMs Exchange, Case 2. All nodes are neighbors of each other.

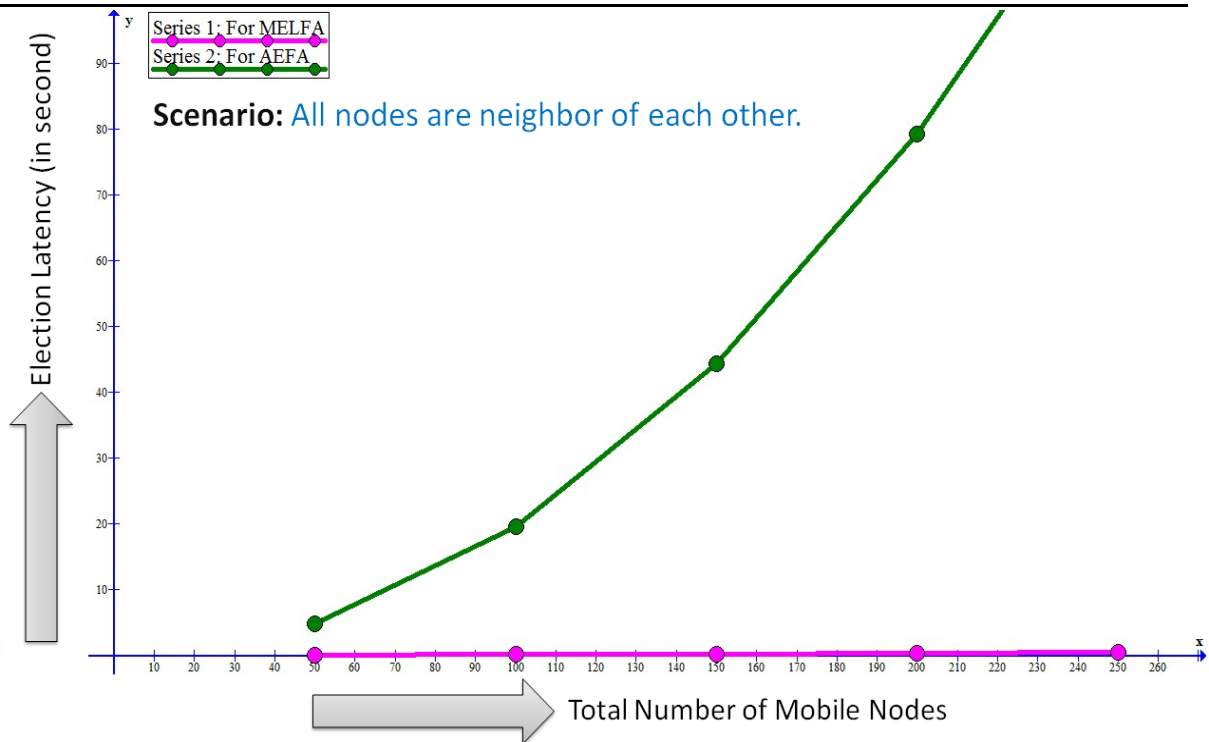


Figure 6.3: MELFA and AEFA - Number of Nodes v/s Election Latency, Case 1. All nodes are neighbors of each other.

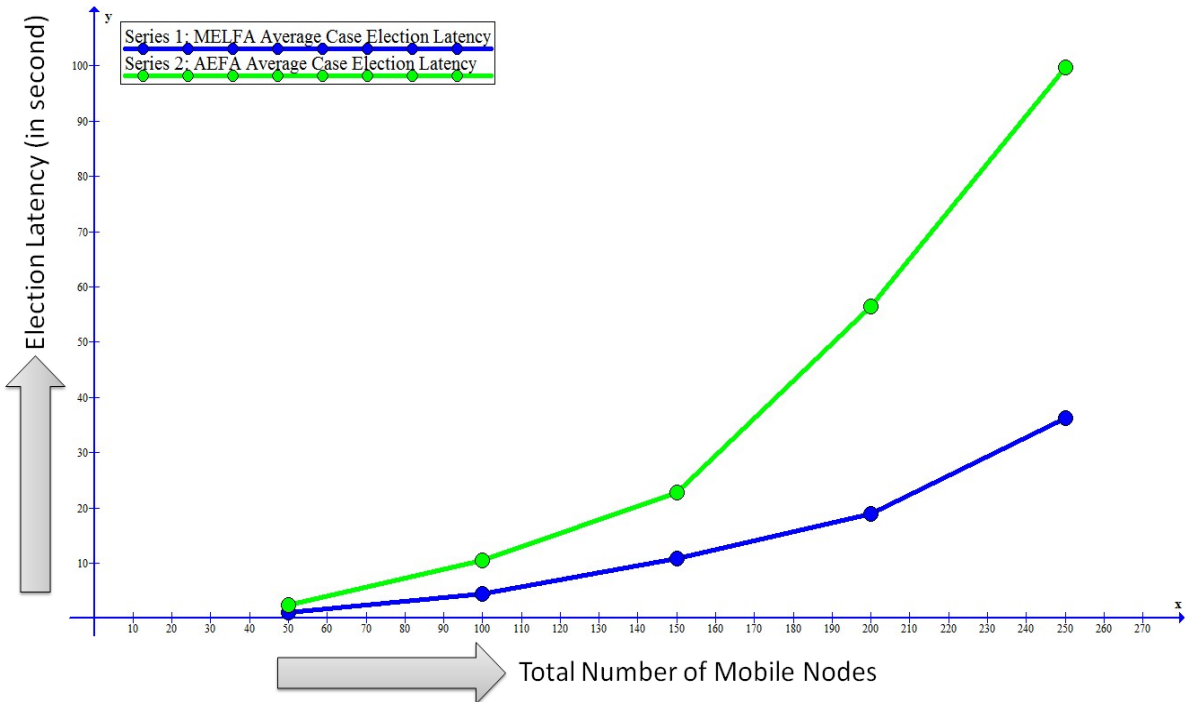


Figure 6.4: MELFA and AEFA - Number of Nodes v/s Election Latency, Case 2. Average case of MELFA.

3. Node Density (Number of Nodes v/s Election Latency):

In order to examine the impact of node density in AEFA and MELFA, we use 250×250 meter² simulation area for 100-700 nodes under the assumption that all nodes are not the neighbors of each other. In the graphs, Figure 6.5, we observe that the election latency of AEFA is much higher than MELFA. However, curve of MELFA has some critical points like A₁, B₁, C₁ and D₁. At point A₁, election latency of MELFA is lower for 100 nodes, and then increases for 200 nodes, refer graph point B₁. Again, election latency of MELFA decreases for graph point C₁, where the total number of nodes are 300 in the simulation area. This represents the encroachment of node compactness. When only 100 nodes are there, election latency is low due to small number of nodes. Subsequently, as the number of nodes increases up to 200, election latency also increases. However, at graph point C₁, election latency decreases because, due to higher node density, all nodes become the neighbors of each others. On the other hand, when the number of node lies between 400 to 700, election latency increases due to increase in number of EMs; though all the nodes are still neighbors of each other.

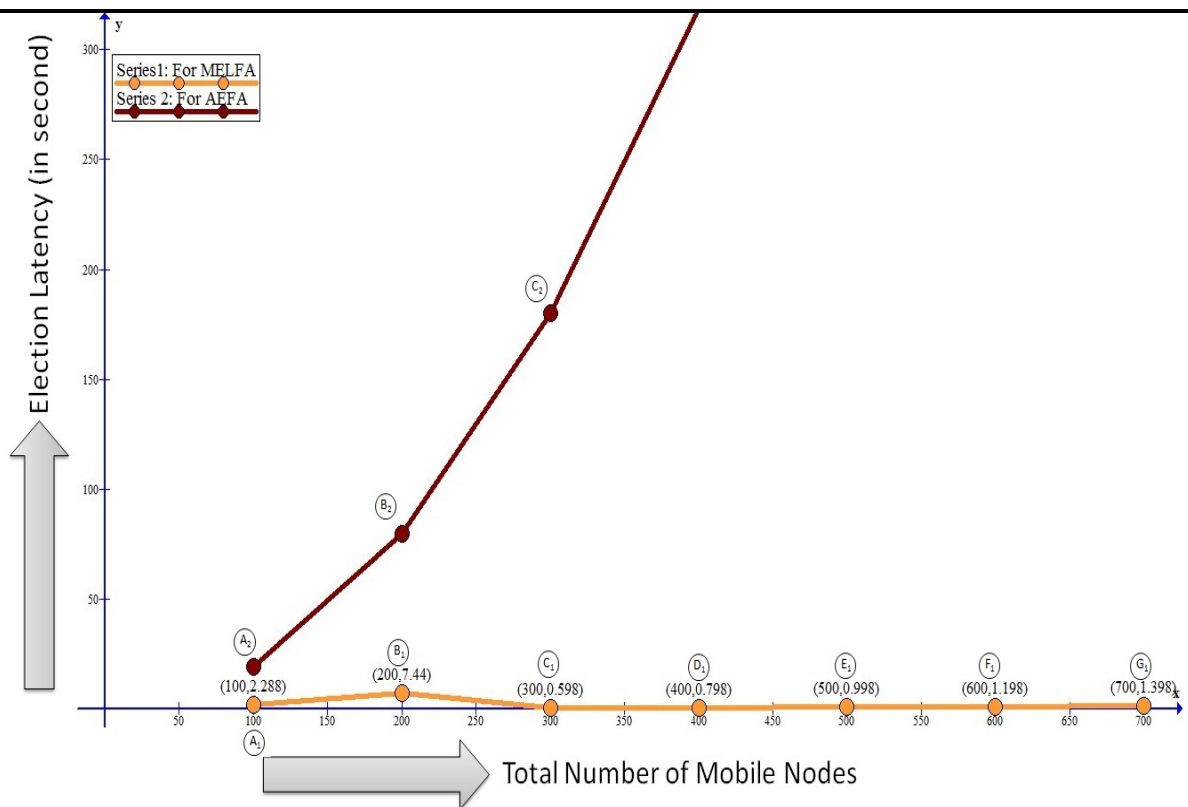


Figure 6.5: MELFA and AEFA - Node Density.

4. Impact of Protocol (Number of Nodes v/s Election Message):

In order to analyze the comparison between MELFA, ELFA and DELFA, we place 100-1000 nodes in 500×500 meter² simulation area. Here, we consider best, average and worst case of all these protocols, refer Figure 6.6, Figure 6.7 and Figure 6.8. Analysis of average case for ELFA and DELFA is given with analysis of Figure 6.9.

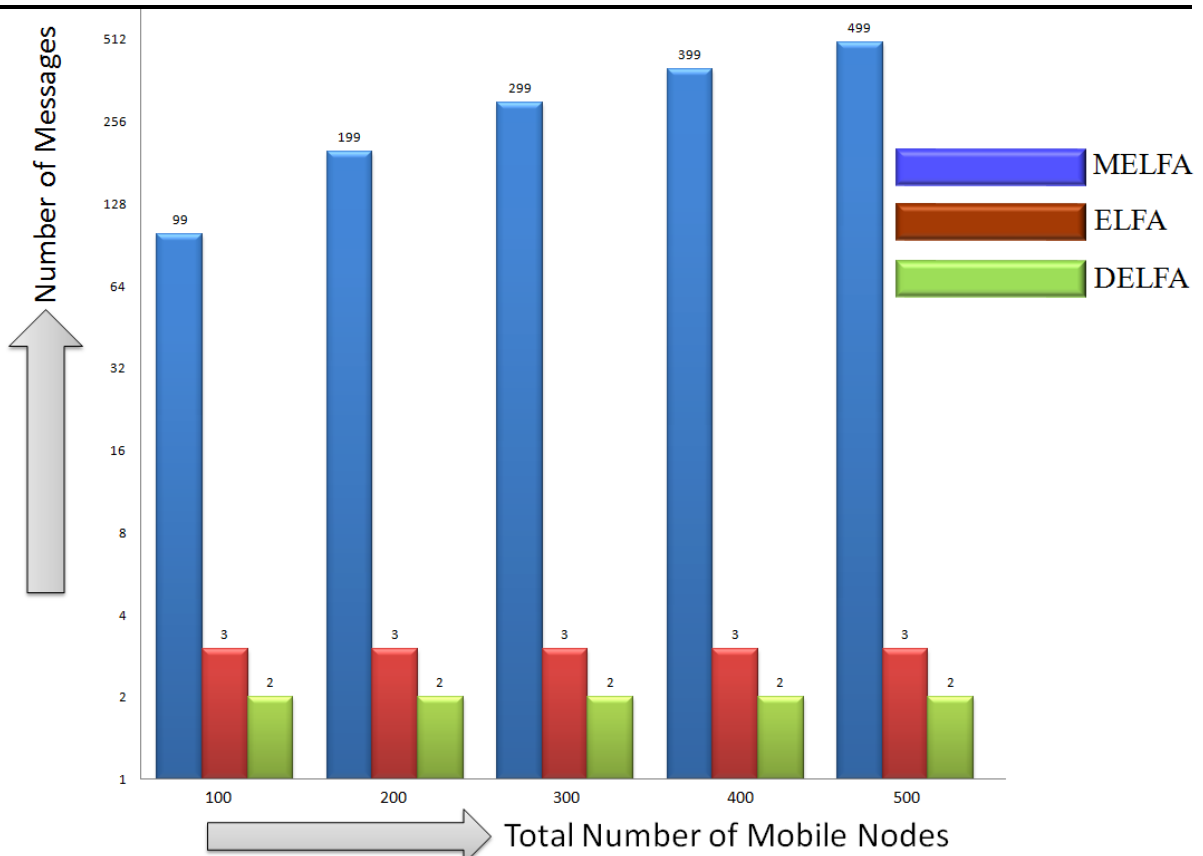


Figure 6.6: Best case of MELFA, ELFA and DELFA.

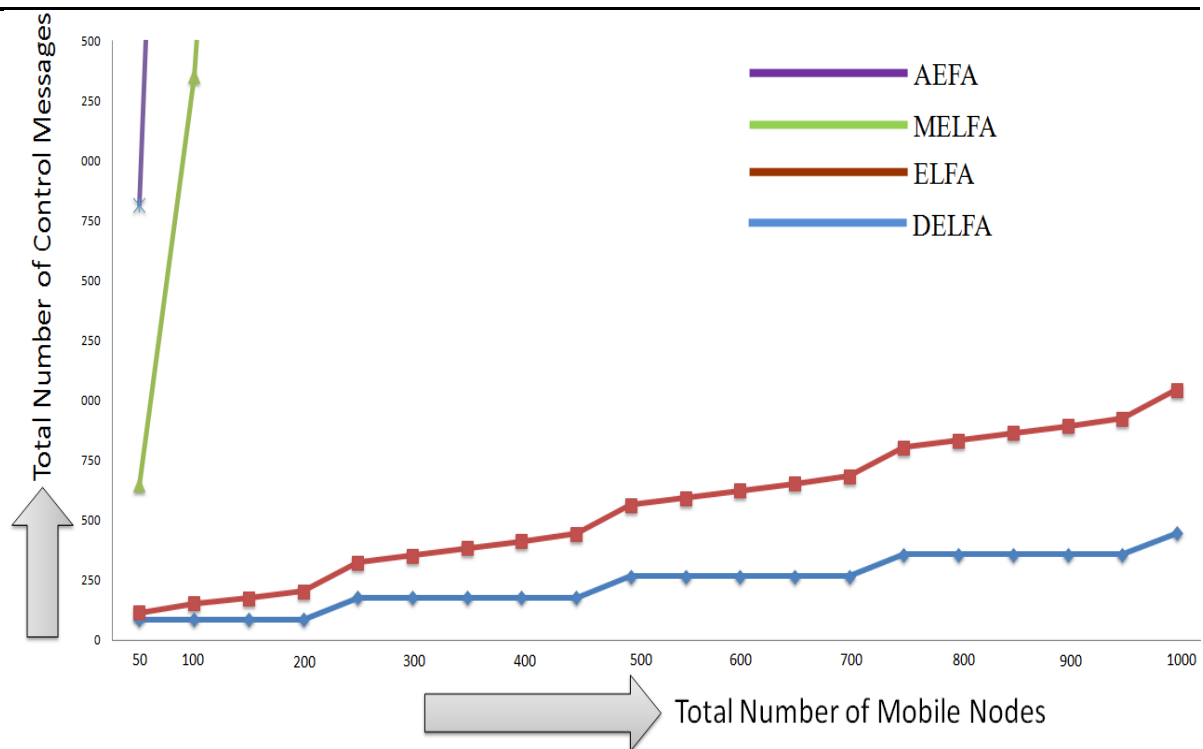


Figure 6.7: Average case of MELFA, ELFA and DELFA.

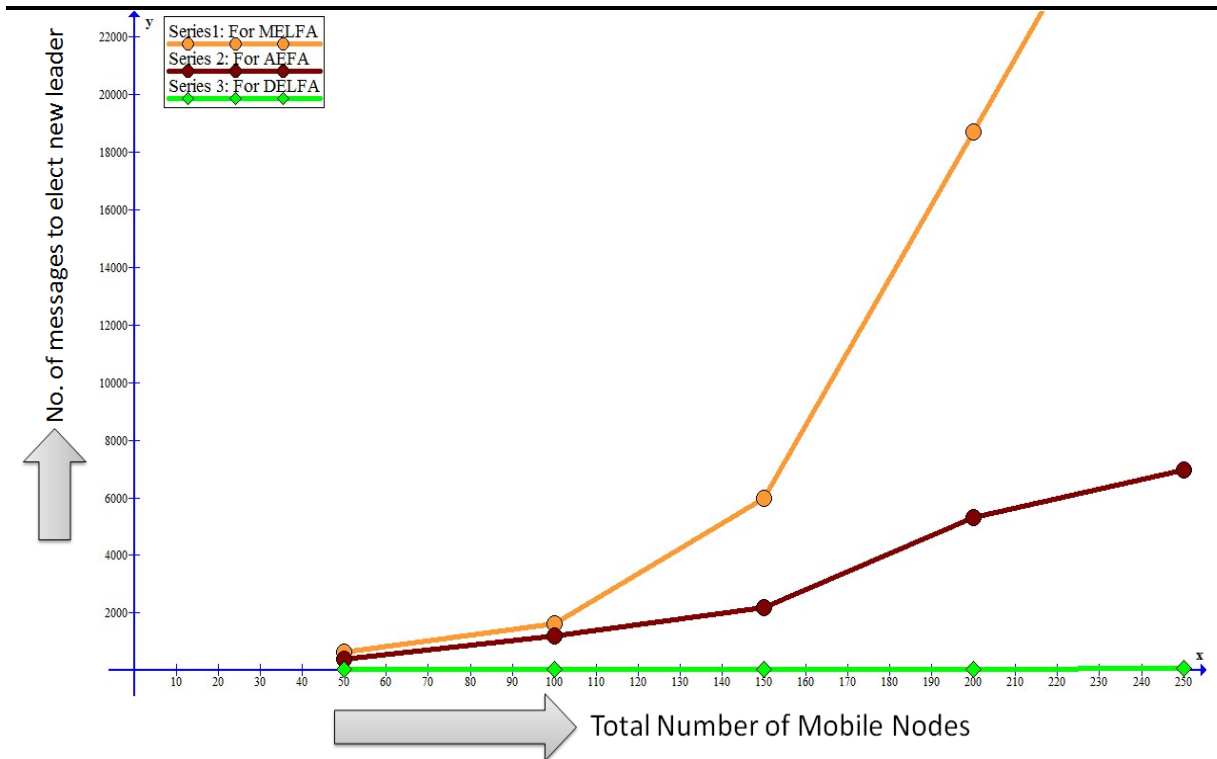


Figure 6.8: Worst Case of MELFA, ELFA and DELFA.

5. Impact of House Concept (Number of Nodes v/s Messages Exchange):

Under this section, we analyze the impact of Lower House concept. We consider our both fault tolerant protocols. Here, we consider total number of messages to elect new leader *i.e.*, EM, AM and CM. Although, DELFA and ELFA provides availability of leader in the event of coordinator crash, the graph, in Figure 6.9 and Figure 6.10, represents total number of messages exchange to elect new vice-leader.

Figure 6.9 demonstrates total number of messages exchange in the presence of leader, in average case, where we consider network partition too. Further, we assume that the cabinet size should increase with every significant increase in the number of nodes, in context of ELFA. The graph reflects the impact of intromission of cabinet concept in MELFA, which results in very less message transmission to elect new leader. In addition, constant increase in cabinet size results a liner graph. On the other hand, Lower House size should step-up at a very prominent increment of total nodes in the system; refer graph points A₁, B₁, C₁, D₁ and E₁. The graph comprises two critical points namely A₂ and B₂. On these two points, curve of ELFA and DELFA cuts together. In addition, noticeable point is that curve of ELFA up to A₂ is falling down than DELFA and after point B₂, ELFA curve is always above the DELFA curve. The main reason behind this inadequate curve as follow: initially, at the point A₁, there are only 50 nodes in the system. According the concept of ELFA, now cabinet size should be lower unlike DEFLA where lower house size should be greater even in the presence of small number of nodes. As the node will be 200, DELFA operate better than ELFA because now cabinet size

of ELFA increases with regular increment of nodes. The same reason lies for the point B₂ and both curve after point B₂.

Figure 6.10 shows the worst case of both protocols where a leader exists in DELFA, but, not in ELFA. Thus, in ELFA, a leader is absent for certain time, refer Figure 6.11. In fact, the graph, in Figure 6.11, shows higher availability of leader in DELFA protocol (we neglect the time of heartbeat absence, *i.e.*, 190 seconds). The graph, in Figure 6.11, holds a critical point A₁. After the point A₁, curve of ELFA goes down. The fact backside is as: at point A₂, total number of nodes in the system is 450. Whenever, cabinet crashes in ELFA, in the worst case, it leads to a drastic reduction in total number of nodes in the system; hence, to get the new leader as well as new cabinet MELFA have to be executed in a very less number of nodes unlike point A₁. The same argue is sufficient for the point A₃.

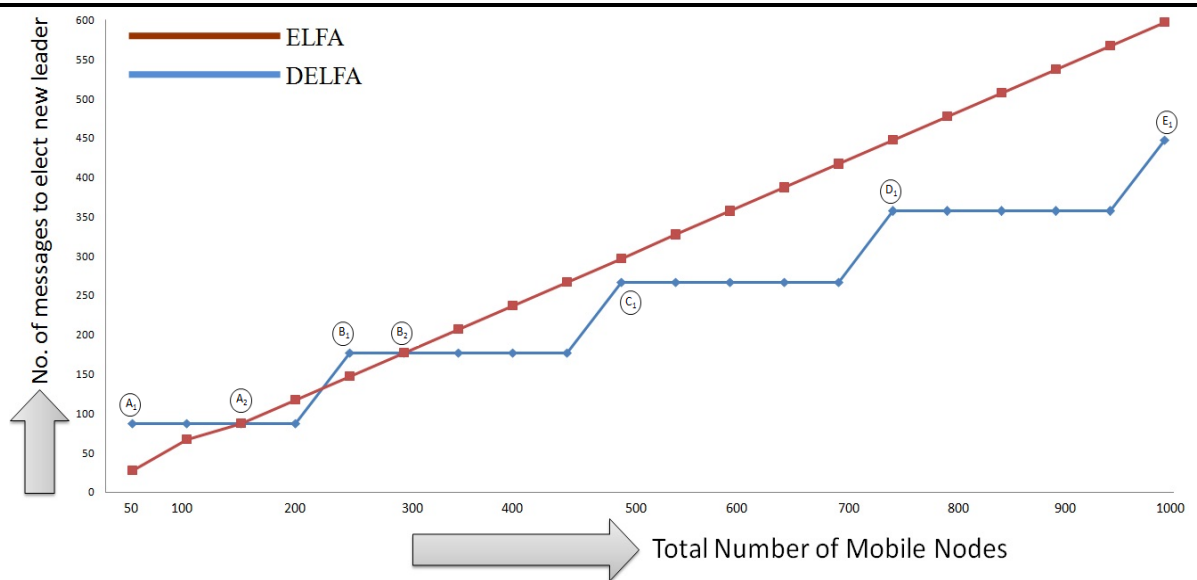


Figure 6.9: Election Message Exchange to Elect Tentative Leader in Average Case of ELFA and DELFA.

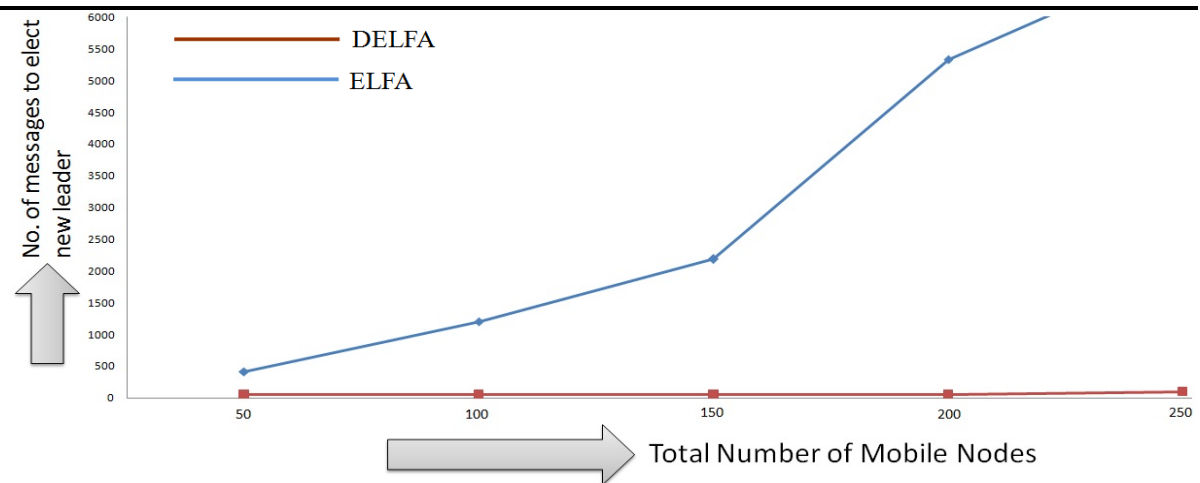


Figure 6.10: Election Message Exchange to Elect Tentative Leader in Worst Case of ELFA and DELFA.

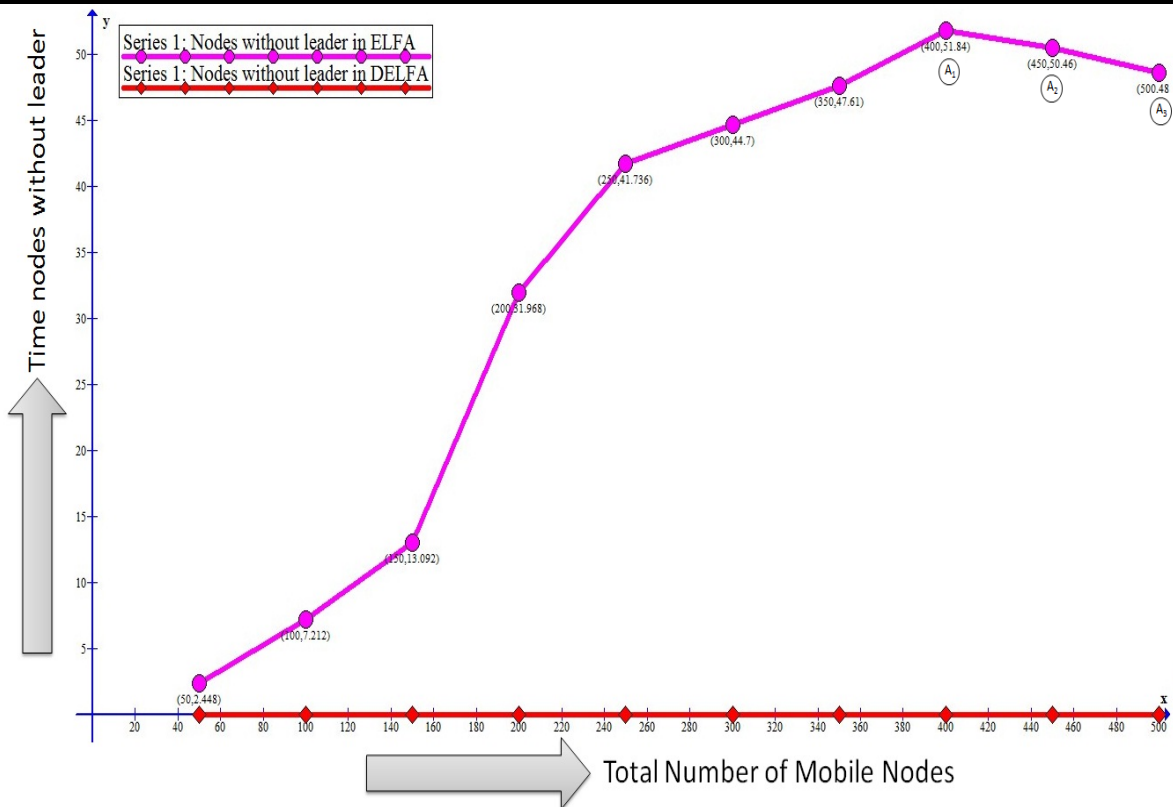


Figure 6.11: Nodes in ELFA without Leader. In this case, DELFA shows higher availability of leader (0 second to elect a leader).

In summary, we have analyzed following points:

1. Our non-fault tolerance protocol, MELFA, is much better than similar class protocol, AEFA, in term of very less control message exchanges, time to elect new leader and node density.
2. ELFA and DELFA is again much better than MELFA in term of minimum number of control message exchanges and time to elect new leader.
3. Finally, DELFA is very much better than ELFA. Graph represents that DELFA provides higher availability of leader. Furthermore, DELFA floods minimum messages to elect new vice-leader in presence of vice-leader-in-charge unlike ELFA, also in the absence of leader.


CHAPTER 7

STOPPING POINTS



“We are not certain, we are never certain. If we were we could reach some conclusions, and we could, at last, make others take us seriously”

- Albert Camus

 Coordinator election is basic to accomplish any application in static as well as dynamic environment. In addition, ad hoc nature of dynamic scenarios throws election problem more challenging. In the dissertation, we have developed and implemented three leader election protocols for mobile ad hoc network that satisfy three major requirements of the work, refer Section 2.3. In term of objective, we have successfully accomplished our primal target, *i.e.*, avoidance of broadcast storm without neighbor discovery. In fact, the dissertation attains following objectives:

1. Firstly, MELFA protocol not only greatly reduces the message overhead in leader election, it also reduces election latency. In fact, to the best of our knowledge, MELFA is the first protocol, since 2003, after the development of AEFA [21], which contributes a lot in leader election through extrema finding approach. Although, several protocols [23, 24, 25] have been developed after AEFA, most of them are incremental contribution only. In MELFA, we piggybacked some information, unlike AEFA, to decrease control message exchange. Furthermore, MELFA does not use Diffusion Computation, unlike AEFA.
2. Secondly, MELFA is not fault-tolerant. Thus, we developed another fault tolerant protocol. The second protocol, ELFA, provides a higher degree of failure resiliency. Again, the concept of elite nodes is novel and it fulfills safety property in each MANET components, unlike [24].
3. Finally, we have implemented entirely new concept of persisting parliament in leader election protocol for mobile ad hoc network. Adapting the similar terminology, we crate Lower House and Upper House with the existence of President, Leader, Vice-Leader and Vice-President which provide fault tolerance up to a saturation point like democratic system, where existence of executive is essential at all time to deal with affairs of state. More importantly, the protocol, DELFA, is a especially designed protocol for large scale MANETs.

Summarily, the protocol suite presented in the dissertation uses altogether novel design concept and to a large extent, it accomplished the objectives set in beginning of the dissertation.

BIBLIOGRAPHY



“Books are the carriers of civilization. Without books, history is silent, literature dumb, science crippled, thought and speculation at a standstill”

- Barbara W. Tuchman

1. G. LeLann, "Distributed Systems-Towards a Formal Approach," Information Processing 77, North-Holland Publishing Company, Amsterdam, pp. 155-160, 1977.
2. E. Chang and R. Robests, "An Improved Algorithm for Decentralized Extrema-Finding in Circular Configurations of Processes," Communication of ACM, vol. 22, no. 5, pp. 281-283, May 1979.
3. D. Hirschberg and J. Sinclair, "Decentralized Extrema-Finding in Circular Configurationsof Processors," Communication of ACM, vol. 23, no. 11, pp. 627-628, Nov. 1980.
4. G. Peterson, "An $O(n \log n)$ Unidirectional Algorithm for the Circular Extrema Problem," ACM Transactions on Programming Languages and Systems, vol. 4, no. 4, pp. 758-762, Oct. 1982.
5. S. Ghosh and A. Gupta, "An Exercise in Fault-Containment: Self-Stabilizing Leader Election," Information Processing Letters, vol. 59, pp. 281-288, 1996.
6. H. Garcia-Molina, "Elections in Distributed Computing System," IEEE Transaction Computer, vol. C-31, pp. 48-59, Jan. 1982.
7. M. Kordafshari, M. Gholipour, M. Jahanshahi, and A. Haghghat, "Two Novel Algorithms for Electing Coordinator in Distributed Systems based on Bully Algorithm," WSEAS Transactions on Systems, vol. 4, no. 1, pp. 10-15, January 2005.
8. M. EffatParvar, M. Effatparvar, A. Bemana, and M. Dehghan, "Determining a Central Controlling Processor with Fault Tolerant Method in Distributed System," In Proceeding of IEEE-International Conference on Information Technology (ITNG), pp. 658-663, 2007.
9. B. Awerbuch, "Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election and Related Problems," ACM, 1987.
10. S. Singh and J. Kurose, "Electing 'Good' Leaders," Journal of Parallel and Distributed Computing, vol. 21, pp. 184-201, 1994.
11. M. Yamashita and T. Kameda, "Leader Election Problem on Networks in which Processor Identity Numbers Are Not Distinct," IEEE Transaction on Parallel and Distributed Systems, vol. 10, no. 9, pp. 878-887, Sept. 1999.
12. C. Fetzer and F. Cristian, "A Highly Available Local Leader Election Service," IEEE Transaction on Software Engineering, vol. 25, no. 5, pp. 603-618, Sept. 1999.
13. M. Shirali, A. Toroghi, and M. Vojdani, "Leader Election Algorithms: History and Novel Schemes," In Proceeding IEEE 3rd International Conference on Convergence and Hybrid Information Technology (ICCIT), pp. 1001-1006, 2008.
14. M. Parvar, N. Parvar, M. Parvar, A. Dadlani, and A. Khonsari, "Improved Algorithms for Leader Election in Distributed Systems," In Proceeding IEEE 2nd International Conference on Computer Engineering and Technology (ICCET), pp. v2:6-v2:10, 2010.
15. N. Malpani, J. Welch, and N. Vaiadya, "Leader Election Algorithms for Mobile Ad Hoc Networks," In Proceeding 4th Workshop Discrete Algorithms and Methods for Mobile Computing and Communication, pp. 96-103, 2000.

16. A. Derhab and N. Badache, "A Self-Stabilizing Leader Election Algorithm in Highly Dynamic Ad Hoc Mobile Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 7, pp. 926-939, 2008.
17. R. Ingram, P. Shields, J. Walter, and J. Welch, "An asynchronous leader election algorithm for dynamic networks," In *Proceeding IEEE 23rd International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1-12, 2009.
18. P. Parvathipuram, V. Kumar, and G-C. Yang, "An Efficient Leader Election Algorithm for Mobile Ad Hoc Networks," In *Proceeding International Conference on Distributed Computing and Internet Technology (ICDCIT)*, pp. 32-41, 2004.
19. M. Shirmohammadi, K. Faez, and M. Chhardoli, "LELE: Leader Election with Load balancing Energy in Wireless Sensor Network," In *Proceeding IEEE International Conference on Communications and Mobile Computing (ICCMC)*, pp. 106-110, 2009.
20. M. Shirmohammadi, M. Chhardoli, and K. Faez, "CHEFC: Cluster Head Election with Full Coverage in Wireless Sensor Networks," In *Proceeding IEEE 9th Malaysia International Conference on Communications (ICC)*, pp. 780-784, 2009.
21. S. Vasudevan, B. DeCleene, N. Immerman, J. Kurose, and D. Towsley, "Leader Election Algorithms for Wireless Ad Hoc Networks," In *Proceeding IEEE DARPA Information Survivability Conference and Exposition (DISCEX)*, pp. 261-272, 2003.
22. S. Park, T. Lee, H-S. Seo, S-J. Kwon, and J-H. Han, "An Election Protocol in Mobile Ad Hoc Distributed Systems," In *Proceeding IEEE 6th International Conference on Information Technology: New Generation (ICITNG)*, pp. 628-633, 2009.
23. G. Zhang, X. Kuang, J. Chen, and Y. Zhang, "Design and Implementation of a Leader Election Algorithm in Hierarchy Mobile Ad hoc Network," In *Proceeding IEEE 4th International Conference on Computer Science & Education (ICCSE)*, pp. 263-268, 2009.
24. S. Lee, R. Muhammad, and C. Kim, "A Leader Election Algorithm within Candidates on AdHoc Mobile Networks," In *Proceeding of the 3rd International Conference on Embedded Software and Systems (ICISS)*, LNCS 4523, pp. 728-738, 2007.
25. Azzedine Boukerche and Kaouther Abrougui, "An Efficient Leader Election Protocol for Mobile Networks," In *Proceeding ACM International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1129-1134, 2006.
26. V. Park and M. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," In *Proceeding IEEE 16th International Conference on Computer Communications (INFOCOM)*, pp. 7-11, 1997.
27. Z. Haas and M. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," Internet Draft, draft-ietf-manet-zone-zrp-02.txt, June 1999.

28. M. J. Handy, M. Haase, and D. Timmermann, "Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection," In Proceeding IEEE 4th International Workshop on Mobile and Wireless Communications Network, pp. 368-372, 2002.
29. K. Hatzis, G. Pentaris, P. Spirakis, V. Tampakas, and R. Tan, "Fundamental Control Algorithms in Mobile Networks," In Processding ACM 11th Symposium on Parallelism in Algorithms and Architectures, pp. 251-260, 1999.
30. E. Gafni and D. Bertsekas, "Distributed Algorithms for Generating Loop-Free Routes in Networks with Frequently Changing Topology," IEEE Transactions on Communications, vol. C-29, no. 1, pp. 11-18, 1981.
31. D. Raz, Y. Shavitt, and L. Zhang, "Distributed Council Election," IEEE/ACM Transaction on Networking, vol. 12, no. 3, pp. 483-492, June 2004.
32. A. Boukerche and K. Abrougui, "An Efficient Leader Election Protocol for Wireless Quasi-Static Mesh Networks: Proof of Correctness," In Proceeding IEEE IEEE International Conference on Communication (ICC), pp. 3491-3496, 2007.
33. V. Raychoudhury, J. Cao, and W. Wu, "Top K-leader Election in Wireless Ad Hoc Networks," In Proceeding IEEE 17th International Conference on Computer Communications and Networks (ICCCN), pp. 87-92, 2008.
34. O. Dagdeviren and K. Erciyes, "A Hierarchical Leader Election Protocol for Mobile Ad Hoc Networks," In Proceeding 8th International Conference on Computational Science (ICCS), LNCS 5101, pp. 509-518, 2008.
35. M. Haddar, A. Kacem, Y. Metivier, M. Mosbah, and M. Jmaiel, "Electing a Leader in the Local Computation Model Using Mobile Agents," In Proceeding IEEE International Conference on Computer Systems and Applications (ICCSA), pp. 473-480, 2008.
36. E.W. Dijkstra, "Self-stabilizing Systems in Spite of Distributed Control," Communications of the ACM, vol. 17, no. 11, 1974.
37. S. Misra, I. Woungang, and S. Misra, Guide to Wireless Ad Hoc Networks. Springer-Verlag, London, 2009.
38. P. Davis and C. McGuffin, Wireless Local Area Networks, McGraw-Hill: New York, 1995.
39. B. Forouzan and S. Fegan, Data Communications and Networking, 4th ed., McGraw-Hill Forouzan Networking Series, 2007.
40. M. Goodchild, D. Johnston, D. Maguire, and V. Noronha, "Distributed and Mobile Computing," In A Research Agenda for Geographic Information Science. Boca Raton: CRC Press, pp. 257-286.
41. N. Qasim, F. Said, and H. Aghvami, "Mobile Ad Hoc Networking Protocols' Evaluation through Simulation for Quality of Service," IAENG International Journal of Computer Science, vol. 36, no. 1, Feb. 2009.

42. Roger Wattenhofer, Lecture notes on Mobile Computing. Distributed Computing Group, Summer 2004. Available at URL: "<http://www.disco.ethz.ch/lectures/ss04/mobicomp/lecture/1/Chapter1IntroductionOriginal.pdf>"
43. I. Chlamtac, and M. Conti, and J. Liu, "Mobile Ad hoc Networking: Imperatives and Challenges," Elsevier Ad Hoc Networks, vol. 1, pp. 13-64, 2003.
44. IBM (2008) Autonomic computing: The 8 Elements. Available at URL: "<http://researchweb.watson.ibm.com/autonomic/overview/elements.html>"
45. L. Gavrilovska and R. Prasad, Ad Hoc Networking Towards Seamless Communications. Springer, The Netherlands, 2006.
46. S. Yu, Y. Zhang, C. Song, and K. Chen, "A Security Architecture for Mobile Ad Hoc Networks," In ACM 2nd Workshop on Wireless Security (WiSE), San Diego, CA, September 19, 2003.
47. M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," In Proceeding ACM 15th Principles of Distributed Computing (PODC), pp. 1-7, 1996.
48. Y. Tseng, W. Liao, and S. Wu, Handbook of Wireless Networks and Mobile Computing. John Wiley & Sons, Inc., 2002.
49. M. Szczodrak and J. Kim, "4G and MANET, Wireless Network of Future Battlefield," In Proceeding International Conference on Security and Management, Las Vegas, Nevada, USA, June 25 ~ 28, 2007.
50. P. Saini and A. K. Singh, "A Primitive View Change for Fault Tolerant Agreement using Single Message Propagation," In Proceeding of International Conference on Advances in Information Technology and Mobile Communication (ICAIM), CCIS 147, pp. 507-512, April, 2011.

LIST OF PAPERS COMMUNICATED

Published

A. Singh and S. Sharma, "Message Efficient Leader Finding Algorithm for Mobile Ad Hoc Networks," in Proceeding of IEEE 3rd International Conference on Communication Systems and Networks (COMSNETS): Workshop on Intelligent Networks: Adaptation, Communication & Reconfiguration (IAMCOM), pages 1-6, Banglore, India, Jan 04-08, 2011.

Accepted

A. Singh and S. Sharma "Elite Leader Finding Algorithm for MANETs," in IEEE 10th International Symposium on Parallel and Distributed Computing (ISPDC), Cluj-Napoca, Romania.

S. Sharma and A. Singh, "Democratic Leader Finding Algorithm for Large Mobile Ad hoc Networks," in Proceeding of IEEE 31st International Conference on Distributed Computing Systems (ICDCS): Workshop on Wireless Ad hoc and Sensor Networks (WWASN), Minneapolis, USA, June 21-24.



© 2011

**National Institute of Technology
Kurukshetra - 136119**