A Distributed Hierarchical Energy-Efficient Scheme for Large Scale Mobile Wireless Ad Hoc Networks

Wasim El Hajj

Western Michigan University

Follow this and additional works at: http://scholarworks.wmich.edu/dissertations

Part of the Computer Sciences Commons

Recommended Citation


http://scholarworks.wmich.edu/dissertations/947
A DISTRIBUTED HIERARCHICAL ENERGY-EFFICIENT SCHEME FOR LARGE SCALE MOBILE WIRELESS AD HOC NETWORKS

by

Wasim El Hajj

A Dissertation
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Doctor of Philosophy
Department of Computer Science

Western Michigan University
Kalamazoo, Michigan
July 2006
In recent years, wireless networks have enjoyed tremendous development and popularity due to the technological advances of wireless radio devices. While interest in conventional wireless networks continues to evolve, new types of wireless networks, such as Mobile Ad Hoc Networks (MANET) and Wireless Sensor Networks (WSN) are evolving fast and receiving much attention from academia, industry, and government.

Scalability and energy-efficiency present two of the most important challenges in Mobile Ad Hoc Networks. Scalability in MANETs can be defined as the capability of the network to provide an acceptable throughput when the network size increases. Energy-efficiency can be defined as using less energy to deliver greater or equal amount of services. MANET nodes rely on portable, limited power resources and they bear the duty of relaying packets for other nodes. The failure of some nodes due to battery drainage might bring the whole network down.

In this dissertation, we propose clustering and routing approaches to overcome scalability and energy constraint problems that exist in large scale MANETs. To handle these problems, we design a hierarchical energy-efficient scheme that can be easily setup and maintained. The scheme has three major components, namely: (1) the hierarchical (clus-
tering) component, (2) the maintenance component, and (3) the routing component. Our research provides a mathematical formulation of the clustering component using integer linear programming - ILP. It also provides centralized and distributed energy-efficient schemes that achieve efficient clustering, efficient maintenance, and efficient routing in large scale MANETs. Our simulation study shows that our suggested approaches outperform well known schemes found in the literature.

The distributed energy-efficient scheme presents one of our main contributions. The distributed scheme builds and maintains a hierarchical network structure using low message and time complexity. Furthermore, it provides an intelligent path selection controller that can be easily incorporated in any existing link state routing protocol to select energy-efficient routes. The hierarchical structure increases scalability, but does not improve energy-efficiency. To achieve energy-efficiency, we propose three fuzzy logic controllers that aggregate multiple network metrics and then decide on the quality of the node/path. This quality is then used as an important parameter in the clustering, maintenance, and routing components. The first controller is used to achieve efficient hierarchical design, the second controller is used to achieve efficient network maintenance, and the third controller is used to achieve efficient routing.
ACKNOWLEDGMENTS

All prays be to God, Allah. Without His mercy and clemency nothing would have been possible.

First and foremost, I would like to express my gratitude to my advisor, Dr. Dionysios Kountanis, for his support, patience, and encouragement throughout my graduate studies. It is not often that one finds an advisor and colleague that always allocate the time for listening to the little problems and roadblocks that unavoidably appear in the course of performing research. His technical and editorial advice was essential to the completion of my dissertation and taught me innumerable lessons and insights while working on my academic research.

I am very grateful for the time and feedback my committee members provided. My special appreciation goes to Dr. Mohsen Guizani, Dr. Ala Al-Fuqaha, and Dr. Song Si. Their help and discussions have enlightened my research and taught me new views that guided me in countless situations.

I owe some colleagues and friends lots of thanks for their help and contribution. I would mention Ghassen Ben Brahim, Osama Awwad, and Mustafa Sanver. It is so good to know that such friends are available around you and ready for any help.

Finally, my special thanks and appreciation are due to my family for their emotional support, prayers, and love. My father and mother are overseas but were with me all along
the way and patient enough for my long-time delayed visit. My brother Hassan El-hajj, his wife, and lovely daughter opened their house and hearts for me. When I was over-tired and could not concentrate on my research any longer, their home was the sanctuary I visited in order to relax and recollect my energy. My uncles Mohsen and Hassan and their families deserve lots of appreciation for their encouragement and motivation at all times.

Wasim El Hajj
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ................................................................. ii

LIST OF TABLES ........................................................................ x

LIST OF FIGURES ....................................................................... xi

1 INTRODUCTION ................................................................. 1

  1.1 MANET Characteristics .................................................. 2

  1.2 MANET Challenges ....................................................... 5

  1.3 Problem Description ...................................................... 6

  1.4 Dissertation Organization .............................................. 10

2 FLAT ROUTING PROTOCOLS OVERVIEW ............................ 13

  2.1 Classification Criteria ...................................................... 13

     2.1.1 Link State (LS) vs. Distance Vector (DV) Routing Protocols ... 13

     2.1.2 Periodical vs. Event Driven Update Routing Protocols .......... 14

     2.1.3 Proactive vs. Reactive Routing Protocols ....................... 15

     2.1.4 Flat Structure vs. Hierarchical Structure ....................... 16

     2.1.5 Decentralized Computation vs. Distributed Computation ...... 17

     2.1.6 Source vs. Hop-by-hop Based Routing Protocols ............... 17

     2.1.7 Single vs. Multiple Paths Routing Protocols ................... 18

  2.2 Flat Routing Protocols Overview ...................................... 18

     2.2.1 Destination Sequenced Distance Vector Routing Protocol (DSDV) 19

     2.2.2 Ad Hoc On-demand Distance Vector Routing ................... 20
2.2.3 Wireless Routing Protocol ........................................ 21
2.2.4 Optimized Link State Routing ................................. 23
2.2.5 Dynamic Source Routing ...................................... 24
2.2.6 Temporally Ordered Routing Algorithm ..................... 26
2.2.7 Associativity Based Routing .................................. 26
2.2.8 Zone Routing Protocol ........................................ 28

3 HIERARCHICAL ROUTING PROTOCOLS OVERVIEW .................. 30

3.1 Dominating Set Based Routing Protocols ......................... 32
3.1.1 Das et al.’s Algorithms ....................................... 34
3.1.2 Wu and Li’s Algorithm ...................................... 37
3.1.3 Alzoubi et al.’s Algorithms .................................. 40
3.1.4 Cheng et al.’s Algorithms .................................... 41
3.1.5 Parthasarathy et al. Algorithm ............................... 42
3.1.6 Stojmenovic et al. Algorithm ................................. 43
3.2 Non-dominating Set Based Routing Protocols .................... 43
3.2.1 Lowest-ID Clustering (LID) .................................. 44
3.2.2 CLUSTERPOW, Tunneled CLUSTERPOW, and MINPOW .... 46
3.2.3 Decentralized Energy-Efficient Cluster Propagation (DEEP) ... 50

4 INTEGER LINEAR PROGRAMMING FORMULATION .................. 53

4.1 ILP Overview .................................................. 54
4.2 Energy-Efficient Clustering with Fully Connected Backbone (EEC-FCB) .................................................. 56
4.2.1 EEC-FCB Definition ............................................... 56
4.2.2 EEC-FCB ILP Formulation ......................................... 57
4.3 Energy-Efficient Clustering with Connected Backbone (EEC-CB) .... 61
4.4 Energy-Efficient Clustering with a Reliable Backbone (EEC-R) ...... 63
  4.4.1 EEC-R Definition .................................................. 65
  4.4.2 EEC-R ILP Formulation ........................................... 65
4.5 Performance Evaluation ................................................ 69
4.6 Summary ............................................................... 70
5 FUZZY LOGIC CONTROLLERS ........................................... 71
  5.1 Fuzzy Logic Overview .................................................. 73
  5.2 Network Setup Controller ............................................ 76
    5.2.1 Residual Energy Membership Functions ......................... 77
    5.2.2 Mobility Membership Functions .................................. 78
    5.2.3 Traffic Membership Functions .................................... 79
    5.2.4 Output Membership Functions ..................................... 80
    5.2.5 Network Setup Controller Rules ................................. 81
  5.3 Network Maintenance Controller ................................... 81
    5.3.1 Setup Quality Membership Functions ........................... 82
    5.3.2 Remaining Capacity Membership Functions ..................... 83
    5.3.3 Received Signal Strength Membership Functions ................ 83
    5.3.4 Output Membership Functions ................................... 84
    5.3.5 Network Maintenance Controller Rules .......................... 85
7.1.5 FDDS is Scalable and Energy-Efficient ............................. 125
7.2 FDDS Complexity Analysis ........................................... 126
  7.2.1 Best Case ............................................................. 127
  7.2.2 Worst Case ........................................................... 128
  7.2.3 Average Case ......................................................... 128
7.3 FDDS Theoretical Analysis ............................................ 133
7.4 Cluster Formation ....................................................... 137
7.5 Cluster Maintenance (FDDS-M) ..................................... 139
  7.5.1 Node Maintenance .................................................. 141
  7.5.2 CH Maintenance ..................................................... 146
  7.5.3 FDDS-M is Scalable and Energy-Efficient ...................... 152
7.6 Routing (FDDS-R) ....................................................... 153
  7.6.1 Routing Controller .................................................. 154
  7.6.2 OLSR Overview ..................................................... 155
  7.6.3 FDDS-R is Scalable and Energy-Efficient ...................... 156
7.7 Simulation Results ..................................................... 157
  7.7.1 Average Number of Hops ........................................ 157
  7.7.2 Network Stretch .................................................... 159
  7.7.3 Backbone Size ...................................................... 161
  7.7.4 Lifetime ............................................................... 163
  7.7.5 Lifetime Under Increasing Traffic Load ......................... 166
  7.7.6 Lifetime with and without Network Setup Controller ........ 167
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.7.7</td>
<td>Lifetime with and without <em>Routing</em> Controller</td>
<td>167</td>
</tr>
<tr>
<td>7.8</td>
<td>Summary</td>
<td>170</td>
</tr>
<tr>
<td>8</td>
<td>CONCLUSION AND FUTURE WORK</td>
<td>172</td>
</tr>
<tr>
<td></td>
<td>BIBLIOGRAPHY</td>
<td>180</td>
</tr>
</tbody>
</table>
LIST OF TABLES

6.1 Nodes’ power level properties ........................................ 106

7.1 Complexities of well known approaches suggested in literature........ 125

7.2 Message and time complexities of FDDS .......................... 127

7.3 Number of connected unlabeled graphs for $n = 1, 2, \ldots, 16$ .......... 131

7.4 Actions taken by node $i$ when itself or its parent CH $j$ move .......... 145

7.5 Message and time complexities of "node maintenance" across all network nodes......................................................... 145

7.6 Actions taken when CH $j$ moves .................................... 151

7.7 Message and time complexities of "CH maintenance" actions across all network nodes......................................................... 152
LIST OF FIGURES

1.1 Hierarchical network design .......................................................... 8

3.1 Instance for which the size of the solution computed by the Das et al.’s algorithm, \( \{v_1, v_2, \ldots, v_k\} \), is larger than the optimal solution, \( \{u_1, u_2\} \), by a logarithm factor ......................................................... 36

3.2 Creating and reducing the CDS based on Wu and Li algorithm ................. 39

3.3 An example of unit-disk graph \( G \) containing 9 vertices and 12 edges ...... 42

3.4 The computed connected dominating set from Cheng’s algorithm contains vertices \( \{0, 1, 2, 3, 4, 7\} \). The optimal solution contains \( \{1, 2, 3, 4, 7\} \) .... 42

3.5 Network topology ............................................................................. 45

3.6 Clustering using LID algorithm .......................................................... 46

3.7 Routing between \( S \) and \( D \) using CLUSTERPOW. \( N_1, N_2, \) and \( N_3 \) are the intermediate hops .............................................................................. 48

3.8 Tunneled CLUSTERPOW: the 100 mW hop from \( S \) to \( N_1 \) is replaced by three hops of 1 mW, 10 mW, and 1 mW each .......................... 48

3.9 Initial cluster head starts the advertisement process. New cluster head candidates send the exploration signal within the range of \( e_{r2} \) to continue the process of cluster establishment. ................................................. 51

4.1 EEC-FCB for a network of 6 nodes. Nodes \( \{1, 2, 3\} \) are CHs and nodes \( \{4, 5, 6\} \) are normal nodes. ................................................................. 61

4.2 EEC-FCB for a network of 9 nodes. Nodes \( \{1, 2, 3\} \) are CHs and nodes \( \{4, 5, 6, 7, 8, 9\} \) are normal nodes. ................................................................. 61

4.3 EEC-CB for a network of 9 nodes. Nodes \( \{1, 2, 3\} \) are CHs and nodes \( \{4, 5, 6, 7, 8, 9\} \) are normal nodes. ................................................................. 64

4.4 EEC-CB for a network of 8 nodes. Nodes \( \{1, 2, 3, 4\} \) are CHs and nodes \( \{5, 6, 7, 8\} \) are normal nodes. ................................................................. 64
4.5 EEC-R for a network of 7 nodes. Nodes \{1, 4\} are CHs, nodes \{2, 3\} are BCHs, and nodes \{5, 6, 7\} are normal nodes. .......................... 68

4.6 Cost of EEC-FCB, EEC-CB, and EEC-R ................................. 69

5.1 Three membership functions defining the relative "tallness" of people .... 74
5.2 Components of the fuzzy logic controller ................................. 75
5.3 Three RE membership functions representing low, medium, and high RE .. 77
5.4 Two mobility membership functions representing low and high mobility ... 78
5.5 Three traffic membership functions representing low, medium, and high traffic ................................................................. 80
5.6 Three output membership functions indicating whether the aggregated weight is bad, acceptable, or good .................................................. 80
5.7 Three SQ membership functions representing low, medium, and high SQ . 82
5.8 Three RC membership functions representing low, medium, and high RC . 83
5.9 Three RSS membership functions representing weak, average, and good RSS ................................................................. 84
5.10 Three MQ membership functions representing bad, acceptable, and good MQ ................................................................. 84

6.1 Poisson Traffic Model: exponential inter-arrival time and exponential holding time ................................................................. 96
6.2 Elected CHs (gray rectangles) when the CH election phase is complete ... 99
6.3 Cluster formation ................................................................. 101
6.4 Network recovery ................................................................. 104
6.5 Network topology using FEER .................................................. 107
6.6 Network topology using Wu et al.'s approach ............................... 107
6.7 Network lifetime vs. network size. A maximum bit rate of 500kbps is used at each node. .......................................................... 108

6.8 Number of CHs vs. network size ........................................ 109

6.9 Network lifetime vs. bit rate. The experiment is made on a network of 100 nodes .............................................................. 110

7.1 Initial CH Election. The numbers next to the nodes represent the nodes’ weights. Nodes 4, 5, 10, and 15 are elected as cluster heads. .............. 118

7.2 For a given cluster head $u$, $u$ can reach another cluster head in: (a) one hop: $d(u, x) = 1$, (b) two hops: $d(u, y) = 2$, (c) three hops: $d(u, z) = 3$ ......... 119

7.3 Node 1 data structure. N(3) and N(2) represent the neighbors of nodes 3 and 2 respectively. The nodes are sorted according to their weights. Such a data structure enables node 1 to know its 1-hop and 2-hop neighbors. .... 120

7.4 The resultant network after removing some redundant backbone nodes .... 123

7.5 A ring network with constant time complexity ....................... 128

7.6 (a) A complete graph where node 5 is the only cluster head. FDSS runs in $O(n)$ time complexity. (b) The data structure of node 5. ................. 129

7.7 All possible topologies for a network with 4 nodes .................... 130

7.8 Number of connected unlabeled graphs for $n = 1, 2, \ldots 9$ ............. 131

7.9 Degree frequency for networks of sizes 3, 4, and 5 ...................... 132

7.10 (a) A cycle that might lead to redundant CH election. (b) Cluster heads 1 and 6 elect nodes 3 and 5 to act as cluster heads. ...................... 133

7.11 (a) A cycle that leads to redundant CH election. (b) Cluster head 1 elects nodes 2 and 4 to act as cluster heads. Cluster head 6 elects nodes 3 and 5 to act as cluster heads. ........................................ 134

7.12 (a) A cycle where 4 paths exists between cluster heads $u$ and $v$ (b) The cluster heads having the highest weights are elected as CHs .............. 135

7.13 When $n$ paths exist between cluster heads $u$ and $v$ such that all paths have $d(u, v) = 3$, a maximum of 2 cluster heads are redundant ............... 136
7.14 Network after cluster formation ............................................. 139
7.15 Actions taken by FDDS-M when node \( i \) or its parent CH move ........... 142
7.16 (a) Originally node \( i \) was connected to CH \( j \) (b) Node \( i \) moved out from the coverage area of CH \( j \) and got connected to CH \( k \) ............................... 143
7.17 (a) Originally node \( i \) was connected to CH \( j \) (b) Node \( i \) moved out from the coverage area of CH \( j \). Node \( i \) elects node \( p \) to become a new CH in order to connect it back to CH \( j \). ............................................ 144
7.18 Actions taken by FDDS-M when CH \( j \) wants to retire. \( J_1 \) consists of the original CH neighbors of CH \( j \). ..................................................... 147
7.19 (a) Originally CH \( j \) was connected to CHs \( \{k, m, n\} \) (b) CH \( j \) retires and node \( i \) changed its status to CH and elects node \( p \) to become a CH to preserve backbone connectivity ........................................ 148
7.20 Actions taken by FDDS-M when CH \( j \) moves. \( J_1 \) is a set containing the CH neighbors of \( j \). .......................................................... 149
7.21 (a) Originally CH \( m \) was connected to CHs \( \{k, n, j\} \) (b) CH \( m \) moves and gets disconnected from all its original CHs. CH \( m \) elects nodes \( i \) and \( p \) as CHs to preserve backbone connectivity. ........................................ 150
7.22 Average number of hops in flat network vs. FDDS when \( R = 150 \) ......... 158
7.23 Average number of hops in flat network vs. FDDS when \( R = 250 \) ............. 159
7.24 Average number of hops in flat network vs. FDDS when \( R = 350 \) ............. 159
7.25 Hop stretch of networks with different sizes when \( R = \{150, 250, 350\} \) .... 160
7.26 Backbone size when the following approaches are used: FDDS, Alzoubi’s [3], and Cardei’s [16] ............................................................. 162
7.27 Network lifetime for networks of different sizes when \( R = 150 \). The approaches compared are: FDDS, Alzoubi’s [3], Cardei’s [16], and Wu [108]. 164
7.28 Network lifetime for networks of different sizes when node coverage = 300. The approaches compared are: FDDS, Alzoubi’s [3], Cardei’s [16], and Wu [108]. ......................................................... 165
7.29 Network lifetime for a network of 100 nodes under increasing traffic load. The approaches compared are: FDDS, Alzoubi’s [3], Cardei’s [16], and Wu [108].......................... 166

7.30 Network lifetime of networks that use Network Setup controller vs. networks that do not use Network Setup controller ......................... 168

7.31 Network lifetime of networks that use Routing controller vs. networks that do not use Routing controller .............................. 169
CHAPTER 1

INTRODUCTION

In recent years, wireless networks have enjoyed tremendous development and popularity due to the technological advances of wireless radio devices. While interest in conventional wireless networks continues to evolve, new types of wireless networks, such as Mobile Ad Hoc Network (MANET) [71] and Wireless Sensor Network (WSN) [1], are evolving fast and receiving much attention from academia, industry, and government.

A mobile ad hoc network (MANET) is a best effort, multiple hop datagram-forwarding network, consisting of mobile nodes interconnected by wireless links [27]. A MANET is a collection of arbitrarily located wireless hosts (also called nodes), in which an infrastructure is absent. A wireless host might be a sensor, laptop, palmtop, PDA, phone, desktop computer, etc. Two nodes can communicate directly with each other if they are within each others range; otherwise, intermediate nodes have to relay messages for them. Therefore, each node in such a network must provide services such as routing, address assignment, DNS-like name translation, and more.

One of the first applications of MANETs is the DARPA Packet Radio Network PRNET [47, 49] that dates back to 1972. Recently, MANETs are gaining considerable interest due to the increasing demand of people to stay connected all the time. MANETs are suitable in situations where an infrastructure does not exist or is expensive to deploy. MANETs can be used to set up temporary networks in business environments, meeting rooms, airports, etc, where applications involve cooperative data exchange. They can also be used to
build personal area networks connecting cell phones, laptops, sensors, and other wireless devices. When MANETs are properly combined with satellite-based information delivery, they provide an extremely flexible method for establishing communications for fire, safety, and rescue operations. So, a typical application is disaster recovery, where the conventional communication network infrastructure is destroyed and restoring communication quickly is important. Instead of rebuilding the wired infrastructure which might take a long time, a MANET can be deployed in much less time. The most promising use of MANETs is for military purposes such as a network in a battlefield. Troops will be able to set up a communication network using light weight wireless equipment in hostile and rapidly changing environments. The equipment can range from radios mounted on backpacks, laptops, hand held computers to antennas mounted on vehicles and even airborne relays to route data to and from the command center. MANETs are also used for smart dust [48], environmental monitoring (water, air, and soil), precision agriculture, transportation, and inventory tracking.

1.1 MANET Characteristics

MANETs have many special characteristics that distinguish them from other types of networks. These characteristics include: (1) multi-hop routing, (2) dynamic topologies, (3) self-organized architecture, (4) limited security, (5) limited resources, etc. Some details of each of these characteristics are:

1. **Multi-hop routing**: A node in a MANET can communicate with the nodes that are within its transmission range. So, each node only keeps in contact with a lim-
ited number of neighbors, regardless of the network size. If a node wants to send a message to another node that is outside its range, other nodes help in relaying the message to its destination. Unlike conventional cellular wireless networks, multi-hop radio relay is a fundamental property in MANETs.

In addition to saving radio transmission power of the nodes, multi-hop routing has other significant advantages. The first advantage is adaptability. Using a multi-hop path, packets can be routed in such a way to bypass: obstacles [57], security breached nodes, low energy nodes, etc. The second advantage is the spatial reuse [52], where neighboring nodes have the ability to use the same frequency. This allows for simultaneous communication to exist between nodes that are out of each other’s range. The third advantage is energy conservation. Assume that 2 paths exist between a certain source and a certain destination where the first path is composed of many hops each using a small transmission radius, and the second path is composed of fewer number of hops each using a large transmission radius. It was proven that by choosing appropriate small radii, the path with more number of hops achieves higher throughput and less energy consumption [42].

2. **Dynamic topologies**: Nodes in a MANET are autonomous and free to move arbitrarily. Such properties result in a topology that is constantly and rapidly changing.

3. **Self-organized architecture**: Self organization plays a role in adapting to the network dynamics. It allows nodes to collaborate with each other to make appropriate decisions that lead to better network performance. One way to provide self organi-
zation is by using a central node that oversees the whole network. But such a central node is not usually present in a MANET. So, network designers resort to utilizing distributed self organization algorithms that are fast and efficient. Dynamic routing is one example of self-organization. Distributed self organization presents a real challenge in MANETs and it constitutes a major part of this dissertation.

4. **Limited security**: MANETs are more vulnerable than wired networks. This is due to the wireless medium that is shared by all mobile nodes. MANETs are subject to attacks such as: eavesdropping, spoofing, denial-of-service, man-in-the-middle, etc.

5. **Limited resources**: Bandwidth and energy are two of the most important resource constraints in MANETs. MANETs have limited bandwidth because wireless link capacity is usually low. Also, wireless links have low throughput due to multiple access, fading, noise, and interference. MANETs are also energy constrained because most MANET nodes rely on batteries. When the node battery is drained, it dies and might bring the whole network to a halt. So, one of the most important system design criteria is energy conservation and it constitutes a major part of this dissertation.

Despite active research groups working on many of these MANET characteristics, major challenges still exist. These challenges include routing, security, quality of service (QoS) and power management. In fact, one of the goals of the Internet Engineering Task Force MANET Working Group (IETF MANET WG), which was formed in 1996, was to develop a framework for running IP based protocols in ad hoc networks [66]. Routing protocols that aim to achieve this goal [67, 86] are constantly being analyzed and tested by
MANET WG for possible standardization.

1.2 MANET Challenges

The characteristics discussed in the previous section differentiate MANETs from conventional networks. They also cause considerable challenges including scalability, energy-efficiency, and QoS requirements.

Scalability in MANETs can be defined as whether the network can provide an acceptable throughput when the network size increases. Scalability in a MANET is directly related to the network design and to the routing protocol used. For example, the network can become more scalable by reducing the routing protocol overhead [89]. Scalability can also be achieved by designing the network in a hierarchical fashion [83]. Scalability is still an open problem in MANETs and it is receiving much interest from researchers. A good survey discussing routing protocols’ scalability can be found in [88].

Energy-efficiency is of considerable importance in MANETs since nodes rely on portable, limited power resources and they bear the duty of relaying packets for other nodes. The failure of some nodes due to battery drainage might bring the whole network down. Most existing solutions for reserving energy are done in the MAC Layer. They reserve energy by sending the nodes to sleep and then waking them up when it is time for them to send/receive packets. Other solutions for reserving energy are done in the Network Layer where routes with many short hops are chosen instead of routes with fewer longer hops [91]. Few researchers considered the MANET environment aspects (node properties and link properties) as deciding factors in reserving energy. Such factors include: the traffic pattern in
the network, the mobility of the nodes, the residual energy of the nodes, and the density of certain regions in the network. Energy-efficiency remains one of the most important challenges in designing MANETs.

Satisfying the QoS requirements is a major challenge in both wired and wireless networks. In wireless networks, the problem is harder due to the unpredictability of the radio frequency (RF) characteristics and due to sharing the channel medium. Extensive research has been done on QoS provisions in ad hoc networks, such as QoS routing or admission control [19, 61, 63, 120]. Most of the existing research works deal with resource allocation (e.g., scheduling or buffering) or routing for QoS requests.

In this dissertation, we address the first two challenges (scalability and energy-efficiency) in details and we present multiple solutions to these problems. The next section presents the problem description and the approaches used to address the above challenges.

1.3 Problem Description

Two different architectures exist for an ad hoc network: flat and hierarchical [41]. Flat networks are the simplest because all nodes are equal. Flat networks require each node to participate in the forwarding and receiving of packets depending on the implemented routing scheme. Hierarchical networks are more complicated and they conceal the details of the network topology by aggregating nodes into clusters. Some nodes, such as cluster heads and gateway nodes, have higher computation and communication burden than other nodes.

Routing protocols used in flat networks rely on either link-state (LS) or distance-vector
Routing algorithms use the shortest distance to reach the destination. In the DV algorithm, each node communicates with only its neighbors and provides them with the least-cost estimates from its location to all the nodes (that it knows about) in the network. DV requires communication between directly connected neighbors at each iteration. DV converges slowly, might have routing loops, and suffers from the count-to-infinity problem. The LS algorithm overcomes the problems that DV has by maintaining global network information at each node. Each node communicates with all the nodes in the network via broadcast and informs them about the cost of its directly connected links. Unfortunately, LS updates produce a routing overhead of $O(n^2)$ in a network of $n$ nodes.

Therefore, as the number of nodes becomes large, the overhead in computing, storing, and communicating routing information becomes prohibitive. The overhead required to broadcast the LS updates to the whole network leaves no bandwidth to send data packets. Also, a DV algorithm that iterates among such a large number of nodes would surely never converge.

It has been actually proven [39, 40] that a flat network has poor scalability. In [40], theoretical analysis show that the node throughput declines rapidly to zero as the number of nodes in the network increases. It was also shown, that even when the nodes are optimally placed, a network of size $n$ can not provide a per-node throughput of more than $\frac{c}{\sqrt{n}}$ bits/sec, where $c$ is a constant. The experimental results of the scaling law described in [40], employing IEEE 802.11 technologies, are reported in [39]. The results show that the decline in throughput is like $\frac{c}{n^{0.68}}$ bits/sec, which is considerably worse than the theoretical results.
To overcome the scalability problem that exists in flat networks, we adopt a hierarchical network design approach. Most hierarchical design approaches for MANETs are based on the concept of cluster head [7, 8, 37]. We follow the same concept and we divide the network into a group of clusters. Each cluster is represented by a cluster head (CH). Nodes inside a cluster can only communicate with their associated cluster head and cluster heads can communicate with each other. The collection of cluster heads form the backbone network (also known as virtual backbone). The virtual backbone has to be connected at all times because all network traffic has to be routed through it. Figure 1.1 shows an example of our hierarchical design.

![Hierarchical network design](image)

**Figure 1.1** Hierarchical network design

In general, the hierarchical network design approach has many benefits: (1) It facilitates
the special reuse of resources by providing the ability to use the same frequency in non-
neighboring clusters. Resources can also be shared in a controlled fashion within the same cluster [62]. (2) The cluster heads act as a virtual backbone for inter-cluster routing. So, the flooding of routing information will only involve the cluster heads, not all the nodes in the network. (3) Routing is localized within a cluster. Local changes need not to be seen and updated by the whole network. However, the hierarchical approach has some side effects that include: (1) The bandwidth wasted while forming and maintaining the clusters. (2) The ripple effect, i.e., the reformation of one cluster can affect all the clusters in the network. (3) The number of rounds in which the cluster formation can complete. The hierarchical designs that we propose take into account these drawbacks and aim to minimize them as much as possible. Chapters 6 and 7 describe in details two different approaches for achieving an efficient hierarchical design.

While setting up the hierarchical design, nodes’ batteries might be drained leading to a non-energy-efficient network. Moreover, the hierarchical design might place nodes with low batteries in the backbone network. Such nodes would surely die fast since they have higher computation and communication burden than the other regular nodes. For these reasons, we use energy-efficient algorithms in each step of the hierarchical design until the network is fully operational. We also use energy-efficient techniques in network maintenance and routing.

In this dissertation, we propose clustering and routing approaches to overcome scalability and energy constraint problems that exist in large scale MANETs. To handle these problems, we design a hierarchical energy-efficient scheme that can be easily setup and
maintained. The scheme has three major components, namely: (1) the hierarchical (clustering) component, (2) the maintenance component, and (3) the routing component. Our research provides a mathematical formulation of the clustering component using integer linear programming - ILP (Chapter 4). It also provides centralized and distributed energy-efficient schemes that achieve efficient clustering, efficient maintenance, and efficient routing in large scale MANETs (Chapters 6 and 7). Fuzzy logic is one of the techniques we used to achieve our goals. We designed three fuzzy logic controllers (discussed in details in Chapter 5) that aggregate multiple network metrics and then decide on the quality of the node/path. This quality is then used as an important parameter in the clustering, maintenance, and routing components. Our simulation study shows that our suggested approaches outperform well known schemes found in the literature.

1.4 Dissertation Organization

The primary focus of this dissertation is network design and routing in MANETs. Specially, scalability, self-organization, and energy-efficiency are considered. The remainder of this dissertation is organized as follows:

Chapter 2 gives an overview of flat routing protocols in MANETs. We divide the flat routing protocols into many categories depending on the way they operate. Some of these categories are: proactive, reactive, and hybrid schemes.

Chapter 3 gives an overview of hierarchical routing protocols in MANETs. We categorize the hierarchical routing protocols into dominating set and non-dominating set based routing protocols.
Chapter 4 defines the optimal solution to the energy efficient clustering problem by formulating it as an integer linear programming (ILP) problem. Three ILP formulations are discussed: (1) energy efficient clustering with a fully connected backbone (EEC-FCB), (2) energy efficient clustering with a connected (not fully connected) backbone (EEC-CB), and (3) energy-efficient and reliable clustering (EEC-R).

Chapter 5 introduces the technique of using fuzzy logic and its benefits. Three fuzzy logic controllers are developed to evaluate the quality of each node in the MANET. The quality of each node is then used to establish an efficient and robust routing protocol. The fuzzy logic controllers are used in (1) network setup, (2) network maintenance, and (3) routing scheme.

Chapter 6 proposes a centralized hierarchical scheme that clusters the network and establishes routing paths between nodes while maximizing network lifetime. The centralized approach is distinguished by its iterative network recovery algorithm which produces a highly reliable network.

Chapter 7 proposes a distributed hierarchical scheme that maximizes network lifetime; the distributed scheme presents the heart of this dissertation. The distributed scheme is distinguished by its fast convergence. It has a message complexity of $O(n)$ and a time complexity of $O(\Delta^2)$, where $n$ is the number of nodes in the network and $\Delta$ is the maximum node degree. To our knowledge, such complexities are the best achieved among proposed schemes found in the literature.

Chapter 8 discusses the dissertation conclusion and future directions. It also summarizes the accomplishments of this dissertation. It emphasizes on the contributions resulting
from the efficient distributed approach.
CHAPTER 2

FLAT ROUTING PROTOCOLS OVERVIEW

Routing protocols in ad hoc networks can be classified based on several criteria. These include what routing information is exchanged, or when and how the routing information should be exchanged, or when and how routes are computed and so on. In this chapter we discuss these criteria.

2.1 Classification Criteria

In this section, we discuss (1) link state vs. distance vector routing protocols, (2) periodic vs. event-driven update routing protocols, (3) proactive vs. reactive routing protocols, (4) flat structure vs. hierarchical structure, (5) decentralized computation vs. distributed computation, (6) source vs. hop-by-hop based routing protocols, and (7) single vs. multiple paths routing protocols.

2.1.1 Link State (LS) vs. Distance Vector (DV) Routing Protocols

Distance vector routing (DVR) and Link state routing (LSR) are two underlying mechanisms for routing in wireless ad hoc networks. In DVR [96], every node maintains a distance vector which includes information about the destination ID, next hop, and distance to the destination. This information is periodically exchanged between neighboring nodes. Upon the reception of distance vectors from its neighbors, the node computes new routes and updates its distance vector. The complete route from a source to a destination is formed, in a distributed manner, by combining the next hop of nodes on the path from the
source to the destination. DVR protocols have several drawbacks. They include the slow convergence and the tendency of creating routing loops. This family of routing protocols is usually referred as Bellman Ford routing protocols. On the other hand, LSR [96] protocols, and contrarily to DVR protocols, routing information is exchanged in the form of link state packets (LSP). The LSP of a node includes link information about its neighbors. Any link change will cause LSPs to be flooded into the entire network immediately. Every node can construct and maintain a global network topology from the LSPs it receives, and compute, by itself, routes to all other nodes. The problem with LSR is that excessive routing overhead may be incurred because nodes in a wireless ad hoc network move quickly causing a change in the network topology.

2.1.2 Periodical vs. Event Driven Update Routing Protocols

In case of LSR protocols, routing information needs to be disseminated to network nodes in order to ensure that the knowledge of network topology remains up-to-date. Based on when the routing information will be disseminated, we can classify routing protocols as periodical update and event-driven update protocols. Periodical update protocols disseminate routing information periodically. Periodical updates will simplify protocols and maintain network stability, and most importantly, enable all nodes within the network to be aware of any network topology change. However if the period between updates is large, the protocol may not keep the information up-to-date. On the other hand, if the period is small, too many routing packets will be disseminated causing a lot of overhead, which is not a desirable feature in a wireless network, where the bandwidth is scarce. However,
in case of event-driven update protocol, when events occur, (such as when a link fails or a new link appears), an update packet will be broadcast and the up-to-date status can be disseminated over the network soon. The problem might be that if the topology of networks changes rapidly, a lot of update packets will be generated and disseminated over the network which will use a lot of precious bandwidth, and furthermore, may cause too much fluctuation of routes. Periodical update and event-driven update mechanisms can be used together, forming what is called a hybrid update mechanism. For example, in DSDV [81], a node broadcasts its distance-vector periodically.

2.1.3 Proactive vs. Reactive Routing Protocols

This type of classification is based on when the route is computed. In proactive routing, also called pre-computed or table-driven routing, routes to all destinations are computed in advance. Therefore, nodes need to store the entire or partial information about the network topology. In order to keep this information up to date, nodes need to update their information periodically or whenever the network topology changes. The advantage of such scheme is that when a source node needs to send packets to a destination node, the route is already available, i.e., there is no latency. The major problem with this type of routing protocols is the fact that the dissemination of routing information will consume a lot of the scarce wireless network bandwidth when the link state and network topology change fast, which is a true fact in a wireless ad hoc network. Another problem is that some routes already computed may never be used. On the other hand, reactive routing protocols, also called on-demand routing, route to a destination node may not exist in advance and
is computed only when the route is needed. When a source node needs to send packets to a destination, it executes the route discovery protocol, and then the source transmits packets along the discovered route. The major advantage over proactive protocols is that the precious bandwidth of wireless ad-hoc networks is greatly saved [68] because it limits the amount of bandwidth consumed in the exchange of routing information by maintaining routes to only those destinations to which the routers need to forward data traffic. The major problem with reactive routing protocols is the large latency at the beginning of the transmission caused by route discovery protocol execution.

2.1.4 Flat Structure vs. Hierarchical Structure

Based on how the overall network is structured, routing protocols can be either with flat or hierarchical structure. In a flat structure, all nodes are at the same level and have the same routing functionality. Flat routing is simple and efficient for small networks. The major problem with such protocols is scalability [39, 40]. As the network size grows, the volume of routing information that need to be exchanged will be large and it will take a long time for routing information to arrive at remote nodes. Hierarchical protocols, which are also called cluster-based routing protocols, were designed to overcome the aforementioned problem [56, 69]. With this scheme, nodes in the network are dynamically organized into clusters. The clusters are aggregated again into larger clusters called super-clusters and so on. It was proven in [45, 69] that organizing a network into clusters help maintain a relatively stable network topology because the high dynamics of membership and network topology is limited within clusters. Only stable and high level information such as the
cluster level or the super-cluster level will be propagated across a long distance, thus the routing overhead caused by the control messages may be largely reduced. Within a cluster, the nodes may have complete topology information about its cluster and proactive routing may be used. If the destination is in a different cluster from the source, inter-cluster routing is used.

2.1.5 Decentralized Computation vs. Distributed Computation

Based on how and where routes are computed, routing protocols can either be with decentralized or distributed computation. In a decentralized computation-based protocol, every node in the network maintains global and complete information about the network topology such that the node can compute the route to a destination solely. The route computation in link state routing protocols is a typical example of decentralized computation. On the other hand, in a distributed computation-based protocol, every node in the network only maintains partial and local information about the network topology. When a route needs to be computed, many nodes collaborate to compute the route. The route computation in distance vector routing protocols belong to this category.

2.1.6 Source vs. Hop-by-hop Based Routing Protocols

In source based routing, the entire route in placed in the headers of data packets so that the intermediate nodes only forward these packets according to the route in the header. With this scheme, the intermediate nodes do not need to maintain up-to-date routing information in order to route the packets they forward, since the packets themselves already contain all the routing decisions. This fact, when coupled with on-demand route computation,
eliminates or reduces the need for the periodic route advertisement and neighbor detection packets required in other [69] kinds of protocols. The major drawback with source based routing arises when the network is large and the route is long. This makes the placement action of the entire route in the header of every packet very costly in terms of network resources used. On the other side, in hop-by-hop routing, the route to a destination is distributed in the "next hop" of the nodes along the route. When a node receives a packet to a destination, it forwards the packet to the next hop corresponding to the destination. The problem with such scheme is the fact that all nodes need to maintain routing information and routing loops are likely to occur.

2.1.7 Single vs. Multiple Paths Routing Protocols

Based on the number of paths from a source to destination pair of nodes, routing protocols can be classified into single path or multiple paths based routing protocols. Finding a single path from a source to destination pair, results in simple protocol design and helps in saving storage space. On the other hand, finding multiple routes has the advantages of easy recovery from a route failure and being more reliable and robust. Moreover, the source can select the best route among multiple available paths.

2.2 Flat Routing Protocols Overview

In this section, we discuss some well known flat routing protocols including: (1) destination sequenced distance-vector routing protocol (DSDV), (2) ad hoc on-demand distance vector Routing (AODV), (3) wireless routing protocol (WRP), (4) optimized link state routing (OLSR), (5) dynamic source routing (DSR), (6) temporally-ordered routing algorithm
TORA), (7) associativity-based routing (ABR), and (8) zone routing protocol (ZRP).

2.2.1 Destination Sequenced Distance Vector Routing Protocol (DSDV)

The Destination-Sequenced Distance-Vector (DSDV) Routing Algorithm [81] is based on the idea of the classical Bellman-Ford Routing Algorithm with certain improvements. The improvements made to the Bellman-Ford algorithm include freedom from loops in routing tables. Every node in the network maintains a routing table that lists all available destinations, the number of hops to reach the destination, and the sequence number assigned by the destination node. The sequence number is used to distinguish old routes from new ones and thus help avoiding the formation of loops. The network nodes periodically transmit their routing tables to their immediate neighbors. A node decides to transmit its routing table if a significant change in the network topology has occurred from the last update sent. Therefore, the update is both time-driven and event-driven. The routing table updates can be sent in two ways: either as a "full dump" or as an "incremental update". In a full dump approach, a node sends the full routing table to its neighbors and could span many packets. However, in incremental update scheme, only entries from the routing table that have been changed since the last update are sent. If there is space in the incremental update packet, those entries whose sequence number has changed may be included. When the network is relatively stable, incremental updates are sent to avoid traffic overhead and full dump are relatively infrequent. In a network where the topology changes frequently, full dumps will be more frequent. In addition to the routing table information, the route update packet contains a unique sequence number assigned by the transmitter which will
be used to distinguish new routes from old ones. The route labeled with the highest sequence number is used. If two routes have the same sequence number then the route with the best metric is used. Based on the past history, the nodes estimate the settling time of routes. Network nodes also keep track of the settling time of routes, which is nothing but an estimate of a weighted average time that routes to a destination will fluctuate before the route with the best metric is received [81]. By delaying the broadcast of a routing update by the length of the settling time, mobiles can reduce network traffic and optimize routes by eliminating those broadcasts that would occur if a better route was discovered in the future.

2.2.2 Ad Hoc On-demand Distance Vector Routing

Ad hoc On-demand Distance Vector Routing (AODV) [82] is an improvement on the DSDV algorithm discussed earlier. AODV minimizes the number of broadcasts by creating routes on-demand as opposed to DSDV that maintains the list of all the routes. The process of finding a path consists on the following. The source node broadcasts a route request packet that will be intercepted by all neighboring nodes. The neighbors in turn broadcast the packet to their neighbors until it reaches an intermediate node with recent route information about the destination or until it reaches the destination. A node discards a route request packet that it has already seen. The route request packet uses sequence numbers to ensure that the routes are loop free and only recent information is exchanged. When a node forwards a route request packet to its neighbors, it also records in its table the node from which the first copy of the request was originated. This information is used
to construct the reverse path for the route reply packet. For this reason, AODV uses only symmetric links because the route reply packet follows the reverse path of route request packet. As the route reply packet traverses back to the source, the nodes along the path enter the forward route into their tables. If one of the intermediate nodes fails, the node’s neighbor realizes the link failure and sends a link failure notification to its upstream neighbors until it reaches the source, which in turn trigger the route discovery protocol looking for a new route. The AODV protocol has the advantage of efficiently using the bandwidth by minimizing the network load for control and data traffic and ensure a loop free routing. However, this protocol presents the drawback of using routing caches to reply to route queries, which results in an uncontrolled replies and repetitive updates that are flooded all over the network. Besides, AODV uses periodic beaconing to keep routing tables updated, which in turns creates a significant message overhead.

2.2.3 Wireless Routing Protocol

The WRP described in [72] is a table-based distance vector routing protocol with the goal of maintaining routing information among all nodes in the network. Each node in the network is responsible for maintaining the following four tables: (1) distance table, (2) routing table, (3) link-cost table, and (4) message retransmission list table. Each entry of the message retransmission list contains the sequence number of the update message, a retransmission counter, an acknowledgment-required flag vector with one entry per neighbor, and a list of updates sent in the update message. The message retransmission list records which updates in an update message need to be retransmitted and which neighbors should
acknowledge the retransmission [72]. Similarly to most routing protocols, update messages are exchanged between nodes in case of any change in the network topology. The scope of update message exchange is restricted to only neighboring nodes and contains a list of information that include the following: the destination, the distance to the destination, the predecessor of the destination, and a list of responses indicating which mobiles should acknowledge the update message by sending an ACK message. In the event of a loss of a link between two nodes, the nodes send update messages to their neighbors. The neighbors then update their distance table entries and check for new possible paths through other nodes. Any new paths are relayed back to the original nodes so that they can update their tables accordingly. Nodes learn of the existence of their neighbors from the receipt of an ACK message. With the WRP protocol, nodes need to, periodically, send hello messages ensuring the connectivity of the sending node to the rest of the nodes having received the hello message. However, the failure to send hello messages to the neighbors indicates the failure of that link and results in an update of the routing table within each node. When a node receives a hello message for the first time from a new node, the ID of the new node is added to the routing table then a copy of the routing table is forwarded to the new node. In WRP, routing nodes communicate the distance and second-to-last hop information for each destination in the wireless networks. WRP belongs to the class of path finding algorithms with an important feature, which consists of its ability to avoid the count-to-infinity problem by forcing each node to perform consistency checks of predecessor information reported by all its neighbors. This ultimately eliminates looping situations and provides faster route convergence when a link failure event occurs.
2.2.4 Optimized Link State Routing

The Optimized Link State Routing Protocol (OLSR) is developed for mobile ad hoc networks. The protocol is documented in the experimental Request For Comment (RFC) 3626 [85]. OLSR is table-driven and pro-active and utilizes an optimization called Multi-point Relaying (MPR) for control traffic flooding.

Optimized link state routing (OLSR) is based on the link state routing concept applied to the upgraded ARPANET. OLSR includes a heuristic designed technique to optimize for the shared wireless media environment. Essentially, the heuristic consists of restricting the flooding of information to a set of selected neighboring nodes called the Multipoint Relaying (MPRs). Each node selects a set of nodes from its one-hop neighbors, which retransmits its broadcast packets. These selected neighbors consist of an MPR set of nodes. For a node to decide if it should retransmit packets coming from other nodes, each node maintains information about those neighbors in its MPR set, which are called the MPR Selectors of the node. Following this scheme, the size of control messages needed to be exchanged by each node is reduced since it only populates the link with its MPR selectors instead of all network links. Along the same way, the OLSR protocol minimizes flooding of the control messages as it only uses its MPRs to retransmit the broadcast messages in a MANET. Each node selects its MPRs from its one-hop neighbors with a bi-directional link. The selection of an MPR set of a node follows a process ensuring that the set covers all nodes that are two hops away from the node. The MPR nodes are selected as intermediate nodes in the routing path.
In OLSR, four steps are needed for a node to create the route table: (1) neighbor sensing, (2) MPR selection, (3) MPR information declaration, and (4) route table calculation. In the neighbor sensing step, each node periodically sends a HELLO message containing information about its one-hop neighbors and their link status. The HELLO messages are received by all one-hop neighbors, and are not retransmitted to further nodes. During the MPR selection step, each node independently selects its MPR set according to the MPR selection scheme. As a result, all two-hop neighbors of each node are contained in the union of the neighbor sets of its MPRs. Then, each node declares its MPRs in the subsequent HELLO messages. From the HELLO messages each node can inform its MPR Selectors and construct its MPR Selector table. In the MPR information declaration step, each node broadcasts specific control messages called Topology Control (TC) messages to declare its MPR Selector set. The TC messages are forwarded through MPR nodes and transmitted to all nodes in the MANET. According to the MPR selectors and the information in TC messages, a node maintains a network topology table to record the MPRs of other nodes. In the last step, the information existing in the network topology table as well as the information existing in the neighbor table is used to build the routing table.

2.2.5 Dynamic Source Routing

The Dynamic Source Routing (DSR) protocol [46, 86] is a source based on-demand routing protocol. Nodes within the network are required to maintain route caches that contain all information about the already established routes. Entries in the route cache are continuously updated as new routes are discovered. The protocol consists of two major
steps: route discovery phase and route maintenance phase. When a node has a packet to send to a specific destination, it first checks its route cache to determine whether an entry to the destination already exists. If it has an unexpired route to the destination, it will use this route to send the packet. On the other hand, if no entry exists, the node initiates the route discovery mechanism by broadcasting a route request packet along with the address of the destination, the source node’s address, and a unique identification number. Each node receiving the packet checks its routing table for an entry to the destination node. If no entry exists, it adds its own address to the route record of the packet and then forwards the packet along its outgoing links. To limit the number of route requests messages propagated on the outgoing links, the node only forwards the route request if the request has not yet been seen and if the node’s address does not already appear in the route record. When the route request message reaches the destination node or when it reaches an intermediate node which contains in its route cache an unexpired route to the destination [12], a route reply message is generated and propagated back to the source node. In case the route reply message was generated by destination node, then the whole route record is included in the message before being sent. However, in the other case, i.e., if the responding node is an intermediate node, it will append its cached route to the route record and then generate the route reply. On the other hand, the second step of the DSR protocol execution consisting of the route maintenance is accomplished through the use of route error packets and acknowledgments. Route error packets are generated at a node when the data link layer encounters a fatal transmission problem. When a route error packet is received, the node in error is removed from the node’s route cache and all routes containing the hop are truncated at that
point. The DSR protocol has the drawback of using routing caches to reply to route queries, which results in an uncontrolled replies and repetitive updates that are flooded all over the network. Furthermore, DSR, also suffers from the scalability problem. As the network becomes larger, the control packets and message packets also become larger which will consume a lot of the scarce wireless network bandwidth.

2.2.6 Temporally Ordered Routing Algorithm

Temporally-Ordered Routing Algorithm (TORA) is highly adaptive, loop-free, distributed routing algorithm based on the link reversal concept. TORA was designed to operate in a highly dynamic mobile networking environment. It is source initiated and provides multiple routes for any desired (source, destination) pair. This algorithm requires the need for synchronized clocks. This protocol consists of the execution of the following three phases: (1) route creation, (2) route maintenance, and (3) route erasure. The advantage of this protocol consists of its ability to provide the following: (1) loop free paths at all instants, (2) multiple paths which are needed in case of a failure in the primary path, (3) fast convergence time, and (3) relatively reduced amount of control messages overhead. The major drawback of TORA is the ”count-to-infinity” problem, which is caused by the use of internodal co-ordination.

2.2.7 Associativity Based Routing

A totally different approach in mobile routing is presented in [99, 100]. The Associativity-Based Routing (ABR) protocol is free from loops, deadlock, and packet duplicates, and defines a new routing metric for ad-hoc mobile networks. This metric is known as the de-
gree of association stability. In ABR, a route is selected based on the degree of association
stability of mobile nodes. Each node periodically generates a beacon signal to signal its
existence. Upon the reception of the beacon signal by the neighboring nodes, the asso-
ciativity tables are updated. For each beacon received, the associativity tick of the current
node with respect to the beaconing node is incremented. Association stability is defined by
connection stability of one node with respect to another node over time and space. A high
degree of association stability may indicate a low state of node mobility, while a low degree
may indicate a high state of node mobility. Associativity ticks are reset when the neighbors
of a node or the node itself moves out of proximity. A fundamental objective of ABR is to
derive longer-lived routes for ad-hoc mobile networks. The three phases of the ABR pro-
tocol are: (1) route discovery, (2) route re-construction (RRC), and (3) route deletion. The
route discovery phase is accomplished by a broadcast query and await-reply (BQ-REPLY)
cycle. A node desiring a route broadcasts a BQ message in search of nodes with a route
to the destination. All nodes receiving the query that are not the destination append their
addresses and their associativity ticks with their neighbors along with QoS information to
the query packet. A successor node erases its upstream node neighbors’ associativity tick
entries and retains only the entry concerned with itself and its upstream node. In this way,
each resultant packet arriving at the destination will contain the associativity ticks of the
nodes along the route to the destination. The destination is, then able to select the best route
by examining the associativity ticks along each of the paths. In case multiple paths have
the same overall degree of association stability, the route with the shortest path in terms
of number of hops is selected. The destination then sends a REPLY packet back to the
source along this path. Nodes propagating the REPLY mark their routes as valid. All other routes remain inactive and the possibility of duplicate packets arriving at the destination is avoided. Route re-construction phase (RRC) may consist of partial route discovery, invalid route erasure, valid route updates, and new route discovery, depending on which node(s) along the route move. Movement by the source results in a new BQ-REPLY process. When the destination node moves, the immediate upstream node erases its route and determines if the node is still reachable by a localized query process. If the destination receives the LQ packet, it sends a REPLY message back with the best partial route; otherwise, the initiating node times out and the process backtracks to the next upstream node. If this process results in backtracking more than halfway to the source, the LQ process is discontinued and a new BQ process is initiated at the source. When a discovered route is no longer desired, the source node initiates a route delete (RD) broadcast so that all nodes along the route update their routing tables. The RD message is propagated by a full broadcast, as opposed to a directed broadcast, because the source node may not be aware of any route node changes that occurred during the route re-constructions phase.

2.2.8 Zone Routing Protocol

The zone routing protocol (ZRP) [72, 86] represents a hybrid routing protocol in that it employs proactive routing within a k-hop radius about each node and reactive (on demand) routing to acquire routing information about nodes lying outside of the proactive routing zone. The primary benefit of this hybrid approach is that the routing protocol is responsive to mobility conditions in the network. That is, if node mobility is low and links are fairly
stable and long enduring then a large proactive routing radius is employed. On the other hand, when nodes are highly mobile and link states fluctuate often, ZRP resorts to a small proactive routing radius. Due to the fact of using both proactive and reactive schemes, the ABR protocol exhibits better performance. However since hierarchical routing is used, the path to a destination may be suboptimal. On the other side, since each node has higher level topological information, memory requirement is greater.

The flat routing protocols discussed in this chapter suffer from poor scalability. To overcome the scalability problem, we adopt a hierarchical network design approach. In the next chapter, we give an overview of some important hierarchical routing protocols. We categorize these hierarchical routing protocols into dominating set and non-dominating set based routing protocols. We concentrate more on discussing the hierarchical routing protocols because they are more related to our proposed approaches.
CHAPTER 3
HIERARCHICAL ROUTING PROTOCOLS OVERVIEW

All the flat routing protocols discussed in Chapter 2 suffer from poor scalability. To overcome the scalability problem, we adopt a hierarchical network design approach. Hierarchical techniques have long been known to increase network scalability [51, 55, 105]. In this chapter, we give an overview of some important hierarchical routing protocols. We categorize these hierarchical routing protocols into dominating set and non-dominating set based routing protocols. Before discussing the various protocols, we present a mathematical model for MANETs and introduce some useful graph theory definitions.

MANETs can be represented as a graph $G = (V, E)$, where $V$ is the set of vertices and $E$ is the set of time-varying edges. Two vertices are joined by a link if and only if they are within each other’s transmission coverage. The structure of the MANET is dynamic and consistently changing especially due to node movement (such consistent change is reflected by $E^t$). If all nodes in the MANET use omni-directional antennas (all having the same transmission range), the network is said to have the property of a unit-disk graph (UDG), in which there is an edge between two nodes if and only if their distance is at most one. Many of the dominating set algorithms use the properties of UDG’s to prove their performance bounds.

Now we present some useful graph theory definitions.

- A dominating set (DS) of a graph $G = (V, E)$ is a set $V’ \subset V$ such that each node in
A connected dominating set (CDS) is a dominating set which also induces a connected subgraph. Another variation of CDS is called the weighted connected dominating set (WCDS) where each node in the graph is associated with a cost.

A minimum connected dominating set (MCDS) is a connected dominating set with minimum cardinality. Another variation of MCDS in called the minimum weighted connected dominating set (MWCDS) where each node in the graph is associated with a cost.

An independent set (IS) of a graph $G = (V, E)$ is a set $I \subset V$ such that no two vertices in $I$ are adjacent. In mathematical terms:

$$I \subset V \text{ s.t. } \forall u, v \in I, \Rightarrow \{u, v\} \notin E.$$  

A maximal independent set (MIS) is an independent set which is not a proper subset of any other independent set. Another common definition is: A maximal independent set is an independent set such that adding any vertex not in the set breaks the independence property of the set [10].

A maximum independent set is the largest independent set for a given graph.

As mentioned in Chapter 1, our hierarchical design divides the network into normal nodes and CHs where all CHs are connected. The CHs form the network’s virtual backbone. The construction of such a virtual backbone is the primary application of Connected
Dominating Sets (CDSs) and Maximal Independent Sets (MISs) in MANETs. Unfortunately, finding the CDS, the MCDS, and the MCDS in unit-disk graphs were proven to be NP-hard problems [25, 35, 38]. Many approximation algorithms differing in their time complexities, message complexities, and objectives have been proposed to find a CDS in MANETs.

Routing protocols that use a CDS to route message from source to destination are usually called dominating set based routing [28, 34, 108, 111], backbone based routing [30], or spine based routing [29, 92]. The rest of this chapter is divided in two sections discussing some important dominating set and non-dominating set based routing protocols.

3.1 Dominating Set Based Routing Protocols

Even though MANETs have no physical backbone, a virtual backbone can be constructed by finding the CDS of the network graph. Utilizing CDS in MANET routing has many advantages including:

- It improves the efficiency of multicast/broadcast routing. A serious problem in multicast/broadcast routing occurs when nodes attempt to broadcast a message. Nodes might unnecessarily forward the same message many times. Consequently, nodes might hear the same message many times. This leads to serious redundancy, contention, and collision. This problem is known as the broadcast storm problem [73]. Using CDS as a virtual backbone eliminates most redundant broadcasts [22, 59, 60, 97, 109, 112, 113, 114].

- It increases the network lifetime by applying certain power management techniques.
MANET nodes are usually equipped with batteries having a limited power supply. By using CDS, more nodes can operate in sleep mode and wake up when it is time for them to send/receive messages [17, 32, 115]. CDS was also used to conserve energy by balancing the network management requirements [90, 108, 110, 112, 113, 114].

- It reduces the control message overhead. Using CDS, only a small percentage of the network nodes is used to communicate control messages [69, 76, 94].

Driven by these advantages, many centralized as well as distributed algorithms have been designed to find a CDS in a graph (network). Theoretically, any centralized algorithm can be implemented in a distributed fashion, with the tradeoff of higher protocol overhead. In [38], Guha and Khuller proposed two centralized greedy heuristic algorithms to construct a CDS. They also discussed the performance bounds of their algorithms. These two heuristic algorithms motivated the design of many other algorithms such as the ones described in [20, 104]. In [87], Ruan et al. proposed a one-step centralized approximation algorithm with performance ratio at most $3 + \ln(\Delta)$, where $\Delta$ is the maximum node degree. In [21], Cheng et al. proposed a centralized greedy heuristic algorithm, based on the MIS, in order to approximate the MCDS in unit-disk. In [70], Min et al. proposed a centralized approach that finds the maximal independent set in the first pass and in the second pass it connects the MIS nodes using a Steiner tree with minimum number of Steiner nodes [18, 33, 64]. Many other centralized algorithms have been suggested to approximate the CDS. However centralized approaches do not scale well. For this reason and because distributed approaches are more practical especially in MANETs, we emphasize in this
section on some well known distributed approaches for finding a CDS. A comprehensive survey discussing CDS construction is found in [10].

Some of the important dominating set based routing protocols include:

- Das et al.’s Algorithms [28, 29, 92, 93]
- Wu and Li’s Algorithm [110, 111, 113, 114]
- Alzoubi et al.’s Algorithms [4, 5, 6, 104]
- Cheng et al.’s Algorithms [13, 14, 20, 22]
- Parthasarathy et al.’s Algorithm [77]
- Stojmenovic et al.’s Algorithm [98]
- Krishna et al.’s Algorithm [53]
- Corson and Ephremides’s Algorithm [26]
- Core Extraction Distributed Ad hoc Routing (CEDAR) [94]

In the following sections, we give a detailed description of the algorithms suggested by Das et al., Wu and Li, Alzoubi et al., Cheng et al., Parthasarathy et al., and Stojmenovic et al.

3.1.1 Das et al.’s Algorithms

Das et al. [28, 29, 92, 93] proposed two algorithms based on the algorithms suggested by Guha and Khuller in [38].
The first algorithm requires each node to know if it has the maximum degree among all nodes in the network. It also requires the nodes chosen as part of the CDS to know which 1-hop and 2-hop neighbors are marked. These two requirements force the flooding of degree information in the whole network. The algorithm starts by growing the CDS starting from the node with the highest degree. In each step, a one-edged or two-edged path emanating from the current CDS is selected until all the network is covered. The algorithm has time complexity of $O(|C|(\Delta + |C|))$ and message complexity of $O(n|C|)$, where $n$ is the total number of nodes, $C$ is the set of nodes constituting the CDS, and $\Delta$ is the maximum node degree. The algorithm has an approximation ratio of $2H(\Delta)$, where $H$ is a harmonic function such that $H(\Delta) = \sum_{i=1}^{\Delta} \frac{1}{i} \leq \ln(\Delta) + 1$.

The second algorithm contains three stages: approximating the minimum dominating set, constructing a spanning forest of stars, expanding the spanning forest to a spanning tree. In the first stage, each node is assigned a weight equal to the number of its unmarked neighbors. The dominating set ($D$) is initially empty. An unmarked node compares its weight with the weights of its 1-hop and 2-hop neighbors. The node with the maximum weight is included in $D$. The algorithm continues by iteratively adding the node with the maximum weight to $D$. This stage terminates when $D$ becomes a dominating set. The stage just described is a translation of Chvátals greedy algorithm [23] for Set Cover, and thus it guarantees an approximation factor of $H(\Delta)$. In the second and third stages, each edge is assigned a weight equal to the number of endpoints not in $D$. Then a distributed minimum spanning tree algorithm is used to connect all the nodes in $D$ and thus forming a CDS. The algorithm has time complexity of $O(\Delta(n + |C|))$ and message complexity of
\(O(n|C| + m + n \log(n))\), where \(m\) is the cardinality of the edge set [10].

In [2], Alzoubi showed that the algorithm proposed by Das et al. [28, 29, 92, 93] approximates the MCDS with a ratio of \(\Omega(3H(\Delta))\). Figure 3.1 shows an instance where Das et al.’s algorithm produces a solution that is larger than the optimum solution by a logarithm factor.

![Figure 3.1](image)

**Figure 3.1** Instance for which the size of the solution computed by the Das et al.’s algorithm, \(\{v_1, v_2, \cdots, v_k\}\), is larger than the optimal solution, \(\{u_1, u_2\}\), by a logarithm factor.

In figure 3.1, all points lie in a rectangle whose horizontal side has length one and whose vertical side has length \(2\sqrt{1 - \left(\frac{1}{2(k-1)}\right)^2}\). \(v_1\) and \(v_k\) are the centers of the left and right vertical sides respectively. \(v_2, v_3, \cdots, v_{k-1}\) are evenly distributed within the line segment between \(v_1\) and \(v_k\) from left to right. \(u_1\) and \(u_2\) are the centers of the two sub-rectangles above and below the segment between \(v_1\) and \(v_k\) respectively. The rest of the points lie...
in the two horizontal sides. In each horizontal side, $2^0 = 1$ node lies to the left of (and excluding) the perpendicular bisector of $v_1v_2$, $2^{k-1}$ nodes lie to the right of (and excluding) the perpendicular bisector of $v_{k-1}v_k$, and $2^{i-1}$ nodes lie between (and excluding) the perpendicular bisector of $v_{i-1}v_i$ and the perpendicular bisector of $v_iv_{i+1}$. Thus, the total number of nodes is:

$$n = k + 2 + 2 \sum_{i=1}^{k} 2^{i-1} = k + 2^{k+1}$$  \hspace{1cm} (3.1)$$

Using similar analysis, Alzoubi [2] showed that the maximum degree is $\Delta = 2^k + k + 1$. Note that $\{u_1, u_2\}$ form the MCDS since $u_1$ and $u_2$ are connected and they cover all the nodes in the network. On the other hand, the Das et al.’s algorithm would add $v_k, v_{k-1}, \cdots, v_1$ sequentially to the dominating set in the first stage and produce a CDS composed of $\{v_1, v_2, \cdots, v_k\}$ at the end of the second and third stages.

Since $n = k + 2^{k+1}$, $\Delta = 2^k + k + 1$, $k > \log(n) - 2$, and $k > \log(\Delta) - 1$, the approximation factor of the algorithm proposed by Das et al. is between $\frac{\log(\Delta)}{2} - \frac{1}{2}$ and $3H(\Delta)$. Both time and message complexities are $O(n^2)$. As for CDS maintenance, local update is used in case of a single node movement. When many nodes move at the same time, local updates are used only if the nodes have no overlapping neighborhoods. If they have overlapping neighborhoods, the entire CDS algorithm might be executed.

### 3.1.2 Wu and Li’s Algorithm

Wu and Li [110, 111, 113, 114] proposed an energy efficient clustering based on the dominating set (DS) marking algorithm. While the algorithm proposed by Das et al. first finds a DS and then grows it into a CDS, the algorithm proposed by Wu and Li first finds a
CDS and then remove certain redundant nodes from the CDS based on a certain objective. The network is represented as a graph \( G = (V, E) \), where \( V \) is the set of vertices and \( E \) is the set of edges. The vertices in the graph represent the nodes of the MANET. An edge between hosts \( \{u, v\} \) indicates that both hosts \( u \) and \( v \) are within their wireless transmission range. Let \( N(v) = \{u|\{v, u\} \in E\} \) be the open neighbor set of vertex \( v \). Initially all vertices are unmarked. They exchange their open neighborhood information with their one-hop neighbors. After the exchange, each node knows all of its two-hop neighbors. The suggested distributed algorithm marks a node as a gateway if two of its neighbors are not directly connected.

After finding the DS, two approaches are suggested to remove some nodes from the DS. In the first approach, a vertex \( v \) is removed from the DS if the closed neighbor set of \( v \) is covered by a neighbor \( u \) (\( u \in DS \)) with a higher ID. In the second approach, a vertex \( v \) is removed from DS if the closed neighbor set of \( v \) is covered by two (or more) of its connected neighbors (\( \text{neighbors} \in DS \)) with higher IDs. The ID of the nodes is used to break the ties. The ID of the node can be replaced by its residual energy to produce an energy-efficient backbone.

Figure 3.2 is presented by Wu and Li as an illustration of the elimination approaches. In figure 3.2(a), since \( N(v) \subset N(u) \), vertex \( v \) is removed from the DS if \( ID(v) < ID(u) \) and vertex \( u \in DS \). In figure 3.2(b), either vertex \( u \) or \( v \) can be eliminated depending on which vertex has the lower ID. In figure 3.2(c), since \( N(v) \subset N(u) \cup N(w) \), vertex \( v \) is removed if \( ID(v) < ID(u), ID(v) < ID(w) \), and \( \{u, w\} \in DS \).

After removing some of the DS nodes based on the approach discussed above, the
routing is done in three steps: (1) If the source is not a gateway, it forwards its packets to the gateway associated with it. (2) The gateway acts as a new source and uses the network induced by the DS nodes to route the packets to the destination gateway. (3) If the destination gateway is not the final destination, the gateway sends the packets to the destination host.

The major problem in this approach is the way the original DS is calculated. The DS is generated based on the node degree and its connectivity with its neighbors. Residual energy is not considered as one of the deciding factors when finding the DS nodes. So, the resultant DS might be composed of nodes that has low residual energy while other nodes in the network have high residual energy. At this point, the elimination of DS nodes with low residual energy (according to the approach described above) does not improve the lifetime of the network. Moreover, the battery of the DS nodes will deplete very fast. Therefore the algorithm suggested by Wu and Li cannot balance the great difference of energy consumption between dominating nodes and non-dominating nodes because its main objective is to minimize the DS updates rather than to balance the energy consumption among all mobile nodes. Also, it is not stated how a node chooses the gateway it wants to connect to. Most probably many nodes can be connected to more than one gateway. Therefore, a certain
criteria should be followed in order to choose the best gateway out of the candidate ones.

The approximation factor of the distributed algorithm proposed by Wu and Li is $\frac{n}{2}$. Its message complexity is $O(n)$ and its time complexity is $O(\Delta^3)$.

### 3.1.3 Alzoubi et al.’s Algorithms

Alzoubi et al. [4, 6, 104] proposed a distributed CDS construction based on computing and then connecting a maximal independent set (MIS). The algorithm starts by electing a leader $v$ in a distributed fashion [24]. The leader election algorithm requires $O(n)$ time complexity and $O(n \log(n))$ message complexity. A spanning tree $T$ rooted at node $v$ is then constructed in order to define the nodes’ ranks as follows: $v$ announces that it has level 0. When the children of $v$ receive the announcement, they increase their parent’s level by 1 and send an announcement. This process continues until the announcement reaches the leaf nodes. When the leaf nodes receive the announcement, they transmit a LEVEL-COMPLETE message which propagates up the tree until it reaches the root.

Each node sets its rank to be the ordered pair of its level and its ID. The labeling process begins from the root node and finishes at the leaves. The node with the lowest rank marks itself black and broadcasts a DOMINATOR message. The marking process then continues according to the following rules:

- If the first message that a node receives is a DOMINATOR message, it marks itself gray and broadcasts a DOMINATEE message.

- If a node received DOMINATEE messages from all its lower rank neighbors, it marks itself black and sends a DOMINATOR message.
The nodes marked as black constitute the MIS. In [4, 6, 104], Alzoubi et al. showed that the distance between any subset of the MIS and its complement is exactly two hops. The second phase constructs a tree spanning all the black nodes, thus creating the CDS. The phase starts by the root joining the CDS and broadcasting an invite message. When a black node joins the CDS, it sends an invitation to all black nodes that are two hops away and outside the current CDS to join the CDS. When a black node receives the message, it joins the CDS along with the gray node that relayed the message. The process terminates when all the black nodes join the CDS. The algorithm has $O(n)$ time complexity and $O(n \log(n))$ message complexity. In [5], Alzoubi et al. extended this work by suggesting a distributed algorithm that does not rely on electing a single leader.

3.1.4 Cheng et al.’s Algorithms

Cheng et al. [13, 14, 20] proposed a distributed CDS construction algorithm that has some similarities with the algorithm described by Azoubi et al. [4, 6, 104]. The algorithm begins by finding a MIS and then it transforms the set into a CDS. Initially every node has white color. When the algorithm terminates, the MIS nodes become black and the rest of the nodes become gray. The algorithm requires a leader election phase [24] with $O(n)$ time complexity and $O(n \log(n))$ message complexity.

The first phase of the algorithm is similar to the first phase in [4, 6, 104], with the level changed to effective degree (number of white neighbors). It requires the effective degree of a node to be broadcasted as many times as its degree. In the second phase, an approximation of the Steiner tree is used to connect the nodes of the MIS. The approximation is based on
the distributed depth-first search spanning tree algorithm. Gray nodes with maximum black degree are considered interconnecting nodes. Figures 3.3 and 3.4 are taken from [10] and they illustrate Cheng’s Algorithm. Figure 3.3 presents a graph with 9 vertices and 12 edges. By applying Cheng’s algorithm to figure 3.3, vertices \{0, 1, 2, 3, 4, 7\} are elected as CDS members (figure 3.4). The algorithm has \(O(n)\) time complexity and \(O(n \log(n))\) message complexity. In [22], Cheng et al. proposed another distributed algorithm that does not elect a single leader and tries to use multiple leaders.

![Figure 3.3](image1.png)

**Figure 3.3** An example of unit-disk graph \(G\) containing 9 vertices and 12 edges

![Figure 3.4](image2.png)

**Figure 3.4** The computed connected dominating set from Cheng’s algorithm contains vertices \{0, 1, 2, 3, 4, 7\}. The optimal solution contains \{1, 2, 3, 4, 7\}.

### 3.1.5 Parthasarathy et al. Algorithm

Parthasarathy et al. [77] proposed two algorithms for finding the virtual backbone. The first algorithm has message and time complexity of \(O(n \log^2 n)\) and \(O(\Delta \log^2 n)\) respectively. The second algorithm has message and time complexity of \(O(n \log n)\) and \(O(\log^2 n)\)
respectively. The authors assume that each node knows (approximately) the number of its neighbors, the maximum degree, and the size of the network. But, there is really no fast way to know the number of nodes in the network or the maximum degree. Acquiring such information requires some kind of flooding which increases the complexity of the algorithm.

3.1.6 Stojmenovic et al. Algorithm

Stojmenovic et al. [98] proposed a scheme that is very similar to Wu’s scheme [110, 111, 113, 114] except that it requires neighborhood topology which may be achieved by GPS or other location technique. Stojmenovic’s approach has a very high message complexity of $O(n^2)$ and a time complexity of $\Omega(n)$. Both approaches do not consider message losses due to collisions in their model.

In this section, we described some important dominating set based routing protocols. It is worth noting that our proposed distributed approach out performs the approaches discussed above by having a message complexity of $O(n)$ and a time complexity of $O(\Delta^2)$. In the next section, we give an overview of some important non-dominating set based routing protocols and we comment on their weaknesses.

3.2 Non-dominating Set Based Routing Protocols

Many existing hierarchical routing protocols do not use a CDS to route messages between nodes. We call such protocols non-dominating set based routing protocols. An important subset of these protocols include:

- Lowest-ID clustering (LID) [62]
In the following sections, we give a detailed description of (1) LID, (2) CLUSTERPOW, tunneled CLUSTERPOW, MINPOW, and (3) DEEP.

### 3.2.1 Lowest-ID Clustering (LID)

In [62], the authors proposed the Lowest-ID clustering algorithm (LID), where nodes are organized into non-overlapping clusters. The clusters are independently controlled and are dynamically reconfigured as nodes move. LID assumes that each node has a unique id and that the nodes do not move during the cluster formation process. A node is a cluster-head if and only if it has the smallest id among nodes in its immediate neighborhood that have not joined any cluster. The clustering algorithm is guaranteed to terminate, and each node is assigned a role of ordinary-node (also known as *cluster member*) or *cluster head*.
LID is a distributed clustering algorithm and it is initiated by all nodes whose ID is the lowest among all their neighbors. Nodes that elected themselves as a CH, broadcast their decision to all their neighbors in order to create clusters. Nodes hearing many broadcast messages select the CH with the lowest ID as their parent CH. If all the node’s neighbors, having lower ID, sent their decisions and none declared itself as a CH, the node elects itself as a CH and broadcasts its ID as cluster ID. Otherwise, it chooses the neighboring node having the lowest ID as a CH and broadcasts such decision. Thus each node broadcasts its clustering decisions after all its neighbors with lower IDs have already done so. Every node associates itself with only one cluster. Also LID requires every node to send only one message during the algorithm. Figure 3.6 presents the outcome of applying the LID algorithm to figure 3.5. LID produced six non-overlapping clusters presented by C1, C3, C5, C10, C14, and C18.

![Network topology](image)

**Figure 3.5** Network topology

LID’s cluster maintenance was designed to minimize the number of node transitions from one cluster to another. An important property of LID is that nodes within a cluster
can communicate with each other in at most two hops. When a node moves from one position to another, it can join the closest cluster if it preserves that property. If this property is violated, the highest connectivity node and its neighbors stay in the original cluster and other nodes are removed. Recall that each node only keeps the information of its “locality”, that is, one hop and two hop neighbors. Upon discovering that a member, say $x$, of its cluster is no longer in its locality, node $y$ should check if the highest connectivity node is a one hop neighbor. If so, $y$ removes $x$ from its cluster. Otherwise, $y$ changes cluster. The maintenance algorithm might lead to single node clusters. Thus it seems that additional procedures for merging or rearranging clusters may be desirable.

### 3.2.2 CLUSTERPOW, Tunneled CLUSTERPOW, and MINPOW

In [50], the authors suggested the CLUSTERPOW, tunneled CLUSTERPOW, and MINPOW routing and power control protocols. CLUSTERPOW and tunneled CLUSTERPOW focus on maximizing the network capacity by increasing spatial reuse, while MINPOW focuses on minimizing the energy consumption. The goal of the above protocols is to choose
the transmit power level, so that most of the intra-cluster communication is at lower transmit power levels, and a higher transmit power level is used only when going to a different cluster. The clusters created has no leaders (cluster heads). Clusters are created by the way the routing is done.

A route in CLUSTERPOW consists of hops of different transmit power such that the clustered structure of the network is respected. The network has three levels of clustering corresponding to power levels of 1 mW, 10 mW, and 100 mW. The route from a source to a destination starts by using a power level of 100 mW at each hop until it reaches the 10 mW cluster to which the destination belongs. Then 10 mW is used at each hop until it reaches the 1 mW cluster to which the destination belongs. Then 1 mW is used at each hop to reach the final destination. Figure 3.7 shows how routing is done using CLUSTERPOW. CLUSTERPOW provides implicit, adaptive, and distributed clustering based on transmit power. The routes discovered consist of a non-increasing sequence of transmit power levels. It is loop free and can be used with any routing protocol, reactive or proactive.

Since numerous low power hops are preferable than fewer high power hops, CLUSTERPOW has a disadvantage because it starts the route by using high power level (100 mW). For this reason, an extension of CLUSTERPOW is suggested that replaces high power hops with two or more lower power hops. Figure 3.8 shows a route calculated by Tunneled CLUSTERPOW, where the 100 mW hop from $S$ to $N_1$ is replaced by three hops of 1 mW, 10 mW, and 1 mW each. But, this extension might cause an infinite loop. To resolve the infinite loop problem another version of CLUSTERPOW protocol called Tunneled CLUSTERPOW is used. This protocol resolves the loop problem by tunneling the
Routing between $S$ and $D$ using CLUSTERPOW. $N_1$, $N_2$, and $N_3$ are the intermediate hops. The packet to its next hop using lower power levels, instead of sending the packet directly. In [50], the authors proved that the Tunneled CLUSTERPOW power control protocol ensures the delivery of packets to their destination.

The third protocol described in [50], MINPOW, minimizes the total power consumption
(for communication) on a route. It is essentially the distributed Bellman-Ford algorithm [9] with sequence numbers where the cost is the total power consumption instead of the hop count. Any shortest path algorithm can be used. The contribution of MINPOW is that the implementation is done completely at the network layer without requiring any support from the physical layer for estimating per link power cost. The link cost has three components: $P_{Tx_{elec}}$, $P_{Rx_{elec}}$, and $P_{Tx_{Rad}}(p)$ where, $P_{Tx_{elec}}$ is the power consumed by the transmitter electronics, $P_{Rx_{elec}}$ is the power consumed by the receiver electronics, and $P_{Tx_{Rad}}(p)$ is the power consumed by the power amplifier to transmit a packet at power level $p$. To estimate the link cost, each node sends a hello packet at each of the transmit power levels. Only the hello packet with the maximum power level contains the routing updates. The rest are only "beacons" which contain: (1) the address of the originator, (2) the total power consumed, $P_{Tx_{total}} = P_{Tx_{elec}} + P_{Tx_{Rad}}(p)$, in transmitting that packet, (3) the transmit power level $P$, and (4) the sequence number of the corresponding maximum power level hello packet. The neighbors receiving these beacons set the link cost to: $\text{linkcost} = \min_{\text{beacons}}(P_{Tx_{total}}) + P_{Rx_{elec}}$. This link cost is used in the distance vector algorithm for computing the routes.

MINPOW provides a globally optimal solution with respect to the total power consumed in communication, provides loop free routes, and requires no support from the physical layer. But, MINPOW has a big disadvantage in the sense that it only relies on the link cost to determine the route. A better approach is to take into account the link cost and the node cost. The node cost can be measured by the residual energy on the node. Since MINPOW only uses the link cost as a deciding factor, it might choose a route that utilizes nodes with low residual energy. This leads to the depletion of the battery on these nodes.
and thus lowering the lifetime of the network. MINPOW uses hello messages to establish a path with minimum power consumption between a source and a destination. After the path is constructed, traffic can flow from the source to the destination. This approach has a problem because it does not consider the traffic load in establishing the route. This means that the nodes on the route might not be able to handle the traffic flow. Therefore, the traffic load should also be considered when establishing the route.

### 3.2.3 Decentralized Energy-Efficient Cluster Propagation (DEEP)

In [103], the authors suggested a Decentralized Energy-Efficient cluster Propagation protocol which is used to manage the communication of data while minimizing energy consumption across sensor networks. Even though DEEP was initially proposed for sensor networks, it can be easily migrated to MANETs. Moreover, DEEP proposes an energy-efficient technique that has some flaws. For these reasons, we decided to include it in this overview and critique its ideas.

DEEP is based on the idea of starting with an initial cluster head and forming new cluster head candidates gradually by controlling relative distance between a pair of cluster heads and circular radius of each cluster. Since this approach tends to load balance all the clusters in the network, periodic re-clustering is not needed.

DEEP starts by manually choosing a cluster head prior to the network deployment. This initial cluster head starts the cluster setup phase by sending a CH declaration signal within a predefined declaration range $d_r$. Let $E_{rc1}$ and $E_{rc2}$ be two predefined exploration ranges. All nodes with signal strengths $E_r$, $E_r > E_{rc1}$ and $E_r < E_{rc2}$, who did not act as CHs
declare themselves as CH candidates (figure 3.9). The new CH candidates repeat the same process by transmitting a CH declaration signal within the range of $d_r$. If the declaration signal is received by another CH candidate, one of them is declared as CH by negotiation. At the end of this process, if the total number of nodes in a cluster is less than a predefined threshold $m_n$, the cluster will be dissolved and its members will join other clusters. After the clustering process is complete, an Inter-Cluster Routing protocol (ICR) is used to route traffic between CHs and base station. ICR is a destination-initiated reactive protocol where the local base station initiates the route discovery. The re-clustering process is based on a small shift in the initial CH. The nearest neighbor of the CH that had never acted as a CH before is chosen to be the new CH. The new CH starts the clustering process and creates a totally different cluster head constellation.

**Figure 3.9** Initial cluster head starts the advertisement process. New cluster head candidates send the exploration signal within the range of $e_{r2}$ to continue the process of cluster establishment.

DEEP has some major drawbacks; (1) it is not completely distributed, because the initial CH is manually chosen before the initial deployment of the network. (2) DEEP
optimizes the energy consumption while routing and does not pay attention to the residual energy of the nodes. The choice of CHs is made in order to load balance the number of nodes in each cluster. Residual energy is not considered at all when declaring CHs. This leads to quick energy depletion among CHs. (3) Re-clustering is applied when the CH dies or is very close to die. This is not a good strategy, because the death of some CHs might partition the network. Also the re-clustering process might cause a ripple effect, i.e., when a new CH is chosen, some nodes become uncovered and covering these nodes might lead to re-clustering the whole network.

As mentioned in Chapter 1, our main objectives are to overcome the scalability and the energy-efficiency problems that exist in MANETs. To achieve these goals, we designed an energy-efficient hierarchical scheme that can be easily setup and maintained. The scheme has three major components, namely: (1) the hierarchical (clustering) component, (2) the maintenance component, and (3) the routing component. In the next chapter (Chapter 4), we formulate the first component (clustering component) mathematically using integer linear programming (ILP) formulation. The ILP formulation provides us with the optimal hierarchical design which we use later to evaluate our heuristic approaches.
Recall that our objective is to design a hierarchical energy-efficient scheme for large scale MANETs that aims to maximize network lifetime. A major component in our scheme is the clustering component. In this chapter, we define the clustering component and formulate it using integer linear programming (ILP). The ILP formulation provides us with the optimal energy-efficient hierarchical structure.

The clustering component is responsible for designing the hierarchical structure of the network. The clustering component is supposed to: (1) elect a group of connected CHs to act as the network’s virtual backbone and (2) connect each normal node to one and only one CH; the group of normal nodes connected to the same CH forms the cluster. Moreover, the resultant network structure has to be energy-efficient.

In this chapter, we define the optimal solution (using ILP) to the energy-efficient clustering problem when the network has a fully connected backbone (EEC-FCB). To make the problem more practical, an extension to EEC-FCB is made in order to reduce the backbone connections (EEC-CB). We also introduce another extension in order to make the network more reliable (EEC-R). EEC-R introduces a backup CH for each cluster. In a situation where the CH fails, weakens, or moves outside its cluster coverage, the backup cluster head (BCH) takes over immediately. In the following sections, we formulate and explain each problem in details. We start by giving a brief overview about ILP.
4.1 ILP Overview

Linear programming (LP) is an important field in operations research that deals with solving optimization problems of a particular form. Linear programming problems consist of a linear cost function (consisting of a certain number of variables) which is to be minimized or maximized subject to a certain number of constraints. The constraints are linear inequalities of the variables used in the cost function (also called the objective function).

If the unknown variables in the linear programming problem are all integers, the problem is called an "integer programming (IP)" or "integer linear programming (ILP)" problem. If only a subset of the variables are integers, the problem is called a "mixed integer programming problem". If all the variables are restricted to 0 or 1, the problem is called "binary integer programming" problem [107].

In contrast to linear programming, which can be solved efficiently in the worst case, IP problems are in the worst case undecidable, and in many practical situations NP-hard. Every ILP problem falls into one of three categories:

1. Infeasible: The problem is infeasible if a feasible solution to the problem does not exist.

2. Unbounded: The problem is unbounded if the constraints do not sufficiently restrain the cost function so that for any given feasible solution, another feasible solution can be found that makes a further improvement to the cost function.

3. Optimal solution: The problem has an optimal solution when the cost function has a unique minimum (or maximum) cost function value. This does not mean that the
values of the variables that yield that optimal solution are unique. A problem that has an optimal solution is a problem that is not infeasible and not unbounded.

The simplex method is usually used to solve LP problems. It is composed of two basic steps. The first step is to find a feasible solution to the problem. The second step is to iteratively improve the value of the cost function. This is accomplished by finding a variable in the problem that can be increased, at the expense of decreasing another variable, such that the cost function is improved.

The traveling salesperson (TSP) is a famous problem that can be easily modeled as an ILP problem. Consider a traveling salesperson who must visit each of \( n \) cities, numbered from 1 to \( n \). For each pair of cities \((i, j)\), let \( c_{ij} \) be the cost of going from city \( i \) to city \( j \) (and vice versa). Let \( x_{ij} = 1 \) if the person travels from city \( i \) to city \( j \) (and vice versa). This problem is known as the symmetric TSP. In the asymmetric TSP, the cost to travel in one direction may differ from the cost to travel in the other. Clearly the asymmetric problem is the more general. The TSP can be modeled as an ILP problem as follows [102]:

\[
\text{Min} \quad \sum_{i=1}^{N} \sum_{j=1}^{i-1} c_{ij} x_{ij}
\]

Subject to

\[
\sum_{j \neq i} x_{ij} = 2 \quad \forall i \quad (4.1)
\]

\[
\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 2 \quad \forall S \subset N \quad (4.2)
\]

\[
x_{ij} \in \{0, 1\} \quad \forall i, \forall j \quad (4.3)
\]

Constraint 4.1 indicates that every city must be visited. This constraint is not sufficient to ensure that TSP is satisfied because it is possible to have multiple cycles (subtours),
rather than one big cycle (tour) through all the cities. Constraint 4.2 handles this situation by eliminating subtours. It indicates that for any subset of cities, S, the tour must enter and exit that set. Constraint 4.3 limits the solution space (values of $x_{ij}$) to either "0" or "1". Having explained what ILP is, we discuss next the ILP formulations of the clustering problem.

### 4.2 Energy-Efficient Clustering with Fully Connected Backbone (EEC-FCB)

In this section, we define and formulate the energy-efficient clustering problem when the network has a fully connected backbone (EEC-FCB). We formulate EEC-FCB as an ILP problem, where every function must be linear, and the solutions of every variable must be integer. In EEC-FCB, every variable must be either 0 or 1.

#### 4.2.1 EEC-FCB Definition

Since flat networks have poor scalability [39, 40], we designed our network in a hierarchical fashion. Two hierarchical levels are used: cluster heads (CHs) level and normal nodes level. Normal nodes can only communicate with their corresponding CHs, while CHs communicate with each other. The EEC-FCB problem is to group a large set of mobile ad hoc nodes into clusters, elect a CH for each cluster, connect cluster nodes to the chosen cluster heads, and fully connect CHs with each other (this condition will be relaxed in section 4.3) while maximizing the network lifetime. The radio model we use follows the most commonly used power-attenuation model [84]. The signal power falls as $1/r^k$, where $r$ is the distance between the transmitter/receiver nodes and $k$ is a real constant dependent
on the wireless environment, typically between 2 and 4. In our case, we set \( k = 2 \) for communication between normal nodes and CHs, and \( k = 3 \) for communication between CHs. The rationale for using more power consumption between CHs \((k = 3)\) is that CHs carry more burden than normal nodes and they are supposed to route all the traffic that is generated from their cluster members.

### 4.2.2 EEC-FCB ILP Formulation

To make the ILP formulation easy to follow, we start by introducing the notations used in the formulation. The constants/variables and their definitions are presented as follows:

- \( N \): Total number of nodes in the network - predetermined
- \( P \): Number of clusters heads - predetermined
- \( d_{ij} \): Euclidean distance between nodes \( i \) and \( j \)
- \( K_j \): Max number of nodes that can be connected to \( CH_j \) - predetermined
- \( c_{ij} \): Cost of connecting a regular node \( i \) to \( CH_j \) (proportional to \( d_{ij}^2 \))
- \( h_{jk} \): Cost of connecting \( CH_j \) to \( CH_k \) (proportional to \( d_{jk}^3 \))
- \( b_j \): Weight associated with \( CH_j \)
- \( x_{ij} \): Variable. 1 if node \( i \) is connected to \( CH_j \); 0 otherwise
- \( z_{jk} \): Variable. 1 \( CH_j \) is connected to \( CH_k \); 0 otherwise
- \( y_j \): Variable. 1 if node \( j \) is chosen to be a \( CH \); 0 otherwise
• $w_{ij}$: Variable. 1 if $x_{ij} = 1$ and $y_j = 1$; 0 otherwise.

Note that $b_j$ can be any meaningful weight assigned to a node (an appropriate weight is discussed in [34]). The higher the value of $b$, the better the node is. Since the objective function is a minimization function, each value in the $b$ array is multiplied by -1. EEC-FCB can be formulated as a binary ILP problem as follows:

$$
\text{Min}_{x,y,z} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij}x_{ij} + \sum_{j=1}^{N} b_j y_j + \sum_{j=1}^{N} \sum_{k=1}^{N} (h_{jk} - c_{jk}) z_{jk} \quad (4.4)
$$

Subject to

$$
\sum_{i=1}^{N} x_{ij} \geq (P - 1)y_j + (1 - y_j) \quad \forall j \quad (4.5)
$$

$$
\sum_{i=1}^{N} x_{ij} \leq (K_j + P - 1)y_j + (1 - y_j) \quad \forall j \quad (4.6)
$$

$$
\sum_{i=1}^{N} \sum_{j=i+1}^{N} x_{ij} = (N - P) + P(P - 1)/2 \quad (4.7)
$$

$$
\sum_{j=1}^{N} y_j = P \quad (4.8)
$$

$$
\sum_{k=1}^{N} z_{jk} = (P - 1)y_j \quad j \neq k, \forall j \quad (4.9)
$$

$$
\sum_{k=1}^{N} z_{jk} = (P - 1)y_k \quad j \neq k, \forall k \quad (4.10)
$$

$$
\sum_{k=1}^{N} x_{ii} = 0 \quad (4.11)
$$
\[ x_{ij} = x_{ji} \quad \forall i, \forall j \] (4.12)

\[ w_{ij} \leq x_{ij} \quad \forall i, \forall j \] (4.13)

\[ w_{ij} \leq y_j \quad \forall i, \forall j \] (4.14)

\[ w_{ij} \geq x_{ij} + y_j - 1 \quad \forall i, \forall j \] (4.15)

\[ \sum_{i=1}^{N} w_{ij} \geq 1 \quad \forall j \] (4.16)

\[ \sum_{i=1}^{N} w_{ij} \leq (P - 1)y_j + (1 - y_j) \quad \forall j \] (4.17)

\[ x_{ij} \in \{0, 1\} \quad \forall i, \forall j \] (4.18)

\[ z_{jk} \in \{0, 1\} \quad \forall j, \forall k \] (4.19)

\[ y_j \in \{0, 1\} \quad \forall j \] (4.20)

Note that variables \( i, j, \) and \( k \) are between 1 and \( N \). The objective function (equation 4.4) is divided into three parts. The first part calculates the cost of connecting normal nodes to CHs. The second part calculates the cost of electing CHs. The third part calculates the cost of connecting CHs with each other. By minimizing the objective function, the power consumption rate is optimized and the lifetime is maximized.
Constraint 4.5 indicates that if node \( j \) is chosen to be a CH, it needs to support at least \((P - 1)\) other CHs. Constraint 4.6 indicates that if node \( j \) is chosen to be a CH, it should be connected to at most \((P - 1)\) CHs and \( K_j \) regular nodes. Constraints 4.5 and 4.6 also indicate that if node \( j \) is a normal node, it should be connected to only one CH. Constraint 4.7 indicates that the total number of interconnections is equal to the number of connections between regular nodes and CHs plus the number of connections that fully connect the CHs. Constraint 4.8 indicates that the total number of CHs is \( P \). Constraints 4.9 and 4.10 make sure that matrix \( z \) (backbone) contains only the connections that interconnect CHs. Constraints 4.11 and 4.12 indicate that matrix \( x \) is symmetric and its diagonal is 0. Constrains 4.13, 4.14, and 4.15 indicate that \( w_{ij} = 1 \) only if both \( x_{ij} = 1 \) and \( y_j = 1 \), otherwise. \( w_{ij} \) is used to simulate \( x_{ij} \times y_j \).

Constraints 4.16 and 4.17 ensure that if a node is chosen to be a regular node, it should be connected to only one CH and not to another regular node. Constraints 4.18, 4.19, and 4.20 indicate that the variables have binary values. Figure 4.1 shows the outcome of EEC-FCB on a network of 6 nodes. Nodes 1, 2, and 3 are elected as CHs. Node 4 joined CH 1, node 5 joined CH 2, and node 6 joined CH 3. Figure 4.2 shows another example of EEC-FCB on a network of 9 nodes. Nodes 1, 2, and 3 are elected as CHs. Nodes 4 and 5 joined CH 1. Nodes 6 and 7 joined CH 2. Nodes 8 and 9 joined CH 3. In the next section, we extend EEC-FCB such that the backbone connections are reduced.
Figure 4.1  EEC-FCB for a network of 6 nodes. Nodes \{1, 2, 3\} are CHs and nodes \{4, 5, 6\} are normal nodes.

Figure 4.2  EEC-FCB for a network of 9 nodes. Nodes \{1, 2, 3\} are CHs and nodes \{4, 5, 6, 7, 8, 9\} are normal nodes.

4.3 Energy-Efficient Clustering with Connected Backbone (EEC-CB)

EEC-FCB (section 4.2) assumes that the backbone network is fully connected. To make the network more practical, we relax this constraint and reduce the backbone connections
while ensuring the network connectivity. One way to ensure the backbone connectivity is to connect one CH with all the other CHs. We call such a CH the main cluster head (MCH).

Note that, if a node is a MCH, then it is definitely a CH. If a node is a CH, then it might or might not be a MCH. If a node is a normal node, then it is neither a CH nor a MCH.

We introduce a new variable called $M_j$, where $M_j = 1$ if node $j$ is a MCH; 0 otherwise.

EEC-CB is presented by the following binary ILP problem:

$$\text{Min}_{x,y,z} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij}x_{ij} + \sum_{j=1}^{N} b_jy_j + \sum_{j=1}^{N} \sum_{k=1}^{N} h_{jk}z_{jk}$$  \hspace{1cm} (4.21)

Subject to

$$M_j \leq y_j \quad \forall \ j$$  \hspace{1cm} (4.22)

$$M_j \in \{0, 1\} \quad \forall \ j$$  \hspace{1cm} (4.23)

$$\sum_{j=1}^{N} M_j = 1$$  \hspace{1cm} (4.24)

$$\sum_{i=1}^{N} x_{ij} \geq (P - 2)M_j + 1 \quad \forall \ j$$  \hspace{1cm} (4.25)

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} x_{ij} \leq (N - P) + P(P - 1)/2$$  \hspace{1cm} (4.26)

$$\sum_{k=1}^{N} z_{jk} \geq (P - 2)M_j + y_j \quad j \neq k, \forall \ j$$  \hspace{1cm} (4.27)

$$\sum_{k=1}^{N} z_{jk} \geq (P - 1)y_j \quad j \neq k, \forall \ j$$  \hspace{1cm} (4.28)
\[ z_{jk} = z_{kj} \quad \forall i, \forall j \] (4.29)

Constraints 4.6, 4.8, and 4.11 through 4.20 are also needed to formulate EEC-CB. Constraints 4.22 and 4.23 indicate that if a node is chosen as a MCH \((M_j = 1)\), then it must be a CH \((y_j = 1)\) but the inverse is not true (discussed above). Constraint 4.24 indicates that there should be one MCH in the network. This condition ensures the connectivity of the backbone. Constraint 4.25 indicates that if a node was chosen as a MCH, then it must be connected to at least \((P - 1)\) other CHs. Otherwise, the node is just a normal node or a CH. If it was a normal node \((M_j = 0)\), it should be connected to exactly one CH. If it was a CH, it should be connected to at least one other CH. Constraint 4.26 is the same as constraint 4.7, but with the equal sign changed to less than or equal. In this case, the backbone network is not fully connected. Constraints 4.27 and 4.28 ensure that the MCH is connected to all other CHs, while normal CHs can be connected to a minimum of one other CH. Constraint 4.29 indicates that the \(z\) matrix (backbone) is symmetric. Figures 4.3 and 4.4 display two sample executions of EEC-CB where the backbone in not fully connected. In the next section, we further extend EEC-FCB (section 4.2) in order to increase the reliability of the network.

### 4.4 Energy-Efficient Clustering with a Reliable Backbone (EEC-R)

In this section, we further extend EEC-FCB (section 4.2) so that each cluster head is provided by a backup cluster head (BCH). Having a BCH is very useful in many situations such as: (1) CH’s energy becomes scarce and the CH can not handle the cluster demand. (2) CH is not able to cover all the nodes in its cluster because of mobility. (3) Connectivity
Figure 4.3  EEC-CB for a network of 9 nodes. Nodes \{1, 2, 3\} are CHs and nodes \{4, 5, 6, 7, 8, 9\} are normal nodes.

Figure 4.4  EEC-CB for a network of 8 nodes. Nodes \{1, 2, 3, 4\} are CHs and nodes \{5, 6, 7, 8\} are normal nodes.

between CHs is broken also because of mobility. (4) CH fails.
4.4.1 EEC-R Definition

EEC-R is to group a large set of nodes into clusters, elect each CH, elect each BCH, connect nodes to the chosen CHs and to the chosen BCHs, and fully connect CHs and BCHs while minimizing the power consumption rate and maximizing the network lifetime. Note that the backbone connections can be reduced using the same technique used in section 4.3.

4.4.2 EEC-R ILP Formulation

The constants/variables notations given in section 4.2 are still the same with \( w_{ij} \) removed and three new variables added. The new variables are presented as follows:

- \( w_j \): Variable 1 if node \( j \) is chosen to be a BCH, 0 otherwise.
- \( t_{ij} \): Variable. 1 if both \( x_{ij} = 1 \) and \( y_j = 1 \), 0 otherwise.
- \( v_{ij} \): Variable. 1 if \( x_{ij} = 1 \) and \( w_j = 1 \), 0 otherwise.

The objective function presented in section 4.2 is still the same. EEC-R can be formulated as the following binary ILP problem:

\[
\min_{x, y, z} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij}x_{ij} + \sum_{j=1}^{N} b_jy_j + \sum_{j=1}^{N} \sum_{k=1}^{N} (h_{jk} - c_{jk})z_{jk} \quad (4.30)
\]

Subject to

\[
y_j + w_j \leq 1 \quad \forall j \quad (4.31)
\]

\[
y_j \geq 0 \quad \forall j \quad (4.32)
\]
\[
\sum_{i=1}^{N} x_{ij} \geq (2P - 1)(y_j) + (2P - 1)(w_j) + (2)(1 - y_j - w_j) \quad \forall \ j
\]  
(4.33)

\[
\sum_{j=1}^{N} x_{ij} \leq (K_j + 2P - 1)(y_j) + (w_j)(2P + K_j - 2) + (2)(1 - y_j - w_j) \quad \forall \ j
\]  
(4.34)

\[
\sum_{i=1}^{N} \sum_{j=i+1}^{N} x_{ij} = (N - P) + (N - 2P) + 2P(P - 1)
\]  
(4.35)

\[
\sum_{j=1}^{N} w_j = P
\]  
(4.36)

\[
\sum_{k=1}^{N} z_{jk} = 2(P - 1)y_j + 2(P - 1)w_j \quad j \neq k, \forall \ j
\]  
(4.37)

\[
\sum_{j=1}^{N} z_{jk} = 2(P - 1)y_k + 2(P - 1)w_k \quad j \neq k, \forall \ j
\]  
(4.38)

\[
t_{ij} \leq x_{ij} \quad \forall \ i, \forall \ j
\]  
(4.39)

\[
t_{ij} \leq y_j \quad \forall \ i, \forall \ j
\]  
(4.40)

\[
t_{ij} \geq x_{ij} + y_j - 1 \quad \forall \ i, \forall \ j
\]  
(4.41)

\[
\sum_{i=1}^{N} t_{ij} \geq 1 \quad \forall \ j
\]  
(4.42)

\[
\sum_{i=1}^{N} t_{ij} \leq (P - 1)y_j + (1 - y_j) \quad \forall \ j
\]  
(4.43)
\[ v_{ij} \leq x_{ij} \quad \forall i, \forall j \tag{4.44} \]

\[ v_{ij} \leq w_j \quad \forall i, \forall j \tag{4.45} \]

\[ v_{ij} \geq x_{ij} + w_j - 1 \quad \forall i, \forall j \tag{4.46} \]

\[ \sum_{i=1}^{N} v_{ij} \geq 1 \quad \forall j \tag{4.47} \]

\[ \sum_{i=1}^{N} v_{ij} \leq (P - 1)w_j + (1 - w_j) \quad \forall j \tag{4.48} \]

\[ w_j \in \{0, 1\} \quad \forall j \tag{4.49} \]

\[ t_{jk} \in \{0, 1\} \quad \forall j, \forall k \tag{4.50} \]

\[ v_{jk} \in \{0, 1\} \quad \forall j, \forall k \tag{4.51} \]

Constraints 4.31 and 4.32 make sure that if a node is chosen to be a CH, it can not be a BCH and if a node is chosen to be a BCH, it can not be a CH. Constraint 4.33 puts a lower bound on the number of connections that node \( j \) can make with other nodes. If node \( j \) was elected as a CH \((y_j = 1)\) or as a BCH \((w_j = 1)\), then node \( j \) should be connected to at least all the other CHs and BCHs \((2P - 1 \text{ connections})\). If node \( j \) was a normal node, then it should be connected to its CH and its BCH \((2 \text{ connections})\). Constraint 4.34 puts an upper bound on the number of connections that node \( j \) can make with other nodes. The
same reasoning applied in constraint 4.33 applies for this constraint. Constraint 4.35 sums up the connections made between normal nodes and their associated CHs/BCHs plus the connections that fully interconnects the CHs/BCHs.

Constraint 4.36 indicates that the number of BCH is equal to the number of CHs. Constraints 4.37 and 4.38 make sure that matrix $z$ contains only the connections that interconnect CHs and BCHs (backbone network). Constraints 4.39, 4.40, and 4.41 make sure that $t_{ij} = 1$ only if $x_{ij} = 1$ and $y_j = 1$, 0 otherwise. $t_{ij}$ simulates $x_{ij} \times y_j$. Constraints 4.42 and 4.43 are used to ensure that if a node is chosen to be a regular node, then it should be connected to one CH and not to another regular node. Constraints 4.44, 4.45, and 4.46 make sure that $v_{ij} = 1$ only if $x_{ij} = 1$ and $w_j = 1$, 0 otherwise. $v_{ij}$ simulates $x_{ij} \times w_j$. Constraints 4.47 and 4.48 ensure that if a node is chosen to be a regular node, then it should be connected to one BCH and not to another regular node. Constraints 4.49, 4.50, and 4.51 are used to indicate that the variables are binary values.

![Figure 4.5](image)

**Figure 4.5** EEC-R for a network of 7 nodes. Nodes {1, 4} are CHs, nodes {2, 3} are BCHs, and nodes {5, 6, 7} are normal nodes.

Figure 4.5 shows the output of EEC-R on a network of 7 nodes. Nodes 1 and 4 are elected as CHs. Nodes 2 and 3 are elected as BCHs for CHs 1 and 4 respectively. Nodes
5 and 6 are connected to CH 1 and BCH 2. Node 7 is connected to CH 4 and BCH 3. The backbone, composed of CHs and BCHs, is fully connected. And the normal nodes are connected to both CHs and BCHs.

### 4.5 Performance Evaluation

In this section we evaluate the performance of EEC-FCB, EEC-CB, and EEC-R. Networks of different sizes and considered. For each network, $n$ nodes are uniformly distributed in a $500^2$ unit space. The average of 10 runs is calculated for each network. Because EEC-FCB, EEC-CB, and EEC-R take lots of time to execute (NP-complete), we only considered networks with less than ten nodes. Figure 4.6 shows the cost of each approach.

![Performance Evaluation of EEC-FCB, EEC-CB, EEC-R](image)

**Figure 4.6** Cost of EEC-FCB, EEC-CB, and EEC-R

Recall that the cost of each approach (objective function) depends on the CHs’ weight, the communication cost between CHs, and the communication cost between normal nodes and CHs. Also note that the communication cost between CHs is more than the commu-
nication cost between normal nodes and CHs (section 4.2). So the smaller the backbone size, the smaller the cost. All approaches tend to choose as CHs nodes that are close to each other. As the distance between CHs increases, the objective function increases and thus the network cost increases. According to this discussion, EEC-CB has the lowest cost because it designs a network with relaxed backbone i.e. less connections between CHs which translates to less network cost. Since EEC-FCB designs a network with fully connected backbone, it has a higher cost than EEC-CB. EEC-R adds a BCH for each CH. It fully connects the BCHs and CHs forming an extended backbone. This increases the communication cost in the backbone and thus increasing the total network cost. For this reason EEC-R has the highest cost among the other approaches.

4.6 Summary

In this chapter, we defined the optimal solution to the energy-efficient clustering problem when the network has a fully connected backbone (EEC-FCB) by formulating it as an ILP problem. To make the problem more practical, an extension to EEC-FCB is made in order to reduce the backbone connections (EEC-CB). We also introduced another extension in order to make the network more reliable (EEC-R). EEC-R introduces a backup CH for each cluster. BCHs are very useful in situations where the CH fails, weakens, or moves outside its cluster coverage. It is worth noting that the ILP formulation is used as a theoretical basis and it is not intended to be used in practice. Our simulation results show that EEC-CB has the lowest cost followed by EEC-FCB and EEC-R respectively.
CHAPTER 5
FUZZY LOGIC CONTROLLERS

In Chapter 4, we defined the clustering problem and formulated it using ILP. The objective function that governed the hierarchical design was to minimize the total power consumption in the network. Therefore, our main objectives (scalability and energy-efficiency) are met: (1) scalability is increased by designing the network in a hierarchical fashion, and (2) energy-efficiency is increased by making energy conservation the design’s objective function. In the coming chapters, we design centralized and distributed approaches that strive to meet the same objectives.

Recall from the ILP formulation (Chapter 4) that each node has a certain weight. When designing the centralized approach (Chapter 6) and the distributed approach (Chapter 7), this weight plays an important role in deciding which node should be a CH and which node should not. For such weight to be meaningful, it should capture the most important parameters of the mobile node. These parameters are: (1) node’s residual energy in Joules, (2) node’s traffic load in bps, (3) node’s mobility in meters, and (4) node’s degree. Combining these parameters using a mathematical formula is not an easy task especially because each parameter has a different unit. Here comes the job of Fuzzy Logic. We designed a fuzzy logic controller that combines these parameters and produces a single value defining the quality of the node. We call this quality, setup quality ($SQ$). We then use $SQ$ as a deciding factor in the network setup phase in order to decide which node should be a CH and which node should not. Note that both the centralized and the distributed approaches use $SQ$ in
As time passes, the configuration of the network changes. The two factors that cause the network configuration to change are when CHs retire/move and when nodes move. When a node moves out from its cluster, it has to associate itself with another cluster head in order to preserve connectivity. But what if the node can hear signals from many CHs? In this case, the node has to choose one of these CHs as its new parent. The parameters that differentiate some CHs from others are: (1) the setup quality of the CH ($SQ$: obtained from the first fuzzy logic controller), (2) the remaining capacity that the CH can handle in bps ($RC$), and (3) the received signal strength of the CH ($RSS$) in meters; note that RSS’s unit is not originally in meter, but we used a simple formula to convert it to meters. In order to let the node decide on which CH it should join, we designed another fuzzy logic controller that combines the above 3 parameters and produces a single value defining the maintenance quality ($MQ$) of each CH. The node joins the CH with the highest $MQ$.

We also designed a third fuzzy logic controller which is used to define the quality of a certain route ($RQ$). When sending a message from a certain source to a certain destination, many paths are calculated to route the message. However, only one path should be selected. Each path is associated with a cost equal to the weight of the weakest node along the path (the node with the minimum residual energy). In addition to the cost of the path, the path has a certain length ($PL$) equal to the number of hops along the path. Both, the path cost and the path length are inputed to a fuzzy logic controller and a single output is produced that defines the route quality ($RQ$). The path with the higher $RQ$ is chosen as the routing path.
To summarize, we designed three fuzzy logic controllers with different objectives. The first controller is used in the initial setup phase and in subsequent phases and it is called the "Network Setup controller". The second controller is used in the network maintenance phase and it is called the "Network Maintenance controller". The third controller is used in the routing phase and it is called the "Routing controller". In this chapter, we explain each controller in details. But, before we do so, we give an overview of fuzzy logic and its uses.

5.1 Fuzzy Logic Overview

The idea of fuzzy logic was introduces by Prof. Lotfi Zadeh [117, 118, 119] and it continues to evolve till this day. Fuzzy logic is a generalization of standard logic. While standard logic applies only to concepts that are completely true or completely false, fuzzy logic applies to concepts that possess a degree of truth anywhere between 0 and 1. Fuzzy logic is supposed to be used for reasoning about inherently vague concepts, such as "tallness". In standard logic, one might say that people taller than 6 feet are tall, while people shorter than six feet are short. But, what about a person with height 5 feet 11 inches? We can not consider such a person short. In fuzzy logic, one might define overlapping regions to define whether a person is tall or short.

In figure 5.1, three triangular membership functions are defined to represent the height of a person. The x-axis is the person’s height. The membership functions can be of any shape (trapezoidal, gaussian, bell, etc.). The membership functions can be explained as follows:

1. Less than 4 feet, the person is "short".
Figure 5.1 Three membership functions defining the relative “tallness” of people

2. Between 4 feet and 4.5 feet, the person is ”short” with a high degree and ”average” with a low degree.

3. Between 4.5 feet and 5.5 feet, the person is ”short” with a low degree and ”average” with a high degree.

4. Between 5.5 feet and 6.5 feet, the person is ”average” with a high degree and ”tall” with a low degree.

5. Between 6.5 feet and 7 feet, the person is ”average” with a low degree and ”tall” with a high degree.

6. More than 7 feet, the person is ”tall”.

Figure 5.2 shows the steps that the fuzzy logic controller is composed of. The steps can be summarized as follows: (1) Receiving of one or more input values representing the measurements of the parameters to be analyzed or aggregated. (2) Subjecting the input
values to fuzzy "If-Then" rules. The rules can be expressed in plain language words, for example, "If a person is tall, back-pain is high". (3) Averaging and weighting the resulting outputs from all the individual rules into one single output decision. (4) Defuzzification of the output to get a "crisp" value between 0 and 1.

In general, two major components are needed to develop the fuzzy logic controller: (1) define membership functions for each input/output parameter and (2) design the fuzzy rules. The membership function is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs, define functional overlap between inputs, and determines an output response. The fuzzy logic rules use the input membership values as weighting factors to determine their influence on the output sets. The membership functions, discussed in this chapter, were designed to satisfy the following two conditions: (1) Each membership function overlaps only with the closest
neighboring membership function; (2) for any possible input data, its membership values in all the relevant fuzzy sets should sum to 1 (or nearly so). Next, we discuss the fuzzy logic controllers we designed along with their membership functions and rules.

5.2 Network Setup Controller

One of the major steps in our hierarchical network design is to select nodes that act as CHs. Many factors affect the choice of a CH. The CH should be able to handle the traffic \(T\) generated to/from its cluster nodes. Therefore, it should have high residual energy \(RE\). Also, the CH should not be too mobile \(M\) because this leads to high packet loss rate. There is a close correlation between the values of these parameters (residual energy, traffic, mobility). Because these parameters have different units and their values can be defined in ranges, we used fuzzy logic to express the effect of their interaction. Another important parameter which is not part of the controller is the node degree. We observed that the node degree is directly related to \(RE\). The higher the node degree, the higher the number of neighbors. Having more neighbors means that more traffic is going to be generated from them, thus putting more burden on the CH. Therefore, we combine the node’s RE and degree based on the following formula: 

\[
RE = \alpha RE + (1 - \alpha) Degree.
\]

Increasing \(\alpha\) gives more weight to RE and less weight to the degree, while decreasing \(\alpha\) gives less weight to RE and more weight to the degree. In our work we set \(\alpha\) to 0.7.

Note that \(RE\), \(T\), \(M\), and \(Degree\) are all normalized with respect to \(RE_{\text{max}}, T_{\text{max}}, M_{\text{max}}, \text{and } Degree_{\text{max}}\) respectively. In the centralized approach, these maximum values are considered to be the maximum among all network nodes. In the distributed approach, these
maximum values are considered to be the maximum among the node’s locality. Let $RE_i$, $M_i$, and $T_i$ be the residual energy, the mobility, and the traffic load of node $i$ respectively. Let $RE_{max}$ be the maximum residual energy among the 1-hop neighbors of $i$ including $i$. Similarly, let $M_{max}$ and $T_{max}$ be the maximum mobility and the maximum traffic load among node $i$ and its 1-hop neighbors. Node $i$ normalizes its own values with respect to the maximum values i.e.

$$RE_{n_i} = \frac{RE_i}{RE_{max}}, \quad Mn_i = \frac{M_i}{M_{max}}, \quad Tn_i = \frac{T_i}{T_{max}}$$

### 5.2.1 Residual Energy Membership Functions

![Figure 5.3](image)

**Figure 5.3** Three RE membership functions representing low, medium, and high RE

Residual energy is represented by 3 triangular membership functions as shown in figure 5.3. The triangular membership function is specified by three parameters $(a, b, c)$ as follows:

$$\text{triangular}(x : a, b, c) = \begin{cases} 
0 & x < a \\
\frac{(x - a)}{(b - a)} & a \leq x \leq b \\
\frac{(c - x)}{(c - b)} & b \leq x \leq c \\
0 & x > c 
\end{cases}$$  \ (5.1)
The x-axis represents the value of the residual energy in Joules (after applying $RE = \alpha RE + (1 - \alpha)Degree$). $RE_{avg}$ represents the average residual energy. If the controller was used in the centralized approach, $RE_{avg}$ is the average RE of all nodes in the network. If the controller was used in the distributed approach, $RE_{avg}$ is the average RE of all the node’s neighbors. The three triangular membership functions representing the $RE$ are marked by low, medium, and high. For example, the “medium” membership function (displayed in figure 5.3) is represented by $a = 20\% RE_{avg}$, $b = RE_{avg}$, and $c = RE_{max} - 20\% RE_{avg}$. Hence, $RE_{avg}$ is the center of the medium range. Nodes with REs less than $RE_{avg}$ are classified as medium and/or low, while those larger than $RE_{avg}$ are classified as medium and/or high.

5.2.2 Mobility Membership Functions

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image1.png}
\caption{Two mobility membership functions representing low and high mobility}
\end{figure}

Mobility is represented by 2 trapezoidal membership functions as shown in figure 5.4.
A Trapezoidal membership function is specified by four parameters \((a, b, c, d)\) as follows:

\[
\text{triangular}(x : a, b, c, d) = \begin{cases} 
0 & x < a \\
(x - a) / (b - a) & a \leq x \leq b \\
1 & b \leq x \leq c \\
(d - x) / (d - c) & c \leq x \leq d \\
0 & x \geq d 
\end{cases} \quad (5.2)
\]

For example, the "high" membership function is represented by \(a = 10\%\), \(b = 40\%\), \(c = D_{\text{max}}\), and \(d = D_{\text{max}}\). The mobility is measured by the change in the average received signal strength (RSS) between the node and its neighbors as it moves from one location to another. Each node translates the received signal strengths (RSS) to distances. Then, the average distance is calculated. A new average is taken after the node moves to a new location. The difference between both averages is fed to the fuzzy logic controller. Note that the difference computed can not be more than the distance a node can cover when it uses its highest power level. Therefore, the maximum distance is represented by \(D_{\text{max}}\) which is the distance covered when a node uses its maximum power level \((P_{\text{max}})\). The mobility model is designed to assign low mobility to nodes that have less than 10\% difference and high mobility to nodes that have greater than 40\% difference. A transition between low and high mobility occurs at 25\% difference.

### 5.2.3 Traffic Membership Functions

Traffic is represented by 3 triangular membership functions specified by the parameters given in equation 5.1. Figure 5.5 shows the graphical representation of the membership functions. The x-axis indicates the normalized value of the input traffic load. The max bit rate is considered to be 11Mbps (when using 802.11g). The three triangular functions
Figure 5.5 Three traffic membership functions representing low, medium, and high traffic determine a smooth transition between low, medium, and high traffic.

5.2.4 Output Membership Functions

Figure 5.6 Three output membership functions indicating whether the aggregated weight is bad, acceptable, or good.

The output (setup quality: $SQ$) is represented by 3 trapezoidal membership functions specified by the parameters given in equation 5.2. Figure 5.6 shows the graphical representation of the membership functions. They indicate whether the output is good, acceptable, or bad. The value of the output is a number between 0 and $O_{max}$. The smooth transition from bad and acceptable occurs between 5% and 45%, while the smooth transition from acceptable to good occurs between 55% and 95%. The higher the value of the output, the better the node quality.
5.2.5 Network Setup Controller Rules

The fuzzy logic rules are used to combine the residual energy, mobility, and traffic keeping in mind the synergy between them. Six fuzzy logic rules are defined as follows:

1. If RE is high, \( SQ \) is good
2. If RE is low, \( SQ \) is bad
3. If RE is medium AND mobility is low, \( SQ \) is acceptable
4. If RE is medium AND traffic is medium, \( SQ \) is acceptable
5. If RE is medium AND traffic is low, \( SQ \) is acceptable
6. If mobility is high OR traffic is high, \( SQ \) is bad

The above rules summarize the CH properties, i.e. the CH is preferred to have: (1) high residual energy, (2) low mobility, and (3) low traffic.

5.3 Network Maintenance Controller

The first fuzzy logic controller was used to determine the quality of a node (\( SQ \)). \( SQ \) is then used by our centralized and distributed approaches (discussed later) in order to cluster the network. Initially each node in the network is associated with a CH. As time passes, the configuration of the network changes and some nodes might move away from their CH’s proximity. In order to preserve the hierarchical structure, such nodes should change their old CH and join a new one (cluster maintenance is discussed in details in Chapter 7). In most cases a moving node will be able to hear signals from many new CHs. But, the node
has to choose only one CH. The node evaluates the CHs it can hear and then decides on joining one of them. The evaluation criteria is based on the following parameters: (1) \( SQ \) of the CH, (2) the remaining capacity that the CH can handle (\( RC \)), and (3) the received signal strength of the CH (RSS). The moving node uses a fuzzy logic controller, called Network Maintenance controller, to combine these parameters and produce a single value defining the maintenance quality (\( MQ \)) of each CH. The CH with the highest \( MQ \) is chosen as the node’s new parent. In the following sections, we discuss the membership functions of each parameter and provide the fuzzy logic rules that combine these parameters.

### 5.3.1 Setup Quality Membership Functions

![Figure 5.7](image)

**Figure 5.7** Three \( SQ \) membership functions representing low, medium, and high \( SQ \)

\( SQ \) is represented by 3 triangular membership functions specified by the parameters given in equation 5.1. Figure 5.7 shows the graphical representation of the membership functions. Recall that \( SQ \) is the output of the first fuzzy logic controller (Network Setup controller). The x-axis indicates the normalized value of \( SQ \). The membership functions present a perfect transition from low, medium, to high \( SQ \). \( SQ \) is very important to the maintenance phase because it indicates how powerful the CH is. However, it is not suffi-
cient. The CH might have a high $SQ$, but it might not be able to handle any extra traffic. Moreover, the CH might have a high $SQ$, but it might be far away from the node. For these reasons, we consider the remaining capacity and the received signal strength of the CH as deciding factors.

### 5.3.2 Remaining Capacity Membership Functions

![Remaining Capacity Membership Functions](image)

Figure 5.8 Three $RC$ membership functions representing low, medium, and high $RC$.

RC is represented by 3 triangular membership functions specified by the parameters given in equation 5.1. Figure 5.8 shows the graphical representation of the membership functions. The x-axis indicates the remaining capacity that the CH can handle. If the remaining capacity is less than 20% of the total capacity, we assume that the CH has a low capacity. This means that the CH can barely handle any extra traffic. If the remaining capacity is more than 20%, we assume that the CH can handle extra traffic. A smooth transition exists when going from low, medium, to high capacity.

### 5.3.3 Received Signal Strength Membership Functions

RSS is represented by 3 trapezoidal membership functions specified by the parameters given in equation 5.2. Figure 5.9 shows the graphical representation of the membership
Figure 5.9 Three RSS membership functions representing weak, average, and good RSS functions. The x-axis indicates the RSS of the CH. RSS indicates the proximity of the node with respect to the CH. If the node is close to the CH, RSS will be good. If the node is far away from the CH, RSS will be weak. The higher the RSS, the closer the node is to the CH and the lower energy it consumes to communicate with it. If RSS is greater than 60%, we assume that the node is close to the CH and therefore it matches the "good" membership function. On the other hand, if RSS is less than 10%, we assume that the node is barely reaching the CH and therefore it matches the "weak" membership function. A transition between the "weak" and the "good" membership functions exists between 10% and 60%.

5.3.4 Output Membership Functions

Figure 5.10 Three MQ membership functions representing bad, acceptable, and good MQ

The output (MQ: maintenance quality) is represented by 3 triangular membership func-
tions specified by the parameters given in equation 5.1. Figure 5.5 shows the graphical representation of the membership functions. They indicate whether the output is good, acceptable, or bad. The value of the output is driven solely by the fuzzy logic rules (discussed next). The node joins the CH with the highest $MQ$.

### 5.3.5 Network Maintenance Controller Rules

The following rules are used to combine $SQ$, $RC$, and $RSS$. They indicate that a candidate CH should have: (1) high $SQ$, (2) high $RC$, and (3) high $RSS$. Five rules are defined as follows:

1. If $SQ$ is low OR $RSS$ is weak OR $RC$ is low, $MQ$ is bad
2. If $SQ$ is medium AND $RSS$ is average AND $RC$ is medium, $MQ$ is acceptable
3. If $SQ$ is medium AND $RSS$ is good AND $RC$ is medium, $MQ$ is good
4. If $SQ$ is high AND $RSS$ is average AND $RC$ is high, $MQ$ is acceptable
5. If $SQ$ is high AND $RSS$ is good AND $RC$ is high, $MQ$ is good

### 5.4 Routing Controller

So far, we discussed the design of two fuzzy logic controllers. The first one is used in network setup and the second one is used in network maintenance. In this section, we discuss the third and last controller in our work. This controller is used to make routing decisions. When sending a message from a certain source to a certain destination, many paths are calculated to route the message. However, only one path should be selected. Each
path is associated with a cost (path cost) equal to the residual energy of the weakest node along the path (the node with the minimum residual energy). Note that the higher the path cost, the better the route is. Moreover, each path has a length (PL) equal to the number of hops along the path.

Most routing protocols use the shortest path to route message between source and destination pairs. Choosing the shortest path is not a good idea specially when energy conservation is one of the network design goals. Nodes along the shortest paths will carry high computation and communication burden and their batteries will deplete quickly. For this reason, we do not use the shortest path to route messages. Instead, we try to find a balance between the path length and the path cost. A preferable path is the one that has few number hops consisting of nodes with high RE, i.e., low PL and high path cost. In order to achieve such balance, we design a fuzzy logic controller, called Routing controller, that combines the path cost and the path length keeping in mind the synergy between them. The controller produces a single output that defines the route quality (RQ). The path with highest RQ is chosen to route messages between (s, d) pairs.

5.4.1 Input/Output Membership Functions

The "path cost" membership functions have the same representation as that in figure 5.3. Path cost is measured relative to the weakest node along the path, because the quality of the path is as good as the quality of the weakest node. The same discussion presented in section 5.2.1 applies here.

The "path length (PL)" membership functions have the same representation as that in
figure 5.8 with the following differences: (1) $R_{\text{max}}$ changed to $PL_{\text{max}}$, (2) "low" changed to "short", (3) "medium" changed to "average", and (4) "high" changed to "long". $PL$ is simply the number of hops along the path. In many cases short $PL$s are preferable, but not in all cases. Routing through the shortest path all the time will drain the nodes’ batteries along the path. On the same hand, having a long path with powerful nodes is not preferable, because a large number of nodes will be affected. If $PL$ is less than 20% the longest path, we assume that $PL$ is short (good criteria). If $PL$ is more than 80% the longest path, we assume that $PL$ is long (bad criteria). Otherwise, $PL$ is a combination of short, average, and long.

The "output ($RQ$)" membership functions have the same representation as that in figure 5.10. The functions indicate whether the output is good, acceptable, or bad. The path with the highest $RQ$ is chosen as the routing path.

5.4.2 Routing Controller Rules

The following rules combine path cost and path length and decide when is the route bad, acceptable, or good. A preferable path is the one having high cost (high cost means that the nodes along the path have high residual energy) and few number of hops (short $PL$). Five rules are defined as follows:

1. If path cost is low, $RQ$ is bad
2. If path cost is medium AND $PL$ is short, $RQ$ is good
3. If path cost is medium AND $PL$ is average, $RQ$ is acceptable
4. If path cost is medium AND PL is long, RQ is acceptable

5. If path cost is high, RQ is good

5.5 Controllers Achieve Energy-Efficiency

Each of the controllers discussed above is used in a different phase of the network lifetime. The first controller (Network Setup controller) is used in the setup phase. The second controller (Network Maintenance controller) is used in the maintenance phase. And the third controller (Routing controller) is used in the routing phase. Even though the controllers are used in different phases, they all share the Energy-Efficiency property, which is one of our important design objectives. Next we explain how each controller achieves energy-efficiency.

- Network Setup controller is used to aggregate the node’s residual energy (RE), traffic (T), and mobility (M). The controller’s Fuzzy Rules puts more emphasize on RE. If RE is low, the quality of the node is bad no matter what the values of T and M are. If RE is high, but T and/or M are high, the quality (SQ) of the node is pulled down. Having high T and M is not preferable because they lead to high energy consumption. This fuzzy logic controller gives a high weight for nodes that have high residual energy, low traffic, and low mobility. Therefore, the objective of the first controller is to give high quality for nodes that can survive a long time. Thus, achieving energy-efficiency and prolonging network lifetime.

- Network Maintenance controller is used by moving nodes in order to evaluate the
CHs they can hear. CH evaluation depends on $SQ$, remaining capacity ($RC$), and received signal strength ($RSS$). The controller gives more weight for CHs that have: high $SQ$, high $RC$, and high $RSS$. High $SQ$ means that the CH can survive for a long time and thus prolonging network lifetime (discussed above). High $RC$ means that the CH can handle extra traffic, i.e., the CH is powerful enough to take more duties. This is important in balancing the traffic load among all the CHs in the network. Balancing the traffic load among all CHs results in a more survivable network [90, 108, 110, 112, 113, 114]. High $RSS$ means that the node is close to the CH. It also means that the node requires less energy to reach the CH and vice versa. So, high $RSS$ translates to energy conservation. Therefore, the objective of the second controller is to associate a moving node with an appropriate CH such that network lifetime is prolonged.

- **Routing** controller is used by CHs to choose an appropriate path to route messages between (s, d) pairs. The choice is governed by the length of the path and its cost. If the path is long, more CHs are involved in relaying the packets. So, the residual energy of many CHs will be affected. For this reason, a short path is preferred. The second parameter, Path cost, is decided by the residual energy of the weakest node along the path. Paths with low cost (a node with low RE along the path) are not preferable. The Routing controller combines the path length and cost giving more preference for paths that are short with powerful nodes. Therefore, the objective of the third controller is to choose energy-efficient routes between (s, d) such that CHs’
life are prolonged.

5.6 Summary

In this chapter, we described three fuzzy logic controllers each having a different objective. The controllers are: (1) "Network Setup controller", (2) "Network Maintenance controller", and (3) "Routing controller". The first controller is used to assign a weight for every node in the network by aggregating the node’s residual energy, traffic load, and mobility. The calculated weight is then used to elect CHs, create clusters, and setup the initial hierarchical structure. The second controller is used by moving nodes in order to evaluate the CHs they can hear by aggregating the CHs’ SQ, remaining capacity, and received signal strength. The higher the output of the controller, the better the CH candidate. The third controller is used as part of the routing protocol. When more than one path exist between (s, d) pairs, the controller evaluates each path and gives a higher weight to the better path. A good path is a short path with powerful nodes.

The most important property of the fuzzy logic controllers is that of energy-efficiency, i.e., powerful nodes are elected in the setup phase, powerful nodes are selected in the maintenance phase, and energy-efficient routes are chosen in the routing phase. The controllers defined in this chapter are used in Chapters 6 and 7 to design a hierarchical routing protocol that maximizes network lifetime.
CHAPTER 6
CENTRALIZED APPROACH

Recall from Chapter 1 that our objectives are to overcome the scalability and the energy constraint problems that exist in MANETs. In this chapter, we propose a fuzzy-based hierarchical energy-efficient design (FEER) that aims to maximize network lifetime. Network lifetime is defined by the time when the network is partitioned or nonoperational. The network might get partitioned because of nodes’ malfunction (battery drainage) and/or nodes’ mobility. So, how do we provide scalability and energy-efficiency?

We provide scalability by dividing the network into a group of clusters. Each cluster is represented by a cluster head (CH). Nodes inside a cluster (members) wishing to communicate with nodes outside the cluster, have to start by relaying their messages to their parent CH. CHs can then communicate with each other to deliver the message. Nodes inside the same cluster that can reach each other can communicate directly without contacting their parent CH. The collection of CHs form the backbone network, which has to be connected at all times. As mentioned in Chapter 3, hierarchical approaches are well-known for increasing scalability [51, 55, 105]. They also have other benefits such as: (1) they facilitate the special reuse of resources [62], (2) data flooding only involves CHs, and (3) routing is localized within a cluster. However, the hierarchical approach has some side effects that include: (1) setup time, (2) ripple effect, and (3) maintenance.

We provide energy-efficiency by employing fuzzy logic techniques and using fast custom made algorithms. FEER utilizes the Network Setup controller (section 5.2) to aggre-
gate the following parameters: node’s residual energy (RE), node’s traffic (T), and node’s mobility (M). It also produces a weight, named $SQ$, that represents the quality of the node. A node with high $SQ$ is a candidate of becoming a CH. Network Setup controller plays an important role in achieving energy-efficiency by giving high quality for nodes that can survive a long time (section 5.5). FEER also uses some algorithms, modified to work well in our design, such as: dominating set approximation, min-cut, and max-flow.

Each node in the network starts by approximating its mobility. Mobility is measured by the change in the average received signal strength (RSS) between the node and its neighbors as it moves from one location to another (section 5.2.2). Each node then passes its parameters (RE, T, and M) to the Network Setup controller, which returns a weight $SQ$. All nodes in the network send their weight and neighbor list to a central processing node (CPN). The CPN can be a fixed base station. In the absence of a base station, the CPN can be the node with the highest $SQ$. A distributed leader election algorithm [24] is used to find a CPN. The CPN runs FEER and obtains the following information: (1) which nodes are CH, (2) which nodes are connected to which CHs, and (3) which CHs are connected to each other. The CPN then advertises this information allowing each node to know its role in the network. FEER can be divided into four parts: (1) elect nodes to act as CHs (CH Election), (2) associate each node in the network with a CH (Cluster Formation), (3) introduce a network recovery approach to ensure a fault tolerant backbone (Network Recovery), and (4) design energy-efficient routing between nodes (Routing). Before we start explaining each step, we present our system models.
6.1 System Models

This section describes the models we use in our work. The models are: (1) energy model, (2) mobility model, and (3) traffic model. Note that these models are also used in the distributed approach (Chapter 7).

6.1.1 Energy Model

In our work, we assume a model where the radio dissipates more energy while transmitting than receiving. Each node has a battery with limited residual energy. Each node, equipped with antennas, can control its transmission power (power level). The higher the power level, the more distance a node covers and the more energy it consumes. The lifetime of a node depends on: (1) the traffic load the node is routing, (2) the energy consumed while transmitting or receiving the traffic load, and (3) the residual energy on the node. These parameters should satisfy the following inequality:

\[
load(b/s) \times Communication(J/b) \times t(s) \leq RE(J)
\]  

(6.1)

Where, \(load\) is the amount of traffic passing through a node in bits per second (b/s), \(Communication\) is the amount of energy dissipated by the node when transmitting, receiving, or both in joules per bit (J/b), and \(RE\) is the node’s residual energy in joules (J). Let \(t\) be the lifetime of the node in seconds (s), then the above inequality can be rewritten as:

\[
t(s) \leq \frac{RE(J)}{load(b/s) \times Communication(J/b)}
\]  

(6.2)

i.e., the node dies when the energy consumed by communication exceeds its own residual energy. The equation used to calculate the energy consumed when a node communicates is
given by:

\[ E_{\text{consumed}} = (\text{packetLength/bitRate}) \times P_c \] (6.3)

Where, \( P_c \) is the power consumed if the node is transmitting a packet, receiving it, or both.

### 6.1.2 Mobility Model

Evaluating the performance of a mobile Ad Hoc network highly depends on the mobility model used. The mobility model should dictate the movement of the mobile nodes in a realistic way. Two of the most mobility models used by researchers are the Random Walk Mobility Model [31] and the Random Waypoint Mobility Model [11, 46]. Each one of these models generates unrealistic scenarios that make them inappropriate for mobile Ad Hoc network simulation. An alternative is to use the Gauss-Markov Mobility Model [101] that fixes the problem encountered by the previous two models.

The Random Walk Mobility Model was developed to mimic the erratic movement of entities in nature that move in unpredictable ways [31]. A mobile node moves from one location to another by choosing two random values corresponding to speed and direction. Speed and direction are chosen to be within predefined ranges, \([\text{speedmin}, \text{speedmax}]\) and \([0, 2\pi]\) respectively. After a certain time period \( t \), or a distance \( d \), new values for speed and direction are generated. No relation exists between the current and the past movements of the node. This might lead to unrealistic scenarios, where a node stops suddenly or makes sharp turns. Also, if the time period \( (t) \) or the distance \( (d) \) were small values, the node will be moving abruptly in a small region.

The Random Waypoint Mobility Model uses pause time between changes in speed.
and/or direction [11, 46]. A mobile node starts by pausing for a certain time period. Then it moves from one location to another by choosing two random values corresponding to speed and destination. Speed is chosen to be uniformly distributed between $[\text{minspeed}, \text{maxspeed}]$. The mobile node travels towards the new destination at the selected speed. Upon arrival, it pauses for a certain period of time and then starts the process again. The Random Waypoint movement is similar to the Random Walk movement when the pause time is 0. Hence, the Random Waypoint Mobility Model suffers from the same problems that the Random Walk Mobility Model suffers from.

To eliminate the problems (sudden stops and sharp turns) encountered by the Random Walk and the Random Waypoint mobility models, we use the Gauss-Markov Mobility Model. The Gauss-Markov Mobility Model was originally proposed for the simulation of a personal communications service (PCS) [58]; however, this model has been used for the simulation of an ad hoc network protocol [101]. The main advantage of this model is allowing past velocities (and directions) to influence future velocities (and directions). A mobile node starts moving using a current speed and direction. At fixed intervals of times, $n$, new speed and direction values are assigned to the mobile node. These values are calculated based on the values used in the previous time interval and a random variable. The speed and direction at the $n^{th}$ instance are given by the following equations:

$$s_n = \alpha s_{n-1} + (1 - \alpha) \bar{s} + \sqrt{(1 - \alpha^2) s_{x,n-1}}$$

(6.4)

$$d_n = \alpha d_{n-1} + (1 - \alpha) \bar{d} + \sqrt{(1 - \alpha^2) d_{x,n-1}}$$

(6.5)

where $s_n$ and $d_n$ are the new speed and direction of the mobile node at time interval $n$;
\( \alpha, 0 \leq \alpha \leq 1 \), is the tuning parameter used to vary the randomness; \( \bar{\alpha} \) and \( \bar{d} \) are constants representing the mean value of speed and direction as \( n \to \infty \); \( s_{x_{n-1}} \) and \( d_{x_{n-1}} \) are random variables from a Gaussian distribution. Totally random values (or Brownian motion) are obtained by setting \( \alpha = 0 \) and linear motion is obtained by setting \( \alpha = 1 \). Intermediate levels of randomness are obtained by varying the value of \( \alpha \) between 0 and 1.

At each time interval, the next location is calculated based on the current location, speed, and direction of movement. Specifically, at time interval \( n \), the mobile node’s position is given by the equations:

\[
\begin{align*}
  x_n &= x_{n-1} + s_{n-1} \cos(d_{n-1}) \\
  y_n &= y_{n-1} + s_{n-1} \sin(d_{n-1})
\end{align*}
\]

(6.6) 
(6.7)

where \((x_n, y_n)\) and \((x_{n-1}, y_{n-1})\) are the \( x \) and \( y \) coordinates of the mobile node’s position at the \( n^{th} \) and \((n - 1)^{st} \) time intervals, respectively, and \( s_{n-1} \) and \( d_{n-1} \) are the speed and direction of the mobile node, respectively, at the \((n - 1)^{st} \) time interval [15]. The Gauss-Markov Mobility Model can eliminate the sudden stops and sharp turns encountered by the models discussed above by allowing past velocities (and directions) to influence future velocities (and directions).

6.1.3 Traffic Model

![Poisson Traffic Model: exponential inter-arrival time and exponential holding time](image)

**Figure 6.1** Poisson Traffic Model: exponential inter-arrival time and exponential holding time
Traffic generation is simulated using the Poisson distribution. Traffic modeled after the Poisson distribution has exponential inter-arrival time and exponential holding time as shown in figure 6.1.

Exponential distributions are a class of continuous probability distribution. They are often used to model the time between events that happen at a constant average rate [106]. The probability density function of the exponential distribution is given by:

\[
f(x; \lambda) = \begin{cases} 
\lambda e^{-\lambda x}, & x \geq 0 \\
0, & x < 0 
\end{cases}
\] (6.8)

where \( \lambda > 0 \) is a parameter of the distribution, often called the rate parameter. The exponential distribution can alternatively be parameterized by a scale parameter \( \mu = \frac{1}{\lambda} \).

6.2 CH Election

Given a random configuration of an Ad Hoc network, the first task is to adjust the power level of each node in order to get a connected network. Each node can switch between six power levels. The algorithm starts with all the nodes using their minimum power level. If node A is reachable by node B, a communication link is created between A and B. The topology created is tested for connectivity using the "disjoint set union" algorithm which is almost linear. The algorithm runs in \( O(n \alpha) \), where \( n \) is the number of nodes in the network and \( \alpha \) is a small number (\( \alpha \leq 5 \) for all \( n \leq 2^{16} \)). If the network is not connected, the power level at each node is incremented and the process is repeated until a connected network is established (if there is one). Note that the power level at each node will be lowered as much as possible after constructing the hierarchical structure.
Having established a connected network, we propose a simple algorithm to elect the CHs. Since the CHs form the backbone network, they must be connected. Algorithm 6.1 shows the pseudo code for electing the virtual backbone. Note that the algorithm is executed by the CPN. The CPN knows the weight \((SQ)\) of every node in the network; \(SQ\) is the weight obtained from the Network Setup controller. After executing the algorithm, the CPN possesses the hierarchical structure of the algorithm.

\begin{verbatim}
begin
  x = ID of the node having the maximum SQ
  Elect node x as a CH
  Add neighbors of x to vector Ne
  repeat
    x = ID of the node having the maximum SQ in Ne
    if x covers new nodes then
      Elect node x as a CH
      Add neighbors of x to vector Ne
      Remove x from vector Ne
    endif
  until Network is fully covered
end

Algorithm 6.1  CH Election
\end{verbatim}

The node with the highest \(SQ\) is chosen to be the first cluster head. The neighbors of the first CH are stored in vector \(Ne\). From the nodes in \(Ne\), the node with the highest \(SQ\) is chosen to be the next CH. If the new chosen CH does not cover any new nodes, it is ignored and another CH is elected. The neighbors of the newly elected CH are added to vector \(Ne\). This step is repeated until all the network nodes are covered. After each iteration, the elected CHs are removed from \(Ne\). In the worst case, the algorithm needs to visit all \(n\) nodes in the network. In the worst case its complexity is \(O(n)\). The advantage
of this approach is that there is no need to know the number of CHs a priori. The CHs are chosen automatically by the algorithm.

Figure 6.2  Elected CHs (gray rectangles) when the CH election phase is complete

Figure 6.2 shows the elected CHs when algorithm 6.1 is executed. The number enclosed in a circle/rectangle represents the node ID. The number next to the node represents the weight ($SQ$) of the node. The nodes enclosed in gray rectangles are the elected CHs. Node 8 is elected first as a CH because it has the highest weight in the network. From the neighbors of node 8, node 10 is elected next as a CH. Then nodes 6, 4, and 12 respectively, thus covering the whole network.

6.3 Cluster Formation

After the selection of the CHs, each node associates itself with a CH to form a cluster. Nodes in a cluster are of two types: one-hop nodes and two-hop nodes. One-hop nodes are the direct neighbors of the CH. Two-hop nodes, also called guests, are nodes that can reach the cluster head through a one-hop neighbor.
The idea of a guest node is introduced by [116]. One reason for using guest nodes is that 1-hop clustering schemes form a highly overlapping cluster structure with a large number of small clusters. Such a structure may cause difficulties in the channel spatial reuse and thus leading to low network capacity. So, using cluster guests reduces the number of clusters. Another reason for using cluster guests is to avoid the ripple effect (re-clustering the whole network from scratch). A mobile node that moves out of the CH range can join a close cluster as a guest rather than re-clustering.

Nodes connect to the CH with the maximum lifetime (equation 6.2). A node might have the ability to reach more than one CH using 1-hop or 2-hops. The lifetime of each CH is calculated and the one with the maximum lifetime is chosen. The algorithm connects nodes having higher traffic first. If the chosen CH was 1-hop away from the node, the node connects directly to that CH and becomes part of the cluster. If the chosen CH was 2-hops away from the node, another node (connector) is needed to connect the node to the CH. Thus, the node and the CH should share at least one neighbor. The neighbor with maximum lifetime is chosen to act as a connector. Note that the lifetime of the nodes is calculated considering whether the node is: transmitting, receiving, or both transmitting and receiving. A node not assigned to a cluster is connected to a neighboring CH with maximum lifetime.

Figure 6.3 presents the cluster formation of the network presented in figure 6.2. Random traffic patterns have been produced to simulate the idea presented above. In most cases, nodes join the CH that has a high weight and is close to them. This is not the case for node 5. Node 5 was supposed to join CH 4. But because CH 4 already has nodes 1, 2,
Figure 6.3  Cluster formation

3 and 7 as members, it needs to handle lots of traffic and thus its weight dropped bellow 49 (the weight of CH 6). When node 5 executed the Cluster Formation algorithm, it found out that it is better to join CH 6 rather than CH 4. Note that no 2-hop neighbors exist in the initial setup. But this will change when nodes start moving.

At this point each node is associated with a cluster. Nodes can not communicate directly unless they go through the CHs they are connected to. The above algorithm achieves a hierarchical structure that maximizes the network lifetime, because nodes with minimum lifetime are avoided. The power of this algorithm lies in the fact that the traffic demand and mobility are integrated within the algorithm and the clusters are chosen to best handle the traffic demand. The algorithm also determines the traffic load that each CH needs to handle. These loads are routed through other CHs in order to reach the destination. But, what if the channel capacity between CHs can not handle those loads? Or, what if one of the links was down because of the channel impairment? These issues can be solved by providing the recovery algorithm (discussed next).
6.4 Network Recovery

Having designed an energy efficient hierarchical structure, it is important to make the network resilient to link failure and that the CHs can handle the traffic flow. To check for resiliency and traffic demand, the min-cut and the max-flow algorithms are used (defined later). The min cut in the network should be greater than or equal to the network fault tolerance requirement \( M \) and the maximum flow should be satisfied. If one or both conditions fail, appropriate recovery techniques are deployed. The network created by CHs is represented by a weighted graph \( G = (V, E) \), where \( V \) is the set of vertices and \( E \) is the set of edges and the weight on each edge represents its capacity.

Min Cut recovery: The min cut of a graph \( G \) is the minimum number of edges needed to disconnect the graph. We used a simple randomized min-cut algorithm to find the min-cut. Repeatedly select an edge randomly and collapse it, reducing the number of vertices by 1 each time, until 2 edges are left. The number of edges that exist between the 2 nodes is the min-cut. The pseudo code is given in algorithm 6.2.

\begin{verbatim}
begin
  G = (V, E), |V| = n
  H = G
  repeat
    Choose a random edge e in H
    Contract edge e
  until 2 nodes are left
  Return the number of edges between the 2 nodes
end
\end{verbatim}

**Algorithm 6.2** Randomized min-cut algorithm
But, what is the probability that the resultant min-cut is the right one? If the min cut has $k$ edges, the graph must have at least $\frac{nk}{2}$ edges (each vertex must have at least $k$ edges connecting to it, or it would be a smaller cut by itself). The probability $p$ of picking one of those $k$ edges in the first merging step is thus $p \leq \frac{k}{nk/2} = \frac{2}{n}$. So, the probability $q$ of not picking one of the $k$ edges is $q \geq 1 - \frac{2}{n}$. The probability $q_1$ of not picking one of the $k$ edges in the second step is $q_1 \geq 1 - \frac{2}{n-1}$. Repeating this argument $n-2$ times gives $\frac{2}{n(n-1)}$. Therefore, the probability of finding the min-cut in the first round is $\frac{2}{n(n-1)}$. The algorithm can be repeated many times ($n^2$ is an appropriate number) in order to increase the probability of getting the correct min-cut. The complexity of the min-cut algorithm is $O(n)$. The total complexity depends on how many times the algorithm is repeated.

If the min-cut was greater than or equal to 2 ($M = 2$), then the network is considered to be reliable and the max-flow is checked. If the min-cut was equal to one, one link failure can bring the network down. Two recovery approaches can be used to increase the min-cut and thus increase the reliability of the network. The first approach is an iterative approach that increases the number of cluster heads without increasing the power level. Suppose that cluster heads $A$ and $B$ are connected by a single link, the approach finds all the neighbors of $A$ and $B$ and chooses the neighbor with the maximum lifetime as a new CH. This algorithm is repeated for any two CHs having a single link between them. Thus, more paths are generated between CHs and fault tolerance is increased. If the first approach was not able to increase the min-cut, the power level of each node is increased and the whole process is repeated.

It can be clearly seen from figure 6.3 that the min-cut is 1. Neighboring CHs (those
that have only edge between them) having the highest weight start the network recovery phase. CHs 8 and 10 elect CH 9 as a new CH. Electing CH 9 increases the number of links between CHs (8, 10) and CHs (10, 12) by 1. Also CHs 6 and 4 run the network recovery algorithm and elect node 5 as a CH. Electing CH 5 increases the number of links between CHs (4, 6) and CHs (6, 8) by 1. Note that node 11 changed its parent CH to CH 8 to CH 5. Figure 6.3 shows the final hierarchical network which is fully operational.

Max-flow recovery: If the min-cut condition was satisfied \( (\text{min-cut} > 1) \), a max-flow algorithm is applied between every \((s, d)\) pairs to check if the traffic flow can be handled by the links. The algorithm used is the Edmonds-Karp algorithm and it runs in \( O(ve^2) \). If the max-flow is satisfied, the network hierarchical structure becomes complete; else the second recovery method is used, i.e., the power level at each node is increased and the whole process is started from scratch.

The complexities of the min-cut and max-flow algorithms are very high. But, both algorithms are applied on the network created by the cluster heads which has a small number
of nodes compared to the number of nodes in the whole network. At this point the energy efficient hierarchical design is almost complete. Two more steps are left: lowering the power level at each node and routing between cluster heads.

6.5 Routing

Remember that our approach started by assigning the same power level to all the nodes. Also the power level might be incremented in the recovery algorithm. After the whole design is complete, every node in the network belongs to a certain cluster. If the node is 1-hop away from the cluster head, its power level is lowered until the node can reach the cluster head with minimum power. If the node is 2-hops away from the cluster, the node will be connected to a connector. The power level of the node is decreased to the lowest power level it needs to reach the connector node. The connector’s power level is decreased to a point where it can reach both the node and the cluster head.

The routing is composed of three steps: (1) if the source is not a CH, it sends its message to its representative CH, else it does nothing. (2) Now the CH acts as the new source. The $k$-shortest path algorithm is used to get different paths from the source CH to the destination CH. Each path is assigned a cost which is equivalent to the minimum lifetime of a node across the path. The path with the max cost (max lifetime) is used to route the traffic to the destination. Along the path, each node updates its lifetime. (3) If the destination CH is not the final destination, it relays the message to the final destination node in its cluster.
Table 6.1  Nodes’ power level properties

<table>
<thead>
<tr>
<th>Power Level</th>
<th>Transmit Power (dBm)</th>
<th>Receive Power (dBm)</th>
<th>Distance (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>20</td>
<td>-70</td>
<td>302</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>-70</td>
<td>240</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>-70</td>
<td>170</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>-70</td>
<td>135</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>-70</td>
<td>107</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>-70</td>
<td>76</td>
</tr>
</tbody>
</table>

6.6 Simulation Results

In this section, we evaluate the performance of FEER via simulation. We compare our approach to the dominating set approach suggested by Wu et al. in [108]. Each node in the network can transmit data using 6 different power levels. The higher the power level the more energy the antenna needs to transmit the signal and the more distance the signal can reach (Table 6.1). The power consumed by the battery when transmitting data is 1749 mW and when receiving is 930 mW. The battery of each node has energy up to 10000 Joules.

Figure 6.5 shows a network of 40 nodes designed using FEER. The black nodes are CHs. Each cluster is composed of the CH and the nodes associated with it. These nodes can be connected to a CH using 1-hop or 2-hop nodes. 9 nodes are elected to act as CHs. Figure 6.6 shows the design of the same network using Wu’s approach. Wu’s approach elects 25 nodes to act as CHs which is almost 3 times more that FEER.
The first experiment we conducted varies the network size and measures the network lifetime. Each node sends packets at a rate up to 500kbps. Each packet ranges in length from 50 bytes to 2400 byte. The energy consumed at each node is calculated using equation
Figure 6.7 Network lifetime vs. network size. A maximum bit rate of 500kbps is used at each node.

6.3 (section 6.1.1). Figure 6.7 shows that as the network size increases, FEER produces longer lifetime than Wu’s approach. Note that networks with large sizes have low lifetimes. This is true because in large networks, the cluster heads will be representing a larger number of nodes and thus their energy is quickly depleted. Using FEER, the lifetime of the network is prolonged by almost 20%. Note that in order to speed up the process of collecting results, we lowered the RE of each node. Then we calculated the percentage improvement of FEER over Wu’s approach.

Figure 6.8 shows the number of elected cluster heads using FEER and Wu’s approach. FEER chooses far fewer cluster heads than Wu’s approach. Even with fewer CHs, FEER still improves the network lifetime. For instance, when the network size is 100, FEER elects 24 cluster heads while Wu’s approach elects 74.

Another experiment was conducted to check the lifetime of the network under heavy
traffic. A network of 100 nodes is tested under different data rates that range from 50kbps to 500kbps. Figure 6.9 shows that as the data rate increases, FEER still achieves longer lifetime (20% better) than Wu’s approach. Both approaches though feature a decreasing lifetime as the data rate increases.

6.7 Summary

In this chapter, we explained our proposed centralized approach (FEER). FEER is a fuzzy-based energy-efficient hierarchical routing protocol that aims to maximize the network lifetime. Network lifetime is defined by the time when the network is partitioned or nonoperational. FEER is composed of four steps: (1) CH election, (2) cluster formation, (3) network recovery, and (4) routing. Each step has unique properties that distinguishes it from the approaches suggested in literature.

CH Election (section 6.2) is an approximation of the weighted connected dominating
Figure 6.9  Network lifetime vs. bit rate. The experiment is made on a network of 100 nodes.

set (CDS). Nodes that belong to the CDS are considered to be CHs. This step is unique in the way it assigns weights to network nodes. FEER utilizes the Network Setup controller (section 5.2) to produce a weight, named $SQ$, that represents the quality of the node. A node with high $SQ$ is a candidate of becoming a CH. Network Setup controller plays an important role in achieving energy-efficiency by giving high quality for nodes that can survive a long time (section 5.5).

Cluster Formation (section 6.3) connects each node in the network to a CH. This step uses the idea of guest nodes, which is very useful for cluster maintenance and spacial reuse. 1-hop nodes connect to the CH with the maximum lifetime (equation 6.2). 2-hop nodes connect to the connector with the maximum lifetime. The algorithm connects nodes having higher traffic first. So, FEER takes into account the traffic flow and the capability of the CHs when forming the clusters which adds to the energy-efficiency of FEER.
Network Recovery (section 6.4) aims to design a backbone that is resilient to link failure. It also makes sure that all CHs can handle the traffic demand generated from their clusters. FEER uses the min-cut and the max-flow algorithms to check the resiliency and the capacity of the backbone. If any one of the tests fails, FEER uses a unique iterative approach that increases the number of cluster heads without increasing the power level. Network recovery adds more paths between CHs. This enables the routing protocol to choose energy-efficient routes between (s, d) pairs.

The routing protocol (section 6.5) completes the design by introducing energy-efficient paths between (s, d) pairs. One objective of the network recovery phase (discussed before) is to create more paths between backbone nodes. The routing protocol evaluates each path and chooses the path that is most energy-efficient. Simulation results (section 6.6) show that FEER succeeds in prolonging the network lifetime. Moreover, it outperforms other approaches suggested in literature.

Even though FEER seems to be a good energy-efficient protocol, it is still a centralized approach. Centralized approaches do not scale well, and this contradicts with one of our objectives; designing a protocol for large scale networks. We used the centralized approach as an intermediate step to get to our ultimate goal which is “designing a distributed approach with the same goals”. Our proposed distributed approach is distinguished by its (1) fast and efficient setup, (2) fast and efficient maintenance, and (3) efficient routing. Chapter 7 discusses the distributed approach in details. It is worth noting that the distributed approach presents the heart of this dissertation.
CHAPTER 7

DISTRIBUTED SCHEME

This chapter discusses our proposed distributed scheme which presents the heart of this dissertation. The distributed scheme is composed of three components, namely: (1) the hierarchical (clustering) component, (2) the maintenance component, and (3) the routing component.

1. The hierarchical component achieves energy-efficiency by using a customized fuzzy logic controller called Network Setup controller (section 5.2). It is responsible for designing a hierarchical network structure in a distributed way, i.e., no central node is needed. Each node makes decisions based on its own locality. The resultant network must have a connected backbone, composed of CHs, where each node is connected to one and only one CH. Even though MANETs have no physical backbone, a virtual backbone can be constructed by finding a connected dominating set (CDS) in the network graph. Connected dominating sets (CDS) are the earliest structures proposed as candidates for virtual backbones in ad hoc networks. A dominating set (DS) is a set \( D \) of vertices of \( G \) such that every vertex of \( G \) is either in \( D \) or adjacent to a vertex in \( D \). A CDS is a DS, where the elements of \( D \) are connected. A minimum connected dominating set (MCDS) is a CDS, where \(|D|\) is minimum. Unfortunately, finding the CDS and the MCDS were proven to be NP-hard problems [25, 35, 38]. More information about CDS can be found in Chapter 3.
To achieve the hierarchical structure, we propose a fast distributed connected dominating set (FDDS) construction in MANETs. FDDS has message and time complexity of $O(n)$ and $O(\Delta^2)$, where $n$ is the number of nodes in the network and $\Delta$ is the maximum node degree. According to our knowledge, FDDS achieves the best message and time complexity combinations among the previously suggested approaches.

2. The maintenance component achieves energy-efficiency by using a customized fuzzy logic controller called Network Maintenance controller (section 5.3). It is responsible for maintaining the hierarchical structure created by FDDS. As time passes, the configuration of the network changes. The network configuration changes as the CHs retire/move and as nodes move. To achieve network maintenance, we propose a distributed maintenance protocol, called FDDS-M, that preserves the integrity of the hierarchical structure. Periodically, CHs (elected by FDDS) send a beacon message to their neighbors. Each node/CH receiving the message sends an acknowledgment (ACK) back. According to the beacon and the ACK messages, each node/CH decides whether it is necessary to invoke the maintenance procedure (FDDS-M). If any node/CH decided to invoke FDDS-M, FDDS-M ensures the integrity of the hierarchical structure. FDDS-M has a low message and time complexity of $O(n)$ and $O(\Delta^2)$ respectively.

3. The routing component achieves energy-efficiency by using a customized fuzzy logic controller called Routing controller (section 5.4). It is responsible for choosing efficient routes between (s, d) pairs. To achieve efficient routing, we propose a pro-
tocol, called FDDS-R, that extends the well known link state protocol named "optimized link state routing (OLSR)". Details about OLSR can be found in section 2.2.4. FDDS-R uses an intelligent path selection controller (section 5.4) that can be easily incorporated in any existing link state routing protocol to select energy-efficient routes. The path selection controller (also called routing controller) takes as input the path’s length and cost. It outputs a number representing the quality of the path. The path with the highest quality is selected as the routing path.

We conducted extensive simulation that compares the structural properties (network stretch, backbone size) and the operational properties (energy-efficiency, lifetime) of our scheme with other schemes. The results show that our proposed distributed scheme outperforms some well known approaches suggested in literature.

In the following sections, we explain in details the various components of our proposed distributed scheme. We explain: (1) FDDS which is used to handle the hierarchical component, (2) FDDS-M which is used to handle the maintenance component, and (3) FDDS-R which is used to handle the routing component. We also provide detailed analysis of each protocol.

7.1 Fast Distributed Connected Dominating Set (FDDS)

FDDS is used to handle the hierarchical component in a distributed way. Its job is to elect CHs and connect normal nodes to CHs such that the network composed of CHs is connected. We assume that each node knows its own ID, residual energy (RE), and traffic load (T). A node can calculate its mobility (M) by measuring its own displacement with
respect to its neighbors at different time periods. At time $t_1$, node $X$ measures the average distance ($D_{1_{avg}}$) between itself and its neighbors. $X$ repeats the same calculation at time $t_2$ in order to obtain $D_{2_{avg}}$. $X$ can then estimate its mobility by calculating $(D_{2_{avg}} - D_{1_{avg}})$. Note that $X$ estimates the distance to its neighbors by measuring their received signal strengths (RSS). FDDS is divided into four steps. The first step performs a simple neighbor discovery protocol and assigns a weight for each node. The second step elects an initial set of cluster heads. The third step connects the cluster heads together (those elected in the second step) forming a connected dominating set. The last step eliminates some redundant cluster heads. In our approach, we consider that message collisions are handled by the MAC layer. In the following subsections, each step is described and analyzed.

7.1.1 Step 1: Neighbor Discovery and Weight Generation

Before FDDS is executed, each node needs to know its 1-hop information. To acquire the 1-hop information, a simple neighbor discovery protocol is performed by each node. Each node sends a message containing its ID, RE, mobility, and traffic (send $nodeInfo\{ID, RE, M, T\}$). Every node that receives the $nodeInfo$ message extracts the data and stores it in a special data structure (Vector).

After collecting the $nodeInfo$ messages, each node knows the ID, RE, M, and T of each of its 1-hop neighbors. Let $RE_i$, $M_i$, and $T_i$ be the residual energy, the mobility, and the traffic load of node $i$ respectively. Let $RE_{max}$ be the maximum residual energy among the neighbors of $i$ including $i$. Similarly, let $M_{max}$ and $T_{max}$ be the maximum mobility and the maximum traffic load among node $i$ and its 1-hop neighbors. Node $i$ normalizes its own
values with respect to the maximum values i.e.

\[ REn_i = \frac{RE_i}{RE_{\text{max}}}, \quad Mn_i = \frac{M_i}{M_{\text{max}}}, \quad Tn_i = \frac{T_i}{T_{\text{max}}} \]

In section 5.2, we designed a fuzzy logic controller (Network Setup controller) which is used to calculate the node’s quality. \( REn_i, Mn_i, \) and \( Tn_i \) are fed to this controller and a single value \( SQ_i \) is returned. \( SQ_i \) represents the quality of node \( i \). Note that Network Setup controller combines the node’s residual energy (RE), mobility (M), and traffic (T) according to certain rules keeping in mind the synergy between them. It tends to give a high weight for nodes that have: (1) high residual energy, (2) low mobility, and (3) low traffic. Its objective is to give high quality for nodes that can survive a long time.

When \( SQ_i \) is generated, node \( i \) sends \( SQ_i \) (send nodeWeight\( \{i, SQ\} \)) to its 1-hop neighbors. A neighbor receiving the message, records the weight of node \( i \). After the completion of this phase, each node knows the ID’s and weights of its neighbors. A node with a high weight is a cluster head candidate.

Step 1 requires each node to send 2 messages \( (O(1) \) message complexity). The first message is used to send the node’s initial information \( (ID, RE, M, T) \) and the second message is used to send the node weight \( (SQ) \). Let \( \Delta \) be the maximum node degree (maximum number of neighbors). The time complexity of Step 1 is \( O(\Delta) \) because each node searches its neighbors to find \( RE_{\text{max}}, M_{\text{max}} \) and \( T_{\text{max}} \).

### 7.1.2 Step 2: Initial Cluster Head Election

Algorithm: In this step, nodes cooperate with each other in order to elect cluster heads. The cooperation is established when a node asks another node to become a cluster head.
The node receiving the request should agree on becoming a cluster head. Algorithm 7.1 presents the pseudo code of this step. A node checks if its weight ($SQ$) is the maximum among its neighbors. If the node has the maximum weight, it sets itself as a cluster head. If it does not have the maximum weight, it asks the neighbor having the maximum weight to become a cluster head. Each node in the network executes algorithm 7.1. Algorithm 7.1 takes as input $\Gamma = \{\text{list of the node's neighbors}\}$. It then decides whether the node should be elected as a CH, or should elect one of its neighbors to be a CH.

begin
  if myWeight is the maximum weight among the nodes in $\Gamma$ then
    1. set myself as a CH
    2. send a message to my neighbors informing them of my decision
  else
    3. Let $j$ be the neighbor that has the maximum weight
    4. send a message to node $j$ asking it to become a CH
  endif
end

**Algorithm 7.1**  Initial CH Election

Figure 7.1 shows the elected cluster heads (nodes enclosed in gray rectangles). Node 4 has $SQ_4 = 57$ which is the maximum weight among its neighbors. So, node 4 sets itself as a cluster head. Same for nodes 10 and 15. Node 5 has $SQ_5 = 50$ and it does not have the maximum weight among its neighbors. But it was elected as a cluster head because node 11 sent a message to node 5 asking it to become a cluster head.

Analysis: This step requires each node to send one message ($O(1)$ message complexity). If the node has the maximum weight among it neighbors, it sends a message informing them that it declares itself as a cluster head. If the node does not have the maximum weight...
Figure 7.1  Initial CH Election. The numbers next to the nodes represent the nodes’ weights. Nodes 4, 5, 10, and 15 are elected as cluster heads. among its neighbors, it requests from the neighboring node having the maximum weight to become a cluster head. The time complexity of this step is $O(\Delta)$ because a node needs to search its 1-hop neighbors looking for the node having the maximum weight. If the 1-hop neighbors are sorted according to their weight, the time complexity of this step becomes $O(\Delta \log(\Delta))$.

Figure 7.1 shows that the elected cluster heads do not form a connected backbone. Let $d(u, v) = k$ represent the number of hops between nodes $u$ and $v$. For example, cluster head 4 is 3-hops away from cluster head 10 i.e. $d(4, 10) = 3$.

**Theorem 1**  \( \forall \) normal node $u$, \( \exists \) cluster head $v$ such that $d(u, v) = 1$.

**Proof 1** The proof is extracted directly from the algorithm. Lines 3 and 4 in algorithm 7.1 indicate that if a node is not a cluster head, it asks one of its neighbors to become a cluster head. Therefore, every node in the network is either a cluster head or a neighbor of a cluster head.
For a given cluster head $u$, $u$ can reach another cluster head in: (a) one hop: $d(u, x) = 1$, (b) two hops: $d(u, y) = 2$, (c) three hops: $d(u, z) = 3$

**Theorem 2** \forall cluster head $u$, \exists cluster head $v$ such that $d(u, v) = 3$ or less, and the intermediate hops are normal nodes.

**Proof 2** Assume that $u$ is a cluster head. Let $C(u) = \{x \mid x$ is a cluster head neighbor of $u\}$. \forall $x \in C(u)$, \ $d(u, x) = 1$ (figure 7.2(a)). If $x \notin C(u)$, then $x$ is a normal node (figure 7.2(b)). Let $C(x) = \{y \mid y$ is a cluster head neighbor of $x\}$. \forall $y \in C(x)$ and $y \notin C(u)$, $d(u, y) = 2$. If $y \notin C(x)$, then $y$ is a normal node (figure 7.2(c)). But according to algorithm 1, $y$ must have at least one cluster head neighbor. Let such a cluster head be $z$, then $d(u, z) = 3$. Therefore, any cluster head can reach another cluster head in at most three hops.

### 7.1.3 Step 3: Connect Cluster Heads

Algorithm: According to theorem 2, each cluster head is 1-hop, 2-hops, or 3-hops away from another cluster head. In figure 7.1, cluster head 15 is 2-hops away from cluster head 10. Nodes 12 and 13 can potentially connect both cluster heads. Cluster head 4 is 3-hops away from cluster head 10. In order to connect cluster heads 4 and 10, 2 normal nodes have to be elected as cluster heads. In the following section, we present the algorithm that elects...
new cluster heads leading to a connected backbone.

The decision of electing new cluster heads is made by the cluster heads that were elected in step 2 (section 7.1.2). Referring to figure 7.1, only cluster heads 4, 5, 10, and 15 execute the algorithm. The algorithm requires each cluster head to know its 2-hop neighbors. Each node sends its 1-hop information to its neighbors. A node receiving the 1-hop information, stores the data in its data structure. Figure 7.3 shows the data structure that node 1 uses. All other nodes use similar data structure.

![Figure 7.3](image)

**Figure 7.3** Node 1 data structure. N(3) and N(2) represent the neighbors of nodes 3 and 2 respectively. The nodes are sorted according to their weights. Such a data structure enables node 1 to know its 1-hop and 2-hop neighbors.

Algorithm 7.2 is used to generate a connected backbone. The Algorithm is divided into two parts. Assume that cluster head $v$ is executing the algorithm. Lines 1 through 8 are responsible for finding the cluster heads and the normal nodes that can be reached by $v$ using 2-hops. Line 1 checks every 1-hop neighbor ($i$) of $v$. Cluster head neighbors are marked as $CHR_i = true$; meaning that cluster head $v$ can reach cluster head $i$. Normal node neighbors are marked as $NR_i = true$; meaning that cluster head $v$ can reach node $i$. Line 4 checks the neighbors of every 1-hop neighbor. Such neighbors are represented by $j$, where $j$ is 2-hops away from $v$. If $j$ is a cluster head that can be reached by $v$, it is
marked as $CHR_j = true$. If $j$ is a normal node that can be reached by $v$, it is marked as $NR_j = true$.

begin
1. for each node $i$ in $\Gamma_v$ do
2. if $i$ is a cluster head then
3. $CHR_i = true$
4. for each node $j$ in $\Gamma_i$ do
5. if $j$ is a cluster head then
6. $CHR_j = true$
else
7. $NR_j = true$
endif
endfor
else
8. $NR_i = true$
endif
endfor
end
begin
9. for each node $i$ in $\Gamma_v$ do
10. if $i$ is not a cluster head then
11. for each node $j$ in $\Gamma_i$ do
12. if $j \neq i$ && $j$ is a cluster head && $CHR_j = false$ then
13. ask $i$ to become a cluster head
endif
14. if $j$ is not a cluster head && $NR_j = false$ then
15. ask $i$ to become a cluster head
16. ask $j$ to become a cluster head
endif
endfor
endif
endfor
end

Algorithm 7.2    Connecting the backbone

Lines 9 through 16 loop across the 1-hop and 2-hop neighbors and elect new cluster heads. If cluster head $v$ is connected to a normal neighbor $i$ and $i$ is connected to a cluster head $j$, but cluster head $v$ can not reach cluster head $j$, then elect $i$ as a cluster head (lines 12-13). This process connects a cluster head to other cluster heads that are 2-hops away from it. In figure 7.1 cluster head 15 elects node 12 to be a new cluster head so that it can
connect to cluster head 10. If cluster head $v$ is connected to a normal neighbor $i$ and $i$ is connected to a normal node $j$, but cluster head $v$ can not reach node $j$, then elect $i$ and $j$ as new cluster heads (lines 14-16). This process connects a cluster head to other cluster heads that are 3-hops away from it. This election process might elect redundant cluster heads. But, sometimes it is good to have more cluster heads because more cluster heads translate to a more reliable network. Also more paths exist between source and destination pairs. If the cluster head made its decision using 3-hop information (larger locality) rather than 2-hop information, less cluster heads would have been elected. In figure 7.1 cluster head 5 elects nodes 6 and 8 to be new cluster heads. Nodes having higher weight are chosen first to act as cluster heads.

Analysis: This step requires each node to send one message ($O(1)$ message complexity). Each normal node sends a message containing its 1-hop information to its neighbors. After executing the algorithm, each cluster head sends a maximum of 2 messages asking some nodes to become cluster heads. The time complexity of the algorithm is $O(\Delta^2)$ because a cluster head needs to loop across its 1-hop and 2-hop neighbors. Note that algorithm 7.2 is only executed by the cluster heads.

7.1.4 Step 4: Reduce Cluster Heads

Algorithm: Step 3 elected new cluster heads based on the 1-hop and the 2-hop information. Such a small locality fails to produce an optimal global result. So, some newly elected cluster heads might be redundant. Algorithm 7.3 presents a very simple algorithm to eliminate some unnecessary cluster heads. If cluster head $v$ and its 1-hop neighbors are
fully covered by a neighboring cluster head, then cluster head $v$ changes its status to become a normal node. In case of a tie (i.e. both neighboring cluster heads have the exact same neighbors), the cluster head having the lower weight switches to become a normal node. Figure 7.4 shows the resultant network after executing all the steps.

\begin{algorithm}
\begin{algorithmic}
\For {each node $i$ in $\Gamma_v$}
\If {$i$ is a cluster head}
\If {$N_v \subseteq N_i$}
\State cluster head $v$ becomes a normal node
\EndIf
\If {$N_v = N_i$}
\If {$W_v < W_i$}
\State cluster head $v$ becomes a normal node
\EndIf
\EndIf
\EndIf
\EndFor
\end{algorithmic}
\end{algorithm}

\textbf{Algorithm 7.3}  \hspace{1cm} \text{Cluster head reduction}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{network.png}
\caption{The resultant network after removing some redundant backbone nodes}
\end{figure}

\textbf{Analysis:} Each cluster head that is reduced sends one message informing its neighbors
that it seized being a cluster head \((O(1)\) message complexity). The time complexity of
the algorithm is \(O(\Delta^2)\) because a cluster head needs to loop across its neighboring cluster
heads to check if anyone of them can entirely cover it along with its neighbors.

**Theorem 3** The elected cluster heads form a connected backbone (connected dominating
set).

**Proof 3** Prior to executing algorithm 7.2, theorem 2 proved that any cluster head can reach
another cluster head in at most 3 hops, where the intermediate hops are normal nodes. If
we prove that algorithm 7.2 changes such intermediate nodes to cluster heads, then all the
cluster heads should be connected. We split the proof into three parts. Given cluster heads
\(u\) and \(v\):

1. If \(d(u, v) = 1\), then both cluster heads are already connected.

2. If \(d(u, v) = 2\), then \(\exists\) a normal node \(i\) than can connect \(u\) and \(v\). Line 13 in algorithm
   7.2 elects such a node as a cluster head.

3. If \(d(u, v) = 3\), then \(\exists\) two normal nodes \(i\) and \(j\) that can connect \(u\) and \(v\). Lines
   15-16 in algorithm 7.2 elects such nodes as cluster heads.

Therefore, any 2 cluster heads that are at most 3-hops away from each other can be joined
together by executing algorithm 7.2. Thus all the elected cluster heads are connected. After
executing the backbone reduction phase, the cluster heads are still connected because the
cluster heads removed are already covered by one of their cluster head neighbors.
7.1.5 FDDS is Scalable and Energy-Efficient

FDDS achieves scalability since it organizes the network in a hierarchical structure using a completely distributed protocol. Each node makes decisions based on its own locality (1-hop and 2-hop neighbors). Hierarchical structures have long been known to increase network scalability [51, 55, 105]. Moreover FDDS designs a hierarchical structure using low message and time complexities.

Table 7.1 Complexities of well known approaches suggested in literature

<table>
<thead>
<tr>
<th>approach</th>
<th>[5]</th>
<th>[16]</th>
<th>[77]</th>
<th>[98]</th>
<th>[111]</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg complexity</td>
<td>(O(n))</td>
<td>(O(n\Delta))</td>
<td>(O(n \log n))</td>
<td>(O(n^2))</td>
<td>(\Theta(m))</td>
</tr>
<tr>
<td>time complexity</td>
<td>(O(n))</td>
<td>(O(n))</td>
<td>(O(\log^2 n))</td>
<td>(\Omega(n))</td>
<td>(O(\Delta^3))</td>
</tr>
</tbody>
</table>

Table 7.1 presents the complexities of some well known approaches suggested in literature. More details about each approach can be found in Chapter 3. FDDS provides better message complexity (\(O(n)\)) and time complexity (\(O(\Delta^2)\)) combination than the approaches presented in table 7.1. Such low complexities are especially important when the network is deployed in large scale. Low message complexity guarantees high QoS, because control messages will only consume a small portion of the channel bandwidth. Low time complexity guarantees a fast setup of the network, thus making the network operational in a short period of time.

FDDS achieves energy-efficiency because it takes into account: (1) the energy restriction imposed on the wireless nodes, (2) the mobility of the wireless nodes, and (3) the
traffic pattern of the wireless node. Moreover, FDDS provides a reliable backbone suitable for mobile ad hoc network applications. FDDS uses a fuzzy logic controller, called \textit{Network Setup} controller, that is responsible for evaluating each node in the network (section 5.2). The controller aggregates the node’s residual energy (RE), traffic (T), and mobility (M) keeping in mind the synergy between them. It gives a high weight for nodes that have high residual energy, low traffic, and low mobility. The objective of the \textit{Network Setup} controller is to give high quality for nodes that can survive a long time. Nodes that are given high evaluation will be, most likely, elected as CHs. In fact, the second step of FDDS elects the best nodes in the network as CHs. Moreover, the third step of FDDS connects such CHs using again the best nodes in the network. So, the backbone network is composed of the most powerful nodes in the network.

Our simulation results (section 7.7) confirm the above discussion and show that FDDS generates network backbone that has (1) low and constant stretch and (2) small backbone size. The results also show that networks designed using FDDS are more survivable than networks designed using other well known approaches. Furthermore, networks designed using FDDS along with the \textit{Network Setup} controller are more survivable than networks designed using FDDS without the controller. Therefore, FDDS is scalable and it succeeds in achieving energy-efficiency and in prolonging network lifetime.

\section{FDDS Complexity Analysis}

The distributed algorithm we propose (FDDS) requires 4 steps. In each step, a constant number of messages is sent by each node. So, the overall message complexity of FDDS is
$O(n)$. The time complexity of FDDS is dominated by steps 3 and 4. Step 3 (section 7.1.3) connects the backbone network and step 4 (section 7.1.4) reduces some redundant cluster heads. Both steps are only executed by the cluster heads. Each cluster head loops across its 1-hop and 2-hop neighbors in order to make the appropriate decisions. If the max number of the 1-hop neighbors is $\Delta$, then the number of the 2-hop neighbors is $\leq \Delta$. Therefore, the cluster heads execute FDDS in $O(\Delta^2)$ time complexity. Table 7.2 shows the message and time complexity of each step.

<table>
<thead>
<tr>
<th>FDDS Step #</th>
<th>Msg Complexity</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>$O(n)$</td>
<td>$O(\Delta)$</td>
</tr>
<tr>
<td>Step 2</td>
<td>$O(n)$</td>
<td>$O(\Delta \log \Delta)$</td>
</tr>
<tr>
<td>Step 3</td>
<td>$O(n)$</td>
<td>$O(\Delta^2)$</td>
</tr>
<tr>
<td>Step 4</td>
<td>$O(n)$</td>
<td>$O(\Delta^2)$</td>
</tr>
<tr>
<td>Total</td>
<td>$O(n)$</td>
<td>$O(\Delta^2)$</td>
</tr>
</tbody>
</table>

Based on the above discussion, the time complexity of the algorithm is governed by the value of $\Delta$. The time complexity can be categorized into: (1) best case when $\Delta$ is constant, (2) worst case when $\Delta$ is $n - 1$, and (3) average case when $\Delta$ is average.

### 7.2.1 Best Case

If $\Delta$ is constant, the time complexity of FDDS is $O(1)$. Figure 7.5 shows a ring topology where FDDS runs in constant time. Nodes 2, 5, and 7 are elected as cluster heads by step
2. Then, each cluster head loops across its 1-hop and 2-hop neighbors in order to elect new cluster heads. Each cluster head has 2 1-hop neighbors and 2 2-hop neighbors. Therefore, the time complexity of FDDS when applied on such networks is $O(\Delta^2) = O(2^2) = O(1)$.

![Figure 7.5](image)

**Figure 7.5** A ring network with constant time complexity

### 7.2.2 Worst Case

Figure 7.6(a) shows a scenario where the graph is completely connected. In such networks, only one node is elected as a cluster head (node 5). Figure 7.6(b) shows the data structure of cluster head 5. The total number of entries in node’s 5 data structure is 49. In general, in a completely connected network, the cluster head holds a data structure that contains $(n - 1)^2$ entries. When step 3 of FDDS is executed, each entry in the cluster head’s data structure is visited bringing the complexity to $O((n - 1)^2) = O(n^2)$, which is the worst case.

### 7.2.3 Average Case

Optimal Calculation of $\Delta$: In actual networks, the best case and the worst case scenarios rarely occur. We are more interested in finding the complexity of the algorithm for random
topologies. In order to calculate the complexity of the algorithm, we need to find the average number of neighbors (Δ). The average degree can be calculated using the following formula:

\[ \Delta = \frac{\sum_{i=1}^{n-1} i \times n_i}{K_n} \]  (7.1)

where \( K_n \) is the number of vertices across all (non-isomorphic) connected graphs, and \( n_i \) is the number of nodes across all graphs having degree \( i \). Figure 7.7 shows all possible configurations for a connected network of 4 nodes. Equation 7.1 can be applied as follows:

\[ \Delta = \frac{1(n_1) + 2(n_2) + 3(n_3)}{K_n} = \frac{1(6) + 2(10) + 3(8)}{24} \approx 2 \]

There are 6 nodes having degree 1, 10 nodes having degree 2, and 8 nodes having degree three. The total number of nodes in all the configurations is 24. The average number of neighbors is 2.
However, counting the number of (non-isomorphic) planar graphs with $n$ nodes is a well-known long-standing unsolved graph-enumeration problem ([65]). There is no known closed formula for counting the number of unlabeled planar graphs. In [95], the authors calculate the number of connected unlabeled graphs for graphs with sizes $n = 1, 2, \ldots, 16$.

Figure 7.8 and table 7.3 indicate that the number of connected unlabeled graphs grows exponentially as the number of vertices increases. Therefore, finding the average number of neighbors using equation 7.1 is not practical because all the connected unlabeled graphs should be known.

Probabilistic Calculation of $\Delta$: Another approach to estimate the average degree in the network is to use a probabilistic approach. For a given network, we calculated the number of occurrences of each degree across all network configurations. For example, if the network size is 4 (figure 7.7): 6 nodes have degree 1, 10 nodes have degree 2, and 8 nodes have degree 3. Degree 2 has the highest frequency. So, with a high probability, the average degree for a network of size 4 is 2. Figure 7.9 shows the number of occurrences of
Figure 7.8  Number of connected unlabeled graphs for \( n = 1, 2, \ldots, 9 \) each degree for networks of sizes 3, 4, and 5. For a network of size 3, the highest degree frequency is 2. For a network of size 4, the highest degree frequency is also 2. And for a

Table 7.3  Number of connected unlabeled graphs for \( n = 1, 2, \ldots, 16 \)

<table>
<thead>
<tr>
<th>number of nodes</th>
<th>number of connected unlabeled graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>112</td>
</tr>
<tr>
<td>7</td>
<td>853</td>
</tr>
<tr>
<td>8</td>
<td>11117</td>
</tr>
<tr>
<td>9</td>
<td>261080</td>
</tr>
<tr>
<td>10</td>
<td>11716571</td>
</tr>
<tr>
<td>11</td>
<td>1006700565</td>
</tr>
<tr>
<td>12</td>
<td>164059830476</td>
</tr>
<tr>
<td>13</td>
<td>50335907869219</td>
</tr>
<tr>
<td>14</td>
<td>29003487462848061</td>
</tr>
<tr>
<td>15</td>
<td>31397381142761241960</td>
</tr>
<tr>
<td>16</td>
<td>63969560113225176176277</td>
</tr>
</tbody>
</table>
network of size 5, the highest degree frequency is 3. We can generalize by saying that for a network of size \( n \), the highest degree frequency is \([n]\). Therefore, with a high probability, the average degree for a network of size \( n \) is \([n]\).

Heuristic Approximation of \( \Delta \): The probabilistic approach discussed above has some problems: (1) it does not consider the area where the network is deployed. (2) It does not consider the transmission radius of each node. (3) It produces misleading value for the average degree in a network. Since the optimal calculation of \( \Delta \) is not practical and since the probabilistic approach has many problems and is misleading, we use a heuristic approach to estimates the average degree in the network. Let \( S^2 \) be the area covered by the network, then the expected number of neighbors of a given node is:

\[
\Delta = \Pi \frac{R^2}{S^2} n \tag{7.2}
\]

where \( R \) is the transmission radius and \( n \) is the number of nodes in the network [121]. If
a network of 5000 nodes is placed in a 5Km x 5Km area and the transmission range is 170m (802.11b), then the average number of neighbors for a given node is approximately 18 ($\Delta = 18$). Having analyzed the complexity of FDDS, we now present some of FDDS unique properties.

### 7.3 FDDS Theoretical Analysis

In this section, we provide some of FDDS unique properties. For example, in which circumstances does FDDS elect redundant CHs? Each property is presented as a theorem along with its proof.

**Theorem 4** The sufficient condition that produces redundant cluster heads is to have at least one cycle $C$ such that: $C$ contains two cluster heads $u$ and $v$ and $\exists$ exactly 2 paths $P_1$ and $P_2$ between $u$ and $v$ such that both paths have $d(u,v) = 3$.

![Figure 7.10](image)

**Figure 7.10** (a) A cycle that might lead to redundant CH election. (b) Cluster heads 1 and 6 elect nodes 3 and 5 to act as cluster heads.

**Proof 4** Figure 7.10(a) shows a cycle $C$ that consists of cluster heads 1 and 6. Cluster
head 1 is 3-hops away from cluster head 6. Recall that each cluster head in the network executes step 3 of the protocol discussed above. When cluster head 1 executes step 3, it has the choice to elect nodes (2 and 4) or nodes (3 and 5) to be cluster heads. Cluster head 1 elects node 3 first because it has higher weight than node 2, then it elects nodes 5 because it is the only node that can connect cluster head 1 to cluster head 6. So, cluster head 1 elects nodes 3 and 5 to act as cluster heads. Similarly, cluster head 6 starts by electing node 5 because it has higher weight than node 4, then it elects node 3. In this case, both cluster heads elect the same nodes to act as cluster heads (figure 7.10(b)).

**Theorem 5** Given 2 cluster heads $u$ and $v$ and exactly 2 paths $P_1$ and $P_2$ from $u$ to $v$ such that $d(u, v) = 3$, $P_1 = \{a, b, v\}$, and $P_2 = \{c, d, v\}$. The necessary condition that produces redundant cluster heads is when $W_a > W_c$ and $W_d > W_c$.

![Diagram](image)

**Figure 7.11** (a) A cycle that leads to redundant CH election. (b) Cluster head 1 elects nodes 2 and 4 to act as cluster heads. Cluster head 6 elects nodes 3 and 5 to act as cluster heads.

**Proof 5** Figure 7.11(a) shows a network with 2 cluster heads 1 and 6 such that the distance between them is 3-hops. Starting from cluster head 1, $P_1 = \{a, b, v\} = \{2, 4, 6\}$ and
\[ P_2 = \{c, d, v\} = \{3, 5, 6\}. \] Cluster head 1 starts by electing node 2 to act as a cluster head because node 2 has a greater weight than node 3 \((W_2 > W_3)\). Then node 4 is elected as a cluster head in order to establish a connection to cluster head 6. Similarly, cluster head 6 starts by electing node 5 to act as a cluster head because node 5 has a greater weight than node 4 \((W_5 > W_4)\). Then node 3 is elected as a cluster head in order to establish a connection to cluster head 1. In this case, 2 redundant cluster heads are elected (figure 7.11(b)). In general, if \(W_a > W_c\) and \(W_d > W_c\), \(u\) elects nodes \(a\) and \(b\) to be cluster heads and \(v\) elects nodes \(c\) and \(d\) to be cluster heads. Two of the cluster heads are redundant.

**Corollary 1** If the cycle presented in theorem 4 contains 4 paths between \(u\) and \(v\) such that all paths have \(d(u, v) = 3\), then no redundant cluster heads will be elected.

\[
\begin{align*}
\text{(a) A cycle where 4 paths exists between cluster heads } u \text{ and } v & \text{ (b) The cluster heads having the highest weights are elected as CHs} \\
\text{Proof 6} & \text{ Let the 4 paths between } u \text{ and } v \text{ be } P_1 = \{a, b, v\}, P_2 = \{c, d, v\}, P_3 = \{a, d, v\}, \\
& \text{ and } P_4 = \{c, b, v\} \text{ (figure 7.12(a)). Cluster head } u \text{ starts by choosing either } a \text{ or } c \text{ to act} \\
\end{align*}
\]
as a cluster head depending on which one has a higher weight. Cluster head \( u \) finishes by selecting either \( b \) or \( d \) to become a cluster head (the one having the higher weight is chosen). Therefore, cluster head \( u \) elects as cluster heads the nodes having the highest weights along the 4 paths. If \( W_a > W_c \) and \( W_d > W_b \), then cluster head \( u \) elects nodes \( a \) and \( d \) to become cluster heads (figure 7.12(b)). Cluster head \( v \) runs the same algorithm to elect 2 new cluster heads. Cluster head \( v \) starts by electing either \( b \) or \( d \) to become cluster head. It finishes by electing either \( a \) or \( c \) to become cluster heads. Cluster head \( v \), also, elects as cluster heads the nodes having the highest weights along the 4 paths. Cluster head \( v \) elects nodes \( d \) and \( a \) to become cluster heads. In such scenarios, both cluster heads elect the same nodes to act as cluster heads.

**Corollary 2** If the cycle presented in theorem 4 contains up to \( n \) paths between \( u \) and \( v \) such that all paths have \( d(u, v) = 3 \), then a maximum of 2 redundant cluster heads will be elected.

**Figure 7.13** When \( n \) paths exist between cluster heads \( u \) and \( v \) such that all paths have \( d(u, v) = 3 \), a maximum of 2 cluster heads are redundant

**Proof 7** Such a scenario is presented in figure 7.13. Starting from \( u \), let \( P_i = \{u_i, v, v\} \) where \( W_{u_i} \) is the maximum weight among all \( u_i \)’s. According to DE-CDS, cluster head \( u \)
elects as cluster heads the nodes that belong to $P_i$. Starting from $v$, let $P_j = \{v_j, u_j, u\}$ where $W_{v_j}$ is the maximum weight among all $v'$s. According to DE-CDS, cluster head $v$ elects as cluster heads the nodes that belong to $P_j$. Two redundant cluster heads will be elected only if $P_i \neq P_j$.

### 7.4 Cluster Formation

Having designed a connected backbone, composed of CHs, the next step is to create the clusters (cluster formation). Cluster formation involves associating each normal node in the network with a unique CH. A normal node can not connect to more than one CH. All nodes connected to a single CH form a cluster. Cluster formation is driven by the following observations:

- Each node in the network can communicate using a certain capacity (for example 11 Mbps). In order for a CH to be beneficial, it should be able to support the traffic generated by its cluster members. So, it is necessary for the traffic generated by the cluster members to be less than or equal to the capacity of the CH. If this was not the case, the CH would not be able to handle the traffic generated by its cluster members and some traffic would be lost.

- It is always preferable for a CH to receive a signal with high strength, because high received signal strength (RSS) translates to low bit error rate (BER) that translates to better QoS. It is mostly the case that CHs receive signals with high RSS from nodes that are close (distance wise) to them *. The closer the node is to a CH, the

---

*CHs can also receive signals with high RSS from distant nodes who are using high power level
less power it needs to reach the CH. Less power needed by a node translates to less energy consumed, that translates to a more survivable node.

Each node in the network uses the *Network Maintenance* controller, discussed in section 5.3, to evaluate the CHs it can hear from. The evaluation criteria is based on the following parameters: (1) $SQ$ of the CH, (2) the remaining capacity that the CH can handle ($RC$), and (3) the received signal strength of the CH. $SQ$ is the value obtained from the *Network Setup* controller (section 5.2). A CH with high $SQ$ is a CH with high residual energy, low traffic, and low mobility.

Assume that node $a$ can hear signals from CHs $x$, $y$, and $z$. Each CH sends a "join" message containing its $ID$, $SQ$, and $RC$ ($join (ID, SQ, RC)$). When node $a$ receives the join messages, its preference is to choose the closest CH that has high $SQ$ and high $RC$. Node $a$ measures the RSS of each CH and passes RSS, SQ, and RC to the *Network Maintenance* controller. The controller returns a weight ($MQ$) for each CHs. Node $a$ chooses the CH having the highest $MQ$ as its parent CH by sending to it a request to join message. The CH receiving the request adds node $a$ to its cluster members. Every node in the network does the same procedure. Nodes left without a parent join the CH that is closest to them. Figure 7.14 shows the final topology after applying cluster formation to figure 7.4.

In figure 7.14, node 5 can hear from CHs 4 and 6. CH 4 has higher $SQ$ (57) than CH 6 (49). Logically, node 5 should join CH 4. But, node 5 is closer to CH 6 (high RSS). Also CH 4 already has 3 members (nodes 1,2, and 3). So its capacity is most probably smaller than that of CH 6. When the *Network Maintenance* controller evaluated each CH, it gave
higher weight for CH 6. Therefore, node 5 joined CH 6 and not CH 4.

The message complexity of cluster formation is $O(n)$, since each CH sends a "join" message and each node replies back with a "request to join" message. So the total number of messages needed is exactly $n$ messages. Let $B_s$ be the number of CHs in the network. In the worst case, the time complexity of cluster formation is $O(B_s)$, since a node might evaluate all the CHs in the network in order to choose one parent. Each evaluation takes a constant time. In this case, the node should be able to hear all the CHs in the network which is a very rare case especially in large scale networks.

7.5 Cluster Maintenance (FDDS-M)

At this point, the network is operational and traffic is flowing between (s, d) pairs. But as time passes, the configuration of the network changes. The two factors that cause the network configuration to change are when CHs retire/move and when nodes move. Periodically (after $T$ seconds) each CH sends a control message to its neighbors to ensure that it has up to date information. The periodic message contains the $ID$ of the CH, its weight
(SQ), and its remaining capacity (RC). Each node/CH receiving the message evaluates the quality (MQ) of the CH who sent the message. MQ is calculated by following the same procedure discussed in cluster formation (section 7.4), i.e., the CH’s SQ, RC, and RSS are passed to the Network Maintenance controller and the CH’s quality MQ is obtained.

Cluster maintenance can be invoked by a normal node or a CH. A normal node invokes cluster maintenance if it receives a weak or no signal from its parent CH. A CH invokes cluster maintenance if itself or any of its CH neighbors contribute to the possibility of having a disconnected backbone. The protocol that handles cluster maintenance is called FDDS-M and it is responsible for preserving the hierarchical structure of the network.

Before we start explaining FDDS-M, we present a theorem that calculates the maximum distance a node/CH can travel between 2 periodic message updates.

**Theorem 6** Let \( R \) be the coverage radius of a mobile node. During a periodic update period (40 seconds), the maximum distance a node can move is \( R \).

**Proof 8** Let \( d_{\text{max}} \) and \( v_{\text{max}} \) be the node’s maximum distance and speed respectively. We would like to have \( d_{\text{max}} = R \), i.e., the node cannot be displaced from its original location by more than its own coverage area.

\[
d_{\text{max}} = v_{\text{max}} * T \tag{7.3}
\]

\[
\Rightarrow T = \frac{d_{\text{max}}}{v_{\text{max}}} = \frac{R}{v_{\text{max}}} \tag{7.4}
\]

Assuming that the maximum speed is 5 \text{ meters/sec} and the node coverage area is 200 \text{ meters}, then the update message should be sent after every \( \frac{200}{5} = 40 \) seconds. In this case,
we know for sure that after 40 seconds the node did not move more than 200 meters. This helps us in developing the cluster maintenance actions.

FDDS-M is composed of 2 parts. One part is used to handle normal node maintenance and the other part is used to handle CH maintenance. In the following sections, we explain both parts and analyze their complexities.

7.5.1 Node Maintenance

Assume that CH \( j \) is the parent of normal node \( i \), i.e., \( i \in \text{members}(j) \). CH \( j \) sends a periodic message to its neighbors. Let \( t \) be a predefined threshold that is used by each node to determine whether a received message has a strong or a weak signal. Figure 7.15 shows a flow chart presenting all the actions that node \( i \) might perform when itself or its parent CH move. All these actions are part of the FDDS-M protocol. A detailed explanation of Figure 7.15 is presented as follows:

1. Node \( i \) receives a message from its parent CH \( j \). Node \( i \) sends an ACK to CH \( j \) indicating that it received the message. Two possibilities might occur.

   (a) Node \( i \) receives one and only one message from its parent CH \( j \). In this case, node \( i \) has no choice but to stay with its parent CH.

   (b) Node \( i \) receives a message from its parent CH \( j \) and messages from other CHs. Node \( i \) measures the signal strength \((RSS_j)\) of the message that was originated from its parent CH \( j \). If \( RSS_j > t \), i.e., node \( i \) is within the coverage area of CH \( j \), node \( i \) does not invoke FDDS-M and it extends its association with its
Node i uses the fuzzy logic controller to get a weight for every CH.

Node i chooses the CH with the highest weight (k).

Do Nothing

Node i sends a periodic message to its neighbors

I sends ACK

Node i gets messages from other CHs

Node i belongs to CH j members

Node i gets messages from other CHs

Node i sends a signal to its neighbors

Node i sends a signal to its neighbors

Any node responds

Node i searches its 2-hop neighbors to find a CH. The connector with the highest weight is elected as a CH to connect i to the backbone.

Node i searches its 2-hop neighbors to find a CH. The connector with the highest weight is elected as a CH to connect i to the backbone.

Node i sends ACK

Figure 7.15 Actions taken by FDDS-M when node i or its parent CH move parent CH.

If $RSS_j \leq t$, i.e., node i is barely covered by CH j, node i invokes FDDS-M. Recall that each periodic message originating from a certain CH contains the CH’s ID, SQ, and RC. Node i feeds the SQ, RC, and RSS of each CH to the Network Maintenance controller defined in section 5.3. The controller returns a weight $MQ$ for each CH. Let the CH with the highest $MQ$ be CH $k$. i changes its association from CH $j$ to CH $k$. It also sends a message to CH $k$ informing it that it is now one of its members. CH $k$ adds node i to its table, and sends a message through the backbone to CH $j$ asking it to remove i from its table.
2. Node $i$ does not receive any message from its parent CH $j$. CH $j$ will time out and remove node $i$’s entry from its table. Two possibilities might occur:

(a) Node $i$ receives messages from other CHs. Node $i$ does the same actions presented in the previous section when $RSS_j \leq t$; in this case $RSS_j = 0$. Figure 7.16 shows an example where node $i$ was connected to CH $j$. When the CHs send the periodic update, node $i$ was able to hear from CH $k$ but not from its own parent. So it connected to CH $k$. CH $k$ then informs CH $j$ to remove $i$ from its membership.

(b) Node $i$ does not receive any message from any CH. Node $i$ sends a message to its neighbors and waits for an ACK. Each ACK contains the 1-hop neighbors of the node that is sending the ACK. (1) When one or more ACKs are received by $i$, the ACKs contents are searched for CHs. Note that the contents of the

![Figure 7.16](image-url)
Figure 7.17  (a) Originally node $i$ was connected to CH $j$ (b) Node $i$ moved out from the coverage area of CH $j$. Node $i$ elects node $p$ to become a new CH in order to connect it back to CH $j$.

ACKs represent the 2-hop neighbors of node $i$. Node $i$ connects to one of the discovered CHs by electing the 1-hop neighbor with the highest weight. Figure 7.17 shows an example of such scenario. (2) If no ACKs are received by $i$, node $i$ becomes completely disconnected from the network. As mentioned earlier, CH $j$ will time out and remove $i$ from its table. In addition to the time out period, CH $j$ waits for $\Delta t$. If no ACK was received after $\Delta t$, CH $j$ sends a message to all the backbone nodes telling them that node $i$ has been disconnected from the network. Such declaration is very useful and energy-efficient because messages destined to $i$ are blocked immediately and no burden is put on the CHs to try to find $i$. When node $i$ connects again to the network, the CH that it is connected to, informs all other CH that $i$ is not disconnected anymore. A Vector data structure can be used, at each node, to hold the ID’s of
the nodes that got disconnected from the network.

Table 7.4  Actions taken by node $i$ when itself or its parent CH $j$ move

<table>
<thead>
<tr>
<th>Action</th>
<th>Number of Messages</th>
<th>Msg Complexity</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$n + B_s$</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>2</td>
<td>$n + B_s$</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>3</td>
<td>$n + B_s + 2$</td>
<td>$O(n)$</td>
<td>$O(\Delta)$</td>
</tr>
<tr>
<td>4</td>
<td>$B_s + 2$</td>
<td>$O(B_s)$</td>
<td>$O(\Delta)$</td>
</tr>
<tr>
<td>5</td>
<td>$B_s + \Delta + 2$</td>
<td>$O(n)$</td>
<td>$O(\Delta^4)$</td>
</tr>
<tr>
<td>6</td>
<td>$B_s + B_s$</td>
<td>$O(B_s)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>$O(n)$</td>
<td>$O(\Delta^2)$</td>
</tr>
</tbody>
</table>

Table 7.5  Message and time complexities of "node maintenance" across all network nodes

1. $j$ sends a message. $i$ receives the message and sends an ACK. $i$ receives a message only from its CH $j$. $i$ does nothing.

2. $j$ sends a message. $i$ receives the message and sends an ACK. $i$ receives a message from CH $j$ and from other CHs where $RSS_j > t$. $i$ does nothing.

3. $j$ sends a message. $i$ receives the message and sends an ACK. $i$ receives a message from CH $j$ and from other CHs where $RSS_j \leq t$. $i$ invokes cluster maintenance. $i$ sends an update message to its new CH. The new CH sends an update message to $j$.

4. $j$ sends a message. $i$ does not receive the message. $i$ receives messages from other CHs. $i$ invokes cluster maintenance. $i$ sends an update message to its new CH. The new CH sends an update message to $j$.

5. $j$ sends a message. $i$ does not receive the message. $i$ does not receive any message from other CHs. $i$ sends a message to its neighbors. Some nodes (maximum $\Delta$) send an ACK back to $i$. $i$ invokes cluster maintenance. $i$ sends an update message to the newly elected CH.

6. $j$ sends a message. $i$ does not receive the message. $i$ does not receive any message from other CHs. $i$ sends a message to its neighbors. No neighbors send an ACK back to $i$. $i$ is disconnected. CH $j$ times out and sends a message to all backbone nodes saying that $i$ is disconnected.
Tables 7.4 shows all possible actions that node \( i \) performs when FDDS-M is invoked. Table 7.5 presents the time and message complexities of each action across all network nodes. Let \( B_s \) be the number of cluster heads in the network. The tables show that the node maintenance algorithm has \( O(n) \) messages complexity and \( O(\Delta^2) \) time complexity. These are the same complexities achieved by FDDS.

### 7.5.2 CH Maintenance

A CH can invoke cluster maintenance depending on two conditions: (1) it wants to retire because its residual energy became low and/or (2) it receives a weak or no signal from a neighboring CH.

**CH retires:** Figure 7.18 shows the actions taken by FDDS-M when CH \( j \) wants to retire. Let \( J_1 = \{j_1, j_2, \ldots, j_\Delta\} \) be the original CH neighbors of CH \( j \). CH \( j \) starts by sending \( J_1 \) to all its neighbors. Let \( i \) be one of \( j \)'s neighbors, such that \( i \) can cover \( j \cup J_1 \). If such an \( i \) exists, \( i \) changes its status to become CH, and sends a message to CH \( j \) telling it to become a normal node.

If non of \( j \)'s 1-hop neighbors can cover \( j \cup J_1 \), then FDDS-M searches for more than one node that can cover \( j \cup J_1 \). Let \( i \) be a 1-hop neighbor of \( j \) such that \( j \cup J_1 \) belongs to the 1-hop and 2-hop neighbors of \( i \). Let \( \{k_1, k_2, \ldots, k_\Delta\} \) be the connectors that connect the 1-hop nodes of \( i \) with the CHs that belong to \( j \cup J_1 \) (excluding those that belong to \( i \)'s 1-hop neighbors). \( i \) changes its status to CH and sends a message to the connector nodes asking them to become CHs. It also asks CH \( j \) to become a normal node. This way the backbone connectivity is preserved.
(1) i changes its status to CH
(2) i sends a message to
\{k1, k2, \ldots\} asking them
 to be CHs
(3) j changes its status to
normal node

CH j is low
on power

j sends a message containing
J1 to its neighbors.

j i gets
the message
and checks its
1-hop neighbors

Let \{k1, k2, \ldots\} be
the 1-hop neighbors
of i that serve as
connectors to the
CHs in J1 U j

Find an i such that J1
U j belong to i's 1-hop
and 2-hop neighbors

no

yes

(1) i changes its status to CH.
(2) i sends a message to j telling it to change its
status to a normal node.

j i U j belong to
i's 1-hop neighbors

Figure 7.18   Actions taken by FDDS-M when CH j wants to retire. J1 consists of the original
CH neighbors of CH j.

Figure 7.19 shows an example when CH j wants to retire. Initially CH j looks in its
1-hop neighbors to check if any node/CH can cover its original CHs neighbors \(\{k, m, n\}\).
But it does not find such a node. So it asks node i to elect extra CHs to preserve the
backbone connectivity. But, node i can only reach CHs k and m; it can not reach CH n. So
node i asks node p to act as a connector between itself and CH n. Then node i changes its
status to CH. CH j changes its status to normal node and connects itself to CH p. Note that
redundant CHs are removed when step 4 of FDDS is invoked. In figure 7.19(b), CH k will
be removed.

![Diagram showing original and after retirement configurations](image)

**Figure 7.19** (a) Originally CH $j$ was connected to CHs $\{k, m, n\}$  (b) CH $j$ retires and node $i$ changed its status to CH and elects node $p$ to become a CH to preserve backbone connectivity

The procedure discussed above requires, in the worst case, 3 messages. One message sent by the CH who wants to retire. Another message sent by the node that can cover $j \cup J_1$. And a third message needed to ask the connector nodes to become CHs. The procedure requires $O(\Delta^2)$ time complexity because a neighbor of $j$ needs to check whether its 1-hop neighbors can cover $j \cup J_1$. So the overall message and time complexity of the "CH retires" procedure are $O(n)$ and $O(\Delta^2)$.

CH moves: Figure 7.20 shows the actions taken by FDDS-M when CH $j$ moves. Let $J_1$ be the set containing the original CH neighbors of CH $j$. CH $j$ starts by sending a periodic message to its neighbors. Each neighbor who receives the message, replies back with an ACK. The following scenarios might occur:

1. All the CHs that belong to $J_1$ send an ACK to CH $j$. In this case, the original connections of CH $j$ are preserved and no action is needed.
2. Only part of the CHs that belong to $J_1$ send an ACK to CH $j$. CH $j$ might also receive some ACKs from CHs that do not belong to $J_1$. CH $j$ starts by establishing connections with the new CHs that send it an ACK (if any). Then CH $j$ sends a message through the backbone destined to all the CHs in $J_1$. The message is sent through the backbone because there is a possibility that some CHs in $J_1$ are not anymore 1-hop neighbors of CH $j$, but might still be reached by $j$ through 2-hops or 3-hops. If all the CHs in $J_1$ replied with an ACK, the backbone would be preserved.
and no action is needed.

![Diagram](image)

**Figure 7.21**  (a) Originally CH $m$ was connected to CHs $\{k, n, j\}$  (b) CH $m$ moves and gets disconnected from all its original CHs. CH $m$ elects nodes $i$ and $p$ as CHs to preserve backbone connectivity.

If only part or none of the CHs in $J_1$ send an ACK to CH $j$, the backbone is disconnected. New CHs must be elected in order to connect CH $j$ with the CHs in $J_1$ that got disconnected from $j$. CH $j$ executes step 3 of FDDS (section 7.1.3) in attempt to connect to the CHs in $J_1$ that are not 1-hop away from CH $j$. According to theorem 6, CH $j$ can be a maximum of 3-hops away from its original CH neighbors, even when all the CHs move in opposite directions. If CH $j$ moved a distance $R$, and one of its CH neighbors moved a distance $R$ in the opposite direction, both CHs can not be more than 3-hops away. The CHs elected by executing step 3 of FDDS change their status to CH and update their tables. CH $j$ also updates its own tables and the backbone is preserved.

Figure 7.21 shows an example of cluster maintenance where CH $m$ gets disconnected from its original CHs ($\{k, n, j\}$). CH $m$ does not receive ACKs from its original CHs,
and thus it executes step 3 of FDDS. Nodes $i$ and $p$ change their status to CH. CHs $i$, $p$ and $m$ update their tables and the backbone network stays connected.

3. If neither CHs nor normal nodes send an ACK to CH $j$, CH $j$ becomes disconnected from the network. One of $j$’s neighbors times out after $\Delta t$ and sends a message to all the backbone nodes telling them that CH $j$ has been disconnected from the network. CH $j$ changes its status to normal node. When CH $j$ connects again to the network, the CH that it is connected to informs all other CHs that $j$ is not disconnected anymore.

Table 7.6  Actions taken when CH $j$ moves

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$j$ sends a message. All the CHs in $J_1$ send an ACK to $j$. Backbone is preserved and no actions are taken.</td>
</tr>
<tr>
<td>2.</td>
<td>$j$ sends a message. Part of the CHs in $J_1$ and other CHs send an ACK. $j$ connects to new CHs. $j$ sends a message through the backbone to the CHs in $J_1$. All CHs in $J_1$ reply. Backbone is preserved and no actions are taken.</td>
</tr>
<tr>
<td>3.</td>
<td>$j$ sends a message. Part of the CHs in $J_1$ and other CHs send an ACK. $j$ connects to new CHs. $j$ sends a message through the backbone to the CHs in $J_1$. Part or none of the CHs in $J_1$ reply. $j$ performs step 3 of FDDS and elects new CHs. Backbone is preserved.</td>
</tr>
<tr>
<td>4.</td>
<td>$j$ sends a message. No ACKs are sent back to $j$. $j$ becomes disconnected. A CH neighbor of $j$ times out and sends a message to all CHs informing them that $j$ is disconnected.</td>
</tr>
</tbody>
</table>

Table 7.6 summarizes the actions taken when a single CH moves. Each CH reacts the same way in case of movement. After the maintenance step is done, step 4 of FDDS is invoked in order to eliminate some redundant CHs. Table 7.7 shows the complexity of each action when all CHs invoke the "CH moves" procedure. The total message and time
complexity of the procedure are $O(n)$ and $O(\Delta^2)$ respectively.

To summarize, cluster maintenance is composed of (1) Node maintenance and (2) CH maintenance. CH maintenance has 2 procedures: (a) CH retires and (b) CH moves. Both "CH retires" and "CH moves" have message and time complexity of $O(n)$ and $O(\Delta^2)$ respectively. Consequently, "CH maintenance" has the same complexities. On the same hand, "Node maintenance" has the same complexities. Therefore FDDS-M has message and time complexity of $O(n)$ and $O(\Delta^2)$ respectively; same complexities achieved by FDDS.

**Table 7.7**  
Message and time complexities of "CH maintenance" actions across all network nodes

<table>
<thead>
<tr>
<th>Action Number</th>
<th>Number of Messages</th>
<th>Msg Complexity</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$n + B_s$</td>
<td>$O(n)$</td>
<td>$O(\Delta)$</td>
</tr>
<tr>
<td>2</td>
<td>$n + 3B_s$</td>
<td>$O(n)$</td>
<td>$O(\Delta)$</td>
</tr>
<tr>
<td>3</td>
<td>$n + 3B_s + \Delta$</td>
<td>$O(n)$</td>
<td>$O(\Delta^2)$</td>
</tr>
<tr>
<td>4</td>
<td>$B_s$</td>
<td>$O(B_s)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>$O(n)$</td>
<td>$O(\Delta^2)$</td>
</tr>
</tbody>
</table>

**7.5.3 FDDS-M is Scalable and Energy-Efficient**

FDDS-M uses a fuzzy logic controller, called Network Maintenance controller, that is responsible for evaluating the network CHs. The controller is used by moving nodes in order to evaluate the CHs it can hear. CH evaluation depends on: (1) $SQ$ (the output of the Network Setup controller), (2) remaining capacity ($RC$), and (3) received signal strength ($RSS$). The controller gives more weight for CHs that have: high $SQ$, high $RC$, and high $RSS$. 
Each one of the above parameters translate to energy-efficiency. High $SQ$ means that the CH is powerful and more survivable than other CHs. High $RC$ means that the CH can handle extra traffic. But, what is more important is that the controller uses $RC$ as means to balance the traffic load on each CH. Balancing the traffic load among all CHs results in a more survivable network [90, 108, 110, 112, 113, 114]. High $RCC$ means that the CH is close to the node. If a CH is close to a certain node, less power is needed for both of them to communicate with each other. All these parameters translated directly to the energy-efficiency of FDDS-M.

FDDS-M achieves scalability since it is completely distributed. In most cases, decisions are made based on 1-hop information. In rare cases, decisions are made based on 2-hop information. Moreover, FDDS-M has low message and time complexity. The complexities are the same as that of FDDS. This means that not too much overhead is produced when FDDS-M is invoked. Our simulation results confirm the above discussion and show that the network lifetime is prolonged when FDDS-M is used. Therefore, FDDS-M is scalable and it succeeds in achieving energy-efficiency and in prolonging network lifetime.

7.6 Routing (FDDS-R)

Up to this point, we proposed two protocols that design and maintain a hierarchical network structure in MANETs. FDDS is responsible for network construction and FDDS-M is responsible for network maintenance. Both protocols achieve scalability and energy-efficiency. To complete our distributed scheme, we designed an energy-efficient routing protocol, called FDDS-R. FDDS-R is in fact an extension of the well known Optimized
Link State Routing Protocol (OLSR). We extended OLSR by designing an intelligent path selection controller that chooses an appropriate path to route messages between (s, d) pairs. In this section, we discuss the fuzzy logic controller (Routing controller), give a quick overview about OLSR, and analyze FDDS-R.

7.6.1 Routing Controller

The Routing controller is discussed in details in section 5.4. It is responsible for choosing appropriate routes between (s, d) pairs. Whenever a route request is established, many paths are available to deliver the message to its final destination. However, only one path should be selected. Most routing protocols use the shortest path to route message between source and destination pairs. Choosing the shortest path is not a good idea specially when energy conservation is one of the network design goals. Nodes along the shortest paths will carry high computation and communication burden and their batteries will deplete quickly. In our design, we prefer to choose the path that is the most energy-efficient. But, how can this be done?

Each path is associated with a cost (path cost) equal to the residual energy of the weakest node along the path (the node with the minimum residual energy). Note that the higher the path cost, the better the route. So choosing the path with the highest path cost is a logical choice. But, such path might be very long in terms of the number of hops. So, the residual energy of all nodes along the path will be affected. For this reason, we might want to choose a path that has a lower path cost with a shorter length.

Let \( PL \) be the path length in terms of the number of hops. The Routing controller
combines the path cost and the path length keeping in mind the synergy between them. The controller produces a single output that defines the route quality ($RQ$). The path with highest $RQ$ is chosen to route messages between $(s, d)$ pairs. So, the *Routing* controller provides an intelligent path selection criteria that achieves a balance between the residual energy of the nodes along the path and the path length.

When a certain path is selected, the routing is composed of three steps: (1) if the source is not a CH, it sends its message to its representative CH, else it does nothing. (2) Now the CH acts as the new source and forwards the message to its destination CH. (3) If the destination CH is not the final destination, it relays the message to the final destination node in its cluster.

The real power of the *Routing* controller lies in the fact that it can be easily incorporated in any existing link state routing protocol to select energy-efficient routes. We chose to use OLSR because of its efficient flooding algorithm. Next we give a brief overview about OLSR.

### 7.6.2 OLSR Overview

OLSR is a table-driven, pro-active, link state routing protocol. OLSR includes a heuristic designed technique to optimize for the shared wireless media environment. Essentially, the heuristic consists of restricting the flooding of information to a set of selected neighboring nodes called the Multipoint Relaying (MPRs). Along the same way, the OLSR protocol minimizes flooding of the control messages as it only uses its MPRs to retransmit the broadcast messages in a MANET. A detailed description of OLSR can be found in
We pick OLSR mainly because of its efficient flooding mechanism. OLSR is also known to perform better in large scale networks, which coincides with our network model. Note that we do not use OLSR’s routing protocol. We only use OLSR to generate a network map on every CH. After the network map is obtained by every CH, the Routing controller is used to choose an appropriate path to the destination. As mentioned earlier, any link state routing protocol can be used to replace OLSR. In the next section, we discuss the energy-efficiency and scalability of FDDS-R.

### 7.6.3 FDDS-R is Scalable and Energy-Efficient

Scalability of FDDS-R comes from using OLSR. As mentioned above, OLSR uses MPRs; which minimizes the flooding of control messages in the network. Also OLSR is known to perform well in large scale and dense networks. In our design, we only need to flood information in the backbone network. The backbone network size is very small compared to the total size of the network. In our simulation results (section 7.7), we show that FDDS elects a relatively small backbone. So, FDDS-R achieves scalability.

Energy-efficiency of FDDS-R is directly related to the Routing fuzzy logic controller. The Routing controller takes as input the length of the path and its cost. It returns the quality of the path. The Routing controller gives more preference for paths that are short with powerful nodes. Short path means that a small number of nodes will be affected when the message gets routed. A path consisting of powerful nodes translates to having a high quality path that can handle more traffic. Because the Routing controller balances these
2 criteria, it extends the life of the CHs along the path. Therefore, the objective of the 
Routing controller is to choose energy-efficient routes between (s, d) such that CHs lives are prolonged.

Our simulation results show that networks that use the Routing controller are more survivable than networks that do not use it. In the next section, we present the simulation results that show the power and efficiency of our proposed distributed scheme.

7.7 Simulation Results

In this section, we present the performance evaluation of FDDS through simulation. We focus on evaluating the structural properties (network stretch, backbone size) and the operational properties (energy-efficiency, lifetime) of FDDS. We also compare FDDS to other approaches. FDDS out performs well known approaches suggested in literature. All data points were averaged over 10 simulation runs.

7.7.1 Average Number of Hops

In this experiment, a uniform random generator is used to generate networks of different sizes such that the resultant networks are connected. Networks with sizes $n = 10 \rightarrow 100$ are generated in a 500 square units space. For a given network $A$, $n$ (source, destination) pairs are randomly generated. The shortest paths between all pairs are calculated and then averaged to obtain the average number of hops in network $A$. We then vary the transmission range ($R$) of every node and compare the average number of hops in flat network $A$ with the average number of hops in the network generated using FDDS. Figures 7.22, 7.23, and 7.24 show this comparison for different network sizes when $R = \{150, 250, 350\}$. 

157
All these figures show that the average number of hops in a flat network is less than that in FDDS. This is true because FDDS reorganizes the whole network by removing certain links. So, some paths that exists in a flat network might seize to exist in FDDS. The elimination of such links makes FDDS use slightly longer paths to connect a certain source to its destination. When the transmission range increases, the network becomes more dense. The more dense the network is, the lower the average number of hops is. This can be clearly seen by observing figures 7.22, 7.23, and 7.24. The most interesting observation that we can obtain from these figures is that the difference between the average number of hops in both networks (flat and FDSS) is very small even when the network size increases.

![Average Number of Hops vs. Network Size when R=150](image)

**Figure 7.22** Average number of hops in flat network vs. FDDS when $R = 150$
7.7.2 Network Stretch

Let $S_{ij}$ be the length of the shortest path between nodes $i$ and $j$ in flat network $A$. Let $S'_{ij}$ be the length of the shortest path between nodes $i$ and $j$ in the network generated using
FDDS. Network stretch is defined as:

\[ NS = \text{Max}(\frac{S'_{ij}}{S_{ij}}) \quad \forall \ i, j \]  

(7.5)

In general, a route is efficient if its total cost (or total hop number) is no more than a constant factor of the minimum total cost (or total hop number) needed to connect the source and the destination in the original (flat) communication graph. The constant is called cost (or hops) stretch factor. So, any hierarchical design is preferred to have low stretch. In this experiment, we use the same parameters as those in section 7.7.1, i.e, network size = \( n \), grid size = 500 square units, and \( n \) (source, destination) pairs are randomly generated. For a given network, we vary \( R \) and calculate the network stretch. \( R \) can take values of 150, 250, or 350.

Figure 7.25  Hop stretch of networks with different sizes when \( R = \{150, 250, 350\} \)

Figure 7.25 shows the hop stretch of networks with different sizes and different transmission ranges. As the transmission range increases (network is more dense), network
stretch decreases. This is true because when the nodes’ coverage increases, more shorter paths can be achieved between (source, destination) pairs. This can be also seen by looking at figures 7.22, 7.23, and 7.24 and noticing that the average number of hops decreases as the node transmission range increases. The very important observation that can be made from figure 7.25 is that no matter what the node transmission range is and the network size is, the network stretch is always very close to 2. Having the network stretch to be a constant factor and a very small constant indeed, proves the efficiency of FDDS in terms of constructing the virtual backbone (connected dominating set). We can not really generalize this observation and make it a theorem, but one of our future goals is to theoretically calculate the order of complexity of the network stretch.

7.7.3 Backbone Size

The virtual backbone is mainly used to route traffic between all (source, destination) pairs. Therefore, the virtual backbone nodes (CHs) have higher computation and communication burden than other nodes. Thus one of the important parameters is the number of nodes in the virtual backbone after it is constructed. If the virtual backbone size is large, many nodes might be extensively used leading to their battery drainage. So, the smaller the size of the backbone is, the better the network performance is. But a smaller size backbone means that all the burden is only put on a small set of nodes. This burden might overwhelm the CHs and bring them to a halt. Therefore, sometimes it is better to have more backbone nodes with each one having less duties. This technique will distribute the load on many CHs and more importantly it will allow for multiple paths to exist between (source,
destination) pairs.

![Backbone Size vs. Network Size when R=25](image1)

(a) $R = 25$

![Backbone Size vs. Network Size when R=50](image2)

(b) $R = 50$

**Figure 7.26** Backbone size when the following approaches are used: FDDS, Alzoubi’s [3], and Cardei’s [16]

In this experiment, we change the simulation parameters. We randomly generate net-
works of sizes 20, 50, 80, and 100 and place them in a 100 square units space. The transmis-
sion range of every node can be either 25 or 50 units. We vary the transmission range and
compare the size of the backbone when networks are generated using different approaches.

We implemented the approaches suggested by Alzoubi [3] and Cardei [16]. Figures
7.26(a) and 7.26(b) present the backbone sizes of different networks when the approaches
mentioned above are used. When the node transmission range is small (figures 7.26(a)), our
approach (FDDS) generates a backbone size that is less than that of Alzoubi’s, but larger
than that of Cardei’s. When the node transmission range is large (figures 7.26(b)), our
approach (FDDS) generates a backbone size than is less than that of Alzoubi’s and very
close to that of Cardei’s. The following important observation can be made from figure
7.26(b): as the network size increases, our approach generates backbones with sizes that
are very close (equal or smaller) to that of Cardei’s. By looking at these figures, we can say
that FDDS performs better when the network is large and more dense (R is large). Note
that we could have easily reduced the backbone size when using FDDS by improving the
reduction algorithm described in section 7.1.4. But, we chose to keep the backbone intact
in order to provide more paths between (source, destination) pairs. This is important when
designing an energy efficient routing protocol. For example, paths that contain CHs with
low energy are avoided.

7.7.4 Lifetime

In the following sections, we analyze the energy-efficiency of our distributed approach.
We refer to the whole approach as FDDS; this includes FDDS, FDDS-M, and FDDS-R.
Networks of sizes $10 \rightarrow 100$ are uniformly randomly generated in a 500 square units grid. Traffic is generated based on Poisson traffic distribution (section 6.1.3). Mobility is modeled after the Gauss-Markov model discussed in section 6.1.2. The energy model used is the same model presented in section 6.1.1. The power consumed by the battery when transmitting data is 1749 mW and when receiving is 930 mW. The battery of each node has energy that varies between 1000 and 10000 Joules.

For this experiment, we implemented the approaches suggested by Alzoubi [3], Cardei [16], and Wu [108]. For a given network, we vary $R$ and calculate the lifetime of each network. $R$ can take values of 150 and 300. Figures 7.27 and 7.28 presents the lifetime of networks with different sizes when the approaches mentioned above are used. The bit rate used in this experiment is 250 kbs.

![Network lifetime vs. Network Size when $R=150$](image)

**Figure 7.27** Network lifetime for networks of different sizes when $R = 150$. The approaches compared are: FDDS, Alzoubi’ [3], Cardei’s [16], and Wu [108].
Figure 7.27 presents the lifetime of different network sizes when the following approaches are used: FDDS, Alzoubi’s [3], Cardei’s [16], and Wu [108]. The coverage of each node is set to 150 units. When the network size is small (< 40), all approaches perform the same. For networks of large size, FDDS has the best lifetime and Wu’s approach has the worst lifetime. FDDS has a lifetime that is almost 10 minutes more than the other approaches. Figure 7.28 repeats the same experiment, but with the nodes’ coverage set to 300 units. Such a large coverage produces dense networks. FDDS achieves much better lifetime (almost double) when the network size is 100. We can conclude from this experiment that FDDS performs better when the network is large and more dense. We can also conclude that FDDS achieves its scalability and energy-efficiency objectives.

![Network lifetime vs. Network Size when R=300](image)

**Figure 7.28** Network lifetime for networks of different sizes when node coverage = 300. The approaches compared are: FDDS, Alzoubi’s [3], Cardei’s [16], and Wu [108].
7.7.5 Lifetime Under Increasing Traffic Load

In this experiment, we evaluate the lifetime of a network with 100 nodes under various traffic loads. The coverage of each node is set to $R = 300$. The approaches compared are the same approaches used in the previous section. Figure 7.29 presents the network’s lifetime under increasing traffic load. The traffic loads used vary between 100 and 400 kbs.

Figure 7.29 shows that the other approaches perform better than FDDS when the traffic load is small. But, as the traffic load increases FDDS outperforms the other approaches. When the traffic load is 400 kbs, FDDS’s lifetime is almost double that of the other approaches. This experiment proves that FDDS continues to perform good when the network has high traffic.

![Network lifetime vs. Traffic load when network size = 100](image)

**Figure 7.29** Network lifetime for a network of 100 nodes under increasing traffic load. The approaches compared are: FDDS, Alzoubi’s [3], Cardei’s [16], and Wu [108].
7.7.6 Lifetime with and without Network Setup Controller

In this experiment, we evaluate the effect of using the Network Setup controller on the performance of FDDS. Recall that the Network Setup controller is used to evaluate the network nodes. It gives a high weight for nodes that have the following properties: (1) high residual energy, (2) low traffic, and (3) low mobility.

We start by calculating the lifetime of a certain network, when each node’s weight is calculated using the Network Setup controller. We then calculate the lifetime of the same network, when each node’s weight is set to be its own residual energy. The bit rate is set to 250 kbs and the coverage area of each node is set to $R = 300$. Figure 7.30 presents the results. The results show that networks with small sizes are not affected by the fuzzy logic controller. But, as the network size increases, networks that use the Network Setup fuzzy logic controller survive more than networks that do not use the controller (almost double survivability when network size is greater than 80). This experiment shows the role that the Network Setup controller plays in increasing the network lifetime. It confirms the statement that we made in section 7.1.5: ”FDDS is Scalable and Energy-Efficient”.

7.7.7 Lifetime with and without Routing Controller

In this experiment, we evaluate the effect of using the Routing controller on the performance of FDDS. Recall that the Routing controller is responsible for choosing appropriate routes between (s, d) pairs. Each path is associated with a cost (path cost) equal to the residual energy of the weakest node along the path (the node with the minimum residual energy). Also each path has a certain length. The Routing controller provides an intelli-
Figure 7.30  Network lifetime of networks that use *Network Setup* controller vs. networks that do not use *Network Setup* controller.

Gent path selection criteria that achieves a balance between the residual energy of the nodes along the path and the path length.

Two routing schemes are used in this experiment. The first scheme uses the *Routing* controller to evaluate each path. It then chooses the path with the highest weight. The second scheme calculates the *path cost* (residual energy of the weakest node) of each path. It then chooses the path with the highest *path cost*. We start by calculating the lifetime of a certain network, when the first routing scheme is used. We then calculate the lifetime of the same network, when the second routing scheme is used. The bit rate is set to 250 kbs and the coverage area of each node is set to $R = 150$.

Figure 7.31 shows the results. When the network size is small, networks that do not use the *Routing* controller have higher lifetime than networks that use the controller. When
the network size becomes greater than 60, networks that use the Routing controller start and continue to perform better. When the network size is large, the path length between \((s, d)\) becomes longer. The Routing controller takes into account the path length. It tries to balance the path length and the path cost, while the other routing scheme only considers the path cost. For this reason, when the network size increases our scheme extends the network lifetime.

This experiment shows the role that the Routing controller plays in increasing the network lifetime. It confirms the statement that we made in section 7.6.3: "FDDS-R is Scalable and Energy-Efficient".

**Figure 7.31** Network lifetime of networks that use Routing controller vs. networks that do not use Routing controller
7.8 Summary

In this chapter, we presented a distributed energy-efficient scheme for large scale MANETs. The distributed scheme is composed of three components, namely: (1) the hierarchical component, (2) the maintenance component, and (3) the routing component. Each component achieves energy-efficiency by using a customized fuzzy logic controller.

FDDS is used to handle the hierarchical component in a distributed way. Its job is to elect CHs and connect normal nodes to CHs such that the network composed of CHs is connected. FDDS uses the Network Setup controller, discussed in section 5.2, to combine the node’s residual energy (RE), mobility (M), and traffic (T) according to certain rules keeping in mind the synergy between them. It tends to give a high weight for nodes that have: (1) high residual energy, (2) low mobility, and (3) low traffic. Its objective is to give high quality for nodes that can survive a long time. Our simulation and analytical results show that FDDS is scalable and energy-efficient. FDDS has a low message and time complexity of $O(n)$ and $O(\Delta^2)$ respectively.

FDDS-M is used to handle network maintenance in a distributed way. It is responsible for maintaining the hierarchical structure created by FDDS. As time passes, the configuration of the network changes due to CH retirement and to node/CH movement. FDDS-M uses the Network Maintenance controller, discussed in section 5.2, to evaluate the CHs that maintain the backbone connections. The controller combines the CH’s $SQ$, $RC$, and $RSS$ keeping in mind the synergy between them. Our simulation and analytical results show that FDDS-M is scalable and energy-efficient. FDDS-M has a low message and time complexity of $O(n)$ and $O(\Delta^2)$ respectively.
complexity of $O(n)$ and $O(\Delta^2)$ respectively.

FDDS-R is used to handle network routing. It is responsible for choosing efficient routes between (s, d) pairs. FDDS-R uses an intelligent path selection controller (section 5.4) that can be easily incorporated in any existing link state routing protocol to select energy-efficient routes. We chose to use a well known link state protocol named "optimized link state routing (OLSR)". The power of OLSR lies in its efficient flooding mechanism. The Routing controller combines the path cost and the path length keeping in mind the synergy between them. Its goal is to achieve balance between the residual energy of the nodes along the path and the path length. Our simulation results show that FDDS-R plays a major role in achieving energy-efficiency.

In the next chapter, we present the thesis contribution and discuss future directions.
CHAPTER 8

CONCLUSION AND FUTURE WORK

In this dissertation, we proposed clustering and routing approaches to handle two of the most important challenges in MANETs; scalability and energy-efficiency. To handle these problems, we designed a hierarchical energy-efficient scheme that can be easily setup and maintained. The scheme is composed of three components: (1) clustering component, (2) maintenance component, and (3) routing component.

To obtain the optimal hierarchical structure, we formulated the clustering component using integer linear programming - ILP (Chapter 4). The clustering component is supposed to: (1) elect a group of connected CHs to act as the network’s virtual backbone and (2) connect each normal node to one and only one CH; the group of normal nodes connected to the same CH forms the cluster. We proposed three ILP formulations to the clustering problem: EEC-FCB, EEC-CB, and EEC-R. EEC-FCB considers a fully connected backbone. EEC-CB presents an extension to EEC-FCB where the backbone is connected (not necessarily fully connected). EEC-R presents another extension to EEC-FCB where a backup cluster head is elected for every cluster head. EEC-R designs a reliable and fault tolerant network. It is worth noting that the ILP formulations are used as a theoretical basis and are not intended to be used in practice.

Since the ILP formulations are very slow and not intended to be used in practice, we suggested centralized and distributed heuristic schemes to achieve (1) efficient clustering, (2) efficient maintenance, and (3) efficient routing in large scale MANETs. Energy-
efficiency is mainly achieved by utilizing three custom made fuzzy logic controllers. The first controller (Network Setup controller) is used to achieve efficient hierarchical design. The second controller (Network Maintenance controller) is used to achieve efficient network maintenance. The third controller (Routing controller) is used to achieve efficient routing.

Network Setup controller is used to aggregate the node’s residual energy ($RE$), traffic ($T$), and mobility ($M$). It then produces a single value ($SQ$) that defines the quality of the node. This quality is then used as a deciding factor in the network setup phase in order to decide which node should be a CH and which node should not. The controller uses a set of rules that aggregates the parameters mentioned above keeping in mind the synergy between them. The fuzzy logic rules put more emphasize on $RE$. If $RE$ is low, the quality of the node is bad no matter what the values of $T$ and $M$ are. If $RE$ is high, but $T$ and/or $M$ are high, the quality ($SQ$) of the node is pulled down. Having high $T$ and $M$ is not preferable because they lead to high energy consumption. This fuzzy logic controller gives a high weight for nodes that have high residual energy, low traffic, and low mobility. Therefore, the objective of the Network Setup controller is to give high quality for nodes that can survive a long time. Thus, achieving energy-efficiency and prolonging network lifetime.

Network Maintenance controller is used to achieve energy-efficiency when trying to preserve the network hierarchical structure. It is used by moving nodes in order to evaluate the CHs they can hear. CH evaluation depends on $SQ$, remaining capacity ($RC$), and received signal strength ($RSS$). The controller gives more weight for CHs that have: high $SQ$, high $RC$, and high $RSS$. High $SQ$ means that the CH can survive for a long time and
thus prolonging network lifetime. High $RC$ means that the CH can handle more traffic, i.e., the CH is powerful enough to take more duties. This is important in balancing the traffic load among all the CHs in the network. Balancing the traffic load among all CHs results in a more survivable network. High $RSS$ means that the node is close to the CH. It also means that the node requires less energy to reach the CH and vice versa. So, high $RSS$ translates to energy conservation. Therefore, the objective of the *Network Maintenance* controller is to associate a moving node with an appropriate CH such that network lifetime is prolonged.

*Routing* controller is used by CHs to choose an appropriate path to route messages between (s, d) pairs. The choice is governed by the length of the path and its cost. If the path is long, more CHs will be involved in relaying the packets. So, the residual energy of many CHs will be affected. For this reason, a short path is preferred. The second parameters, Path cost, is decided by the residual energy of the weakest node along the path. Paths with low cost (a node with low RE along the path) are not preferable. The Routing controller combines the path length and cost giving more preference for paths that are short with powerful nodes. Therefore, the objective of the *Routing* controller is to choose energy-efficient routes between (s, d) such that CHs lives are prolonged.

Both the centralized and the distributed schemes we proposed use the above fuzzy logic controllers. In the centralized approach (FEER), each node passes its parameters (RE, T, and M) to the *Network Setup* controller, which returns a weight $SQ$. All nodes in the network send their weight and neighbor list to a central processing node (CPN). The CPN runs FEER and obtains the following information: (1) which nodes are CH, (2) which
nodes are connected to which CHs, and (3) which CHs are connected to each other. The CPN then advertises this information allowing each node to know its role in the network. FEER is divided into four parts: (1) CH Election, (2) Cluster Formation, (3) Network Recovery, and (4) Routing. FEER is distinguished by its network recovery mechanism. *Network Recovery* (section 6.4) aims to design a backbone that is resilient to link failure. It also makes sure that all CHs can handle the traffic demand generated from their clusters. FEER uses the min-cut and the max-flow algorithms to check the resiliency and the capacity of the backbone. If any one of the tests fails, FEER uses a unique *iterative approach* that increases the number of cluster heads without increasing the power level. Network recovery adds more paths between CHs. This enables the routing protocol to choose energy-efficient routes between (s, d) pairs. Our simulation study (section 6.6) shows that FEER succeeds in prolonging the network lifetime. Moreover, it outperforms other approaches suggested in literature.

Since centralized approaches do not scale well, we proposed a distributed scheme that achieves energy-efficiency in large scale MANETs. The distributed approach presents the heart of this dissertation. It includes novel approaches to cluster the network, maintain it, and route traffic between network nodes. The clustering approach uses the *Network Setup* controller, the maintenance approach uses the *Network Maintenance* controller, and the routing approach uses the *Routing* controller.

The first component in the distributed scheme is the clustering approach (FDDS). The clustering approach creates a connected backbone composed of CH. The backbone is created using a fast distributed connected dominating set (FDDS) algorithm. FDDS is divided
into four steps. The first step performs a simple neighbor discovery protocol and assigns a weight for each node. The second step elects an initial set of cluster heads. The third step connects the cluster heads together (those elected in the second step) forming a connected dominating set. The last step eliminates some redundant cluster heads. FDDS achieves scalability since it organizes the network in a hierarchical structure using a completely distributed protocol; each node makes decisions based on its own locality (1-hop and 2-hop neighbors). Moreover FDDS designs a hierarchical structure using low message and time complexities; $O(n)$ and $O(\Delta^2)$ where $n$ is the number of nodes in the network and $\Delta$ is the maximum node degree. To our knowledge, such complexities are the best achieved among proposed schemes found in the literature. FDDS achieves energy-efficiency since it takes into account: (1) the energy restriction imposed on the wireless nodes, (2) the mobility of the wireless nodes, and (3) the traffic pattern of the wireless node. Also, FDDS provides a reliable backbone suitable for mobile ad hoc network applications.

Our simulation study (section 7.7) shows that FDDS achieves good structural properties such as low stretch and small backbone size. The study also shows that networks designed using FDDS are more survivable than networks designed using other well known approaches. Furthermore, networks designed using FDDS along with the Network Setup controller are more survivable than networks designed using FDDS without the controller.

The second component in the distributed scheme is the maintenance approach. The maintenance approach (FDDS-M) is responsible for preserving the hierarchical structure of the network at all times. FDDS-M can be invoked by a normal node or a CH. A normal node invokes FDDS-M if it receives a weak or no signal from its parent CH. A CH
invokes FDDS-M if itself or any of its CH neighbors contribute to the possibility of having a disconnected backbone. FDDS-M achieves scalability since it is completely distributed. In most cases, decisions are made based on 1-hop information. In rare cases, decisions are made based on 2-hop information. Moreover, FDDS-M has low message and time complexity of $O(n)$ and $O(\Delta^2)$ respectively. FDDS-M achieves energy-efficiency because of the Network Maintenance controller it uses. Our simulation study shows that the network lifetime is prolonged when FDDS-M is used.

The third component in the distributed scheme is the routing approach (FDDS-R). FDDS-R is in fact an extension of the well known Optimized Link State Routing Protocol (OLSR). We extended OLSR by designing an intelligent path selection controller (Routing controller) that chooses an appropriate path to route messages between $(s, d)$ pairs. The goal of FDDS-R is to achieve balance between the residual energy of the nodes along the path and the path length. FDDS-R achieves scalability because it uses MPRs, which minimizes the flooding of control messages in the network. FDDS-R achieves energy-efficiency because it uses the Routing fuzzy logic controller (discussed above). Our simulation study shows that networks that use the Routing controller are more survivable than networks that do not use it.

In this dissertation, we designed several schemes to handle energy-efficiency and scalability constraints that exist in large scale MANETs. In all schemes, we adopted a hierarchical design of the network. We summarize the dissertation contributions by the following items:
We obtained the optimal energy-efficient hierarchical structure of a network by formulating the clustering problem using integer linear programming. Three variations of the clustering problem are considered and formulated.

We designed a centralized heuristic approach to cluster the network and route traffic between network nodes. The centralized approach achieves energy-efficiency and outperforms well-known approaches suggested in literature.

We designed three fuzzy logic controllers that aggregate various network metrics. One controller (Network Setup controller) is used to achieve efficient clustering. Another controller (Network Maintenance controller) is used to achieve efficient network maintenance. A third controller (Routing controller) is used to achieve efficient routing. All controllers succeed in increasing the network lifetime.

We designed a distributed heuristic approach to cluster the network (FDDS), maintain it (FDDS-M), and route traffic between network nodes (FDDS-R). FDDS, FDDS-M, and FDDS-R use the fuzzy logic controllers discussed above. The distributed scheme has a low message and time complexity of $O(n)$ and $O(\Delta^2)$ respectively. To our knowledge, such complexities are the best achieved among proposed schemes found in the literature.

Our simulation study shows that FDDS, FDDS-M, and FDDS-R are scalable and succeed in achieving energy-efficiency and in prolonging network lifetime.

Our distributed scheme creates a fault tolerant and reliable network by providing
multiple paths between (s, d) pairs.

- Our distributed scheme achieves very good structural properties, such as low network stretch and small backbone size.

- Our distributed scheme performs better when the network is large and more dense.
  
  This satisfies our objective which is to design a scheme for large scale MANETs.

There are a number of interesting areas of future work that can be identified as a result of the work in this dissertation. We summarize our future directions as follows:

- Design a stand alone routing protocol not based on any existing link state protocol such as OLSR.

- Find a theoretical upper bound on the backbone size generated by FDDS.

- Find a theoretical upper bound on the network stretch generated by FDDS.

- Study the performance of FDDS under a variety of node distributions and mobility models.

- Study the performance of FDDS based on network throughput, delay, and congestion.

- Extend FDDS to be a secure and trusted routing protocol.

- Using MIMO technology on each mobile node.

- Investigate the possibility of implementing our schemes in three dimensional space.


64. Lin, G.-H., and Xue, G. Steiner tree problem with minimum number of steiner points and bounded edge-length. Information Processing Letters 69 (1999), 53–57.


85. **RFC 3626.** Optimized link state routing protocol (olsr), October 2003.


94. SIVAKUMAR, R., SIIIHA, P., and BHARGHAVAN, V. Cedar: Core extraction distributed ad hoc routing. *IEEE Journal on Selected Areas in Communications, Special Issue on Ad hoc Networks* 17, 8 (1999).


100. TOH, C.-K. Long-lived ad-hoc routing based on the concept of associativity. IETF Internet Draft, March 1999.


