Sink localization and topology control in large scale heterogeneous wireless sensor networks

Rui Zhang

University of South Florida

Follow this and additional works at: http://scholarcommons.usf.edu/etd

Part of the American Studies Commons

Scholar Commons Citation


This Dissertation is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Sink Localization and Topology Control in Large Scale Heterogeneous Wireless Sensor Networks

by

Rui Zhang

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Miguel A. Labrador, Ph.D.
        Kimon Valavanis, Ph.D.
        Rafael Perez, Ph.D.
        Tapas K. Das, Ph.D.
        Wilfrido A. Moreno, Ph.D.

Date of Approval:
April 23, 2007

Keywords: Location Service, Energy Efficiency, Multiple Moving Sinks and Targets, Data Dissemination, Location-Based Routing, Connectivity, Network Longevity

(c) Copyright 2007, Rui Zhang
ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere appreciation to all the people that have supported and guided me these past four years. I would especially like to acknowledge my mentor and advisor Dr. Miguel Labrador, who has made considerable contributions to both my academic and personal life. The countless insights and invaluable advice that he has provided me over the years have directly helped in the completion of this dissertation. I would also like to sincerely thank Dr. Valavanis, Dr. Perez, Dr. Das, and Dr. Moreno for being in my committee and for their valuable comments and suggestions.

It takes more than words to express my gratitude to my family for their constant and unconditional support and love. My parents are everything to me. I would also like to take this opportunity to thank all my friends for their continuous encouragement and support during those long nights.
# TABLE OF CONTENTS

LIST OF TABLES iii

LIST OF FIGURES iv

ABSTRACT viii

CHAPTER 1 INTRODUCTION 1

1.1 Wireless Sensor Networks 1

1.2 Sink Localization in WSNs 4

1.3 Topology Control in WSNs 6

1.4 Contributions 7

1.5 Organization of the Dissertation 9

CHAPTER 2 LITERATURE REVIEW 10

2.1 Location Service Protocols 10

2.2 Topology Control 15

CHAPTER 3 ANCHOR LOCATION SERVICE PROTOCOL 20

3.1 The Anchor Location Service Protocol 20

3.1.1 ALS Global Grid Construction Process 21

3.1.2 Anchor Selection Process 23

3.1.3 Query and Data Dissemination Processes 25

3.1.4 Sink and Target Mobility and Agent Chain Maintenance 26

3.2 Theoretical Analysis of ALS 27

3.2.1 Scenario and Notation 27

3.2.2 Communication Overhead 28

3.2.2.1 Process #1: Establishment of the Global Grid and Anchor System 28

3.2.2.2 Process #4: Querying the Anchor System 30

3.2.2.3 Process #5: Data Packet Transmission 33

3.2.3 State Overhead 34

3.3 Performance Evaluation 35

3.3.1 Optimal Cell Size 36

3.3.2 Impact of the Number of Sinks and Sources 40

3.3.3 Impact of the Number of Sources 43
3.3.4 Impact of Sensor Density 45
3.3.5 Impact of Network Area 47
3.3.6 Impact of Sink Mobility 50
3.3.7 Total Communication Overhead 53
3.4 Conclusions 55

CHAPTER 4 RESIDUAL ENERGY AWARENESS DYNAMIC TOPOLOGY CONTROL 56
4.1 Network Model 56
4.1.1 Maxpower Graph 57
4.1.2 Energy Model 58
4.1.3 Weighted Cost Function 61
4.2 The Centralized Residual Energy Aware Dynamic Topology Control Algorithm 63
4.2.1 Centralized Residual Energy Awareness Dynamic Algorithm 63
4.2.2 Simulation Results and Evaluation of READ 65
4.2.2.1 Simulation Setup 66
4.2.2.2 Simulation Results and Analysis Without Packet Transmission 67
4.2.2.3 Results and Analysis with Packet Transmission 69
4.3 The Distributed Residual Energy Aware Dynamic Topology Control Algorithm 73
4.3.1 Distributed Residual Energy Aware Dynamic Algorithm 73
4.3.2 Complexity of the DREAD Topology Control Algorithm 81
4.3.3 Simulation Results and Evaluation of DREAD 82
4.3.3.1 Simulation Setup 82
4.3.3.2 Results and Analysis 83
4.3.3.3 Comparison Between READ and DREAD 87
4.4 Conclusions 89

CHAPTER 5 CONCLUSIONS AND FUTURE WORK 90
5.1 Conclusions 90
5.2 Future Work 91

REFERENCES 92

APPENDICES 97
Appendix A Property Proof of ALS and READ 98
A.1 Convergence Proof of ALS’ Anchor Setup Process and Query Process 98
A.2 Connectivity and Symmetric Property Proof of READ 109

ABOUT THE AUTHOR End Page
LIST OF TABLES

Table 4.1  Simulation parameters for each type of device.  66
Table 4.2  One-hop neighbor table.  75
Table 4.3  One-hop edge weight table.  75
Table 4.4  Two-hop neighbor table.  76
Table 4.5  Two-hop edge weight table.  76
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Node and network architectures (figure is from [1]).</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Mechanism and protocols that support the operation of WSNs.</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>A simple example of location service.</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Taxonomy of location service protocols.</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Taxonomy of topology control.</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>The grid node selection process.</td>
<td>22</td>
</tr>
<tr>
<td>3.2</td>
<td>Left and right hand rule in the anchor setup process.</td>
<td>24</td>
</tr>
<tr>
<td>3.3</td>
<td>The anchor system and the query and data dissemination processes.</td>
<td>25</td>
</tr>
<tr>
<td>3.4</td>
<td>Calculation of $c_1$.</td>
<td>29</td>
</tr>
<tr>
<td>3.5</td>
<td>Anchor setup time vs. cellsize $\alpha$.</td>
<td>37</td>
</tr>
<tr>
<td>3.6</td>
<td>Grid and anchor setup overhead vs. cellsize $\alpha$.</td>
<td>38</td>
</tr>
<tr>
<td>3.7</td>
<td>Location time vs. cellsize $\alpha$.</td>
<td>38</td>
</tr>
<tr>
<td>3.8</td>
<td>Location overhead vs. cellsize $\alpha$.</td>
<td>39</td>
</tr>
<tr>
<td>3.9</td>
<td>Data delay vs. cellsize $\alpha$.</td>
<td>39</td>
</tr>
<tr>
<td>3.10</td>
<td>State overhead vs. cellsize $\alpha$.</td>
<td>40</td>
</tr>
<tr>
<td>3.11</td>
<td>Anchor setup time vs. the number of sinks.</td>
<td>41</td>
</tr>
<tr>
<td>3.12</td>
<td>Grid and anchor setup overhead vs. the number of sinks.</td>
<td>41</td>
</tr>
<tr>
<td>3.13</td>
<td>Location time vs. the number of sinks.</td>
<td>42</td>
</tr>
<tr>
<td>3.14</td>
<td>Location overhead vs. the number of sinks.</td>
<td>42</td>
</tr>
<tr>
<td>3.15</td>
<td>State overhead vs. the number of sinks.</td>
<td>43</td>
</tr>
</tbody>
</table>
Figure 3.16 Anchor setup time vs. the number of sources.
Figure 3.17 Grid and anchor setup overhead vs. the number of sources.
Figure 3.18 Location time vs. the number of sources.
Figure 3.19 Location overhead vs. the number of sources.
Figure 3.20 State overhead vs. the number of sources.
Figure 3.21 Average anchor setup time vs. the network density.
Figure 3.22 Grid and anchor setup overhead vs. the network density.
Figure 3.23 Average location time vs. the network density.
Figure 3.24 Anchor setup time vs. the network area.
Figure 3.25 Grid and anchor setup overhead vs. the network area.
Figure 3.26 Location time vs. the network area.
Figure 3.27 Location overhead vs. the network area.
Figure 3.28 State overhead vs. the network area.
Figure 3.29 Anchor setup time vs. sink mobility.
Figure 3.30 Grid and anchor setup overhead vs. sink mobility.
Figure 3.31 Agent chain break times vs. sink mobility.
Figure 3.32 Location time vs. sink mobility.
Figure 3.33 Location overhead vs. sink mobility.
Figure 3.34 State overhead vs. sink mobility.
Figure 3.35 Total communication overhead vs. number of sinks, $\alpha = 200$. 54
Figure 3.36 Total communication overhead vs. number of sources, $alpha = 100$. 54
Figure 3.37 Total communication overhead vs. network area, $alpha = 200$. 55
Figure 4.1 Maxpower graph with 100 nodes.
Figure 4.2 LMST with link addition.
Figure 4.3 LMST with link removal.
Figure 4.4  R&M with link addition.  68
Figure 4.5  R&M with link removal.  68
Figure 4.6  READ with two-degree connectivity.  69
Figure 4.7  READ with one-degree connectivity.  69
Figure 4.8  Average node degree.  70
Figure 4.9  Average link length.  70
Figure 4.10  Number of nodes alive in centralized implementation.  71
Figure 4.11  Successful delivery rate in centralized implementation.  71
Figure 4.12  Number of military nodes alive in centralized implementation.  73
Figure 4.13  Number of robots alive in centralized implementation.  73
Figure 4.14  Number of PDA nodes alive in centralized implementation.  74
Figure 4.15  Number of sensors nodes alive in centralized implementation.  74
Figure 4.16  Establishment of two-hop tables.  77
Figure 4.17  Generation local minimal spanning tree.  78
Figure 4.18  Topology generated by DREAD.  83
Figure 4.19  Number of nodes alive in distributed implementation.  85
Figure 4.20  Successful delivery rate in distributed implementation.  85
Figure 4.21  Number of military nodes alive in distributed implementation.  87
Figure 4.22  Number of robots alive in distributed implementation.  87
Figure 4.23  Number of PDA nodes alive in distributed implementation.  87
Figure 4.24  Number of sensors nodes alive in distributed implementation.  87
Figure 4.25  Number of nodes alive of CREAD and DREAD.  88
Figure 4.26  Successful delivery rate of CREAD and DREAD.  88
Figure 4.27  Number of military nodes alive of CREAD and DREAD.  89
Figure 4.28  Number of robots alive of CREAD and DREAD.  89
Figure 4.29  Number of PDA nodes alive of CREAD and DREAD.  

Figure 4.30  Number of sensors nodes alive of CREAD and DREAD.  

Figure A.1  A graphical representation of real-grid-nodes (solid nodes), void-grid-nodes (hollow nodes), real-edges, void-edges, and void areas.  

Figure A.2  A graphical representation of four Real Polygon examples.  

Figure A.3  A graphical representation of void areas and their envelops.  

Figure A.4  Base case with 1 void-grid-point.  

Figure A.5  Base case 1 with 2 void-grid-points.  

Figure A.6  Base case 2 with 2 void-grid-points.  

Figure A.7  Step with 1 void-grid-node as far neighbor.  

Figure A.8  Step with 1 void-grid-node as neighbor.  

Figure A.9  A simple max power network example.
SINK LOCALIZATION AND TOPOLOGY CONTROL IN LARGE SCALE
HETEROGENEOUS WIRELESS SENSOR NETWORKS

Rui Zhang

ABSTRACT

Wireless Sensor Networks (WSNs) continue to evolve as new applications emerge. In the recent past, WSNs were mostly single sink networks with a few number of homogeneous and static sensor nodes. Now, several applications require networks with multiple and moving sinks and targets as well as thousands of heterogeneous devices. However, the same constraints remain: sensor nodes continue to be very limited in resources, posing new challenges in the design of scalable and energy-efficient algorithms and communication protocols to support these new applications.

This dissertation first addresses the problem of sink localization in large scale WSNs. A scalable and energy-efficient sink localization mechanism, called the Anchor Location Service (ALS), is introduced to support the use of location-based routing protocols. ALS avoids frequent and costly flooding procedures derived from the mobility of the sinks and targets, and utilizes face routing to guarantee the success of localization.

The problem of topology control in heterogeneous environments is addressed next. A new topology control mechanism, the Residual Energy-Aware Dynamic (READ) algorithm, is devised to extend the lifetime of the network while maintaining connectivity. READ extends the lifetime of the network by assigning a more prominent role to more powerful devices.
ALS and READ are evaluated and compared with other well-known protocols using analytical means and simulations. Results show that ALS provides a scalable sink location service and reduces the communication overhead in scenarios with multiple and moving sinks and targets. Results also show that READ increases both the network lifetime and the packet delivery rate.
CHAPTER 1
INTRODUCTION

1.1 Wireless Sensor Networks

Sensors integrated with microelectronic technologies emerged decades ago. Early sensors were used individually in applications to monitor smoke inside residences, collect indoor or outdoor temperature, or collect sound stimulus in stairways. Since then, sensors have evolved from those isolated single sensors to those with the ability to communicate wirelessly and collaborate with each other. The proliferation of sensing and wireless communication technologies in conjunction with the development of microelectronics has made a new breed of more powerful wireless sensor devices available. Their application scenarios have also expanded from simple cases to a countless number of more complicated ones.

Several of these new applications are envisioned with hundreds of thousands of these small wireless devices spread over very large areas to monitor the environment, perform intrusion detection, collect seismic information, etc. One proposed concrete application could be a vast heterogeneous sensor network deployment along the border between the United States and Mexico, from San Diego, California, to Brownsville, Texas, to protect the country from illegal immigration. The border is 1951 miles long and traverses a variety of terrains, which makes it very difficult, if not impossible, to monitor illegal intruders by human power. Wireless Sensor Networks (WSNs) make this very difficult mission much easier to achieve.
Classic wireless sensor node and network architectures are shown in Figure 1.1. All
sensors have the following hardware components: sensing unit, processing unit, with stor­
age, transmission unit and power supply. In terms of the network architecture, sensors auto
organize themselves to collect information and relay packets for one another without the
need of any infrastructure. Messages are forwarded from the Source to the Base Stations
(BS), which may connect the sensor network to the Internet or any other public or private
network.

![Diagram of sensor network architecture]

Figure 1.1 Node and network architectures (figure is from [1]).

Although sensors differentiate from each other in terms of sensing functionality, phys­
ical dimension or even mobility, they all have some common characteristics. For instance,
most sensor nodes have very limited computational capabilities, storage capacity, and en­
ergy resources. Therefore, it is very important to design simple and energy efficient com­
munication protocols and algorithms for WSNs.
Although wireless sensor nodes do consume energy during both sensing and processing tasks, communication is by far the most energy consuming task. For example, the RFM TR1000 radio transceiver included in the Berkeley motes consumes 1\(\mu\)J to transmit one bit and 0.5\(\mu\)J to receive one, while it takes around 8\(n\)J of energy per instruction. This results in a communication to processing ratio of about 190. Other transceivers, such as Rockwell’s WINS nodes and MEDUSA II nodes [2] nodes, have 220 to 2900 ratios. Two main conclusions can be drawn from these power consumption figures. First, communication costs must be minimized. Second, it is worth spending additional processing cycles if they can contribute to communication savings. This is known as ”in-network processing”.

WSNs are supported by many mechanisms and protocols. A sample of the most important ones is shown in Figure 1.2. Most of these mechanisms and protocols are designed with constraints of WSNs in mind. For instance, Routing Protocols utilize either location information or local neighbor tables to route packets from Source to Destination. Schedul-
ing in Physical Layer is a strategy employed to turn nodes on and off in order to reduce energy consumption. Similarly, innovative Smart MAC layer protocols have been proposed to reduce wireless signal collision and energy consumption. In Data Dissemination, information is replicated in certain nodes that act as repositories. One possible approach is to employ quorum-based strategies to update and query this information. Time Synchronization always attracts attention in distributed systems. Time synchronization algorithms are used to keep the sensor clocks as tightly synchronized as possible considering the scalability and energy constraints. Localization Service is another very important research area in sensor networks. Location information is needed to know the location of events and also supports the use of location-based routing protocols. Topology Control Protocols are used to adjust and simplify the network topology to save energy.

This dissertation focuses on Localization and Topology Control. A solution to the problem of scalable and efficient sink localization in large scale WSNs is introduced first. Then, a topology control algorithm that considers the coexistence and cooperation of heterogeneous devices is included. Two mechanisms, ALS and READ, are introduced next.

1.2 Sink Localization in WSNs

Location-based routing has recently emerged as an important approach to address the scalability and energy efficiency concerns for data dissemination in large scale WSNs. For example, in location-based routing, nodes do not need to make complex computations to find the next hop as routing decisions are made based on local information. Moreover, because of the locality, nodes do not need to maintain large data structures. Finally, location-based routing substantially reduces the communication overhead because routing table advertisements, like those found in traditional routing protocols, are unnecessary. For instance, when geographical greedy routing protocol, a location-based routing protocol, is
utilized, packets include destinations’ locations and every node only needs to maintain a one-hop neighbors’ location table. The processing node performs a calculation using local information and selects from its one-hop neighbors a neighbor geographically closest to the packet’s destination to forward the data packet to.

While these are important features, very well in line with the constraints and characteristics of WSNs, location-based routing protocols assume and rely on the existence of a scalable and energy efficient mechanism to distribute the location information of the sinks or destinations. Unfortunately, most of the existing location mechanisms utilize some sort of flooding procedure to spread the sink’s location, which is unsuitable for large scale WSNs. Furthermore, this flooding procedure is frequently repeated in cases with multiple and moving sinks and targets.

Figure 1.3 illustrates an application scenario, where numerous sensors are deployed in a large network area. Without the support of any infrastructures, sensors work collaboratively
to collect elephants’ migration information from the sensor network and then provide that information to multiple rangers by using a location-based routing protocol. Once a target elephant moves the sensors surrounding the elephant collect the stimulus. One of these sensors fuses the data and acts as a source to send fused data to destinations, which in the case are moving rangers.

However, the common assumption held here by almost all the location-based routing protocols is that the destination’s location information, in this instance, the ranger’s location information, has been disseminated into the entire network in a scalable and energy efficient way. In this particular application, however, there could be more than one moving elephant or more than one moving ranger, which increases the difficulty of disseminating destination location information. Frequently repeated flooding queries for rangers’ locations in a large network like this is clearly neither a scalable nor an energy efficient solution.

Although location services have been under investigation for some time, there are few solutions provided in the literature that are suitable for large scale WSNs, especially with the consideration of multiple mobile targets and sinks. The sink location service proposed in this dissertation provides a complete scalable and energy efficient solution for this type of application.

1.3 Topology Control in WSNs

Topology control is one of the most important mechanisms utilized in WSNs to reduce energy consumption. Topology control is well defined in [3] as the art of coordinating nodes’ decisions regarding their transmission ranges in order to generate a network with the desired properties while reducing node energy consumption.

Although topology control has been studied for some years, current topology control approaches only consider homogeneous sensor networks, where the differences of initial
energy of batteries and sensitivity of transceivers are omitted. However, the wide spectrum of possible applications where wireless ad hoc and sensor networks can be applied has increased the possibility of mixed networks, where devices of different types and characteristics co-exist and work in the same application. In this type of heterogeneous environment, it is very important to devise algorithms and mechanisms that will allow different devices to collaborate, each taking advantage of the goodness of the others. It is this approach to topology control algorithms that this research takes, where more powerful devices are set to have a more prominent role in the network connectivity to extend the network’s lifetime.

1.4 Contributions

This research introduces the ALS protocol, a grid-based protocol that provides sink location information in a scalable and efficient manner, and thus supports location-based routing for large scale WSNs. In ALS, each sink builds a global grid made of special location server nodes called anchors that are used by all sources to find its location. Because of this global grid structure, fewer location request messages need to travel through the network. This also reduces the number of collisions in wireless transmissions, which results in additional energy savings. Furthermore, additional processing is done in the anchors to store the location information of all sinks and respond to query messages. Considering the size of large scale WSNs, ALS not only reduces the communication costs but also the location information response time, as sources will only have to find the first global anchor grid.

In this study, the strategies used in the construction of the global grid system and the selection of the anchors are presented. In addition, the location dissemination process and the location query process are described. The case of very large wireless sensor networks with rather static and resource constrained nodes and scenarios with multiple and moving
sinks and targets are considered. This is a very common scenario since many sources from many different places might be transmitting information to one or more sinks at the same time. Using a mathematical approach and simulations, not only is the performance of the location service assessed, but also, by using a location-based routing protocol, ALS’s performance is compared with the well known grid-based TTDD [4–6]. The location time and overhead, as well as and the communication and state overhead of the protocols are considered as main performance metrics, which are presented varying the number of sinks and sources, the network size, the network density and the speed of mobile sink nodes. The results demonstrate that ALS provides a scalable and efficient location service. Compared to TTDD, ALS reduces location query overhead by at least 50% with multiple sinks and 40% with multiple sources. With different network area, ALS reduces location query overhead by at least 70% in the worst case and 90% in the best case.

Although topology control problems have been studied in the context of heterogeneous wireless ad hoc and sensor networks before, most existing mechanisms have focused on varying the nodes’ transmission power based on the assumption that all the wireless devices have identical physical characteristics. As a result, topology control problems have been solved as range assignment problems, which not only neglect the heterogeneity of the network but also don’t take advantage of the unique capabilities of different devices.

In this chapter, the READ topology control algorithm and the DREAD topology control algorithms are presented. Both mechanisms consider the problem of topology control in heterogeneous wireless scenarios, where sensor nodes, ad hoc nodes, robots with communication capabilities and even more powerful military wireless devices work together in the same application. In this heterogeneous scenario, the assumption of identical initial energy, residual energy, receiver sensitivity, and antenna gain for every wireless devices does not hold. Therefore, the topology control problem needs to be formulated as a power assignment problem. READ considers these aspects in the formulation of the optimization
problem to dynamically recruit the links that optimize the workload between different wireless devices while still maintaining network connectivity. DREAD provides a distributed solution with the same considerations in mind.

Both READ and DREAD algorithms are implemented in a simulation tool, and their performances are compared with R&M and LMST, two well known topology control algorithms for wireless ad hoc and sensor networks. It is demonstrated that READ and DREAD extend the network lifetime by making more powerful nodes play a more important role in the network. The network with READ as topology control algorithm can last much longer with better delivery rate.

1.5 Organization of the Dissertation

The remainder of this dissertation is organized as follows: Chapter 2 reviews the existing literature regarding location services and topology control algorithms. Chapter 3 presents the ALS protocol in detail and its theoretical analysis. Simulation results are also presented in this chapter. Chapter 4 presents the Dynamic Residual Energy Awareness topology control algorithm. The network model, details of the algorithm, and its performance evaluation are presented in this chapter. Chapter 5 summarizes the dissertation and presents direction for future research.
CHAPTER 2
LITERATURE REVIEW

2.1 Location Service Protocols

Location service mechanisms can be broadly divided into three categories as shown in Fig. 2.1. They are quorum-based systems, home-based systems, and systems with approximate information. In quorum-based systems, the set of nodes is divided into mutually disjoint subsets where information about each node is replicated within its own subset (quorum). As a result, each element in a quorum can respond to queries coming from a different quorum. These subsets are designed in such a way that their intersection is non-empty and the requesting node finds the desired information.

In one type of quorum-based systems, location information is sent in one direction (e.g., north and south) to update the other nodes (location servers) in the quorum, while query messages are sent in the orthogonal direction (e.g., east and west) from nodes contained in different sets. Several quorum-based mechanisms have been proposed for wireless mobile ad hoc networks. The scheme presented in [7] maintains the quorum structure as the nodes move. The novelty of this scheme lies in the update mechanism, which utilizes link incidents as location updates triggering signals instead of using distance or movement-based approaches, like the ones utilized in [8, 9]. The scheme presented in [10, 11] utilizes the same update strategy but organizes the quorum differently. The scheme utilizes a quorum-based location service that avoids partial flooding overhead, and/or location failures in group movement scenarios. In this scheme, the destination node distributes its location to
all nodes located to the north and south of their current location while sources send messages in the east and west direction to search for the location of the destination, which is finally found at the intersection. The authors utilized face routing in the distribution and search mechanisms to guarantee the success of the location service. They proposed four variants with different update and search strategies and performed simulation experiments to determine the success rate and communication overhead of the strategies in scenarios with different number of nodes and node degree. The scheme is shown to provide a good location service to static and mobile nodes, including nodes that move in groups and toward the same direction. One of the drawbacks of the scheme, however, is that the messages that distribute the location information of the destination and the search messages may travel through the entire network even if the source and destination are relatively close to each other. Other quorum-based schemes utilize a random approach to obtain up-to-date information in several sets [12, 13] while other schemes build and maintain a virtual backbone with server nodes that maintain the location information [14–16]. However, these later schemes need to send location updates and queries through the entire backbone, and it is not clear if this overhead is better than using simple flooding.
Home-based systems are similar to those well known location management mechanisms utilized in cellular networks. Home-based systems divide the network in several zones, and, then, nodes affiliate with a particular zone (home) and share their location information with the other nodes within their zone. Nodes constantly send updates to their home zone so that their current positions are known. In this manner, location queries can be sent to zones instead of individual servers. As it can be inferred, these schemes require a considerable amount of overhead and introduce routing inefficiencies in scenarios with high mobility. In this case, nodes have to send position updates more frequently to keep the nodes’ positions up-to-date, and these messages have to travel through longer routes. In the case of static networks, these schemes are known to work well but use centralized servers per zone, which is not suitable for heterogeneous WSNs because of energy and memory constraints. In the case of large scale WSNs, it is not clear if these schemes provide better performance and consume less energy than other approaches, such as the ones proposed in Section 4.2 and Section 4.3. Among the most relevant home-based schemes proposed in the literature are [17–21].

Several other location services are based on approximate information. In [22], for example, the authors present a scheme by which nodes update their positions in concentric circles of doubling size. Whenever a node moves out of its present circle, it broadcasts its new position to all nodes inside a new circle centered at the current node’s position. The main problem with this scheme is that larger circles may cover the entire network, which introduces a considerable amount of overhead, not particularly suited for large scale WSNs. A similar scheme, which utilizes a hierarchy of square regions instead of circles, is presented in [23]. In the proposed Grid Location Service (GLS) scheme [23], as the distance increases, location updates are sent to fewer number of location servers. Although GLS is more scalable than the scheme presented in [22], it is complex to understand and its efficiency has not been assessed in very large networks. DREAM [8] and LAR [9] are
other schemes in which nodes flood the network with their positions in a proactive or reactive manner, respectively, and the location of the destination is estimated within a region. Of course, the flooding procedure makes these mechanisms unsuitable for large scale networks. The Zone-based Location Service (ZLS) mechanism presented in [24] is another example that belongs to this category.

Perhaps the most similar protocol to ALS is the Two-Tier Data Dissemination (TTDD) protocol for large scale WSNs presented in [4–6]. TTDD is a grid or quorum-based protocol that provides location information and routing in an integrated manner. Upon detection of a stimulus, the source node pro-actively builds a grid structure throughout the sensor field. The procedure selects sensors located at grid points, called dissemination nodes, that the protocol uses to receive and forward information from source to sink. This virtual backbone for routing made of dissemination nodes has been criticized for not providing optimal paths [25]. Once the grid structure has been built, a query from a sink travels through two tiers to reach the source node. If the position of the destination is unknown, TTDD utilizes flooding, otherwise, greedy forwarding is used. Initially, the sink floods its query in the lower tier, which is within the cell of the sink’s current location. The flooding continues until it reaches the closest dissemination node. At this point, the message reaches the higher tier, which is made up of all dissemination nodes from the sink’s cell to the source’s cell. The closest dissemination node to the sink receives the query and forwards it to the next dissemination node located upstream and toward the source. The query is then forwarded through the higher tier until it reaches either the source node or a dissemination node currently receiving data on behalf of the source. This process provides information of the path back to the sink node, enabling the source information to traverse the same two tiers but in reverse order.

Both TTDD and ALS are meant to handle multiple mobile sinks and sources. A sink trajectory forwarding strategy makes sink mobility transparent to the higher tier whenever
the sink moves within the current cell. If it moves beyond the cell, a new dissemination node discovery procedure needs to be triggered with the associated overhead. However, this overhead is expected to be small, as new dissemination nodes are likely to be found in adjacent cells.

There are several similarities and differences between TTDD and ALS. Both are grid-based or quorum-based systems that assume sensor nodes are stationary and aware of their locations. Both protocols are scalable in the sense that they avoid global flooding as the main mechanism to disseminate data and location information. Global flooding is avoided by confining queries to local cells, which has the dual advantage of reducing both energy consumption and network overhead when compared to other protocols. However, the schemes are different in several aspects. For example, TTDD is source (not sink) oriented, as it establishes one grid per source. This is an important difference in terms of the final overhead because the number of sinks is usually known in advance to the network designer, while the number of targets is completely unknown. Also, TTDD utilizes the dissemination nodes to distribute both location and data while ALS only uses the grid to distribute location information. This is also an important difference. ALS decouples the routing and the location functions. As such, once the location of the sink is known to the source, any location-based routing protocol, such as GFG [26, 27], Greedy Perimeter Stateless Routing (GPSR) protocol [28] or Scalable Energy-Efficient Location-Aided Routing (SELR) Protocol [29], may be used. Also, the dissemination nodes are expected to have their energy drained considerably faster than the other nodes, and therefore a strategy to change them frequently must be included. In ALS, the anchors do not bear this load, as they only respond to location queries. Two additional differences are worth mentioning. First, TTDD utilizes greedy forwarding while ALS utilizes greedy forwarding with face routing [26,27]. This is important because greedy forwarding alone does not guarantee packet delivery. Second, ALS uses the "in-network processing" strategy called "propagated fusion" to further
optimize the performance of the protocol. As new anchor systems are being set up, current anchors include the new locations in their memories. Query messages do not need to travel further to find the required location then, and the location time is reduced.

There is a very large amount of literature on routing mechanisms for wireless ad hoc and sensor networks. Since this is not the main topic of this research though, only key references are provided. The interested reader is directed to [30–33] to learn about routing algorithms for wireless mobile ad hoc networks. Survey papers on routing algorithms for WSNs can be found in [1,34–36]. In addition, [37,38] include an extensive and thorough review of location-based routing schemes for wireless ad hoc and sensor networks, and [39] provides a good survey on location-based routing and location services for ad hoc networks.

### 2.2 Topology Control

Current topology control algorithms can be categorized as homogeneous, nonhomogeneous and heterogeneous as shown in Figure 2.2. Homogeneous topology control algorithms assume that all wireless devices use the same transmission range. Correspondingly, the topology control problem becomes a range assignment problem that searches for the minimal transmission range or critical transmitting range (CTR) while maintaining certain network properties, such as network connectivity. Problems of finding CTR are the simplest topology control problems to formulate and solve, and were the first to appear in the literature. The assumption that all nodes use the same transmission power, however, holds only if all the wireless transceivers have no difference in their technology and finding the minimal transmission range is the only way to reduce energy consumption. For instance, the schemes presented in [40] and [41] belong to this category. [40] proposed a distributed topology control algorithm to construct a planar spanner of unit-disk graph. The resulting graph contains all Delaunay triangulation edges from the unit-disk graph. In the resulting
topology, the shortest path between any two nodes $u$ and $v$ is at most a constant factor of the shortest path connecting $u$ and $v$ in unit-disk graph. Several other examples can be found in [3].

It has been proven that the critical transmitting range with preservation of connectivity equals to the length of the longest edge of the Euclidean Minimum Spanning Tree (EMST). However, the knowledge of the exact locations of all the nodes is necessary to calculate the length of the longest EMST. Due to the strong assumption of knowing the exact node’s location and the huge amount of control message overhead to exchange location information network-wide, researchers have devoted their attention to find the CTR with the presence of uncertainty about node positions. The typical approach is to study the conditions for asymptotically almost sure connectivity with a certain node probability density distribution in the area. In dense networks, geometric random graphs theory has been utilized to solve the problem. In 1997, Mathew D. Penrose proved in [42] that if $n$ points are distributed uniformly at random in the unit square $[0, 1]^2$ and letting $M_n$ be the random variable denot-
ing the length of the longest MST edge built on the $n$ nodes, then Equation 2.1 will hold for any $\beta \in R$ as follows:

$$
\lim_{n \to \infty} P[n\pi(M_n)^2 - \log n \leq \beta] = \frac{1}{e^{\exp(e^{-\beta})}}
$$

(2.1)

A further corollary is that if the area is unit square and $n$ nodes are distributed uniformly at random, then the CTR for connectivity is:

$$
R_C = \sqrt{\frac{\log n + f(n)}{n\pi}}
$$

(2.2)

where $f(n)$ could be any function that satisfies the condition $\lim_{n \to \infty} f(n) = +\infty$.

However, the assumption that the network area is always in unit is not very pragmatic. Some researchers, therefore, have added one more parameter, $l$, the side length of the deployment region, to the model. After this modification, the network density $\frac{n}{l^d}$ can vary in from 0 to any constant $c$, where $d$ is the order of dimension of the network space. The proposition was proven in [3] that if the area is the $[0, l]^d$ with $d = 2, 3$ and $n$ nodes are distributed uniformly at random in the area, the CTR for connectivity is

$$
R_C = \frac{k\frac{l^d \log l}{n}}{n}; \quad 0 \leq k \leq 2^d d^\frac{d}{2} + 1
$$

(2.3)

In non-homogeneous topology control, on the other hand, different wireless devices can choose different transmission ranges within the same maximum transmitting range. Under the assumption that all nodes have the same path model with the same parameters, the transmitting range and transmit power level are exchangeable concepts. Compared with homogeneous topology control, non-homogeneous topology control methods seek to find the desired transmission range for each individual wireless device, while maintaining certain network properties and achieving specific objectives. In [3], the range assignment problem
was defined as follows: let \( N \) be a set of nodes in the \( d \)-dimensional space, with \( d = 1, 2, 3 \). Determine a range assignment function \( \overline{RA} \) such that the corresponding communication graph is strongly connected, and \( c(\overline{RA}) = \sum_{u \in N} (\overline{RA}(u))^\alpha \) is minimum over all connecting range assignment functions, where \( \alpha \) is the distance-power gradient. Andrea Clementi in [43] and Lefteris Kirousis in [44] have proven that the Range Assignment problem is an NP-hard problem in two-dimensional networks and in three-dimensional networks, respectively. Soon after that, Mahesh Marina and Samir Das in [45] observed that in the case of routing protocols, unidirectional links incur more overhead, which could override the benefits provided by directional links. Also, due to the wide adoption of IEEE standard 802.11 standard at MAC layer, the Symmetric Range Assignment problem became popular, where the resulting communication graph contains only bidirectional links. In [46], Blough et al. proved that the Symmetric Range Assignment problem is also NP-hard.

Several non-homogeneous topology control mechanisms have been proposed in the literature. For example, the algorithms presented in [47–52] also minimize energy consumption. In [51], the authors consider mobile and static networks and propose two solutions to find the maximum transmission power while maintaining connectivity and bi-connectivity. Two centralized algorithms are proposed for the static version, while two heuristics algorithms are proposed for the mobile version. In [52], the authors present a distributed cone-based topology control algorithm for non-homogeneous multi-hop wireless ad hoc networks. This algorithm aims at guaranteeing network connectivity and increasing the network lifetime by determining the minimal operational power requirement for each node individually. The algorithm is a direction-based topology control scheme and assumes that all the nodes only need to know their neighbors’ direction. The authors use the number of sensors still alive over time to evaluate the performance (network lifetime) of the algorithm.

In [49] and [50], Ning Li and Jennifer C. Hou proposed a distributed neighbor-based topology control algorithm for non-homogeneous wireless networks, in which every sensor
is assigned a different transmission range. In the algorithm, called LMST, each node builds its one-hop local minimum spanning tree with the distance as the weight cost associated with each edge. The topology derived preserves the network connectivity, and the degree of any node in the topology is bounded by 6. Moreover, the topology can be transformed into one with bi-directional links.

In [47], Volkan Rodoplu and Teresa H. Meng proposed a location-based topology control scheme, R&M for ad hoc networks, where all the nodes have very accurate information about their location. In R&M, each node eliminates any nodes in its relay region and only selects the links from its immediate neighborhood that maintain the connectivity within its enclosure. Instead of transmitting directly, a node chooses to relay through other nodes if less power is consumed. If every node maintains links with the nodes in its enclosure, it is shown that the resulting topology is a minimum power, strongly connected topology. Due to these advantages, LMST and R&M have become widely known and benchmark algorithms for performance comparison. However, they still assume that all the devices are similar and start work with the same configuration.

Homogeneous and non-homogeneous topology control algorithms differ from the one presented in Chapter 4. In this dissertation, however, it is no longer assumed that the network devices are similar; instead, wireless devices with different capabilities and characteristics are considered. Therefore, known homogeneous and non-homogeneous algorithms will not provide the expected results. In a heterogeneous network, wireless devices may have different receiver sensitivities, antenna gains, maximal transmission powers, and/or different batteries, and consequently, homogeneous or non-homogeneous algorithms cannot be used directly. Heterogeneous network topology control problems have not been formulated and solved so far.
CHAPTER 3
ANCHOR LOCATION SERVICE PROTOCOL

3.1 The Anchor Location Service Protocol

The ALS protocol proposed in this chapter can be categorized as a quorum-based scheme, as it also utilizes columns and rows to divide the network and selects a set of nodes as location servers (anchors). ALS was designed from the ground up for large scale networks with rather static nodes in mind. As such, the proposed protocol is expected to provide superior performance when compared to the other quorum-based and home-based schemes presented in Section 2.1. For example, the anchor system established in ALS is expected to provide better search times than the schemes presented in Section 2.1, mainly because of proximity. On average, it is expected that search messages will go through fewer number of hops using ALS than the other quorum-based and home-based schemes.

ALS is also expected to substantially reduce overhead compared to any of the approximate information-based schemes since flooding is reduced to local exchanges. Furthermore, most of the schemes included in the related work presented in Section 2.1 were designed with mobile ad hoc networks in mind. As a result, one can argue that in order to handle mobility, they need to be more complex and less efficient than ALS. Finally, neither energy efficiency nor scalability to a very large number of nodes are design considerations in most of those schemes. As a result, it is unknown if these protocols can be used directly in large scale WSNs. These are additional justifications for the comparison of ALS with the TTDD protocol only.
The ALS protocol was first introduced in [53] and then expanded in [54] and ?? In ALS, the sensor field is represented as a two-dimensional plane constructed along an \((x, y)\) axis and divided into equal-sized cells. The predefined geographical crossing points of the grid structure are referred to as grid points. ALS constructs a single global grid structure by assigning sensors nearest to the grid points as grid nodes. Each sink selects the nearest grid node as its sink agent to distribute its location information. Then, the sink agent selects some special grid nodes as anchors and builds an anchor system that contains the location of the sink agent. When the sink moves, a sink agent chain is formed dynamically to keep track of it, and the first immediate agent in the chain is referred to as the primary agent. Upon detection of a stimulus, a source node queries the anchor system to obtain the location of the sink agent. After that, data is transmitted using any location-based routing protocol. In the above case, the GPSR protocol is utilized. In the following sections, the global grid construction process, the anchor selection process, the query and data dissemination processes, and Sink and Target Mobility and Agent Chain Maintenance will be described.

### 3.1.1 ALS Global Grid Construction Process

After all the sensors are deployed, they participate in the global grid construction process. Sensors need two predefined parameters to build the global grid structure: \(\alpha\), which is the size of each grid cell, and \((X_{\text{base}}, Y_{\text{base}})\), the baseline coordinate of a pre-defined coordinate system. These parameters are either hard coded in the nodes or set in the mission messages. The positive directions of \(x\)-axis and \(y\)-axis of the predefined coordinate space are pointing to the east and north, respectively. As in TTD, with this information, the coordinates of the grid points are determined using the baseline coordinate \((X_{\text{base}}, Y_{\text{base}})\) as follows:

\[
\{X_p = X_{\text{base}} + i \times \alpha, Y_p = Y_{\text{base}} + j \times \alpha\}; \quad \{i, j = \pm 0, \pm 1, \ldots\} \tag{3.1}
\]
Once all sensors are in place, every sensor decides if it is a grid node. First, each cell of length $\alpha$ is divided by the two midlines to create four smaller squares of equal size (see Figure 3.1). Every grid node will be in charge of its four nearby squares. Every sensor node then obtains its own location using any existing positioning mechanism and maintains its one-hop neighbors’ geographic locations in its neighbor table. Based on its own location, every sensor makes calculations to know which small square it falls into and which grid point the square belongs to. For example, in Figure 3.1, $G_1..G_9$ are nine grid points. Node $A$ belongs to square 4, which is attached to grid point $G_5$. If in the small squares 4, 7, 10, and 13 (which are also attached to $G_5$), there is no other node that is nearer to $G_5$ than $A$, $A$ will become the grid node of grid point $G_5$.

After the grid node election process, neighboring grid nodes send $ALS.GN.GN.DECLARE$ messages to each other to establish a neighbor grid node table. In the example, after node $A$ confirms its role as grid node, it sends $ALS.GN.GN.DECLARE$ messages to the grid nodes of $G_2, G_4, G_6, G_8$ to inform them that it has taken the role of grid node. Node $A$ also updates its own
table after receiving similar messages from its neighboring grid nodes. These messages are sent using geographic routing using the greedy forwarding strategy, i.e., sending the packets to the closest node to the destination.

3.1.2 Anchor Selection Process

In ALS, the sink agent distributes the sink’s location information using an anchor system, a selected group of grid nodes called anchors that act as location servers.

The anchor selection process is achieved by means of propagating anchor setup messages. At the beginning of the process, the sink agent, using location information about its neighbors, sends out four First Stage Anchor Setup Messages in four straight orthogonal directions (North, South, East, and West) and recruits all the grid nodes that lie along the routing path as anchors. The setup messages are relayed by intermediate sensors between two neighboring grid nodes and the recruited anchors store a copy of the sink agent’s location.

The anchor selection process needs to consider special cases, such as relaying the setup messages around void areas and the border of the network. Face routing mechanism is utilized to guarantee anchor setup messages can route around the void areas and converge with one another. For instance, once a setup message arrives at the border or at a void area, it is divided into two Second Stage Anchor Setup Messages. The border of this void area is then partitioned into two parts and each second stage setup message is routed around one of these partitions using the right-hand rule or left-hand rule [26,27]. (The proofs that this technique guarantees packet delivery are included in the same references.) The process can be better explained by looking at the example shown in Figure 3.2. The North first stage anchor setup message, which is represented by the arrow coming from the bottom, moves along the \( x=1 \) line. At point \( A \), the first-stage anchor setup message checks the local neighbor grid node table and cannot locate the next forwarding grid node in the North
direction. The first stage anchor setup message is then split into two second stage anchor setup messages, which use the right-hand and left-hand rule to navigate around the perimeter of the void area. Once a second stage anchor setup message comes back to the x=1 line, at point B, it is split again. At that point, one second stage anchor setup message continues along x=1, and another second stage anchor setup message keeps using the right-hand or left-hand rule to route around the void area until it reaches point C. Once two North anchor setup messages arrive at particular grid nodes which have already been visited, e.g. points C and D, they stop propagating and the normal setup process resumes.

In the anchor setup process, once a setup message arrives at a grid node, the grid node not only checks if it has received this setup message before, but also adds other already stored sink agent location information to the current passing anchor setup message. This strategy is known as propagated-fusion, which will significantly reduce the time delay of location propagation information and provide better performance. Convergence proof of the anchor setup process is included in Claim 1 and Claim 2 in Appendix A.
3.1.3 Query and Data Dissemination Processes

During the operation of the sensor network, some physical stimulus (target) will appear. At that point, one sensor node will sense the target and will become the source node that will transmit the sensed information to the sink. In order to do that, the source node will register itself with the nearest grid node, which is known as the source agent. The source agent will then send four query packets to find the location of the sink agent. Once the source agent receives the sink agent’s point location, it forwards it to the source, which finally sends the data packets to the sink agent using the GPSR protocol [28].

The query process is quite similar to the anchor system setup process. First stage query packets are sent to four orthogonal directions using the same strategy utilized in the anchor setup process. Also, these messages are split into second stage messages when they reach the border of the network or a void area and follow the same left-hand and right-hand rule strategy shown in Figure 3.2. The query messages traverse all available anchor systems, and
the process stops when these messages find each other. Upon reaching the anchor system, the anchors provide the source agent with the sink agent’s grid point coordinate. Anchors also send these replies utilizing geographic routing with the greedy forwarding strategy. This process increases the resilience of the scheme because if one query message fails, the other three queries will serve as backups to retrieve the sink’s location information. This is also beneficial because it shortens the query time, as the source will utilize the first response. The global grid construction process, the anchor selection process, and the query and data dissemination processes, the anchors, and the query and data forwarding processes are all shown in Figure 3.3. Convergence proof of the query process is included in Appendix A.

3.1.4 Sink and Target Mobility and Agent Chain Maintenance

As stated before, a sink agent chain is formed dynamically to manage moving sinks and avoid creating a new anchor system every time the sink moves beyond its current cell. After a sink selects its first agent (the primary agent), it keeps updating its instant location until it finds another grid node that is closer to itself. At that point, the sink selects the new grid node as its new sink agent, and this new sink agent in turn notifies the primary agent about its own location. Thus, a sink agent chain is built to keep the anchor system intact while tracking the sink node locally. The protocol is designed to allow the chain to have up to $l$ number of links; after that, ALS sets up a new anchor system and clears up the old one.

ALS does not keep track of moving targets, i.e., no chain is built and maintained as in the case of moving sinks. In this case, sensor nodes detect moving targets and become sources. Then, source nodes find source agents that query the anchor system in search of the sink agent’s location. If the target moves, a new sensor will become the source node. If the new source node is still within the same source agent’s area of coverage, the source agent will not trigger a new anchor query. However, if the target moves beyond the current
cell or out of the range of the current source agent, the new source node will find a new
source agent, and the source agent will query the anchor system again.

It is important to consider how the source agent knows the new position of the new sink
agent when the sink moves beyond its current cell. If the source agent does not have that
information, it will continue sending packets to the old sink agent and packets will be lost.
In order to avoid this situation, the protocol includes a process whereby the source agent
queries the anchor system periodically. The frequency of the query is set considering the
moving speed of the sink node. This information can be dynamically adjusted or statically
set. If the moving speed of the sink node is known in advance (which in most cases it is),
then the query frequency can be set statically. Otherwise, the sink agent can make moving
speed estimations and convey that information back to the source agent periodically. In
this work, the static approach is implemented; the dynamic approach will be part of future
research.

3.2 Theoretical Analysis of ALS

In this section, the scenario and notation utilized to analyze the communication and
state overhead of ALS is described in mathematical terms. The results of this analytical
model will also serve to validate the simulation models and results later. Simulations are
utilized to evaluate the ALS protocol more thoroughly, in particular those aspects that in­
volve time, such as the average location time and anchor system setup time, among others.

3.2.1 Scenario and Notation

The square area under consideration is defined as $A$, and therefore the area has side
length of $\sqrt{A}$. Each square cell within $A$ has the same length side $\alpha$, and the area of each
cell is then $\alpha^2$. The total number of sensors is assumed to be $N$, uniformly distributed so
that the sensor density is $\frac{N}{A}$. It is assumed that there are $K$ sinks and $S$ sources. All sensors and sinks have the same radio range $r$. It is also assumed that every sink receives a total of $D$ data packets.

### 3.2.2 Communication Overhead

The total communication overhead of the ALS protocol consists of the following five processes:

1. The global grid establishment process that takes place during the network set up phase and the establishment of the sink’s anchor system;
2. The local flooding in the sink’s cell that selects the sink agent $G_{Sink}$;
3. The local flooding in the source’s cell that selects the source’s agent $G_{Source}$;
4. The query that the source performs to the anchor system via the source’s agent in order to obtain the location of the sink agent, and the reply messages;
5. The transmission of the packet from the source to the sink;

In this analysis, only the communication overhead of processes 1, 4, and 5 will be considered, as there is no major difference in processes 2 and 3 between ALS and TTD.

#### 3.2.2.1 Process #1: Establishment of the Global Grid and Anchor System

Initially, a global grid structure needs to be built. Grid nodes are then selected, and the sink sets up the anchor system, which is the subset of grid nodes that will serve as location servers on its behalf. The total overhead of these processes is given by:

\[
O_{grid-anchor-system} = K \times (2R(m)+1) \times \left( \left\lceil \frac{c_1 \times x \times \alpha}{r} \right\rceil + 6 \times \left\lceil \frac{a}{r} \right\rceil \times R \times \left( \frac{xA}{\alpha} \right) \right) \\
+ \quad 4 \times \left\lceil \frac{a}{r} \right\rceil \times \left( R \left( \frac{xA}{\alpha} \right) \right) \times \left( R \left( \frac{xA}{\alpha} \right) + 1 \right) \quad (3.2)
\]
where $R$ is the Round function. Equation 3.2 considers that there are $K$ sinks in the system and also the moving sink situation. If it is assumed that sinks move at the average speed of $\bar{v}$ and the length of the sink agent chain is $l$, during an observation time $t$, the sink agent chain will break at most $m$ times and a total of $m+1$ anchor systems will be established, where $m = \lceil \frac{vt}{l} \rceil$. Of course, when sinks are static, $m = 0$.

The first part of Equation 3.2 represents the overhead of building one anchor system times the number of sinks and the number of times the chain is broken due to mobility. The $\lceil \frac{c_{1,v} \times x}{r} \rceil$ factor is the overhead in number of hops incurred in transmitting the message from the sink to the sink agent, and $c_{1,v}$ ($0 \leq c_{1,v} \leq \frac{\sqrt{2}}{r}$) is determined by the relative position between $sink_v$ and $G_{sink_v}$. This is shown in Figure 3.4 and calculated in Equation 3.3. When $sink_v$ is very close to $G_{sink_v}$, $c_{1,v} \approx 0$. If $sink_v$ is very close to the center of the cell, $c_{1,v} \approx \frac{\sqrt{2}}{r}$, which is the maximum value that $c_{1,v}$ can take. Because it is assumed that the
sensor nodes are uniformly distributed, \(c_1.v\)’s expectation is:

\[
E[c_{1,v}] = \frac{\int_0^{\frac{1}{2}} \int_0^{\frac{1}{2}} \sqrt{x^2 + y^2} \, dx \, dy}{\int_0^{\frac{1}{2}} \int_0^{\frac{1}{2}} \, dx \, dy} = 0.3824 \tag{3.3}
\]

Once \(G_{sink_v}\) receives \(sink_v\)’s message, it is ready to establish the anchor system. This overhead is given by \(6 \times [\frac{\alpha}{r}] \times R(\frac{\sqrt{A}}{\alpha})\), the average number of hops that the setup messages will go through. It is worth noting that because of the left or right hand rule, the anchor system will be established using six lines of grid nodes, as shown in Figure 3.3 - three lines of grid nodes traversing the network vertically and three lines of grid nodes traversing the network horizontally. Every traversing path of those six lines is approximately partitioned into \(R(\frac{\sqrt{A}}{\alpha})\) segments. Setup messages visit every grid node in their path in order to recruit the anchors. In every segment, the distance between two grid nodes is \(\alpha\), and setup messages will have to go over \([\frac{\alpha}{r}]\) hops to traverse one segment.

The second part of Equation 3.2 is the overhead for setting up the global grid structure. During that process, \(ALS.GN.GN.DECLARE\) messages are sent among neighboring grid nodes. Every selected grid node sends four declare messages to its four potential neighboring grid nodes in four orthogonal directions. \(2 \times R(\frac{\sqrt{A}}{\alpha}) \times (R(\frac{\sqrt{A}}{\alpha}) + 1)\) approximate the number of segments between all neighboring grid node pairs, i.e., the total number of cell sides. \(\lfloor \frac{\alpha}{r} \rfloor\) is the number of hops between each two neighboring grid nodes. The final coefficient is 4 because transmissions are in both directions. Analyzing Equation 3.2, it can be seen that the complexity of Process #1 is \(O(\theta m(\frac{\sqrt{A} + \alpha}{\frac{\alpha}{r}}) + \frac{A}{\alpha})\).

### 3.2.2.2 Process #4: Querying the Anchor System

This process consists of the communication overhead incurred by the source agent when it queries the anchor system in search of the location of the sink agent and the response from
the anchor system. The overhead of this process consists of four messages from $G_{Source}$ to four anchors and the subsequent replies, as follows:

$$\frac{c_{1,v} \times \alpha}{r} + \left\lceil \frac{c_{1,v} \times \alpha}{r} \right\rceil + 6\left\lceil \frac{a}{r} \right\rceil R\left(\frac{\sqrt{A}}{\alpha}\right) + R\left(\frac{c_{2,x} \sqrt{A}}{\alpha}\right) + R\left(\frac{c_{2,y} \sqrt{A}}{\alpha}\right) \left\lceil \frac{a}{r} \right\rceil$$ (3.4)

The first term, $\left\lceil \frac{c_{1,v} \times \alpha}{r} \right\rceil$, is the overhead when the source sends a query to $G_{Source}$. This overhead is exactly the same as the one already calculated by Equation 3.2, when the sink transmits a message to $G_{Sink}$. This explains why the same variable, $c_{1,v}$, is utilized, which has the same expectation calculated in Equation 3.3. The second term is the overhead incurred by the source agent, $G_{Source}$, when it sends the sink location back to the source, which due to symmetry is $\left\lceil \frac{c_{1,v} \times \alpha}{r} \right\rceil$. The third term of the equation represents the overhead incurred by the query message when it goes from $G_{Source}$ to the anchor system. On average, the overhead of four queries from $G_{Source}$ to anchors is bounded by $6 \times \left\lceil \frac{a}{r} \right\rceil \times R\left(\frac{\sqrt{A}}{\alpha}\right)$, i.e., no matter where the sink is located, $G_{Source}$ transmits two messages in the horizontal axis and two messages in the vertical one. As in the anchor system setup process, when query messages encounter the border of network area or void area, they split into two second stage messages and use the right hand rule and left hand rule to route around, forming six lines of messages over the entire area. This value is at the same time multiplied by the number of segments and the number of hops per segment. The last term in Equation 3.4 is the overhead of the replies coming from the anchors back to $G_{Source}$. Each query message works independently and creates a reply message only if it hits an anchor with the unknown sinks’ location information. The distance reply messages need to travel through depends on the relative location between the source agent and the anchors. Therefore, variables $c_{2x}$ and $c_{2y}$ are introduced to calculate the average number of hops from $G_{Source}$ to the anchor system. $c_{2x}$ and $c_{2y}$ are determined by both the relative position between $G_{Sink}$ and $G_{Source}$
and the relative position between \( G_{\text{Source}} \) and the network border. The four vertices of the
network square are \((0, 0), (0, \sqrt{A}), (\sqrt{A}, \sqrt{A}), (\sqrt{A}, 0)\) and an arbitrary pair of \( G_{\text{Sink}} \) and \( G_{\text{Source}} \)
are located at \((x_1 \cdot \sqrt{A}, y_1 \cdot \sqrt{A}), (x_2 \cdot \sqrt{A}, y_2 \cdot \sqrt{A})\), respectively. Without loss of generality, it
holds that \(0 \leq x_1, x_2 \leq 1\) and \(0 \leq y_1, y_2 \leq 1\). The expectation of \( c_{2x} \) and \( c_{2y} \) is obtained by the
relative position in the \(x\) axis and \(y\) axis, respectively, as follows:

\[
E[c_{2x}] = \int_0^1 \left( \int_0^{x_2} x_2 \, dx_1 + \int_{x_2}^1 (1 - x_2) \, dx_1 \right) \, dx_2 = \frac{2}{3} \quad (3.5)
\]

\[
E[c_{2y}] = \int_0^1 \left( \int_0^{y_2} y_2 \, dy_1 + \int_{y_2}^1 (1 - y_2) \, dy_1 \right) \, dy_2 = \frac{2}{3} \quad (3.6)
\]

In order to calculate the message overhead in a network with \( K \) sinks and \( S \) sources, the
following steps are taken. First, the message overhead incurred by the query from a \( \text{Source} \)
to its \( G_{\text{Source}} \) and the subsequent reply are calculated using Equation 3.7:

\[
O_{\text{Source} \leftrightarrow G_{\text{Source}}} = 2 \times \left[ \frac{c_{1,R} \times \alpha}{r} \right] \quad (3.7)
\]

Then, the overhead of the query messages from \( G_{\text{Source}} \) to all the anchors and the their
subsequent replies is calculated, which is given by Equation 3.8:

\[
O_{G_{\text{Source}} \leftrightarrow \text{anchors}} = 6 \times \left[ \frac{\alpha}{r} \right] \times R\left( \frac{\sqrt{A}}{\alpha} \right) + \sum_{v=1}^{K} R\left( \frac{c_{2x} \sqrt{A}}{\alpha} \right) \times \left[ \frac{\alpha}{r} \right] + \sum_{v=1}^{K} R\left( \frac{c_{2y} \sqrt{A}}{\alpha} \right) \times \left[ \frac{\alpha}{r} \right] \quad (3.8)
\]

The first part of Equation 3.8 is the overhead of the query messages from \( G_{\text{Source}} \) to \( K \)
anchor systems, which traverses the network like the anchor system setup process. This
guarantees that the source will obtain the location of all sinks. The second and third parts
are the overhead of the replies from the anchors to \( G_{\text{Source}} \), two messages in the horizontal
direction and two messages in the vertical direction.
Finally, the total query overhead with $K$ sinks and $S$ sources is:

$$O_{Source(u)\leftrightarrow anchors} = \sum_{u=1}^{S} \left( O_{Source(u)\leftrightarrow G_{Source(u)}} + \sum_{u=1}^{S} \left( O_{G_{Source(u)\leftrightarrow anchors}} \right) \right) \quad (3.9)$$

From Equations 3.5, 3.6, 3.7, 3.8 and 3.9, the expected total overhead is given by:

$$O_{Source(u)\leftrightarrow anchors} = S \times \left( 2 \times \left\lceil \frac{D}{2r} \right\rceil + 6 \times \left\lceil \frac{\alpha}{r} \right\rceil \times R \left( \frac{\sqrt{A}}{\alpha} \right) + 2 \times K \times R \left( \frac{4\sqrt{A}}{3\alpha} \times \left\lceil \frac{\alpha}{r} \right\rceil \right) \right) \quad (3.10)$$

From Equation 3.10, the complexity of Process #4 is $O(S(\frac{\alpha}{r} + K\frac{\sqrt{A}}{r}))$.

### 3.2.2.3 Process #5: Data Packet Transmission

In ALS data packets are transmitted from the source to the sink agent using the GPSR protocol. The associated overhead to transmit one packet between $Source_u$ and $G_{sink_v}$ is:

$$\left\lceil \frac{c_{3,(u,v)} \times \sqrt{A}}{r} \right\rceil \quad (0 \leq c_{3,(u,v)} \leq \sqrt{2}) \quad (3.11)$$

where $c_{3,(u,v)}$ is determined by the straight distance between $Source_u$ and $G_{sink_v}$. However, the calculation of the expected value of $c_{3,(u,v)}$ is still an open problem in Mathematics. Therefore, for the theoretical estimation, a two dimensional space ($n = 2$) is considered and its expectation is calculated, which is equal to 0.5214054331. Considering that every sink receives $D$ data packets, the total data forwarding overhead is:

$$O_{data} = K \times D \times \left\lceil \frac{c_{3,(u,v)} \times \sqrt{A}}{r} \right\rceil \quad (u=1,2,\ldots,S-1,S; \quad v=1,2,\ldots,K-1,K) \quad (3.12)$$

From Equation 3.12, it can be seen that Process #5 has a complexity of $O(KD\frac{\sqrt{A}}{r})$. 

3.2.3 State Overhead

The total state overhead of the ALS protocol is given by the storage space needed by grid nodes, anchors, source agents, and sink agents.

Every grid node needs to maintain the information of its four neighboring grid nodes. In total, there are at most $(R \left( \frac{\sqrt{A}}{\alpha} \right) + 1)^2$ grid nodes in the whole network area, and each spends at most 4 space units to save the neighbors’ information. Thus, the storage complexity for a grid node is $O(1)$.

For $K$ sinks and $S$ sources, the total storage overhead for the grid system is:

\[ 4 \times R \left( \frac{\sqrt{A}}{\alpha} \right)^2 + 1 \]  

which is scalable to the number of sinks, sources, and sensors.

For every sink, there are approximately $6R \left( \frac{\sqrt{A}}{\alpha} \right)$ anchors in the network. Each one will use one space unit to store the location of the sink agent. Therefore, its storage complexity is $O(1)$. For $K$ sinks and $S$ sources, the total storage is:

\[ 6 \times R \left( \frac{\sqrt{A}}{\alpha} \right) \times K \]  

Every $G_{Sink}$ and $G_{Source}$ needs to store its sink’s and source’s location, respectively. Thus, the storage complexity is $O(1)$. For $K$ sinks and $S$ sources, the total storage is:

\[ S + K \]  

The total number of states maintained in anchors, grid nodes, and number of $G_{Sink}$ and $G_{Source}$ is:

\[ 4 \times R \left( \frac{\sqrt{A}}{\alpha} \right)^2 + 6 \times R \left( \frac{\sqrt{A}}{\alpha} \right) \times K + S + K \]  

\[ 3.16 \]
The complexity of the state overhead can be calculated from the equation above as $O\left(\frac{A}{n^4} + K\frac{\sqrt{A}}{n}\right)$.

### 3.3 Performance Evaluation

As stated before, the performance of the ALS protocol from the location service point of view is more interesting, i.e., the establishment of the global grid structure, the anchor setup process, and the location query process (processes 1 and 4). Experiments are performed to determine the grid and anchor system setup time and overhead, the sink location time and overhead, and the state overhead of the protocol, while varying the cell size, the number of sinks and sources, the network size, the network density, and the mobility of the sinks.

The ALS protocol is implemented in the Network Simulator 2 (ns-2) [55] to validate the mathematical model and provide additional results, such as those that involve time. The mathematical results included in [4, 6] and the ns-2 simulation models found in [5] are utilized to compare ALS with TTDD.

Two hundred and fifteen stationary sensor nodes and four stationary sink nodes are uniformly deployed in a two dimensional $1000m \times 1000m$ network area. In this scenario, the average node degree was 33, which corresponds to a very dense network. Four of those 215 sensors were chosen as source nodes. Each experiment is run nine times to avoid cases where the source and sink nodes were very close or very far apart from each other only. As a result, every point drawn in the graphs is based on observations of nine random deployments. In all simulations, control packets were 36 bytes long while data packets were 64 bytes long. For those simulations where the data transfer delay is under investigation, each source node generated one data packet per second. Except in those simulations where the effect of the cell size is under observation, the parameter of cell size $\alpha$ is set to 200 meters since it was found to be the best performing value. Sink mobility
was implemented using the standard random way-point model with zero pause time and a fixed speed from source to destination point taken from a uniform distribution between 0 and maximal speed.

In the simulations, the wireless extensions from the MONARCH project [30] are utilized to provide for node mobility, realistic physical layer modeling including a radio propagation model supporting propagation delay, capture effects and carrier sense, and radio network interfaces. In order to minimize collisions among anchor setup messages originated from different sinks, each sink node is assigned a random back-off time interval before the anchor setup process begins. The extensions implement the IEEE 802.11 Medium Access Control (MAC) [56] protocol using the Distributed Coordination Function (DCF) that models the contention of nodes for the wireless media. The wireless interface worked like the 914 MHz Lucent WaveLAN Direct-Sequence Spread-Spectrum (DSSS) radio interface [57]. The signal propagation model combined both a free-space propagation model and a two-ray ground reflection model. The free-space model was used only when the transmitter was within the cross-over distance of the receiver, otherwise the two-ray ground model was used. The radio transmission range was set to 250m. Since these models are very well known, they are not explained here any further (see [58]). Unless otherwise stated, these parameters are utilized in all experiments.

3.3.1 Optimal Cell Size

The impact of the cell size on the performance of the location service is investigated first and the results presented in Figures 3.5 to 3.10. The default scenario with four sinks and four sources is considered, and the cell size varies from 100m to 1000m in 100m increments. The anchor set up time is analyzed first. As stated before, each point in the plot is the average value from nine experiments with 95% confidence intervals. Figure 3.5 illustrate that the setup times are fairly similar regardless of the value of $\alpha$. Also, it can be noticed that
the largest setup time occurs at $\alpha = 100m$, while the smallest time occurs at $\alpha = 200m$. This has to do with the nodes’ transmission range of 250m. With $\alpha = 200m$, the anchor setup packets can reach the grid nodes in one hop; however in the case of $\alpha = 100m$, there are about three grid nodes within the 250m transmission range, and setup packets need to go through each of them. An additional observation is the short amount of time needed to set up the anchor system, which in the worst case takes 45 ms.

Figure 3.6 depicts the grid and anchor setup overhead. It can be observed that the average protocol overhead reduces slightly with the cell size. This slight reduction is due to the reduced number of grid nodes when the cell size increases. As a result, fewer $ALS\_GN\_GN\_DECLARE$ messages are sent. These results were expected, as the overhead is almost independent of $\alpha$ (see Equation 3.2). The results also show the small overhead introduced by ALS, which is well in line with the constraints of WSNs. Given the results presented in Figures 3.5 and 3.6, it is concluded that cell size has little impact on the grid and anchor setup processes.

In Figures 3.7 and 3.8, the average location time and overhead of ALS are compared with those of TTDD. The location time of the protocols is defined as the time elapsed from the moment the source agent sends out the four location queries until the moment it
receives the location of all the sink agents available in the network. The figures show that the performance of ALS is almost independent of the cell size, as the values remain fairly constant. The slight increase in the location time is due to the relay messages taking longer to reach the source node from the source agent in bigger cells. As it can be observed from the figures, ALS outperforms TTDD by reducing 80% the location overhead and 50% the location time for almost all values of $\alpha$. In TTDD, sink nodes flood messages within one cell to find the nearest data dissemination node; therefore, the bigger the cell size, the more overhead.
The impact of the cell size on the average data delay is studied as well. The average data delay is the time period from the moment when a data packet appears at the source until the packet is received by the sink. It consists of two major parts: *sink location time* and *data propagation time*. Once a source generates the first data packet, it starts the sink’s location process to find the location of all the sinks in the network. After that, the source sends all data packets using the GPSR protocol. The time for every data packet to travel from source to sink is the *data propagation time*. Therefore, the average data delay is calculated as follows:
\[
Average_{Data\_Delay} = \frac{SinkLocation\_Time + \sum_{i=1}^{D} Data\_Propagation\_Time_i}{D}
\]

where \( D \) is the total number of data packets. Figure 3.9 shows the results of these experiments using \( D = 100 \) packets.

![Graph showing state overhead vs. cell size.](image)

Figure 3.10 State overhead vs. cell size \( \alpha \).

Figure 3.10 shows the state overhead of ALS and TTDD. As explained before, the overhead decreases with the cell size because the number of grid nodes also decreases. This can also be explained by looking at Equations 3.13 and 3.14.

### 3.3.2 Impact of the Number of Sinks and Sources

In order to evaluate the impact of multiple sinks, the number of sinks is varied from 1 to 10 in steps of 1, and the default scenario is used for the rest setting, where four sources are deployed and the size of the cell size \( \alpha \) is set to 200m. Figures 3.11 and 3.12 show the average anchor setup time and average grid and anchor setup overhead, respectively. As it can be seen, the anchor setup time remains fairly constant regardless of the number of sinks in the network. This is because in ALS, each sink sets up its own anchor system.
independently and without interfering with other sinks. As expected, the grid and anchor setup overhead increases with the number of sinks. The higher the number of sinks, the higher the overhead, as ALS establishes one anchor system per sink. This can also be explained mathematically by Equation 3.2.

![Figure 3.11 Anchor setup time vs. the number of sinks.](image)

![Figure 3.12 Grid and anchor setup overhead vs. the number of sinks.](image)

Simulations regarding the location time and overhead are also conducted. From Figure 3.13, it can be seen that the location time slightly increases with the number of sinks. This is because when multiple sinks are deployed in the network, location query messages
always have to travel until they find the farthest anchor system. While a longer average location time is expected, the proposed propagated-fusion strategy compensates for this factor. This strategy spreads sink’s location information in other anchors without incurring extra setup overhead. TTDD experiences a 50% higher delay than ALS because it floods the queries instead of the point-to-point query scheme used in ALS. Similar trends can be observed in the case of the average location overhead in Figure 3.14. More sinks imply more anchor systems and more query response messages.
Figure 3.15 State overhead vs. the number of sinks.

Figure 3.15 shows the case of the state overhead of the protocols. From this figure, it can be seen that ALS’s overhead increases linearly with the number of sinks while TTDD’s remains fairly constant. This is because ALS is sink-oriented, while TTDD is source-oriented. Equation 3.14 explains this behavior.

### 3.3.3 Impact of the Number of Sources

Figure 3.16 Anchor setup time vs. the number of sources.
The impact of having multiple sources is also evaluated. As before, the default scenario is used, where cellsize was fixed as $\alpha = 200m$ and the number of sinks was set as four, and were increased from 1 to 10 in steps of 1. The average anchor setup time and average grid and anchor setup overhead are presented in Figure 3.16 and Figure 3.17, respectively. Both of them remain fairly constant because anchor setup processes are initiated by the sinks and have very little impact from the number of sources.

Figure 3.17 Grid and anchor setup overhead vs. the number of sources.

Figure 3.18 Location time vs. the number of sources.
The average location time and overhead are shown in Figures 3.18 and 3.19. The number of sources have little if any effect on the performance metrics because in ALS, query processes are independently conducted by individual sources and do not conflict with each other.

Figure 3.20 shows the state overhead of the protocols. In the case of ALS, the overhead is constant because when sinks are static, the global grid system is constructed once and the anchor system is built $K$ times. However, these two major parts of the state overhead in Equation 3.16 do not depend on the number of sources $S$. On the other hand, TTDD floods the network $S$ times to establish one global grid per source.

### 3.3.4 Impact of Sensor Density

Most results presented so far consider the base scenario where 215 sensor nodes are uniformly deployed over a square area of $1000 \times 1000m^2$. As mentioned before, the deployed network has an average node degree of 33. Although this seems very high, this case scenario has been utilized in many other studies. So for comparison reasons, it is also utilized in this study. This scenario is also being used because the node degree is not ex-
Figure 3.20 State overhead vs. the number of sources.

Figure 3.21 Average anchor setup time. vs. the network density.

...So... expected to play a major role in the performance of the ALS protocol. Unless the network is very sparse, to the point where it may get partitioned, the ALS protocol should be able to establish the anchor system going around void areas without any problem.

In order to test the hypothesis, several simulations were conducted using the base scenario utilized throughout the chapter, but varying the number of nodes, so that average node degrees of 4, 6, 8, 10, 12, 15, 20, 25, 30 and 33 are obtained. From Figure 3.21 to Figure 3.23, the average anchor setup time, the average location time, and the anchor setup overhead simulation results are presented as a function of node degree. The simulation
results confirm the stated hypothesis, i.e., the ALS protocol is unaffected by the density of the network.

### 3.3.5 Impact of Network Area

In this subsection, the scalability of the protocol is explored. The same sensor density is used as in the original scenario, and 54, 215, 464 and 815 sensors are deployed in $500 \times 500m^2$, $1000 \times 1000 m^2$, $1500 \times 1500 m^2$ and $2000 \times 2000 m^2$ network areas with an average node
degree of 26, 33, 35 and 36, respectively. Four sources and sinks are randomly deployed and \( \alpha = 200m \) is set in all experiments.

Figures 3.24 and 3.25 illustrate the linear increase of the average anchor setup time and average grid and anchor setup overhead with the size of the network area. This is expected as these metrics are proportional to the network size and number of nodes. In bigger network areas, anchor setup packets will travel longer to recruit more grid nodes as
anchors. Regarding to the location time, Figure 3.26 illustrates that when a network size increases nine times, the location time only doubles (from 0.02s to 0.04s).

ALS scales well to bigger network areas because of the propagation-fusion strategy. Figure 3.27 shows the location overhead of the protocols. It can be seen in this figure that the location overhead slowly increases four times (from 25 to 100) while the network area increases nine times. Not only was TTDD’s slope bigger, but it also incurred at least 150% more overhead than ALS. On the other hand, Figure 3.28 that the state overhead of both
protocols are fairly similar. Although the protocols’ overhead increase linearly with the network area, both schemes present similar values and slopes.

![State overhead vs. network area]

Figure 3.28 State overhead vs. the network area.

![Anchor setup time vs. sink mobility]

Figure 3.29 Anchor setup time vs. sink mobility.

### 3.3.6 Impact of Sink Mobility

In order to study the impact of moving sinks on the location service, four sinks are set to move at the speed of 5m/s, 10m/s, 15m/s and 20m/s. The maximum sink agent chain
length is set to 2 \((l = 2)\), meaning that when a sink moves, the original one will only be able to chain up to two new agents. At that point, the sink will adopt a new agent, and this new agent will update the original one with its position, so the messages can be relayed along the agent chain. When the moving sink finds its third agent, it will take the new agent as the primary agent, build a new anchor system, and send a cancel message through the previous anchor system to erase it. The additional overhead and time incurred in the cancellation process is considered in the average grid and anchor setup overhead and average anchor
setup time. Figures 3.29 and 3.30 illustrate the anchor system setup time and overhead when the sinks are moving. As expected, these metrics increase with the sinks’ moving speed. The faster the sink moves, the higher the chance to break the sink’s agent chain, which in turn causes more sinks to cancel the previous anchor system and set up a new one. This is shown in Figure 3.31, where it can be seen that when the sinks’ moving speed is 5m/s, each sink only sets up the anchor system once, however when sinks move at 20m/s, each sink breaks the agent chain once.

Figure 3.32 Location time vs. sink mobility.

Figure 3.33 Location overhead vs. sink mobility.
The impact of the sink’s mobility on location time and location overhead is also analyzed. The results are presented in Figure 3.32 and Figure 3.33, respectively. From these figures, it can be found that mobility had a little effect on the metric. The average location time and overhead are fairly constant when the sinks move because the anchor systems do not change unless the agent chain breaks. Sinks move locally and the chain of agents handle the little extra overhead. Similarly, the state overhead in Figure 3.34 presents a fairly constant behavior with $v$. This is because every time the chain is broken due to sink mobility, a new anchor system is setup but the old one is then eliminated. So, at the end, there is just one anchor system per sink. Equation 3.14 also explains this.

### 3.3.7 Total Communication Overhead

In this subsection, the total communication overhead of ALS and TTDD will be reported, and will include, the data transmission overhead. As explained earlier, GPSR is utilized to route packets over WSNs (ALS+GPSR), while TTDD has its own routing mechanism. Figure 3.35 shows the total communication overhead versus the number of sinks. As expected, both protocols’ overhead increase linearly with the number of sinks, but ALS’s
overhead is very small and stable compared with TTDD’s, which utilizes flooding of query messages within the local cell.

In Figure 3.36, the number of sources is varied. The trends are similar to the case with multiple sinks. In TTDD, every source builds its own grid system and the sink query is forwarded through each individual grid system, which is why the communication overhead increases dramatically with the number of sources. On the other hand, ALS builds an anchor system per sink that is shared by all sources.
Finally, the scalability of the protocols with respect to the network area is shown in Figure 3.37. Once more, the scalability of ALS compared to TTDD is shown to be superior. In TTDD, a sink updates its location by flooding a local query to reach an immediate dissemination node, from which the query is further forwarded to the source along the grid. In other words, given a fixed sensor density, the bigger the network area, the bigger the number of sensor nodes involved in the forwarding process.

### 3.4 Conclusions

Proposal of the ALS protocol, a grid-based protocol that provides sink location information in a scalable and efficient manner in large scale WSNs. ALS decouples the location service to the routing protocol, and as a result, supports the use of any existing location-based routing protocol. Compared to any known method, ALS not only reduces the communication costs but also the location information response time.
CHAPTER 4
RESIDUAL ENERGY AWARENESS DYNAMIC TOPOLOGY CONTROL

4.1 Network Model

In the traditional homogeneous network, it is better to transmit packets through many short links rather than directly. However, this approach only considers the distance between the nodes, which is only effective if the network is homogeneous. In heterogeneous networks, however, a new approach to select the edges is needed considering the different characteristics of the devices and the desire to assign more powerful devices a more prominent role in the network. In this chapter, the network model under investigation is formulated first, then the centralized and distributed versions of the READ topology control algorithm are presented along with their performance evaluation.

The network model considered in this chapter is a heterogeneous wireless network represented by a graph $G = (V(G_{max}), E(G_{max}), W(E), P(V), RE(V), \beta(V))$, where the nodes of wireless network are represented by a set, $V = \{v_1, v_2, ..., v_n\}$, and are randomly deployed in a 2-Dimension network area. $E(G_{max})$ is the set of edges. $W(E)$ is the weighted cost value associated with each edge in the graph $G$. The weighted cost function will be explained in great details in subsection 4.1.3. Depending on the specific application scenario, wireless networks can consist of nodes of different types of devices, such as sensors, PDAs, robots, and even more powerful military devices. Because of certain characteristics of heterogeneous networks, the maximum transmission range, residual energy, antenna gain, and receiver sensitivity may vary from one node to another. Therefore, node $v$ is associated with
attributes \( (P_{\text{max},v}, RE_v, \beta_v) \). The notation \( P_{\text{max},v} \) is \( v \)'s maximum transmission power in Watts, and node \( v \) is able to dynamically choose any power \( P_v \in [0, P_{\text{max},v}] \) to transfer its packets. \( \beta_v \) is the sensitivity of \( v \)'s antenna in decibels. \( RE_{E_v} \) is current residual energy in node \( v \) in Joules, and decreases with \( v \)'s activity from initial value \( I_{\text{INITIAL ENERGY}} \) to zero.

Considering two random nodes, \( u \) and \( v \), with Euclidean distance \( d(u, v) \) between them, it holds that \( u \)'s transmission is successfully received at \( v \) if Equation 4.1 holds as follow:

\[
P_{\text{transmit}}(d(u, v), P_u) \geq \beta_v
\]  

(4.1)

where \( P_{\text{transmit}}(d(u, v), P_u) \) is the received signal strength at \( v \) given that \( d(u, v) \) is the Euclidean distance between the nodes and \( P_u \) is the transmission power at \( u \). In addition, two random nodes, \( u \) and \( v \), from set \( V \) are connected by a link \( e(u, v) \in E(G) \) in \( G \) only if \( u \) and \( v \) are connected in both directions. In other words, only bi-directional graph is considered in this dissertation, which is supported by the widely used IEEE 802.11 wireless network Medium Access Control (MAC) protocol. MAC sends link-level acknowledgments for all uni-cast packets, so that all links built on top of 802.11 network must be bi-directional.

### 4.1.1 Maxpower Graph

The maxpower graph represents the network topology generated by having each node communicate with its maximum transmission power. Maxpower graph is denoted as \( G_{\text{max}} = (V(G), E(G), P_{\text{max}}, RE, \beta) \), where \( E(G_{\text{max}}) \) is the edge set when all the nodes work using their maximum transmission power \( P_{\text{max}} \). According to Equation 4.1, \( E(G_{\text{max}}) \) can also
be defined by Equation 4.2 as follows:

\[ E(G_{\text{max}}) = \{(u, v) : P_{r,v}(d(u, v), P_{\text{max} \cdot u}) \geq \beta_v, P_{r,u}(d(v, u), P_{\text{max} \cdot v}) \geq \beta_u\} \quad (4.2) \]

Note that the graph \( G_{\text{max}} \) represents the set of all possible bi-directional communication links between the network nodes. It will be proven later that if the maxpower graph \( G_{\text{max}} \) is strongly connected, the network topology generated by the proposed READ topology control algorithm also preserves the strong connectivity.

### 4.1.2 Energy Model

The main goal of the proposed energy aware topology control algorithm is to prolong the network lifetime by ameliorating energy consumption among different kinds of wireless devices. Depending on the type of device, the amount of energy consumed by the radio transceiver can approach nearly 35% of the total energy dissipated by the node [3]. Thus, optimizing the energy used for communication is an important issue, and the energy model utilized concentrates on the energy spent on communications, since it is the most energy consuming factor [2]. The amount of energy consumed by a wireless interface can be described by the following simple model:

\[ E_{\text{consume}} = E_{\text{elec}} + E_{\text{amp}} + E_{\text{sense}} \quad (4.3) \]

where \( E_{\text{elec}} \) is the energy used to run the transceiver circuitry in signal processing, for instance channel coding, interleaving and modulation; \( E_{\text{amp}} \) represents the energy used by the amplifier to transfer the signal and satisfy the receiver’s sensitivity requirement; and \( E_{\text{sense}} \) denotes the energy for sensing the wireless channel before the transmission takes place. Note that \( E_{\text{amp}} \) is the product of the transmission power and the transmission

58
time in Joules. Usually, $E_{elec}$ and $E_{sense}$ are neglected for the simplicity of discussion. A comprehensive study [59] on the energy consumption of the wireless interface not only affirms that the higher the $E_{amp}$, the higher the energy consumption $E_{consume}$ but also that the amplifier power takes a major stake in the overall power budget. In fact, the increase in $E_{consume}$ is over-proportional to the increase in $E_{amp}$. Therefore, $E_{amp}$ is hereafter used to denote the amount of energy consumed by the wireless interface during communication. Also, the following equations hold:

$$E_{amp_{tx}} = P_{tx} \times T_{tx}$$ \hspace{1cm} (4.4a)

$$E_{amp_{rx}} = P_{rx} \times T_{rx}$$ \hspace{1cm} (4.4b)

How to determine the power consumed during the transmitting and receiving processes still remains unanswered.

As discussed earlier, a radio channel between transmitter $u$ and receiver $v$ is established if the strength of the radio signal $P_{r_{v},u}$ captured by node $v$ is above the sensitivity threshold $\beta_v$ and at the same time $P_{r_{u},v}$ captured by node $u$ is above $\beta_u$.

The relationship between transmission power and transmission range is discussed in [58]. Given that a node transmits a message to another node that is $d$ distance away, the following model from [58] is used to compute the power consumption $P$ needed to send the message:

$$P = kd^c$$ \hspace{1cm} (4.5)

where $k$ and $c$ are up to the environment and are constants for the specific wireless system (usually $2 \leq c \leq 4$). Clearly, the amount of power for transmission monotonically increases with the distance between two communication nodes.
There are several common path loss models evaluated in the literature to describe the propagation in the wireless medium. In the Two-Ray Ground propagation model, the relationship between the power $P_t$ used by the amplifier to transfer packets and the signal strength captured by receiver is described as:

$$P_r = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L}$$  \hspace{1cm} (4.6)

In the Free Space propagation model, this relationship is described as:

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2 L}$$  \hspace{1cm} (4.7)

In Equations 4.6 and 4.7 $G_t$ is the transmitter antenna gain, $G_r$ is the receiver antenna gain, $L$ is the system loss factor and is independent from propagation, and $\lambda$ is the wavelength in meters. $h_t$ and $h_r$ are the antenna heights of the transmitter and receiver, respectively, and $d$ is the Euclidean distance between the two transceivers.

In this study, the Free Space propagation model is used, and Equation 4.7 is further simplified to:

$$P_r(d) = C_f \cdot \frac{P_t}{d^2}$$  \hspace{1cm} (4.8)

where the $C_f$ depends on the characteristics of the transceivers and is not related to propagation. Combining Equations 4.1 through 4.8, it can be concluded that in order for the receiver to receive the signal correctly, the following relationship must hold:

$$P_t \geq \frac{\beta \cdot d^2}{C_f}$$  \hspace{1cm} (4.9)

Given the above equation, the minimum transmission power required for each communication can be computed. The notation $p_{u,v}$ is used to represent the minimum power required
by transmitter $u$ for transmitting a data packet to its neighbor node $v$, and the notation $p'_v$ is utilized to describe the power used by node $v$ to receive this message. Note that:

$$ p_{u,v} = \frac{\beta \cdot d(u,v)^2}{C_f} \quad (4.10) $$

Normally in the literature [3] [59], the receiving power $p'_v$ at node $v$, or the power consumed by the electronics at receiver $v$ during the process of receiving a packet, is proportional to $v$’s maximum transmission power $P_{\text{max},v}$, and is independent of the sender’s transmitting power. This relationship can be characterized by the following equation:

$$ p'_v = C_T \cdot P_{\text{max},v} \quad (0 < C_T < 1) \quad (4.11) $$

where the coefficient $C_T$ is the ratio between the transmission and reception power consumed at the node, which depends on the type of wireless card. For example, the CISCO Aironet 350 wireless card has a transmission/reception ratio of 2.22:1.33, or a $C_T$ of 0.59.

### 4.1.3 Weighted Cost Function

In order to increase network longevity as much as possible, the proposed READ topology control algorithm considers both the energy for sending and receiving data and the current residual energy at each node. Therefore, instead of using the Euclidean distance between the two communicating nodes to define the link cost, as in a homogeneous network, a new *weighted cost* value is introduced for each pair of nodes. Since the maxpower graph is bi-directional, it is only necessary to discuss two asymmetric communication links for each pair of nodes in the heterogeneous network. Assume node $u$ and $v$ are within the neighbor set of each other, for one direction, $w_{u\rightarrow v}(e(u,v))$ represents the weighted cost for transmitting and receiving data from $u$ to $v$; for the reverse direction, $w_{v\rightarrow u}(e(u,v))$
describes the weighted cost for transmitting and receiving data from \( v \) to \( u \). \( w_{u \rightarrow v}(e(u, v)) \) is defined as follows:

\[
w_{u \rightarrow v} = \frac{p_{u,v} \cdot t_{tx}}{RE_u} + \frac{p'_{v} \cdot t_{rx}}{RE_v}
\] (4.12)

where \( RE_u \) and \( RE_v \) are the residual energy at each node; \( p_{u,v} \) is the minimum power for \( u \) to successfully send any packet to \( v \); \( p'_{v} \) is the power for \( v \) to receive data; and \( t_{tx} \) and \( t_{rx} \) are the times for transmitting and receiving a given packet. Similarly,

\[
w_{v \rightarrow u} = \frac{p_{v,u} \cdot t_{tx}}{RE_v} + \frac{p'_{u} \cdot t_{rx}}{RE_u}
\] (4.13)

Hence, the weighted cost function reflects the proportion of energy required at both ends to perform a successful communication. Since only bi-directional links are considered, communication costs in both directions are treated as a whole. Thus, the weighted cost is defined as:

\[
w(e(u, v)) = w_{u \rightarrow v} + w_{v \rightarrow u}
\] (4.14)

Therefore, given two edges \((u_i, v_i)\) and \((u_j, v_j)\), it holds:

\[
w(e(u_i, v_i)) < w(e(u_j, v_j))
\]

\[
\Leftrightarrow
\]

\[
w_{u_i \rightarrow v_i} + w_{v_i \rightarrow u_i} < w_{u_j \rightarrow v_j} + w_{v_j \rightarrow u_j}
\]

or

\[
(w_{u_i \rightarrow v_i} + w_{v_i \rightarrow u_i} = w_{u_j \rightarrow v_j} + w_{v_j \rightarrow u_j})
\]

and \( \max\left(\frac{p_{u_i,v_i} \cdot t_{tx}}{RE_{u_i}}, \frac{p'_{v_i} \cdot t_{rx}}{RE_{v_i}}, \frac{p_{u_j,v_j} \cdot t_{tx}}{RE_{u_j}}, \frac{p'_{v_j} \cdot t_{rx}}{RE_{v_j}}\right) \)

\[
< \max\left(\frac{p_{u_j,v_j} \cdot t_{tx}}{RE_{u_j}}, \frac{p'_{v_j} \cdot t_{rx}}{RE_{v_j}}, \frac{p_{u_i,v_i} \cdot t_{tx}}{RE_{u_i}}, \frac{p'_{v_i} \cdot t_{rx}}{RE_{v_i}}\right)
\]

or

\[
(w_{u_i \rightarrow v_i} + w_{v_i \rightarrow u_i} = w_{u_j \rightarrow v_j} + w_{v_j \rightarrow u_j})
\]

and \( \max\left(\frac{p_{u_i,v_i} \cdot t_{tx}}{RE_{u_i}}, \frac{p'_{v_i} \cdot t_{rx}}{RE_{v_i}}, \frac{p_{u_j,v_j} \cdot t_{tx}}{RE_{u_j}}, \frac{p'_{v_j} \cdot t_{rx}}{RE_{v_j}}\right) \)

\[
= \max\left(\frac{p_{u_i,v_i} \cdot t_{tx}}{RE_{u_i}}, \frac{p'_{v_i} \cdot t_{rx}}{RE_{v_i}}, \frac{p_{u_j,v_j} \cdot t_{tx}}{RE_{u_j}}, \frac{p'_{v_j} \cdot t_{rx}}{RE_{v_j}}\right)
\]

and \( \text{second}_{\max}\left(\frac{p_{u_i,v_i} \cdot t_{tx}}{RE_{u_i}}, \frac{p'_{v_i} \cdot t_{rx}}{RE_{v_i}}, \frac{p_{u_j,v_j} \cdot t_{tx}}{RE_{u_j}}, \frac{p'_{v_j} \cdot t_{rx}}{RE_{v_j}}\right) \)

\[
< \text{second}_{\max}\left(\frac{p_{u_i,v_i} \cdot t_{tx}}{RE_{u_i}}, \frac{p'_{v_i} \cdot t_{rx}}{RE_{v_i}}, \frac{p_{u_j,v_j} \cdot t_{tx}}{RE_{u_j}}, \frac{p'_{v_j} \cdot t_{rx}}{RE_{v_j}}\right)
\]
The above relationship guarantees a unique outcome in the edge selection step in the READ topology control algorithm, which is further discussed in the next two sections.

In summary, in this model, nodes from the set $V$ are randomly deployed in a 2-D network work area, and every node has its own maximum transmission power and sensitivity. All the possible edges are presented in maxpower graph and associated considering the minimum requested power as defined by the weighted cost. Edge weighted cost can be calculated from distance and sensitivities. Every node can adjust its transmission power between zero and its maximum transmission power, therefore different subsets of max-power graph will be produced. In Section 4.2, the centralized version of the topology control algorithm READ is presented. It selects certain edges from maxpower graph by adjusting the nodes’ transmission power to each other to meet the goal of extending the network longevity. In Section 4.3, the distributed version of the topology control algorithm (DREAD) is presented.

4.2 The Centralized Residual Energy Aware Dynamic Topology Control Algorithm

In this section, the Centralized Residual Energy-Aware Dynamic (READ) topology control algorithm is presented in detail. This work was first introduced in [60].

4.2.1 Centralized Residual Energy Awareness Dynamic Algorithm

The centralized READ algorithm has two phases: the Initialization phase and the Topology Construction phase. The Initialization phase consists of the following steps:

1. Construct the maxpower graph $G_{\text{max}}$ by using the maximum transmission power of each transceiver. Note that $G_{\text{max}}$ is bi-directional. Without loss of generality, it is also assumed that $G_{\text{max}}$ is strongly connected.
2. For each edge $e(u, v) \in E(G_{\text{max}})$, compute the weighted cost $w(e(u, v)) = w_{u \rightarrow v} + w_{v \rightarrow u}$.

3. After step 2, each edge $e(u, v) \in E(G_{\text{max}})$ is associated with a weighted cost $w(e(u, v))$. Sort the set $E(G_{\text{max}})$ in increasing order of $w(e(u, v))$ based on the weighted cost relationship described previously in subsection 4.1.3. To resolve the ambiguity, the sorted edge sequence will be renamed as $E_{\text{order}}$.

In the Topology Construction phase, the new network topology is constructed from the empty edge set $E(G_{\text{READ}})$. Consider every node in the original network graph as an isolated component set $G_i$, i.e. $G_i = \{u_i\}$. During the construction process, two component sets will be merged at a time by adding one edge from $E(G_{\text{max}})$ to $E(G_{\text{READ}})$ until all the nodes have been connected and there is only one component set left. The resulting topology is strongly connected. The detailed algorithm is as follows:

Initialization:

1. Construct $G_{\text{max}}$ along with $E(G_{\text{max}})$

2. Compute the $w(e_i)$ for every $e_i \in E(G_{\text{max}})$

3. Sort $E(G_{\text{max}})$ in increasing order to $E_{\text{order}} = (e_1, e_2, ..., e_m)$, where $m = |E(G_{\text{max}})|$;

4. Create $G_i = \{u_i\}$ as single element set for each $u_i \in V(G_{\text{max}})$;

5. $E(G_{\text{READ}}) = \phi$;

Topology Construction:

6. While (there are still more than one isolated set $G_i$) && (not all the edges have been scanned)

   7. Remove the first edge $e_i = (u, v)$ from the head of remaining $E_{\text{order}}$, and suppose $u \in G_\zeta$ and $v \in G_\tau$

   8. If $G_\zeta \neq G_\tau$

   9. Then $G_\zeta = G_\zeta \cup G_\tau$;
\begin{align*}
10: & \quad E(G_{READ}) = E(G_{READ}) \cup \{(u, v)\}; \\
11: & \quad \text{Else if } G_c = G_\tau \\
12: & \quad \text{Then skip the edge } e_i = (u, v); \\
13: & \quad \text{End if} \\
14: & \quad \text{End while}
\end{align*}

Thus \( G_{READ} = (E(G_{READ}), V(G_{max})) \) is the resulting topology, which is also strongly connected and bi-directional.

It is worth pointing out that the above algorithm can be applied \( k \)-times to preserve the inherited \( k \)-connectivity of the maxpower graph. Basically, READ can simply resort all remaining edges - edges which have never been used in previous \( k - 1 \) rounds, and create isolated sets \( G_i \) for every node and merge the sets by adding edges to \( E(G_{READ}) \). Even if the maxpower graph does not have \( k \)-connectivity, the algorithm is guaranteed to be terminated. Thus, the algorithm is designed to be more robust against node failures, which are common phenomena in wireless networks. When some of those components cannot be merged, \( k \)-connectivity can still be established among the rest of the nodes and the algorithm will be terminated when all the remaining edges have been scanned. The proof of the correctness of the algorithm is included in Claim 3 in Appendix A.

\subsection{Simulation Results and Evaluation of READ}

In this subsection, the performance evaluation of the READ algorithm is presented. The simulation setup parameters will be described first, and then the simulation results and the respective analysis considering simulations with and without data transmission will be described.
4.2.2.1 Simulation Setup

In the simulation experiments, a heterogeneous wireless network with four types of devices is considered: sensors, PDAs, Robots, and military devices. One hundred nodes in a 1000m × 1000m network area are randomly distributed, where 5% of them are military devices, 10% of them are robots, 20% PDAs, and 65% of them are sensors. Each node utilizes an uniform distribution to randomly draw its maximal transmission power in Watts, receiver sensitivity in dB, and initial energy in Joules from the pre-defined ranges \([P_L, P_H]\), \([β_L, β_H]\) and \([I_E_L, I_E_H]\) that characterize each type of device (See Table 4.1). The range of those parameters are determined based on [61], [62], [63], and [64]. As discussed previously, in the energy model only the amount of energy consumed by the sender amplifier and receiver during transmission is considered. From Equation 4.8, it can be found that the factor \(C_f\) is not related to propagation but determined by the characteristics of each pair of transceivers. Without loss of generality, only omnidirectional antennae are considered in this simulation and \(G_t\) and \(G_r\) were set to 1. The system loss factor is set to \(L = 1\), and the operational frequency equal to 2.472GHz. Also from Equation 4.11, it is known that \(C_f\) is subject to the wireless interface device itself, and without loss of generality, this parameter is set to a constant value 0.6 for all the nodes. \(t_{tx}\) and \(t_{rx}\) are both set to 0.01 second, meaning that packets are considered of equal and fixed size. The above simulation setting will be referred as the default setting in the remaining part of this chapter.

<table>
<thead>
<tr>
<th>Category</th>
<th>(β_L) (dBm)</th>
<th>(β_H) (dBm)</th>
<th>(P_L) (W)</th>
<th>(P_H) (W)</th>
<th>(I_E_L) ((× 10^3))</th>
<th>(I_E_H) ((× 10^3))</th>
<th>Per</th>
</tr>
</thead>
<tbody>
<tr>
<td>Military</td>
<td>-81</td>
<td>-67</td>
<td>60</td>
<td>75</td>
<td>3000</td>
<td>20000</td>
<td>5</td>
</tr>
<tr>
<td>Robot</td>
<td>-81</td>
<td>-67</td>
<td>0.5</td>
<td>2.0</td>
<td>180</td>
<td>720</td>
<td>10</td>
</tr>
<tr>
<td>PDA</td>
<td>-81</td>
<td>-67</td>
<td>0.1</td>
<td>0.2</td>
<td>36</td>
<td>72</td>
<td>20</td>
</tr>
<tr>
<td>Sensor</td>
<td>-101</td>
<td>-65</td>
<td>0.000004</td>
<td>0.1</td>
<td>0.1</td>
<td>36</td>
<td>65</td>
</tr>
</tbody>
</table>
4.2.2.2 Simulation Results and Analysis Without Packet Transmission

![Maxpower graph with 100 nodes.](image)

In this subsection, the type and characteristics of the topologies built by the READ, LMST and R&M will be studied. Therefore, the default simulation scenario with the above parameter settings is kept, and there are no transmission activities taking place. First, a graphical comparison of the topologies that each algorithm produces is provided. Figure 4.1 shows the maxpower graph generated when each node uses its maximum transmission power. As it can be observed, the maxpower graph is strongly connected. The maxpower graph in Figure 4.1 is used as a reference. Figures 4.2, 4.3, 4.4, 4.5, 4.6 and 4.7 then plot the topologies generated by LMST with link addition (LMST-add), LMST with link removal (LMST-rem), R&M with link addition (R&M-add), R&M with link removal (R&M-rem), READ with bi-connectivity (READ-K2), and READ uni-connected (READ-K1), respectively.

As it can be seen, the topologies generated by LMST-Add and R&M-Add are connected, while LMST-Rem and R&M-Rem do not generate a connected topology even
though the maxpower graph is connected, which contradicts the original goal of extending the network longevity with preservation of network connectivity.

Figure 4.8 shows the average node degree for each type of topology control algorithm. It can be seen that the average degree of R&M-add and R&M-rem increases linearly with
the number of nodes in the network area or the density, while READ-K1, READ-K2, LMST-add and LMST-rem remain fairly constant and with a very low node degree of around 2 to 3. Average degrees of R&M-add and R&M-rem are always higher than READ and LMST, which can also be confirmed by looking at Figure 4.4 and Figure 4.5, as the topologies generated by R&M are denser than the others.

Finally, Figure 4.9 compares the average link length of READ with the other four algorithms. In general, the average link length of the topology control algorithms decreases with the number of nodes in the network area. In this case, READ’s average link length lies between R&M’s and LMST’s.

### 4.2.2.3 Results and Analysis with Packet Transmission

As earlier mentioned, the weighted cost is a function of the residual energy, which changes with time. To compare the performance of the algorithm against other topology control algorithms in terms of network lifetime, the data packets will be transmitted after the topology is built. In contrast to the previous simulation scenario without packet trans-
mission, this part of the study will consider packet traffic as the default simulation scenario. First, the setting without packet transmission is kept as before and four packet collectors, or sinks, are randomly deployed in the network area.

![Figure 4.8 Average node degree.](image)

![Figure 4.9 Average link length.](image)

In this scenario, every device generates four data packets per time unit and each packet is sent to one specific sink. In order to minimize the performance difference caused by the routing algorithm, the Dijkstra algorithm is utilized in all cases but with different link metrics. The link metric utilized for LMST and R&M was the energy consumption, while for READ, the link metric was the *weighted cost* metric defined in Section 4.1.3. During the simulation, the READ algorithm is triggered every time the residual energy in one node
is reduced by 40% of its last recorded value. Afterwards, the Dijkstra algorithm is run again to establish the routing tables according to the new topology. In the case of R&M-add and LMST-add, the topology control algorithms are run every time a node dies, and Dijkstra is also run to re-compute the best path to the sinks.

Figure 4.10 Number of nodes alive in centralized implementation.

Figure 4.11 Successful delivery rate in centralized implementation.

Figure 4.10 depicts the number of nodes alive, which is the performance metric utilized to reflect the active ability of the network. As the figure illustrates, the number of nodes alive of R&M-add and LMST-add start dropping linearly right after the simulation begins, while READ-K2 remains unchanged until \( \text{time} = 9000 \). This is very significant considering that at \( \text{time} = 9000 \) the network has sent around \( 100 \times 4 \times 9000 = 3.6 \times 10^6 \) packets.
This is even more significant if Figure 4.10 is seen along with Figure 4.11, which shows the packet delivery rate of the algorithms. As it can be seen, while READ-K2 remains at 100% delivery rate, both LMST and R&M drop dramatically from the very beginning. At $time = 9000$, LMST-add has less than 75 nodes alive and R&M has only approximately 70 nodes alive, and their delivery rate dropped to 2% and 65%, respectively. One interesting observation is the case of LMST-rem, which maintains the number of nodes alive at a fairly slow decreasing rate compared to the rest of the algorithms. Although it may be considered the best algorithm from this perspective, when looking at Figure 4.11, however, it can be seen that the delivery rate is very low. The LMST' and R&M algorithms present opposite trends. While LMST algorithms tend to keep more nodes alive than R&M algorithms, R&M algorithms provide better delivery rates than LMST algorithms. This basically means that R&M keeps the network more connected than LMST, even with fewer number of nodes. It also can be seen that the number of nodes alive for maxpower graph drops faster than all other algorithms, which is due to that maxpower graph does not conduct any topology control. However, the performance of the LMST algorithms in terms of the packet delivery rate is lower than the performance achieved by using the maxpower graph, meaning that although the algorithms reduce the topology, this reduction does not consider that the energy consumption associated with edges in heterogeneous network is not necessarily proportional to the length of edges. Therefore, the topology reduction in LMST does not translate into a useful performance advantage.

In order to better understand the network topology evolution, Figures 4.12, 4.13, 4.14 and 4.15 are also included to show the number of nodes alive for each type of device. In READ-K1, four types of devices simultaneously drop at $time = 5500$. In READ-K2, the first major drop happens after $time = 9000$, while in LMST and R&M, they drop at different times. In LMST-add, military devices start to drop at $time = 3000$; robot devices and PDAs start to drop at $time = 2000$; and sensors start to drop at $time = 200$. In R&M,
PDAs and sensors start to die before \(time = 1000\); robots start to die after \(time = 5500\); and military devices start to die after \(time = 8200\). At least two important observations can be made from these four figures: 1) In LMST and R&M, four type of devices drop at different stages of the network lifetime. The less the initial energy, the earlier the devices start to die in LMST and R&M; 2) When comparing of these four figures, it can be found that the major drop of different type of nodes in READ happen simultaneously, which further supports the theory even energy consumption among different types of nodes can maximize network longevity.

![Figure 4.12 Number of military nodes alive in centralized implementation.](image)

![Figure 4.13 Number of robots alive in centralized implementation.](image)

4.3 The Distributed Residual Energy Aware Dynamic Topology Control Algorithm

In this section, the DREAD topology control algorithm is presented. DREAD is the distributed version of the READ algorithm introduced in the last section.

4.3.1 Distributed Residual Energy Aware Dynamic Algorithm

The DREAD algorithm is implemented in two phases: the Initialization phase and the Maintenance phase. It is assumed that a group of wireless devices start working approxi-
Figure 4.14 Number of PDA nodes alive in centralized implementation. 
Figure 4.15 Number of sensors nodes alive in centralized implementation.

ultimately at the same time and enter the Initialization phase. The Initialization phase consists of the following steps:

1. Broadcast Self Advertise Message: Right after node \( u \) starts to work, it enters the Self Advertise Period, denoted as \( T_{SAP} \). Node \( u \) randomly chooses a time moment within \( T_{SAP} \) to broadcast Self Advertise Message, denoted as SAM, at its maximal power \( P_{\text{max}} \) to all of its potential neighbors. The SAM contains the following information:

   - node \( u \)'s location information, which is used to calculate the distance from \( u \) to any potential one-hop neighbor. This calculation is performed using Equation 4.10;

   - node \( u \)'s sensitivity threshold and antenna gain, which are used to calculate the minimum transmission power, \( p_{u,v} \), from any potential neighbor \( v \) to node \( u \);

   - \( P'_u \), the power consumed by the electronics when \( u \) receives any packet, used in Equation 4.12 and Equation 4.13; and,

   - node \( u \)'s current residual energy.
2. Establish One-hop Neighbor Table: Upon receiving an SAM, the receiving node $v$ adds the received information to the One-hop Neighbor Table, denoted as Table.ON, which is shown in Table 4.2. Node $v$ performs a simple calculation using Equation 4.8 to check if node $v$ can reach node $u$ by using its maximum transmission power. If not, node $v$ erases the information from node $u$ from Table.ON. Note that by doing this simple check, directional edges will be eliminated from the network.

3. Establish One-hop Edge Weight Table: With the above information from all the one-hop neighbors, the receiving node can calculate the weighted cost for all the one-hop bi-directional edges. The calculation result will be recorded in the One-hop Edge Weight Table, denoted as Table.OEW, which is shown in Table 4.3, where ID.1 and ID.2 are the identification of two end nodes.

<table>
<thead>
<tr>
<th>Neighbor_ID</th>
<th>Position</th>
<th>Sensitivity(dBm)</th>
<th>Re.Power (W)</th>
<th>Re.En(J)</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(18,40)</td>
<td>-70</td>
<td>0.3</td>
<td>200</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3 One-hop edge weight table.

<table>
<thead>
<tr>
<th>ID.1</th>
<th>ID.2</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.02</td>
</tr>
</tbody>
</table>

4. Exchange One-hop Edge Weight Table and One-hop Neighbor Table: Once the T.SAP expires, node $u$ enters the Advertise Neighbor Period, denoted as T_ANP. It chooses a random time to broadcast its Table.OEW and Table.ON at the maximum transmission power.

5. Create Two-hop Edge Weight Table and the Two-hop Neighbor Table: Upon receiving the Table.OEW and the Table.ON from a neighbor, the receiving node $u$ merges
two local one-hop tables with received tables to create a Two-hop Neighbor Table and a Two-hop Edge Weight Table, which include the neighbor information and edge weight information within two hops. These two tables are very similar to the one-hop tables and are denoted as Table.TN and Table.TEW, respectively, which are shown in Table 4.4 and Table 4.5.

Table 4.4 Two-hop neighbor table.

<table>
<thead>
<tr>
<th>Neighbor ID</th>
<th>Position</th>
<th>Sensitivity(dBm)</th>
<th>Re Power (W)</th>
<th>Re En(J)</th>
<th>Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(18,40)</td>
<td>-70</td>
<td>0.3</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>(34,50)</td>
<td>-90</td>
<td>0.5</td>
<td>100</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.5 Two-hop edge weight table.

<table>
<thead>
<tr>
<th>ID .1</th>
<th>ID .2</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.02</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.12</td>
</tr>
</tbody>
</table>

In order to better explain the process of building up two-hop tables, an example is provided in Figure 4.16. A maxpower graph before topology control is displayed in the top part of the figure, where node A has three one-hop neighbors: B, C and D. After step 3, each of these nine nodes has its Table.ON available. The one-hop neighbor tables of nodes A, B, C and D are displayed on the left side of the figure. At step 4, the DREAD algorithm running on each node starts to exchange one-hop neighbor tables. Node A receives one Table.ON from each of its one-hop neighbors, in this case, from nodes B, C, and D, as listed on the left side of the Figure 4.16. Once node A receives these one-hop neighbor tables, it merges all of them into its own Table.TN, as shown at the right side of Figure 4.16. Table.TEW at node A is
built in the same manner. After step 4, node A has collected all the information about the nodes and edges within two hops.

![Diagram of two-hop tables]

Figure 4.16 Establishment of two-hop tables.

6. Create Symmetric Local Minimal Spanning Tree: Once the T_ANP expires, every node uses the Prim algorithm [65] to generate its first one-hop Symmetric Local Minimal Spanning Tree based on the information from Table_TN and Table_TEW. Basically, the Prim algorithm will first generate a two-hop local minimal spanning tree and only the first hop of the tree is kept in the final symmetric local minimal spanning tree. The reason it is called symmetric is because if an edge between node
$u$ and node $v$ is selected by node $u$, it is also selected by node $v$. The proof of this Claim is included in Appendix A.

To illustrate the generation of the local minimal spanning tree using the Prim algorithm, the same example from Figure 4.16 is used as shown in Figure 4.17. After step 4, node $A$ has a maxpower graph within two hops as in part $a$ of Fig. 4.17. Then, node $A$ starts using the Prim algorithm to select edges within two hops, as in part $b$ of Figure 4.17. Finally, node $A$ will only keep one-hop of the generated tree as in part $c$ of Figure 4.17. The same algorithm is conducted in all the other nodes independently. Parts $d$, $e$ and $f$ illustrate the generation of a local minimal spanning tree on node $C$.

![Image of diagrams showing A and C's two max power graphs and their corresponding local minimal spanning trees](image)

Figure 4.17 Generation local minimal spanning tree.

The Maintenance phase starts after the very first DREAD topology is generated. It is responsible for adjusting the topology according to the value of the remaining energy. In the current DREAD implementation, the algorithm is triggered once the node’s residual

78
energy reaches 60% of its previous record value. The Maintenance phase consists of the following steps:

1. Broadcast Update Control Message: Whenever a node $u$’s residual energy drops 60% of its previous value, it uses its maximum transmission power to broadcast an Update Control Message, denoted as a UCM, to its neighbors with the updated residual energy. The structure of the UCM is very similar to an SAM, except it has a special field to denote the message type. A UCM not only informs all the potential neighbors about the node’s residual energy, but also solicits them to reply with their updated residual energy using an SAM.

2. Reply Self Advertise Message: Once node $v$ receives a UCM from any of its potential neighbors, it uses Equation 4.8 to check if it can reach node $u$ using its maximum transmission power. If not, node $v$ would simply omit this UCM. Otherwise, it would use the residual energy information from the UCM to overwrite the corresponding value in Table.ON and update its Table.OEW accordingly. It also broadcasts SAMs with its maximum transmission power.

3. Enter Waiting Reply Period and wait for replies: Once node $u$ broadcasts a UCM, it enters a Waiting Reply Period, denoted as T.WRP, to wait for SAMs replied by all the potential neighbors.

4. Build up new one-hop tables: Upon receiving an SAM, node $u$ records them into its Table.OEW and Table.ON.

5. Exchange one-hop tables: After the T.WRP expires, nodes $u$ enters the T.ANP. It chooses a random time to broadcast its One-hop Tables with Reply Request Message, denoted as an OTRR message, at the maximum transmission power, which includes Table.OEW and Table.ON.
6. Build up two-hop tables: Upon receiving the OTRR, all the nodes will reply with their own Table.OEW and Table.ON during the T.ANP, and then update their two-hop tables. Local minimal spanning tree based on the updated Table.TEW and Table.TN will be calculated.

7. Construct new symmetric LMST: The node $u$ that initiated the topology update process will recalculate its local minimal spanning tree based on all the updated residual energy information as well as Table.TEW and Table.TN.

The detailed complete algorithm for node $u$ is as follows:

**Initialization:**

1. Enter T.SAP period and broadcast SAM message with $P_{\text{max}}$ using random times.
2. If (receive SAM message from node $v$ during T.SAP) and ($v$ can be reached by $u$’s $P_{\text{max}}$)
3. Then put received information into Table.ON and update Table.OEW
4. End if
5. If period of T.SAP expires
6. Then 1) enter T.ANP period
7. 2) choose a random time to broadcast Table.ON and Table.OEW at $P_{\text{max}}$.
8. End if
9. If receive Table.ON and Table.OEW
10. Then merge received tables with local tables to generate Table.TN and Table.TEW
11. End if
12. If T.ANP period expires
13. Then construct symmetric local minimal spanning tree

**Maintenance:**

14. If TC triggering condition is satisfied
15. Then broadcast UCM message and enter $T_{WRP}$ period to wait for SAM messages
16. End if
17: If (receive UCM message from node \( v \)) and (\( v \) can be reached by \( u \)'s \( P_{max} \))
18: Then 1) use received information to update Table.ON and Table.OEW
19: 2) broadcast SAM message
20: Else omit received UCM message
21: End if
22: If (receive SAM message from node \( v \) during T.WRP) and (\( v \) can be reached by \( u \)'s \( P_{max} \))
23: Then put received information into Table.ON and Table.OEW
24: End if
25: If period of T.WRP expires
26: Then enter T_ANP period and choose a random time to broadcast OTRR message
27: End if
28: If receive OTRR message
29: Then 1) update Table.TN and Table.TEW
30: 2) recalculate local symmetric local minimal spanning tree
31: 3) broadcast Table.ON and Table.OEW
32: End if
33: If T_ANP period expires
34: Then recalculate symmetric local minimal spanning tree
34: End if

4.3.2 Complexity of the DREAD Topology Control Algorithm

In order to derive the computational and message complexity of the Maintenance phase, the average number of neighbor nodes from each device is denoted as \( n \), which can vary with the number of total nodes alive. The computational complexity of steps 1, 2, 3 and 5 are constant, and, in the worst case, bounded by \( O(1) \). Step 4 builds the one-hop neighbor tables, which has a computational complexity of \( O(n) \). In step 6, two-hop tables are built
by appending all the received one-hop neighbor tables together, sorting them, and then merging the common items. Since there are \( n \) neighbors for each node, \( n \) one-hop neighbor tables will be received, and the length of each of those tables is \( n \), so after appending them together, there are \( n^2 \) items at the most before sorting. Therefore, by using an appropriate sorting algorithm, the computational complexity of step 6 is bounded by \( O(n^2 \log n^2) = O(2n^2 \log n) = O(n^2 \log n) \). In step 7, the local minimal spanning tree is built based on two-hop neighbor tables using the Prim algorithm. In the worst case, there are \( n^2 \) nodes in a two-hop neighbor table and \( n^2 \) edges in a two-hop edge weight table. By using binary heap, the computational complexity of local Prim algorithm in the worst case is bounded by \( O(n^2 \log n^2) = O(2n^2 \log n^2) = O(n^2 \log n) \). Therefore, the total computational complexity of the Maintenance phase is bounded by \( O(1 + n^2 \log n + n^2 \log n + n) = O(n^2 \log n) \).

A total of three messages are sent by each node during the Maintenance phase: UCM, SAM, and OTRR message. Therefore, the message complexity of the Maintenance phase is bounded by \( O(1) \).

### 4.3.3 Simulation Results and Evaluation of DREAD

In this section, the performance evaluation of the DREAD algorithm is presented. The simulation setup parameters are described first and then the simulation results and the analysis are presented.

#### 4.3.3.1 Simulation Setup

In this simulation experiment, a heterogeneous wireless network with four types of devices are considered: sensors, PDAs, robots, and military devices. One hundred nodes in a \( 1000m \times 1000m \) network area are randomly distributed, where 5% are military devices, 10% are robots, 20% PDAs, and 65% of them are sensors. The same settings used to configure the devices in the earlier READ simulation (See Table 4.1), are used here in the
distributed version. Each node utilizes a uniform distribution to randomly draw its maximal transmission power in Watts, receiver sensitivity in dB, and initial energy in Joules from the pre-defined ranges, \([P_L, P_H], [\beta_L, \beta_H] \) and \([I.E_L, I.E_H] \), that characterize each type of device. The range of these parameters are based on the literature [61–64]. Without loss of generality, omnidirectional antennae are considered and set to \(G_t = G_r = 1 \), the system loss factor \(L = 1 \), and the operational frequency equal to 2.472GHz. \(C_T \) is subject to the wireless interface device itself and is assigned a constant value of 0.6 for all the nodes without loss of generality. The value of \(t_{tx} \) and \(t_{rx} \) are also set to 0.01 seconds, meaning that packets are considered of equal and fixed size.

![Figure 4.18 Topology generated by DREAD.](image)

4.3.3.2 Results and Analysis

The performance of different topology control algorithms are evaluated under the scenario where packets are transmitted and distributed topology control algorithms are running constantly. Four sinks are randomly deployed in the network area. Each of these 100 devices generates four data packets per time unit and sends one packet to each sink. In order to minimize the performance difference caused by the routing algorithm, the Di-
Jkstra algorithm is utilized in all cases but with different link metrics. The link metric utilized for LMST and R&M was energy consumption, while for READ, the link metric was the weighted cost metric defined in Section 4.1.3. Every device is running its own DREAD topology control algorithm independently. During the simulation, the DREAD algorithm is triggered every time the residual energy in one node is reduced by 40% of its last recorded value. Afterwards the Dijkstra algorithm is run again to establish the routing tables according to the new topology.

Both R&M and LMST are implemented in a distributed manner and therefore run on every node independently. The energy consumption of control messages exchange incurred by invoking LMST or R&M topology control algorithms is taken into account. In the distributed LMST, topology control algorithm generates a local minimal spanning tree using the Euclidean distance as metric to select edges and only keeps on-tree nodes that are one-hop away as its neighbors in the final topology. The distributed R&M topology control algorithm selects edges based on the concept of Relay Region, which does not take residual energy or antenna sensitivity into account. LMST and R&M are invoked every time a node dies and Dijkstra is also run to re-compute the least total energy consumption path to the sinks. In DREAD, as explained earlier, the weighted cost function is used as the metric to generate the local minimal spanning Tree and to compute the best path to sinks by Dijkstra. DREAD, LMST and R&M are all distributed, position-based topology control algorithms.

Figure 4.18 illustrates the topology of the network after running DREAD. The figure shows a topology made of 184 edges after 100 time units, while there are 983 edges in maxpower graph at the same moment.

Figure 4.19 depicts the number of nodes alive in the network with consideration of energy consumption incurred by data packet transmission and by invoking topology control algorithms. As it can be seen, the curve follows the trend that is observed in Figure 4.10.
The number of nodes alive in the case of LMST and R&M start dropping linearly right after the simulation begins. At time = 6000, the number of nodes alive in R&M-add and R&M-rem have decreased from 100 to 60 and the number of nodes alive in LMST-add and LMST-rem have decreased from 100 to 70. However, the number of nodes alive in the case of DREAD remains almost unchanged until time = 7000. Considering that at moment time = 7000, 100 × 4 × 7000 = 2.8^6 packets have been sent, this is a significant difference in terms of network lifetime compared with the other algorithm. This network longevity gain is due to the fact that DREAD always keeps the links that have less weighted cost function in the topology so that all the devices contribute fairly considering their residual
energy in the data packet transmission procedure. Figure 4.20 shows the packet delivery rate of the algorithms. As it can be seen, DREAD’s delivery rate remains at 100% until \( time = 7000 \), while both LMST’s and R&M’s delivery rate drop dramatically from the very beginning. For example, at \( time = 5000 \) the delivery rate of LMST and R&M are all below 40%.

It can also be seen that the distributed LMST and R&M algorithms do not outperform the maxpower graph in terms of number of nodes alive and delivery rate. The reasons is the maxpower graph curve in Figure 4.20 is the same one as in Figure 4.11, which does not consider any energy consumption incurred by the exchange of messages during the generation of the maxpower graph. On the other hand, the distributed LMST and R&M algorithms deduct energy every time a topology control message is exchanged.

Figures 4.21, 4.22, 4.23 and 4.24 show the number of nodes alive for each type of device. For DREAD, the number of each type of nodes drops simultaneously around \( time = 7000 \), which occurs because DREAD keeps energy consumption fairness in mind and lets the devices with more residual energy contribute more in the data packet transmission process. In LMST and R&M, different devices drop at different times. In LMST, sensors start to drop at \( time = 100 \); PDAs start to drop at \( time = 500 \); military devices start to drop at \( time = 2000 \); and robots devices start to drop at \( time = 3000 \). In R&M, sensors start to drop at \( time = 150 \); PDAs start to die before \( time = 600 \); robots start to die after \( time = 3000 \); and military devices start to die after \( time = 5000 \). In LMST and R&M, the four type of devices die at different stages of the network lifetime. The lower the initial energy, the earlier the devices start to die in LMST and R&M. In DREAD, all devices die simultaneously, which further confirms that evening out energy consumption among different types of nodes can maximize network longevity.
4.3.3.3 Comparison Between READ and DREAD

To further analyze the performance of READ and DREAD, the number of nodes alive and the delivery rate of DREAD-K2 from Section 4.2 are used to compare to that of DREAD. READ is a centralized topology control algorithm and the energy consumption during the exchange of control messages is not taken into account in the simulation. DREAD, on the other hand, is a distributed topology control algorithm, which runs on each individual node independently and considers all the energy consumption incurred by control packets. Both READ and DREAD use the same weighted cost function metric to select edges. Figure 4.25 and Figure 4.26 demonstrate the number of nodes alive and de-
livery rate. DREAD drops faster than READ in both figures. DREAD starts dropping at time = 7000, while READ starts dropping at time = 9000. The reasons that READ can extend network longevity further are twofold. First, the centralized version selects edges based on the information of the entire network and the resulting global spanning tree is optimal. In contrast, DREAD is distributed, which means every local minimal spanning tree is only optimal locally, but not necessarily optimal in the entire network. Secondly, the energy consumption during the generation of global READ topology is not deducted in the READ simulation, while the energy consumption incurred by the exchange of topology control messages is deducted in the DREAD simulation.

Figures 4.27, 4.28, 4.29 and 4.30 shows the comparison of the number of nodes alive for different type devices between READ and DREAD. In These four figures, DREAD drops earlier than READ for the same reasons stated before. Crossing these four figures, it can be found that the different types of devices in both DREAD and READ drop simultaneously due to the fact that both Residual Energy-Awareness Dynamic topology control algorithms keep energy consumption in mind and use the weighted cost function to select edges.
4.4 Conclusions

Proposal of the READ topology control algorithm, which considers the problem of topology control in heterogeneous wireless scenarios, where sensor nodes, ad hoc nodes, robots with communication capabilities, and even more powerful military wireless devices work together in the same application. This is the first research that takes into account difference in initial energy, residual energy, receiver sensitivity, and antenna gain for every wireless device connected.
CHAPTER 5
CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

This dissertation studies sink localization problem in large scale WSNs as well as topology control in heterogeneous WSNs. Sink localization problems are considered first with the introduction of the ALS protocol, a quorum-based mechanism for large scale, WSNs that allows sources to find the location of the sinks in a scalable and efficient manner. In ALS, each sink selects a set of nodes to establish a global anchor system that facilitates the propagation of its location information. In this manner, global or directed flooding procedures are avoided substantially reducing the communication overhead. Furthermore, multiple sources use the same global anchor system to find the sink’s location, reducing the communication and storage overhead even more. By means of a mathematical model and simulations, we demonstrate the effectiveness and scalability of the location service with multiple and moving sinks and sources, different network densities, and increased network areas. ALS also includes procedures to avoid frequent flooding, caused by the mobility of the sinks and targets. In addition, we show that ALS with GPSR, a location-based routing protocol, reduces the communication overhead by 50% compared with TTDD, a very well known grid-based routing protocol for WSNs.

The challenges of topology control in heterogeneous multi-hop WSNs is considered next. Different types of devices working on the same application are studied, and the problem is solved as a power assignment problem. We propose both centralized and distributed
versions of the READ topology control algorithm, which considers the receiver’s sensitivity, the sender’s maximal transmission power, and the node’s residual energy to determine the final topology.

The READ algorithm dynamically recruits the links that optimize the workload between different wireless devices while maintaining network connectivity. Simulation results demonstrate that READ can efficiently increase the network lifetime, presents a low average node degree and increases the packet delivery rate by 40% over R&M and 90% over LMST, two well known position-based topology control mechanisms.

5.2 Future Work

The research included in this dissertation can be extended in several ways. Examples of possible future work include the following:

- Currently anchor setup and query are carried on in horizontal and vertical directions, therefore, the resulting system in ALS is grid system. In the future, different geometric shapes could be explored to reduce the communication overhead even further.

- In READ, ratios for each different type of devices are fixed at 5% for military nodes, 10% for robot, 20% for PDAs and 65% for sensors. All the performance evaluations in this dissertation were conducted with these ratios. In the future, the composing ratios can be varied to and investigate the performance under different ratios and find out the relationship between performance of the READ algorithm and the composing ratio of different wireless devices.

- An important aspect in the topology control problem is the introduction of mobility in the algorithm. In heterogeneous networks, like the one utilized in this dissertation, it is very likely that robots, military devices, and PDAs will move, which imposes new challenge to topology control algorithm design.
REFERENCES


93


Appendix A Property Proof of ALS and READ

A.1 Convergence Proof of ALS’ Anchor Setup Process and Query Process

As mentioned in Section 3.1, both the anchor system setup and the query processes start sending four first stage orthogonal messages. These messages may later split into second stage messages if they encounter the border of an inner void area or the network’s boundary. The second stage messages move according to the right hand and left hand rule to go around the void areas. Since the area outside the sensor network can be viewed as the largest void area, routing along the boundary of the sensor network is considered a special case of routing around an inner void area. In this section, the convergence of the anchor setup and query processes is proved by showing that “two second setup messages meet each other by using the right hand and left hand rule”. In the proofs that follow, the following definitions are utilized:

Definition 1 (real-grid-point and void-grid-point): Define any grid point \( P(x_p, y_p) \) as a real-grid-point, if there is a sensor node that can be elected as a grid node for the grid point \( P \). Similarly, define any grid point \( P(x_p, y_p) \) as a void-grid-point if there is no such sensor node.

Definition 2 (real-edge and void-edge): Connect every pair of adjacent grid points by a virtual edge, only if those two adjacent grid points are both real-grid-points or void-grid-points. Define the virtual connection between two adjacent real-grid-points as real-edge and the virtual connection between two adjacent void-grid-points as void-edge. Note that real-edges and void-edges are all along the grid lines in the four orthogonal directions. Fig-
Appendix A (Continued)

ure A.1 shows a graphical representation of real-grid-points, void-grid-points, real-edges, void-edges, and four void areas.

Definition 3 (neighbor and far-neighbor): Every grid point has other eight other adjacent grid points around itself. Four of them, located in four orthogonal directions, are called neighbors and the others are called far-neighbors.

For any two void-grid-points, if they are neighbors or far-neighbors, they are considered to be connected and in the same void area, as shown in Figure A.1. On the other hand, real-grid-points are connected only if they are neighbors. Any real-grid-point or void-grid-point is also connected to itself. Therefore, connected is a congruent relation, which satisfies reflexive, symmetric, and transitive conditions.

It is assumed that the whole network is not partitioned and that all real-grid-points are connected. For example, in Figure A.1, there are four void areas, which are disconnected with each other. These void areas are bounded by void-grid-points connected to each other via void-edges, such as cases a, b, and d. If there is no void-edge between two void-grid-points but they are far-neighbors to each other, they are still considered in one void area and connected, as in case c.

Definition 4 (Void Polygon, World Void Polygon, and Snode(VP_i)): Define every void area as Void Polygon, denoted as VP. Define the Void Polygon containing all the void area outside the whole network area as World Void Polygon, or WVP, which is a special case of VP. For any VP_i, define the void-grid-point set which includes all the void-grid-points in VP_i as Snode(VP_i).
Appendix A (Continued)

Figure A.1 A graphical representation of real-grid-nodes (solid nodes), void-grid-nodes (hollow nodes), real-edges, void-edges, and void areas.

In order to simplify the proof, the case a in Figure A.1 will not be considered because one real-grid-point inside the void area is not connected to the rest of the area. Actually, the real-grid-point will be considered as a void-grid-point and therefore, it will not make any difference in the proof.

*Definition 5 (Real Polygon, World Real Polygon, and Universal set of Real Polygons):* Define *Real Polygon*, denoted as *RP*, as any closed area where the boundary of the area consists of real-grid-points and real edges, while means that the inner area of an RP may contain some void polygons. In Figure A.2, there are four RP examples. The Universal Set of RP is denoted as *U(RP)* and the biggest RP in the network area as *World Real Polygon* or *WRP*. 

100
Appendix A (Continued)

Figure A.2 A graphical representation of four Real Polygon examples.

Figure A.3 A graphical representation of void areas and their envelops.
Definition 6 (Cover \((VP_i)\)): Define \(Cover(VP_i)\) the set of RPs that all contain the same \(VP_i\) or \(Cover(VP_i) = \{RP_i | RP_i \in U(RP) \& WRP \supseteq RP_i \supseteq VP_i\}\).

Definition 7 (Envelop \((E(VP_i))\)): Consider any \(VP_i\), \(VP_i\)'s Envelop, denoted as \(E(VP_i)\), as an RP, defined by \(E(VP_i) = \bigcap_{RP_i \in Cover(VP_i)} RP_i\). As a result, \(E(VP_i)\) is not only the smallest RP that contains \(VP_i\) but also unique. Since \(E(VP_i)\) is the smallest RP in the set of \(Cover(VP_i)\), the inner area of \(E(VP_i)\) contains and only contains the void polygon \(VP_i\). Also, using this definition, the boundary of WRP can be referred to as \(E(WVP)\). Figure A.3 shows four void areas and their corresponding \(E(VP_i)\).

Lemma 1: Consider a void polygon \(VP_i\), starting from any one of real-grid-point neighbors of \(VP_i\), the second stage anchor setup or query messages can construct \(E(VP_i)\) by using the right hand rule to route around the \(VP_i\).
Appendix A (Continued)

Proof: Lemma 1 will be proved by induction. Denote the cardinality of $S_{node}(VP_i)$ as $n$, i.e., $|S_{node}(VP_i)| = n$. We represent the forwarding and incoming direction of the message by a vector in the complex plane. When $n = 1$, there is only one void-grid-point in $VP_i$, and therefore, only possible scenario, like the one shown in Figure A.4. In the same figure, when the first stage setup message reaches $VP_i$ at $G_2$ with forwarding direction $(x, yi)$, it splits into two second stage setup messages. The second stage setup message using the right hand rule at $G_2$ utilizes the priority sequence $(x, yi) \times (0, i)$, $(x, yi) \times (0, i)^2$, $(x, yi) \times (0, i)^3$ to select the next grid point as its forwarding destination. Every time we apply cross multiplication $(0, i)$ to a vector, we turn that vector counter-clockwise turn that vector 90 degrees. As a result, $G_3$ is chosen, as illustrated in the figure. Then at $G_3$, the setup message with incoming direction $(x', y'i)$ uses priority sequence $(x', y'i) \times (0, i)$, $(x', y'i) \times (0, i)^2$, $(x', y'i) \times (0, i)^3$ to select the next grid point, which is $G_4$. The process continues in the same manner and the setup message will hop over nodes $G_5$, $G_6$, $G_7$, $G_8$, $G_9$ according to the priority sequence $(x', y'i) \times (0, i)$, $(x', y'i) \times (0, i)^2$, $(x', y'i) \times (0, i)^3$ to construct the Polygon $G_2G_3G_4G_5G_6G_7G_8G_9$. As it can be seen, the setup message only uses the forwarding direction as the base direction to calculate the next forwarding direction at $G_2$ while it uses the incoming direction as the base direction in the rest of the process. The border of $G_2G_3G_4G_5G_6G_7G_8G_9$ consists of real-edges that can be traveled around by setup messages. Hence the polygon is closed. In other words, $G_2G_3G_4G_5G_6G_7G_8G_9$ is an RP and $VP_i \subset \{G_2G_3G_4G_5G_6G_7G_8G_9\}$. From Figure A.4, it can be seen that there is no other RP containing $VP_i$ with fewer number of grid points or real-edges, or smaller area than $G_2G_3G_4G_5G_6G_7G_8G_9$. The reason is that deleting any real-edge or real-grid-point from real-polygon $G_2G_3G_4G_5G_6G_7G_8G_9$ will cause it to be unclosed, which conflicts with Definition 5. Hence the real polygon
Appendix A (Continued)

Figure A.5 Base case 1 with 2 void-grid-points.

When \( n = 2 \), there are two possible scenarios. One of which shown in Figure A.5. In this case, a first stage setup message splits into two second stage setup messages when it meets \( VP_i \) at \( G_2 \) with forwarding direction \((x, yi)\). The second stage setup message using the right hand rule at \( G_2 \) uses the priority sequence \((x, yi) \times (0, i), (x, yi) \times (0, i)^2, (x, yi) \times (0, i)^3\) to select the next grid point as its forwarding destination. As illustrated in the figure, \( G_3 \) is chosen. Then, at \( G_3 \), the setup message with \textit{incoming} direction \((x', y'i)\) uses priority sequence \((x', y'i) \times (0, i), (x', y'i) \times (0, i)^2, (x', y'i) \times (0, i)^3\) to select next grid point, which is \( G_4 \). Continuing the process, the setup message will hop over \( G_5, G_6, G_7, G_8, G_9, G_{10}, G_{11} \) according to the priority sequence \((x', y'i) \times (0, i), (x', y'i) \times (0, i)^2, (x', y'i) \times (0, i)^3\). Polygon \( G_2G_3G_4G_5G_6G_7G_8G_9G_{10}G_{11} \) is an \textit{RP} because its border consists of real-edges and real-grid-points. Also, because its border can be cycled around, starting from and ending at the same real-grid-point, and its inner area contains void polygons \( VP_i \), there is no other
Appendix A (Continued)

*RP* that contains \( V P_i \) with less number of real-grid-points or real-edges or less area than \( G_2G_3G_4G_5G_6G_7G_8G_9G_{10}G_{11} \). This is true, because if any real-edge or real-grid-point from *RP* \( G_2G_3G_4G_5G_6G_7G_8G_9G_{10}G_{11} \) is deleted, it will be unclosed, which conflicts with Definition 5. As a result, real polygon \( G_2G_3G_4G_5G_6G_7G_8G_9G_{10}G_{11} \) is \( E(V P_i) \).

The other scenario is shown in Figure A.6. In this case, the setup message chooses the first real-grid-point \( G_2 \), followed by \( G_3, G_4, G_5, G_6, G_7, G_8, G_9, G_{10}, G_{11}, G_{12} \) and \( G_{13} \). The *RP* \( G_2G_3G_4G_5G_6G_7G_8G_9G_{10}G_{11}G_{12}G_{13} \) is then created. Applying the same deduction, it can be shown that *RP* \( G_2G_3G_4G_5G_6G_7G_8G_9G_{10}G_{11}G_{12}G_{13} \) is the smallest closed real polygon which contains \( V P_i \). Therefore, it is \( E(V P_i) \).

Proofs for the other cases where the anchor setup or query messages start at different nodes in the graph and \( n = 3, 4, 5, 6, 7... \) are omitted here since they can be done following the same procedure. Therefore, if we suppose \( n = k \), a second stage anchor setup or query messages using the right hand rule will create \( E(V P_i) \), which is an *RP* and the smallest *RP* that contains \( V P_i \). When \( n = k + 1 \), we add an additional void-grid-point \( \Phi \) to the \( k \)-void-grid-point \( V P_i \). If \( \Phi \) can be seen as part of the new \( VP \), there must be at least one void-grid-point which belongs to \( VP \), and which is \( \Phi \)'s neighbor or far-neighbor. Also, assuming that all the real-grid-points are connected with each other, there are at most seven void-grid-points which belong to \( VP \), and which are \( \Phi \)'s neighbors or far-neighbors. If \( \Phi \)'s eight neighbors and far-neighbors all belong to \( VP \) and there is a real-grid-point at \( \Phi \)'s position in the \( n = k \) scenario, the original real-grid-point at \( \Phi \)'s position is disconnected with the other real-grid-points, which contradicts the assumption.

Now, we consider the case where only one grid-point belonging to \( VP \) is \( \Phi \)'s neighbor or far-neighbor. There are two cases. Case 1 is shown in Figure A.7. In this case,
Appendix A (Continued)

Figure A.6 Base case 2 with 2 void-grid-points.

Figure A.7 Step with 1 void-grid-node as far neighbor.
Appendix A (Continued)

$G_1, G_2, G_3, G_4, G_5, G_6, G_7$ are real-grid-points. The 4-point star $v_1$ belongs to k-void-grid-point $VP_i$. The 5-point star $v_2$ was a real-grid-point when $n=\overline{k}$, and now becomes a void-grid-point. By induction, when $n=\overline{k}$ Lemma 1 holds, and we have $G_1, G_2, V_2 \in E(\overline{VP}_i)$. For the k-void-grid-point $VP_i$, when the setup message arrives at $G_1$, it transmits from $V_2$ to $G_2$ using the right hand rule. After $v_2$ becomes a void-grid-point, the setup message will be forwarded from $G_1$ to $G_3, G_4, G_5, G_6, G_7, G_2$. The sequence is shown in Figure A.7. Assume that the boundary of the original real polygon constructed by the second stage anchor setup or query message with right hand rule for $VP_i$ has a cycling sequence of $G_1, V_2, G_2, H_1, H_2, ..., H_j$. Now the new cycling sequence of the boundary of the RP constructed by the second stage anchor setup or query message with right hand rule for $VP_i + V_2$ is $G_1, G_3, G_4, G_5, G_6, G_7, G_2, H_1, H_2, ..., H_j$. So the border of the polygon constructed by the right hand rule is composed of real-edges, and it can be cycled around, starting from and ending at the same real-grid-point. The polygon is closed. Hence, it is an RP, and $VP_i + V_2 \subset \{G_1, G_3, G_4, G_5, G_6, G_7, G_2, H_1, H_2, ..., H_j\}$. There is no other RP containing $VP_i + V_2$ with less number of grid points or real-edges, or less area than $G_1, G_3, G_4, G_5, G_6, G_7, G_2, H_1, H_2, ..., H_j$. According to the induction hypothesis, RP $G_1, V_2, G_2, H_1, H_2, ..., H_i$ is $E(\overline{VP}_i)$ which does not contain any redundant area or real-grid-point, therefore the new part $G_1, G_3, G_4, G_5, G_6, G_7, G_2$ does not involve any other new redundant area or real-grid-point or real-edge in the above process. Therefore $G_1, G_3, G_4, G_5, G_6, G_7, G_2, H_1, H_2, ..., H_j$ is the smallest RP which contains $VP_i + V_2$. Thus we have shown that the RP $G_1, G_3, G_4, G_5, G_6, G_7, G_2, H_1, H_2, ..., H_j$ is $E(\overline{VP}_i+V_2)$.

The same holds for the other case when $t = 2$, as shown in Figure A.8. We can also prove this lemma when $t = 3, 4, 5, 6, 7$ in a similar way. (The rigorous proof is again omitted.) So when $n = \overline{k + 1}$, the second stage setup message with the right hand rule will
Appendix A (Continued)

create the $E(VP_i)$, which is an RP and the smallest RP which contains $VP_i$.

Hence, we conclude that using the right hand rule, three setup message will create the $E(VP_i)$ for all $|Snode(VP_i)|$ by induction.

**Lemma 2**: Consider a void polygon $VP_i$, starting from any one of real-grid-point neighbor of $VP_i$, the second stage anchor setup or query messages can construct $E(VP_i)$ by using the left right hand rule to route around the $VP_i$.

Proof: The proof is similar to that of Lemma 1.

**Claim 1**: If two second stage anchor setup or query messages start from a real-grid-point, which is a neighbor of a void-grid-point on a VP, using the right hand rule and left hand rule respectively, they will meet each other somewhere around the VP.

Proof: From Lemma 1 and Lemma 2, the second stage anchor setup or query messages using the right hand rule and left hand rule will each construct an $E(VP_i)$ for $VP_i$. Note
Appendix A (Continued)

that there is only one \( E(V_{Pi}) \) for \( V_{Pi} \) and it is a cycle, so the messages using the right hand rule and left hand rule will construct the same \( E(V_{Pi}) \) RP. If the network size is bounded and two second stage anchor or query messages travel on the \( E(V_{Pi}) \)'s boundary from opposite directions, they will meet at some real-grid-point in a finite number of steps.

**Claim 2**: During the anchor setup or query process, when multiple setup messages meet at the boundary of the WVP, they will terminate the process in a finite number of steps.

Proof: Considering that the boundary of the WRP is the only envelop of the whole void area WVP, if a setup message reaches the border of the WRP, it will split into two second stage setup messages. From Lemma 1 and Lemma 2, the second stage anchor setup or query messages using the right hand rule and left hand rule will each construct an \( E(WVP) \) for \( WVP \). Note that there is only one \( E(WVP) \) for WVP and it is a cycle, so the second stage anchor setup or query messages using the right hand rule and left hand rule will construct the same \( E(WVP) \) RP. Considering that there are at most eight second stage messages and that the number of real-grid-points on \( E(WVP) \)'s boundary is finite, these messages will definitely meet each other. Furthermore, at that point they will stop their propagation.

**A.2 Connectivity and Symmetric Property Proof of READ**

*Claim 3*: If \( G_{max} \) is connected, the resulting topology is also connected.

Proof: We will prove it by contradiction. If the resulting topology is not connected, then at least two nodes, let us say \( u_1 \) and \( u_2 \) are not connected and in different isolated sets. \( u_1 \) is in set \( G_c \). Since \( G_{max} \) is connected, there exists at least one path between \( u_1 \) and \( u_2 \) in \( G_{max} \), let us call it \( \psi \). If we walk along the path \( \psi \) from \( u_1 \) to \( u_2 \), and suppose the first node which is not in set \( G_c \) is the node \( u_i \) in isolated set \( G_c \) and the node prior to \( u_i \) on path \( \psi \) is
Appendix A (Continued)

$u_i$, which is still in set $G_z$. We call the edge between $u_i$ and $u_j$ in path $\psi$ as $e(u_i, u_j)$. The READ algorithm is terminated when either all the node sets are connected or it finishes scanning all the edges. Since we assume that resulting topology is not connected so it means READ has finished scanning all the edges and could not find any edge to connect node sets $G_z$ and $G_\tau$, which contradict with the existence of edge $e(u_i, u_i)$.

Claim 4: Local Spanning Tree constructed by DREAD is symmetric

Proof: Since Onehop Edge Weight Table & Onehop Neighbor Table are broadcasted once, therefore those information is propagated and converged within two hops. We will prove Claim 4 by contradiction. Suppose node A and node B are neighbors to each other in max power graph as shown in Figure A.9. After exchanging Onehop Edge Weight Table & Onehop Neighbor Table, every node only knows neighbors’ and edges’ weight information within two hops and those information is identical. For instance, node A knows information regarding nodes B, C, D, E, F, G, H and edges AB, AC, AD, AE, CH, CE, CB, BH, BG, BF. Node A has no information about node I or edge HI. We assume that A chooses edge AB in its LMST by using DREAD and B does not choose AB in its LMST. Since node B does not choose edge AB in its LMST generated based on two hop information, it means that B could find another path to reach A within two hop with less weight than to reach A by AB directly. Without loss of generality, let us say there is a common neighbor node C within two hops of node B and node A so that Equation A.1 holds as following:

\[
 w(e(A, B)) > w(e(A, C)) + w(e(B, C)) \tag{A.1}
\]
Appendix A (Continued)

Figure A.9 A simple max power network example.

However, since edge AB is selected by node A, which means Equation A.2 holds as well as following:

\[ w(e(A, B)) < w(e(A, C)) + w(e(B, C)) \]  \hspace{1cm} (A.2)

Since neighbor information and edge weight information are propagated and identical within two hops, it can be concluded that Equation A.1 and Equation A.2 conflicts with each other. Therefore, Claim 4 holds.
ABOUT THE AUTHOR

Mr. Rui Zhang is a Ph.D candidate in the Department of Computer Science and Engineering at the University of South Florida since September 2003. His research focuses on wireless networking. Mr. Zhang received the Bachelor of Science degree in July 2002 from the Department of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China. After his undergraduate degree, he worked as a researcher in the School of Computer Engineering, Nanyang Technological University (NTU), Singapore.