Calhoun
Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

2013-12

Tactical network load balancing in multi-gateway wireless sensor networks

White, Kevin A.

Monterey, California: Naval Postgraduate School

http://hdl.handle.net/10945/39036

Downloaded from NPS Archive: Calhoun

DUDLEY KNOX LIBRARY

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TACTICAL NETWORK LOAD BALANCING IN MULTI-GATEWAY WIRELESS SENSOR NETWORKS

by

Kevin A. White

December 2013

Thesis Advisor:	Preetha Thulasiraman
Second Reader:	Rachel Goshorn

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE TACTICAL NETWORK LOAD BALANCING IN MULTI-GATEWAY WIRELESS SENSOR NETWORKS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Kevin A. White	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. government. IRB protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

A tactical wireless sensor network (WSN) is a distributed network that facilitates wireless information gathering within a region of interest. For this reason, WSNs are finding increased use by the Department of Defense. A challenge in the deployment of WSNs is the limited battery power of each sensor node. This has a significant impact on the service life of the network. In order to improve the lifespan of the network, load balancing techniques using efficient routing mechanisms must be employed such that traffic is distributed between sensor nodes and gateway(s). In this thesis, we study load balancing from a cross-layer point of view, specifically considering energy efficiency. We investigate the impact of deploying single and multiple gateways on the following established energy aware load balancing routing techniques: direct routing, minimum transmission energy, low energy adaptive cluster head routing, and zone clustering. Based on the node die out statistics observed with these protocols, we develop a novel, energy efficient zone clustering algorithm called EZone. Via extensive simulations using MATLAB, we analyze the effectiveness of these algorithms on network performance for single and multiple gateway scenarios and show that the EZone algorithm maximizes network lifetime and service area coverage.

14. SUBJECT TERMS Wireless sensor network, load balancing, networking, WSN, mobile ad hoc network, ground sensor network, microsensor, Dijkstra, cluster routing, zone routing, data aggregation, single gateway, multi-gateway	**15. NUMBER OF PAGES** 247
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

TACTICAL NETWORK LOAD BALANCING IN MULTI-GATEWAY WIRELESS SENSOR NETWORKS

Kevin A. White
Lieutenant, United States Navy
B.S., California State Polytechnic University Pomona, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2013

Author:	Kevin A. White
Approved by:	Preetha Thulasiraman, PhD. Thesis Advisor
	Rachel Goshorn, PhD. Second Reader
	Clark Robertson, PhD Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

A tactical wireless sensor network (WSN) is a distributed network that facilitates wireless information gathering within a region of interest. For this reason, WSNs are finding increased use by the Department of Defense. A challenge in the deployment of WSNs is the limited battery power of each sensor node. This has a significant impact on the service life of the network. In order to improve the lifespan of the network, load balancing techniques using efficient routing mechanisms must be employed such that traffic is distributed between sensor nodes and gateway(s). In this thesis, we study load balancing from a cross-layer point of view, specifically considering energy efficiency. We investigate the impact of deploying single and multiple gateways on the following established energy aware load balancing routing techniques: direct routing, minimum transmission energy, low energy adaptive cluster head routing, and zone clustering. Based on the node die out statistics observed with these protocols, we develop a novel, energy efficient zone clustering algorithm called EZone. Via extensive simulations using MATLAB, we analyze the effectiveness of these algorithms on network performance for single and multiple gateway scenarios and show that the EZone algorithm maximizes network lifetime and service area coverage.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

3G	third generation mobil communication technology
ARPANET	Advanced Research Projects Agency Network
ASN	autonomous sensor network
CBR	constant bit rate
CDMA	code division multiple access
CH	cluster head
CMOS	complementary metal oxide semiconductor
Direct-S	single gateway direct transmission
Direct-M	multi-gateway direct transmission
DoD	Department of Defense
FOB	foreign operating base
GPRS	general packet radio service
GSM	Global System for Mobile Communications
HTTP	hypertext transfer protocol
IC	integrated chip
ICMP	internet control message protocol
ICMP4	internet control message protocol version 4
IETF	Internet Engineering Task Force
IPV4	internet protocol version 4
LEACH	low-energy adaptive clustering hierarchy
LEACH_S	single gateway LEACH routing protocol
LEACH_M	multi-gateway LEACH routing protocol
EZONE-S	single gateway zone routing protocol with energy efficient cluster head election
EZONE-M	multi-gateway zone routing protocol with energy efficient cluster head election
MAC	medium access control
MTE	minimum transmission energy
NFC	near field communication
OV1	operational view

PA	power amplifier
RFC	request for comment
RFID	radio frequency identification
ROV	remotely operated vehicle
RV	random variable
SNR	signal to noise ratio
UDP	user datagram protocol
UGS	unattended ground sensor
WSN	wireless sensor network
ZONE_S	single gateway zone routing protocol with random CH election
ZONE_M	multi-gateway zone routing protocol with random CH election

EXECUTIVE SUMMARY

A wireless sensor network (WSN) is a group of sensor nodes that are geographically distributed to provide data gathering and monitoring of tasks and events. Wireless sensor networks are increasing in popularity throughout society. This is a result of the fact that the integrated chip (IC) technology boom during the past 20+ years has miniaturized IC hardware while increasing computational capability. These WSNs can be used in a variety of applications such as atmospheric monitoring, human detection, video surveillance, or virtually any task that involves sensing and communicating information.

As a result of their ubiquitous inclusion in society, WSNs are finding increased applicability to the Department of Defense (DoD) in areas specific to surveillance and reconnaissance. A WSN can be used to remotely monitor a battlespace making the presence of a warfighter unnecessary thereby increasing the safety of our forces. A WSN can be used to remotely monitor deployed systems and trigger alerts at a command-and-control site when certain events occur.

Since WSNs obtain and communicate information individually, sensor nodes are inherently more complex. Each node must have the ability to simultaneously serve as a sensing device and a wireless communication device that can exchange information with nearby nodes. It is critical that information from every node is communicated to a desired destination outside the network. A typical WSN and its associated supporting infrastructure are shown in Figure 1. Individual sensor nodes capture information using the battery energy they are deployed with. Nodes then utilize their peers (if necessary) to pass this information wirelessly to the gateway and then through supporting infrastructure to a command-and-control site for further processing. In the case of Figure 1, data is passed using a minimum transmission energy (MTE) routing algorithm that uses neighboring nodes to establish a path that cumulatively reduces the energy consumption of the nodes.

The focus of this thesis is the deployment of tactical WSNs. Tactical WSNs are remotely deployed in potentially hostile areas with gateway nodes located on the outskirts

of these areas. A key challenge in the deployment of tactical WSNs is the limited battery power of each sensor node. This has a significant impact on the service life of the network. In order to improve the lifespan of the network, load balancing techniques using efficient routing mechanisms to achieve energy efficiency must be employed such that traffic is distributed between sensor nodes and gateway(s).

In order to solve the load balancing problem, it is important to first understand the layout of a networking system. Modern day networks abstract all the processes that take place between any two nodes and represent them in the form of layers. The general network layering construct is shown in Figure 2 and contains the following five layers labeled one through five, respectively: physical, medium access control (MAC), network, transport, and application layers. Generally, layers of one node only rely on information from the layer immediately above or below it, and the information from Layer *i* of *Node X* is only accessed from the same layer *i* of *node Y* (logical links).

In this thesis, we exploit the opportunity to explore a cross-layer solution for the load balancing problem. A cross-layering method does not restrict a layer from utilizing information only from the layer directly above or below it. Specifically, for load balancing and energy efficiency, we allow the network layer access to the physical layer for battery parameters and distance between nodes in performance of energy-efficient routing strategies. Allowing the network layer to access this information provides another level of control that can be incorporated into the network layer protocol. This level of control allows us to create a network layer protocol with two additional aspects of network layer energy efficiency: 1) creating routing paths that conserve transmission power, and 2) favoring those nodes with higher residual energy to perform high energy consumption tasks. In a second cross-layering implementation, we allow the networking layer to access the application layer to perform data aggregation. Performing data aggregation reduces the size of network data packets, which reduces the energy required to transmit each packet through the network.

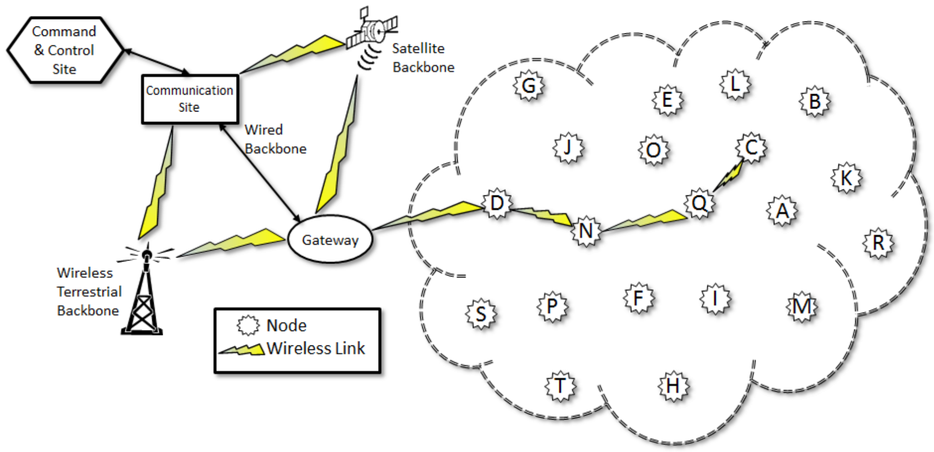


Figure 1. Operational view of a 20 node WSN with supporting infrastructure. Sensor C is shown transmitting its payload to the gateway through nodes Q, N, and D.

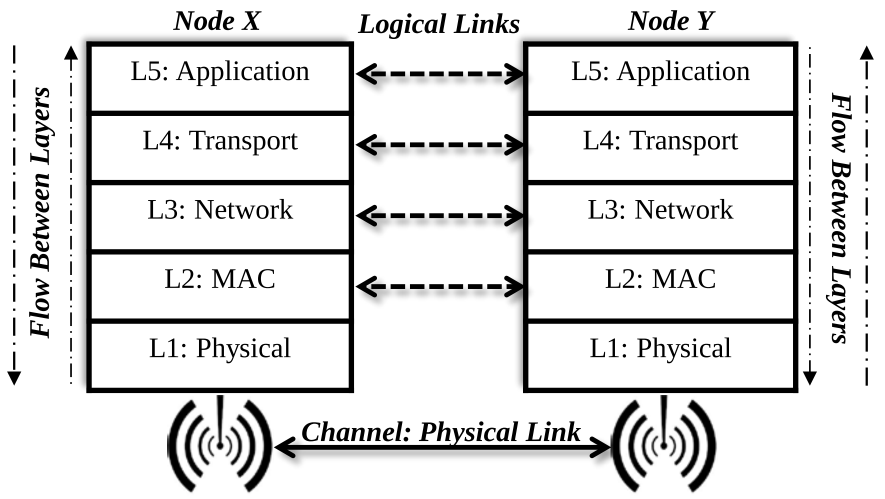


Figure 2. General network layering protocol stack between two communicating nodes. All information flows via the channel (physical layer). The information from layer *i* of node *X* is only accessed from layer *i* of node *Y* (logical links).

The protocols in place at each layer have a dramatic impact on the service life of the network and the coverage area. As node battery levels are depleted, they begin to die out. Thus, various design techniques are needed at each layer to achieve load balancing across the network.

In this thesis we survey the load balancing opportunities at each layer and use these opportunities to build a WSN network protocol stack that extends the service life of the network and controls the topology of live nodes. We control the topology of live nodes as the concentration of dead nodes increases such that we achieve uniform service coverage throughout the area of interest. In addition, the use of an additional gateway to optimize load balancing under the considerations just discussed was investigated in this thesis. As a result, the contributions of this thesis are as follows:

- Survey and identify load balancing techniques for WSNs. Design and simulate various network routing protocols and observe the impacts to the WSN.
- Simulate traditional networking routing protocols and identify performance improvements of adding an additional gateway.
- Develop a novel energy efficient WSN networking algorithm that uses a cross-layer approach and identify performance improvements compared to algorithms that do not consider energy efficiency. We refer to this algorithm as EZone.
- As sensor-node battery levels are depleted and nodes subsequently die out, show how the networking algorithm in operation affects the spatial distribution of alive nodes and dead nodes in the sensor field and how this affects the continuous service coverage throughout the sensor field.
- Show detailed energy statistics for a specific node-gateway(s) arrangement.
- Model network die out statistics as random variables to better characterize the distribution of the algorithm results over thousands of trial. This technique allows us to better substantiate the performance of classical network algorithms and our novel energy efficient algorithm.

As a result of our literature search on load balancing at each layer, we implemented the following models into each layer of the protocol stack. We build these models for both single and multi-gateway implementations of each routing algorithm analyzed:

- Physical layer: Our WSN is comprised of 100 uniformly distributed sensor nodes located in a 50 m x 50 m grid and the gateway (gateways for the multi-gateway simulation) is placed 100 m away from the grid. We utilize a first order power amplifier and sensor model. This model assigns an energy cost-per-bit to collect, transmit and receive information. It considers direct path and multi-path wireless signal propagation theory to identify the amount of information required to transmit one bit of information a certain distance between nodes while guaranteeing adequate signal-to-noise ratio at the receiving node. All our simulations assume that each node is within wireless transmission of the gateway that also means that each node is within communication range of any other node in the WSN.

- MAC layer: We implement a Time-Division Multiple Access (TDMA) scheme that assigns each node in the WSN a timeslot during which it transmits information to the gateway.
- Network layer: We implement several traditional and established routing algorithms observed in the literature. We also develop and implement our own energy efficient routing algorithm (EZone). The routing algorithms we implement are as follows:

  1. Direct: All nodes transmit their data message directly to the nearest gateway during each transmission round.
  2. Minimum Transmission Energy (MTE): All nodes transmit their data to the nearest gateway using a shortest-path MTE route that is calculated using Dijkstra's shortest path routing algorithm[1]. The link cost parameter input into Dijkstra's algorithm is the distance squared between nodes along the path.
  3. Low energy adaptive clustering hierarchy (LEACH) routing: LEACH routing elects a CH and nodes associate with the CH according to the LEACH algorithm [2]. Each node picks a random number between zero and one. Each node also computes a threshold number (*Tn*), which is a number between zero and one and is proportional to the current round. The probability for any node to serve as a CH is denoted as *p*. If a node has been a CH in the last 1/*p* rounds, it is excluded from being a CH during the round. Otherwise, if the temporary random number is less than *Tn*, the node is elected as a CH during the round. Nodes that were not elected as CHs during each round then associate in cluster with the nearest CH. Each node then transmits its data message to its CH The CH collects all the messages of its nodes and retransmits them collectively to the gateway. This process repeats during subsequent rounds until all nodes have died.
  4. Zone: Zone routing with random CH election partitions the network topology into zones. A CH is randomly assigned from the set of nodes in the zone to serve as the intermediate relay [3]. Our zones remain static throughout our simulations.
  5. EZone: Energy efficient zone routing is our novel routing algorithm designed in this thesis. EZone implements Zone routing as described in 4, except EZone elects the node with the most energy at the beginning of each round to be the CH.

- Transport layer: Our transport layer implements a strategy similar to the modern day internet's use of the User Datagram Protocol (UDP). User Datagram Protocol is a connectionless oriented protocol in which the source node gets no feedback that its messages reached the destination. This is applicable for WSNs as it provides a mechanism to prevent feedback transmissions that would unnecessarily deplete WSN energy levels.
- Application layer: Our application layer implements two strategies, 1) use of a traffic generator, and 2) use of a data aggregation technique. The traffic generator of each node generates a 2000 bit data message during each round for transmission to the gateway. Data aggregation is used only for the clustering algorithms and the CH is the only node that can perform data aggregation. The CH receives all the messages from nodes in the cluster. It then includes its own sensor's message, compresses all the messages into one 2000-bit message, and transmits the compressed message to the gateway at the end of each round.

The above layering strategy is implemented on the WSN shown in Figure 3, which is displayed with zone partitions. This zoning arrangement is only used for zone related algorithms; other algorithms make no use of the vertical zone partitions. Gateways are displayed as solid green, and nodes are represented by a blue outline circle. We show the single gateway scenario in Figure 3. The multi-gateway scenario is similar except an additional gateway is placed on top of the network topology at the position (25 m, 150 m). All nodes have a starting energy of 0.5 J, and gateway(s) are assumed to have unlimited energy (they are not energy constrained). We plotted the total WSN system energy level during each transmission round (Figure 4), number of live nodes during each round (Figure 5), and the energy variance that resulted from the distribution of individual node battery levels (Figure 6). We visually observed how nodes geographically die out throughout the simulation. In each legend of Figures 4–6, *S* after an algorithm name refers to the single gateway scenario and *M* refers to the multi-gateway scenario.

The clustering algorithms dramatically outperformed the MTE and direct routing algorithms as a result of rotating and distributing the high energy role of nodes performing a long-range transmission and allowing the CHs to perform data aggregation. The single and multi-gateway clustering algorithms generally displayed similar energy depletion rates that are illustrated in the linear regions of Figure 4. The clustering algorithms minimized the energy variance of the WSN, and our energy efficient zone routing algorithm provided an indistinguishable flat variance plot compared to other algorithms as shown in Figure 5. Our energy efficient zone routing algorithm (EZone) maximized the time when all nodes are alive with the single gateway simulation outperforming other multi-gateway algorithms. This is significant in that it reveals the efficiencies that can be gained by implementing an energy efficient cross-layer approach.

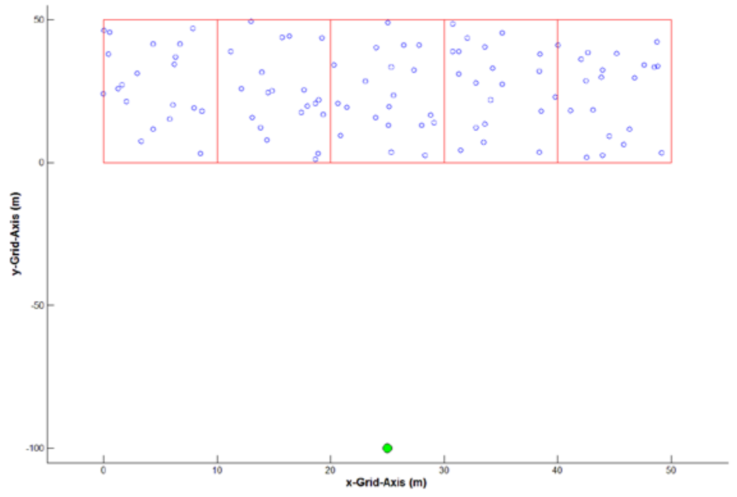


Figure 3. Single gateway network topology for simulations. The node grid is bound by the red perimeter; gateways are represented by a green circle. Vertical zones are displayed and only utilized for zone routing algorithms.

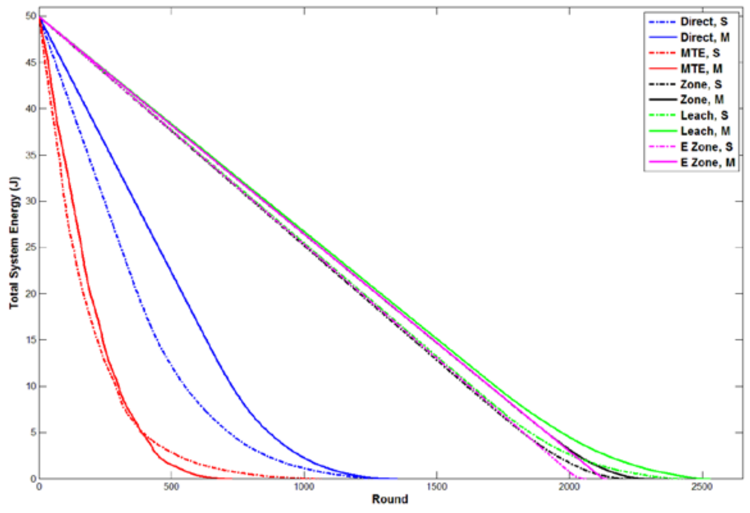


Figure 4. Total WSN system energy versus transmission round for all algorithms simulated.

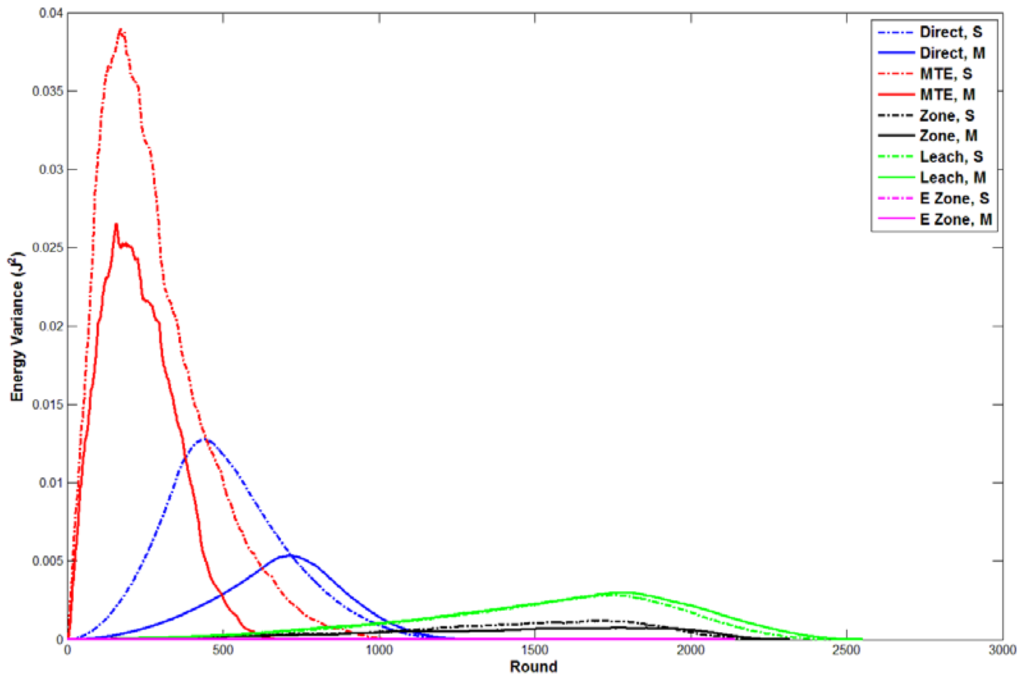


Figure 5. Energy variance versus transmission round for all algorithms simulated.

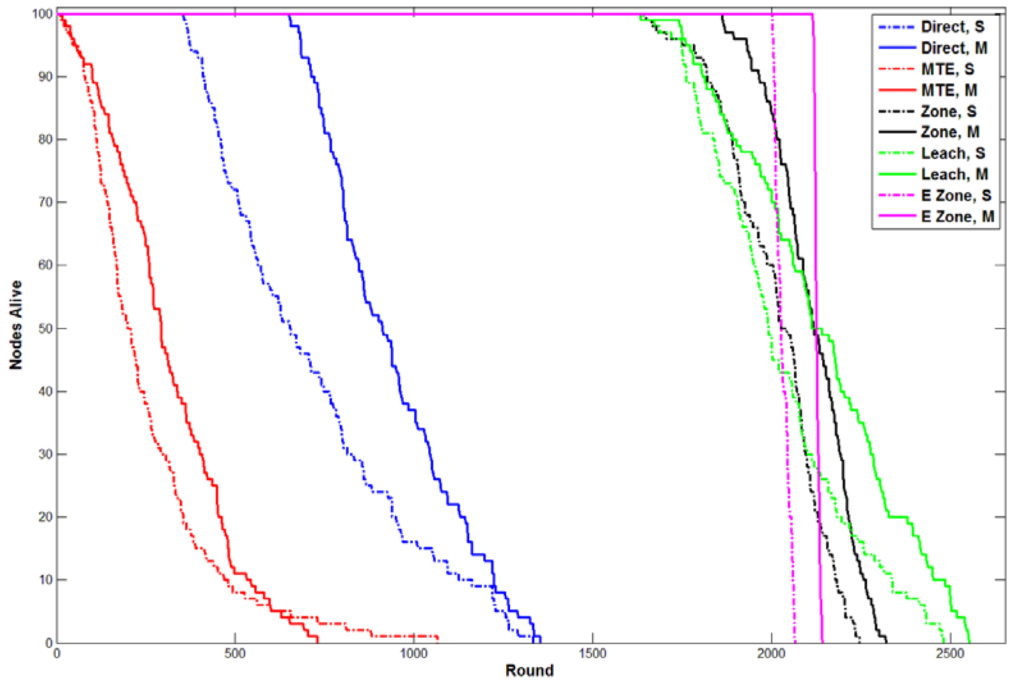


Figure 6. Total number of alive nodes versus transmission round for all algorithm simulated. Our energy efficient zone routing protocol provided the longest timeframe of 100 percent service area coverage.

Our energy efficient zone routing algorithm outperformed all other algorithms from a topology perspective during node die out as well. While other algorithms created a pattern for die out, our energy efficient algorithm caused nodes to quickly die out immediately after the first node died. This also is significant in that we utilized a cross-layer approach to maximize 100 percent service coverage of the WSN. Node die out of other networking algorithms occurred in an unfavorable fashion. For example, in the direct case, live nodes farther from the gateway died first since their energy is depleted proportional to their distance from the gateway. As a result, areas farthest from the gateway lost service first, while areas closest to the gateway remained in service longest. In MTE routing, the node closest to the gateway is always chosen to be included in the route. This node is known as the hot-node. Since the hot-node is the relay point between the gateway and all traffic from other nodes, it is overwhelmed with traffic during each round and dies quickly. Another hot-node is then immediately chosen. This hot-node concept in MTE routing causes nodes that are alive and that are closest to the gateway to die out first. The LEACH algorithm inefficiently creates clusters that cause the network to die out starting in the center of the sensor field and progressing radially outward. As a result of this die out mechanism, we lose coverage in the middle of the sensor field first. These die out mechanisms warrant the choice of our energy efficient zone routing algorithm for a tactical WSN since it preserves 100 percent network coverage the longest.

Detailed statistics for our simulations are provided in Table 1 and Table 2. Statistics of the rounds when specific quantities of nodes are alive are provided in Table 1. This is graphically shown in Figure 6. A comparison of the impact of the additional gateway is also shown in Table 1. The ratio of time the network is depleted from 100 percent to 20 percent (80 percent die out range), and the timeframe that the network provides100 percent service coverage (the round the first node dies) is shown in Table 1.

Instead of solely basing our conclusion on the network arrangement of Figure 3, we allowed the algorithms to run for thousands of times, regenerating a uniformly distributed sensor field to capture the distributions of the statistics we presented. This data is presented in Table 2 and graphically in Figure 7. All algorithms were executed for 5,000 trials except for the MTE algorithms that were executed for 1,000 trials (Dijkstra's algorithm is computationally intensive, thus we limit the simulation to 1,000 trials). The conclusions we made about the performance of the various algorithms are supported by the results shown in Table 2. On average, there is no performance advantage of adding the additional gateway in the LEACH algorithm also noted in Table 2. There is little or no gain in service life statistics by adding the second gateway to the LEACH algorithm.

An energy efficient routing strategy offers quantifiable gains to the service life of tactical WSNs. It balances the use of individual battery levels at the node level to maximize the time when all nodes are fully capable. The techniques investigated in this thesis show the importance of a broader topic, that of load balancing in WSNs, and that design creativity can have significant impacts on achieving a lasting capability of WSN performance.

Table 1. Network statistics that result from the WSN in Figure 3.

Protocol	Metric	Single Gateway	Multi Gateway	% Increase
Direct	Round First Dead	356	652	83
	10% Nodes Dead	410	712	74
	50% Nodes Dead	652	911	40
	80% Nodes Dead	939	1128	20
	Energy Depletion Rate (J^2)	0.0798	0.0554	−31
	80% Dieout Range (rounds)	583	476	−18
	80% Dieout range/Round First Dead	1.6376	0.7301	−55
MTE	Round First Dead	11	17	55
	10% Nodes Dead	77	100	30
	50% Nodes Dead	199	293	47
	80% Nodes Dead	354	453	28
	Energy Depletion Rate (J^2)	0.2140	0.1418	−34
	80% Dieout Range (rounds)	343	436	27
	80% Dieout range/Round First Dead	31.1818	25.6471	−18
LEACH	Round First Dead	1642	1633	-1
	10% Nodes Dead	1760	1805	3
	50% Nodes Dead	1990	2112	6
	80% Nodes Dead	2182	2327	7
	Energy Depletion Rate (J^2)	0.0245	0.0232	−5
	80% Dieout Range (rounds)	540	694	29
	80% Dieout range/Round First Dead	0.3289	0.4250	29

Protocol	Metric	Single Gateway	Multi Gateway	% Increase
Zone	Round First Dead	1649	1862	13
	10% Nodes Dead	1821	1964	8
	50% Nodes Dead	2022	2117	5
	80% Nodes Dead	2140	2215	4
	Energy Depletion Rate (J^2)	0.0248	0.0235	−5
	80% Dieout Range (rounds)	491	353	−28
	80% Dieout range/Round First Dead	0.2978	0.1896	−36
EZone	Round First Dead	2003	2116	6
	10% Nodes Dead	2007	2119	6
	50% Nodes Dead	2026	2126	5
	80% Nodes Dead	2051	2134	4
	Energy Depletion Rate (J^2)	0.0246	0.0235	−4
	80% Dieout Range (rounds)	48	18	−63
	80% Dieout range/Round First Dead	0.0240	0.0085	−65

Table 2. Detailed statistics of random variable testing. All data is mean data averaged from 5,000 trials except MTE data is a result of 1,000 trials.

Protocol	Metric	Single Gateway	Multi Gateway	% Increase
Direct	Round First Dead	350	660	89
	10% Nodes Dead	395	714	81
	50% Nodes Dead	664	942	42
	80% Nodes Dead	1004	1163	16
	80% Dieout Range (rounds)	654	503	−23
	80% Dieout range/Round First Dead	1.87	0.76	−59
MTE	Round First Dead	12	16	33
	10% Nodes Dead	73	97	33
	50% Nodes Dead	202	289	43
	80% Nodes Dead	351	472	34
	80% Dieout Range (rounds)	339	456	35
	80% Dieout range/Round First Dead	28.25	28.50	1
LEACH	Round First Dead	1840	1844	0
	10% Nodes Dead	1987	1995	0
	50% Nodes Dead	2294	2319	1
	80% Nodes Dead	2523	2565	2
	80% Dieout Range (rounds)	683	721	6
	80% Dieout range/Round First Dead	0.37	0.39	5
Zone	Round First Dead	1566	1841	18
	10% Nodes Dead	1777	1976	11
	50% Nodes Dead	2031	2122	4
	80% Nodes Dead	2151	2210	3
	80% Dieout Range (rounds)	585	369	−37
	80% Dieout range/Round First Dead	0.37	0.20	−46
Ezone	Round First Dead	1936	2070	7
	10% Nodes Dead	1944	2076	7
	50% Nodes Dead	2035	2132	5
	80% Nodes Dead	2083	2157	4
	80% Dieout Range (rounds)	147	87	−41
	80% Dieout range/Round First Dead	0.08	0.04	−45

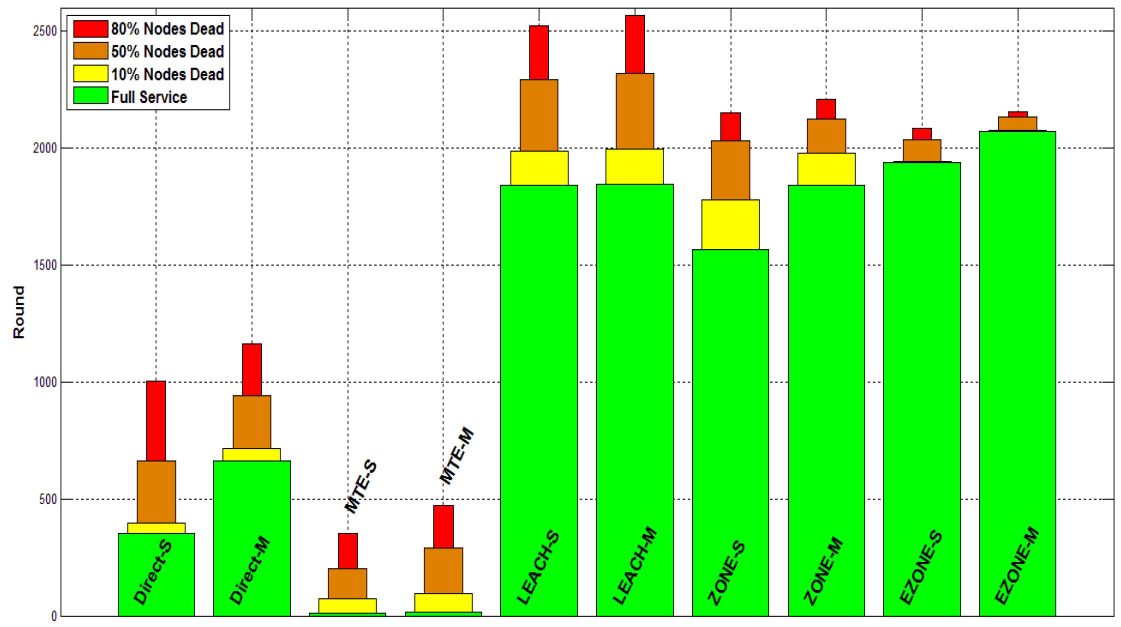


Figure 7. Summary of WSN availability for each algorithm in single and multi-gateway scenarios. Our energy efficient zone routing protocol provided the largest number of transmission rounds with full network service coverage.

LIST OF REFERENCES

[1] W. Stallings, *Data and Computer Communications*, 9th ed., Upper Saddle River, NJ: Prentice Hall, 2011.

[2] W. R. Heinzelman et al., "Energy-efficient communication protocol for wireless microsensor networks," *System Sciences, Proceedings of the 33rd Annual Hawaii International Conference*, Maui, HI, 2000, pp. 1–10.

[3] S. K. Singh et al., "Energy Efficient Homogenous Clustering Algorithm for Wireles Sensor Networks," *International Journal of Wireless Mobile Networks*, vol. 2, no. 3, pp. 49–61, 2010.

ACKNOWLEDGMENTS

I would like to thank the Navy for the time and opportunity to engage in such higher level learning and perform this research during my time at the Naval Postgraduate School. My gratitude and appreciation goes out to my professors and advisors for their countless hours of mentoring and assistance in shaping my education and this research.

Above all, I would like to thank my family whom endured an unthinkable number of hours as I mentally dedicated myself and time to this research. Your support and devotion ensured my success.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION: A BRIEF OVERVIEW AND SURVEY OF THE UTILITY OF WIRELESS SENSOR NETWORKS

Computer networking principles have changed the landscape of global communications since the original Advanced Research Projects Agency Network (ARPANET) was established in the late 1960s. Networking theory, coupled with substantial improvements in integrated chip (IC) hardware, has enabled vast new technologies and capabilities over the past 20+ years. In this thesis, we study one of the significant capabilities enabled by networking and IC improvements: wireless sensor networking (WSN).

Modern warfare is transitioning to a status quo largely occupied by unmanned systems with ubiquitous situational awareness and means to communicate. Our forever-improving technology facilitates real-time knowledge of the environment. This can be readily provided by WSNs. Wireless sensor networks can provide information wirelessly to the warfighter so that other warfighters do not have to. This idea is significant because it ultimately improves the safety of warfighters in areas too hostile or too extreme to provide reliable human data collection.

Our investigation into WSNs is from a tactical perspective in that the networks are remotely deployed. The network must operate reliably and maximize sensor network coverage for the maximum amount of time in the absence of human contact. A key challenge in the deployment of WSNs is the limited battery power of each sensor node. This has a significant impact on the service life of the network. In order to improve the lifespan of the network, load balancing techniques using efficient routing mechanisms must be employed such that traffic is adequately distributed between sensor nodes and the gateway node (relay node to WSN backbone infrastructure).

In this thesis, we show that various traditional routing algorithms have a negative impact on the service life of a WSN because their design does not take into account energy efficient strategies required to extend the life of the WSN. Realistic insights on how to incorporate an energy efficient strategy into a routing algorithm to maximize service life is provided in this thesis. We simulate several classic load balancing routing algorithms for WSNs and introduce our own novel energy efficient routing algorithm that reveals a performance improvement as a result of our energy optimization. We further consider and obtain performance improvements by tactically including an additional gateway in our simulations. Whereas the research community traditionally only models one specific node-gateway network arrangement, we model one specific node-gateway arrangement and extend signal processing methodologies to model WSN die out parameters as random variables. This allows us to generate thousands of data points and draw our conclusions on an expansive subset of data instead of just one trial.

The contributions of this thesis are summarized as follows:

- Survey and identify existing load balancing techniques for WSNs that use efficient routing algorithms. Load balancing is described in detail in Chapter II but generally consists of techniques to extend network service lifetime.
- Simulate traditional network routing algorithms using MATLAB and identify performance improvements of adding an additional gateway. The traditional network routing algorithms studied are:
    - Direct routing: An algorithm in which each node transmits its payload directly to the gateway.
    - Minimum transmission energy (MTE): Each node transmits its payload to its nearest neighbor in a path to the gateway such that the energy consumption along the path is reduced. MTE uses Dijkstra’s classic shortest path routing algorithm to construct the paths.
    - Low energy adaptive clustering hierarchy (LEACH): A popular network clustering algorithm specific to WSNs [1] [2].
    - Zone: A zone-based network clustering algorithm that partitions the network into zones and hierarchically determines node to gateway routes using a cluster-head (CH) assigned in each zone.
- Develop a novel energy efficient WSN networking algorithm, called EZone, using a cross-layer approach and identify performance improvements compared to algorithms that do not consider energy efficiency. EZone partitions the sensor field tactically and assigns a CH in each zone using an energy efficient strategy. The CH serves as an intermediate relay for node-gateway routing in each zone.
- As sensor-node battery levels are depleted and nodes subsequently die out, we show by simulation how the load balancing algorithm affects the

spatial distribution of live nodes and dead nodes in the sensor field and how this affects the continuous service coverage throughout the sensor field.

- Show detailed energy statistics for specific node-gateway(s) arrangement.
- Model network die out statistics as random variables to better characterize the distribution of the algorithms' results over thousands of trials. This technique allows us to better validate the performance of classical network routing algorithms in comparison to our novel energy efficient algorithm (EZone).

A. WIRELESS SENSOR NETWORKS: A BRIEF INTRODUCTION

Terms such as wireless sensor network (WSN), unmanned ground sensor network (UGS), autonomous sensor network (ASN), and ad hoc sensor network all refer to a collective wirelessly inter-connected network of nodes. In this thesis, we refer to this network topology as a WSN. A WSN is a collection of nodes distributed over a geographic area to perform some type of monitoring or data-gathering task [1]. In this thesis, tactical WSNs that are autonomous are considered; that is, they must operate in the absence of physical human interaction, yet their information must traverse the network reliably and efficiently to a desired destination.

An operational view (OV1) of a 20-node WSN along with its significant supporting infrastructure is shown in Figure 1. The WSN is assumed to be deployed in a remote geographic location. Each node must rely solely upon the battery energy it was deployed with and use wireless transmissions to communicate with other nodes within communication range or to the gateway. The supporting infrastructure primarily consists of a backbone, which performs the long-haul communications between the gateway, and a communication site, which is connected to the command and control node. Three possible backbones are displayed in Figure 1: a satellite wireless link, a terrestrial wireless link, and a wired backbone. The type of backbone employed by a WSN is variable depending on how and where the WSN is deployed. For example, an organization desires to deploy a WSN to study CO2 levels in a remote jungle. They intend to deploy their WSN by scattering sensors from an airplane and dropping a gateway into the sensor field. In this example, it would be practical for the gateway to communicate via a commercial satellite since a wireless communication tower in the area is not likely nor would it be practical to run a wired backbone through the jungle. Conversely, in another example a nuclear power-plant design company desires to outfit a nuclear power plant with internetworked sensors that monitor water temperature inside the nuclear reactor. In this case, a wired backbone is optimal to interconnect the sensors to minimize circumstances of an unreliable wireless connection and possible wireless interference.

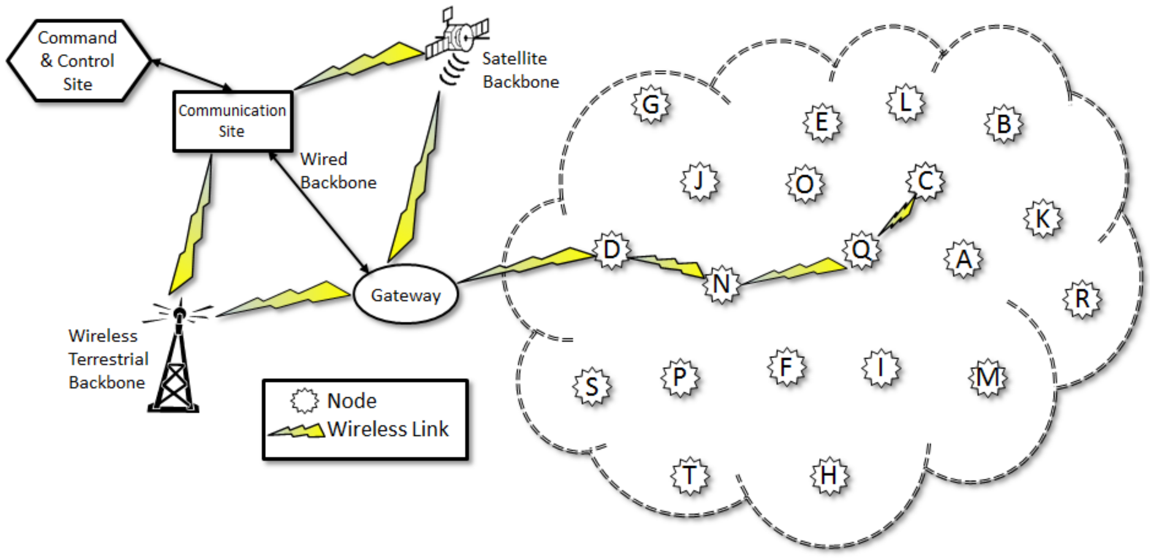


Figure 1. Operational view (OV1) or a 20 node WSN with supporting infrastructure and shown with sensor C transmitting its payload to the gateway through nodes Q, N, and D.

The scenario presented in Figure 1 is that of node C communicating its payload with the gateway such that the routing algorithm in place utilizes the path C → Q → N → D → gateway. This route was decided upon by the protocol implemented at the network layer of each node. We describe our layering construct in Chapter II. It is important to note that the route chosen here may not be the most beneficial route for the WSN as a whole. In particular, the route chosen here was that of a minimum transmission energy (MTE) path, in which the routing algorithm determined that the route along C → Q → N → D → gateway minimizes the total wireless transmission energy and propagation distance along the path. Since a wireless device is only constrained by its transmission radius at any time, there are other creative ways for a node to determine the route its payload takes to the gateway. We discuss the specifics of this further in Chapter IV.

In the construct of our WSN, each node only cares that its information is routed to the gateway (i.e., the destination for a source node's payload is always the gateway). While significant infrastructure and algorithms are required to communicate information beyond the gateway to a command and control site, this is beyond the scope of this thesis. For the purposes of the research presented in this thesis, we are not concerned with communication beyond the gateway.

Every node in the WSN is simultaneously serving as a sensing/communication device and as a router. This can be both an advantage and disadvantage. The advantage is that wireless nodes have a large number of possible communication paths at their disposal for their payload to be routed through the network. Conversely, a disadvantage is that if all nodes previously within communication range are energy depleted (dead nodes) and the gateway is farther than the node's wireless transmission range, the node can no longer communicate with the gateway. Thus, its remaining energy is effectively wasted. In this state, a useful node is partitioned from the network and can no longer serve its purpose.

To understand the framework for our study and simulations, it is worthwhile to discuss the similarities and differences between nodes and gateways.

1. Nodes

Each node is made up of at least four basic subsystems, which are shown in Figure 2 [3]. The four subsystems are power, sensing, processing, and communication subsystems. The power system provides energy for the node to perform its function and may obtain its energy from a number of different sources including battery, solar or hard-wired (i.e., plugged into another source). The most common power source for deployable WSN nodes are batteries.

The sensing subsystem may include any sensor (or sensors) desired for the system. Common sensor examples include atmospheric monitoring (temperature, pressure, humidity), sound, imagery, motion detection, or more elaborate sensors such as a mechanism to detect presence of a particular signal (cellular, short-wave radio). The sensing subsystem receives power from the power subsystem and provides the sensed data to the processing subsystem so it can be packetized for transmission.

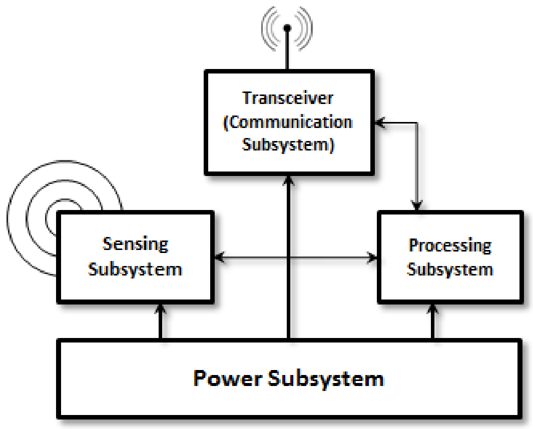


Figure 2. The basic architecture of a WSN node consists of four subsystems including power, sensing, processing, and communication subsystems.

The processing subsystem utilizes a microprocessor to handle all the processing needs of the node. The processing needs of the node are those involved in packetizing data received from the sensor and performing wireless networking functions for the node. The processor utilizes a small software operating system to provide minimal processing capability that utilizes the smallest amount of energy feasible but also provides enough computational capability to satisfy the node's technical requirements. One such example is TinyOS, which is an open source operating system engineered for low-power wireless devices [4].

The communication subsystem includes a transceiver, which includes both a transmitter and receiver to handle send and receive operations. The transceiver enables several modes of operation designed to minimize the energy draw of the node. Specifically, the transceiver may vary the transmission power of a particular signal. We discuss several more energy load balancing techniques later in this thesis.

2. Gateway

The gateway serves as a collection point for all the information that is coming out of the WSN. From the perspective of individual nodes, the gateway represents the destination for a node’s packets. The gateway is not limited to being located at the periphery of the sensor network as we have shown in Figure 1. The gateway could also be located somewhere within the sensor grid. The gateway differs from traditional WSN nodes in two ways: 1) The gateway is *less* energy constrained than other nodes in the network, and 2) the gateway provides the link to the outside of the network for all nodes inside the network. A gateway that is *less* energy constrained refers to the notion that it is not limited by energy. The gateway has enough energy to handle all the communications from the network until the last node in the WSN dies. This means it has a battery energy level a few magnitudes greater than a normal node, or it has an external source of power. We use this assumption to preclude the possibility that the gateway is the limiting factor in the network.

The gateway is of such significance to the network that its location must be considered. Without the gateway, the network is useless. As a result, we focus our attention toward gateway locations on the periphery of the sensor field. Since our research is focused toward tactical WSNs, we assume that a location on the periphery is more likely to be a safe zone compared to that where the sensor nodes are deployed. Our use of safe zone refers to a location where the gateway is outside normal environmental and physical constraints to which sensor nodes may be subjected. The gateway is not a sacrificial device as the sensor nodes are. In this thesis we do not consider nor determine the optimal placement of the gateway on the periphery. We place the gateway(s) on the periphery and use the same location(s) for all simulations contained in Chapters V and VI.

We investigate two types of WSNs: 1) a single gateway, which was shown in Figure 1, and 2) a multi-gateway scenario. The majority of existing research in WSNs generally includes the perspective of a single gateway [1] [2] [5] [6][1]–[9]. Thus, it is important to extend WSN concepts to a multi-gateway framework and identify the resulting performance improvements by including an additional gateway.

B. MULTI-GATEWAY WIRELESS SENSOR NETWORKS

A multi-gateway WSN performs the same functions as the single gateway case, except each sensor now must choose which gateway to send its payload to at any given time. The multi-gateway version of Figure 1 is shown in Figure 3 and two gateways to support the WSN are shown. While more gateways can be used, we limit our research to two gateways, which is practical from a cost and logistics perspective since more gateways are expensive to deploy and maintain. We assume that the WSN in Figure 3 is a tactical network, thus the gateways are located on the periphery in a safe zone.

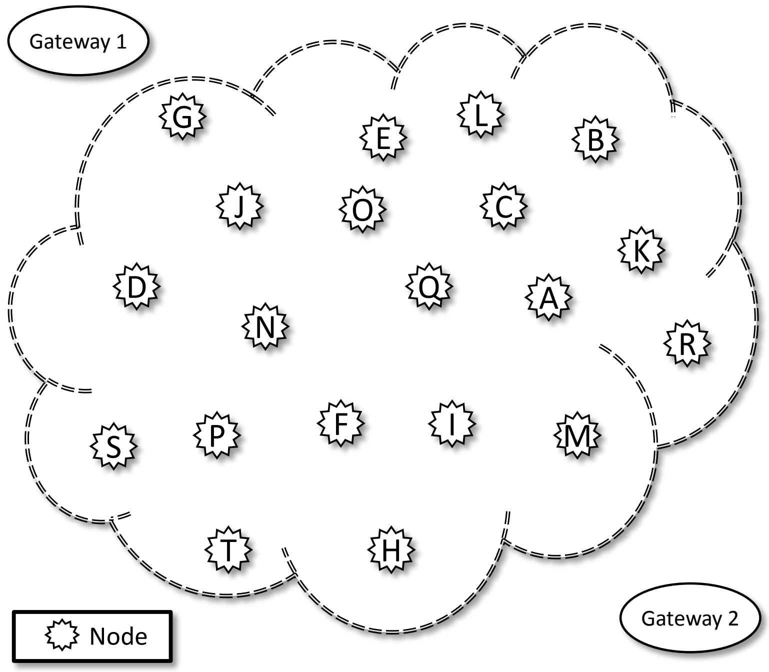


Figure 3. Multi-gateway WSN version of Figure 1.

C. COMMERCIALLY AVAILABLE SYSTEMS

There are several commercially available solutions for WSNs. One such example is the Waspmote from Libelium, which utilizes an ultra-low power housing that provides options for 60 available plug-and play sensors [10]. The Waspmote provides eight physical radio technologies ranging from long range (third generation mobile communication technology, 3G, and general packet radio service, GPRS) to short range (radio frequency identification, RFID, near field communication, NFC and Bluetooth) communication capabilities. A picture of the Waspmote is shown in Figure 4 and is small enough to fit in the palm of the hand.

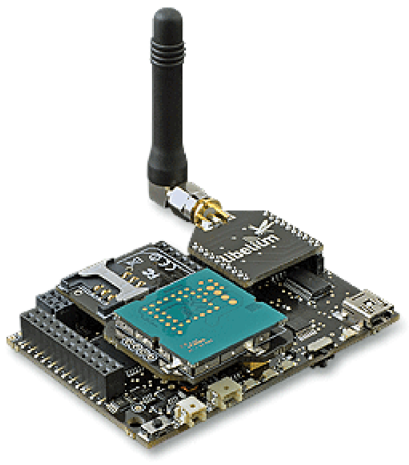


Figure 4. The multi-configurable Waspmote wireless sensor node from Libelium (from [10]).

National Instruments produces WSN starter kits that includes a package of nodes with optional plug and play sensors and a gateway that can be connected to a personal computer to configure the WSN. Each node operates on four AA batteries, which can provide power for up to three years [11]. Their kit is displayed in Figure 5.

Figure 5. National Instruments WSN starter kit consisting of nodes, a gateway and network configuration utility software (from [11]).

D. APPLICATIONS OF WIRELESS SENSOR NETWORKS

Examples of WSNs are common place in society. We present a few examples to demonstrate their utility and show how WSNs can influence our daily activities.

1. Radiation Detection Levels in Fukushima

Wireless sensor networks play a vital role in tracking radiation levels at and around the tragic Fukushima Daiichi reactors as a result of the Japanese tsunami that occurred on March 11, 2011 [12]. The system employs Geiger counters attached to a control circuit board. The Geiger counters measure the number of counts per minute, which represents the amount of radiation in the area, at which point the information is transmitted to a control point (gateway) so it can be monitored by a command and control network. The system is powered with internal batteries and claims an operable lifetime on the order of years of available service [12].

2. Environmental Parameter Monitoring for Smart Cities

Smart cities of the future will employ WSNs to provide real time information to any citizen. Specifically, the digital smart city of Santander, Spain is a test bed for smart city design and implementation. The Santander test bed is comprised of approximately three thousand 802.15.4 devices (Zigbee protocol with medium transmission range capability), two thousand general packet radio service (GPRS) modules and two thousand joint radio frequency identification (RFID) tag labels fixed at street lights, bus stops, and other fixed locations. These devices provide real-time information regarding traffic intensity monitoring, guidance to parking lots with available spaces, mobile environmental monitoring, and control of park and garden irrigation systems [13], [14].

3. Agricultural Monitoring

Wireless sensor nodes can be imbedded near crops and in the soil at farms to monitor conditions such as temperature, humidity, leaf wetness and many other parameters. One such project at a vineyard in Galicia, Spain employs WSNs to predict the onset of a plague, thereby allowing personnel to take action to minimize the plague [15]. Another example uses a WSN incorporating image processing to detect when a vineyard has any type of deficiency to its grapes or leaves [16].

By continuously monitoring parameters of crops, one can better tailor the environmental parameters (soil wetness) in zones that are most favorable to the crop, which can increase the quality of the crop yield. A field test at a Pickberry Vineyard in Sonoma, California covered a 30 acre site by sensors measuring humidity, temperature, soil humidity and soil temperature to control the irrigation system. Using this crop management, they were able to lower their operating costs and increase the vineyard's grape yield [17].

E. APPLICABILITY OF WIRELESS SENSOR NETWORKS TO DEPARTMENT OF DEFENSE

1. Remote Monitoring and Surveillance

Remote monitoring and surveillance by way of WSNs offers the most utility to the Department of Defense (DoD) as many government program offices are engaged in this arena. Video surveillance requires that millions of pixels of information be transmitted at every instance of time, which requires optimized WSN protocols that enable maximum WSN service duration. Wireless sensor networks can be placed in battle as well as in training scenarios to provide useful feedback to commanders. For example, desert training sites could be instrumented with smart sensors similar to the smart city previously discussed to monitor critical training parameters such as atmospheric conditions or more elaborate monitoring such as troop movements, noise levels due to explosions or munitions use and to track area activity. Weaponry could be instrumented to track ammunition use that could feed back to logistical systems to minimize the overhead of and improve efficiency of ammunition availability. Individual soldiers could be instrumented with micro-sensors to track individual fatigue and readiness over time while in battle. This information might not be on demand, but each sensor could transmit its storage payload when it is within range of a gateway, a process that would occur periodically through the performance of daily activity.

Wireless sensor networks can offer remote monitoring of military bases instead of having to rely heavily on personnel to physically patrol a perimeter. The data provided by a WSN can be fused together at a base command and control node to minimize perimeter lookouts, which also minimizes their safety risk. Wireless sensor networks can be deployed to foreign operating bases (FOB) or remote outposts setup by small forces occupying imminently hostile areas to increase the situational awareness of their surroundings and provide better warning in the event of an attack.

2. Vehicular Networks

Wireless sensor networks can be incorporated in military vehicular networks to provide situational awareness. Force movement by individual groups of vehicles can be tracked. When a tracking sensor is within range of a gateway, its position is logged and sent to a database. Over time, such tracking can provide means to balance out force distribution in an area, optimize vehicle and personnel movements, and provide a way to track anyone or anything containing a tracking sensor in an automated fashion.

3. Remotely Operated Vehicles

Wireless sensor networks can provide an inexpensive way to operate remotely operated vehicles (ROVs). As we previously mentioned, since sensor nodes operate as both sensors and routers, an ROV could be controlled by a WSN with minimal infrastructure with sensor nodes serving as relays in remote areas to allow communication with the ROV. The ROV could also be designed to deploy the WSN as it travels to its mission. This could offer tremendous flexibility and range. For example, [18] predicts a trillion sensor world by 2020. In order to achieve the maximum life of sensors under the presence of data throughput requirements, custom and specific protocols must be implemented to achieve a networked energy efficiency.

F. THESIS ORGANIZATION

The remainder of this thesis is organized as follows. The basic layering structure employed by networks and the concept of load balancing is discussed in Chapter II. The models we employ for our physical and medium access layers in our simulations are described in Chapter III. We use mathematical principles of signal propagation theory to show that there are tradeoffs based on physical layer technology that impact the design of the load balancing algorithm. In Chapter IV, we describe the algorithms we implement in the networking layer, which include: direct, MTE, LEACH, Zone and EZone. The simulations and our analysis of the results for one WSN node arrangement are provided in Chapter V. We demonstrate by simulation how load balancing techniques and routing algorithms impact the service life of the WSN, as well as how sensor nodes in the network die out over time. In Chapter V, we focuse on one specific network arrangement. In Chapter VI, the algorithms implemented in Chapter V are executed many times to model network die out parameters as statistical random variables (RV). We then obtain the distribution of these RVs and make observations of the characteristics of each algorithm. We conclude the thesis in Chapter VII and propose topics for future work. The authors biography is included as Appendix A, and all code for the algorithms implemented in this thesis are provided in Appendix B. The appendices also provide insight on how we chose our simulation platform and many of our coding strategies.

G. CHAPTER I SUMMARY

In this chapter, we provided an introduction and overview of WSNs introducing the concepts, contributions, and motivations for this thesis. We briefly looked at the topology and architecture of a typical WSN and the basic process of a node communicating its payload to a gateway. We described the basic components of a WSN and the differences between nodes and gateways. Single gateway versus multi-gateway

WSNs were discussed, which was followed by a brief survey of WSN technologies and capabilities, current WSNs in use, and their utility to DoD.

II. METHODOLOGY FOR LOAD BALANCING IN WSNS

In this chapter we introduce the problem of load balancing in WSNs. Load balancing generally requires optimization of protocols at different layers of the network stack. We describe the layering construct for wireless devices as this is the basis for our models in subsequent chapters. Various load balancing techniques for each layer are presented based on our background research. We conclude the chapter by providing the framework for the load balancing strategies we implement later for single gateway and multi-gateway WSNs.

A. WSN LOAD BALANCING

One substantial problem regarding tactical WSNs is that they are severely energy constrained. Nodes must rely on little energy storage for potentially years of service (i.e., the National Instrument nodes presented in Chapter I lasts up to three years using AA batteries). Improvements and miniaturization of semiconductor chip devices has outpaced the ability to miniaturize equivalent batteries to service those devices. As we previously stated, a node is not useful to its peers when its energy is depleted. As a result, the designer of the protocols and algorithms used by the node must implement a design that extends the nodes' service to subsequently extend the service life of the WSN. We define WSN load balancing, as a group of energy management techniques implemented at each layer with the intent of minimizing the energy depletion rate of the node and that of the WSN.

Load balancing must be considered to achieve optimal performance. We consider optimal performance of the WSN to be when all nodes function for the maximum amount of time. Since nodes rely on their peers to pass information to a gateway, if a peer prematurely dies that wireless communication route is no longer available. If that communication route was the last viable route in the network, then the network becomes partitioned. Nodes with available energy no longer have a communication path to the gateway, which creates suboptimal network performance. In order to understand where load balancing opportunities exist, it is pertinent to present a generic version of the layering construct for networked devices.

B. NETWORK LAYERING CONSTRUCT

The networking layer construct is a conceptual model that describes the internal operation of a node by separating activities into layers of abstraction. The generic network layering construct is shown in Figure 6 for wireless communication between two nodes. Each node, node *X* and node *Y*, contain a stack of layers including physical, medium access control (MAC), network, transport, and application layers numbered L1 to L5, respectfully. The dashed lines represent logical links in that each layer of Node *Y* will only interface with information corresponding to the same layer as node *X*. Information can only flow up or down a device's layer stack. A layer relies upon information only from layers immediately above or below the layer. There is an opportunity to break this restriction, known as a cross-layer approach, which we present later in this thesis.

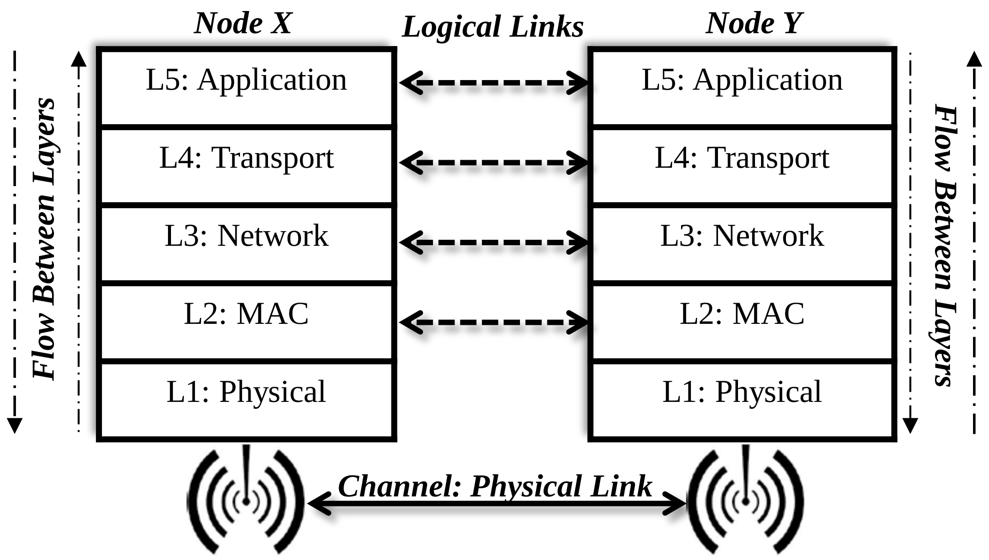


Figure 6. Generic network layering construct between two nodes.

Briefly describing the layers of Figure 6, the physical link (solid line) is the process of node *X* passing data to node *Y* through a wireless medium, also known as the channel. A node's physical layer uses energy (battery power) to transmit and receive information according to the MAC layer. The MAC layer provides a control mechanism for when and how the physical layer accesses the channel, such that many wireless devices have the opportunity to use the same channel in a controlled fashion. The network layer is responsible for deciding which node the packet is sent to next. In our example from Figure 1, each node along the route C → Q → N → D → gateway needs to decide the next node to forward information to since there are potentially many choices (nodes) within wireless range that could be used. The transport layer provides end-to-end communication services in that it keeps track of information coming back and forth to ensure all data is received. The application layer provides services for an application program to communicate with the stack. For example, internet browsing uses an application protocol hypertext transfer protocol (HTTP) to connect the internet browser (the application) to the layering stack for the internet browsing experience.

C. CROSS-LAYER DESIGN

In describing the traditional implementation of the layering construct, information only flows up or down a device's layer stack; a layer only relies upon information from layers immediately above or below. The research presented in this thesis is focused on energy efficiency (physical layer) and routing (network layer). Thus, the network layer must have knowledge of the physical parameters (battery level). A layer relying on information from another layer that is not immediately above or below is known as A cross-layer design. We allow the network layer to directly obtain battery energy level or physical distance between nodes to permit routes be calculated with energy efficiency in mind. From this perspective, the energy aware strategies employed in this thesis use a cross-layer design.

D. LOAD BALANCING OPPORTUNITIES AT EACH LAYER

Load balancing opportunities are based on engineering acumen as well as design creativity. They are present at each layer and can be specific to a capability. We performed an extensive literature search to identify load balancing techniques at different layers of the network stack that could be of use to WSNs.

1. Physical Layer Load Balancing

Physical layer load balancing can be categorized into two groups: software and hardware. The hardware in use has a significant effect on the life of the node. If the transceiver is not being used efficiently when transmitting and receiving, unnecessary power is consumed. An example of an inefficient use of the transceiver is transmitting at a higher power than needed, which could be a result of inefficient implementation of the transceiver. Efficient use of a transceiver is well documented in the literature. For example, a personal cell phone utilizing the Global System for Mobile Communications (GSM) uses the minimum power required for the signal to reach the cell tower. The base station (gateway) can increase the power of the mobile device in increments of 2.0 dBm up to a maximum threshold level [19].

Improvements in transceiver design and threshold has led to low power terminals for WSN applications. For example, a transceiver design where the received signal is amplified and filtered at baseband frequencies instead of some other high intermediate frequency results in a lower current drain in the transceiver’s amplifiers and filters is implemented in [20]. An energy efficient complementary metal oxide semiconductor (CMOS) power amplifier (PA) for WSN nodes that uses high quality bond wire inductors, capacitance compensation technology, and chock inductors to enhance the efficiency and linearity of the PA produced improvements in power efficiency was introduced in [21].

Antenna design also influences WSN power consumption, thus recent work has focused on the design of antennas specific to WSN nodes. A smart, secure, power-efficient, beam-steerable transmission system for WSNs was devised in [22]. The system is programmed to adjust antenna characteristics to transmit a signal in a desired direction such that no signal is sent in many undesired directions.

2. MAC Layer Load Balancing

Medium access control layer load balancing consists of contention-based and contention-free protocols. Contention-based protocols require that when a device is receiving data, transmissions from other devices are impeded. A device senses when no other devices are transmitting, imposes a back-off delay, and proceeds to access the channel. A consequence of this mode is that there may be an uncontrolled delay for a node to transmit, which results in idle energy consumption [23]. As a result, most WSNs employ a contention-free MAC protocol, which divides a period of time known as a frame into small reservation periods by which nodes are assigned to transmit. Also, these time-division multiple access (TDMA) protocols may permit energy efficient techniques, such as transceiver sleep scheduling, to minimize idle energy consumption. The MAC protocols we employ later assume a non-contention TDMA scheme for each node.

Researchers are experimenting with MAC layer protocols specific to WSNs. One implementation, S-MAC, uses three techniques to reduce WSN energy consumption: 1) nodes periodically sleep, 2) neighboring nodes form clusters that synchronize their sleep schedules, and 3) message parsing is applied to reduce contention latency [24]. Networks with a large number of sensor nodes frequently have a lot of duplicate information injected into the network by many sensors that sense the same event, which increases data processing and energy depletion. Another MAC implementation specific to WSNs is called Sift [25], which provides a mechanism to avoid multiple nodes in the same neighborhood from transmitting information for a common detected event.

3. Network Layer Load Balancing

The network layer is a huge research focus area for WSN load balancing because the routes that are chosen have a huge impact on the total energy expended in a WSN over time. Inefficient route determination causes the network to prematurely die out. As long as transmit/receive operations and access to the channel are performed efficiently (those process are well tried and tested) the network layer offers fruits for load balancing opportunities in any custom WSN. Network layer WSN routing protocol surveys have been performed in [5]–[9], [26]–[28]. The literature generally includes two areas of network layer load balancing: the traditional minimum cost path for information to pass through many nodes to the gateway and group clustering. In Figure 1, an example of a minimum cost route in which the route from node C to the gateway was determined to be C → Q → N → D → gateway is provided.

The traditional minimum cost path involves each node determining the next hop to send the information according to an algorithm that incorporates some type of cost parameter. The cost parameter varies significantly in the literature and can be any cost metric of interest to the application being designed. Traditional minimum cost routing can degrade a WSN. In [29], the authors showed that minimum cost routing paths tend to overuse certain nodes. These nodes are then overloaded with traffic from the rest of the network and, thus, experience faster energy depletion compared to other nodes. To address this problem, the authors include a cost metric that accounts for network load where the greater the load, the greater the cost metric. These costs are then used to route around high traffic nodes, which provides an increase in network lifetime and fault tolerance [29].

We implement an energy-efficient version of Dijkstra's algorithm [30] in our simulations to analyze the performance of minimum cost routing in WSNs. Minimum cost routing is generally based on Dijkstra's shortest path algorithm [30]. We describe the specifics of this routing algorithm in Chapter VI.

An alternative to traditional minimum cost routing is clustering. In clustering, a cluster head (CH) is chosen from a group of nodes to serve as an intermediate relay between a group of nodes and the gateway(s). In other words, nodes with data to send forward the data to their respective CH, at which point the CH then forwards the data to the gateway. This scenario is presented in Figure 7, which was adapted from Figure 3. Cluster heads P, M and O are chosen from a clustering algorithm at the network layer followed by an association of child nodes to associate with the CH to form three smaller WSNs.

An illustration of a multi-gateway network is shown in Figure 7, which adds further complexity at the network layer. Whether the network layer is using a traditional minimum cost routing algorithm or a clustering algorithm, the algorithm must be designed to use the gateway that provides the least energy cost to a WSN. In the case of clustering, this may mean that a CH chooses the closer gateway to minimize required transmission power. Another instance may require the CH to choose the gateway that is further away to avoid increased interference at the closer gateway.

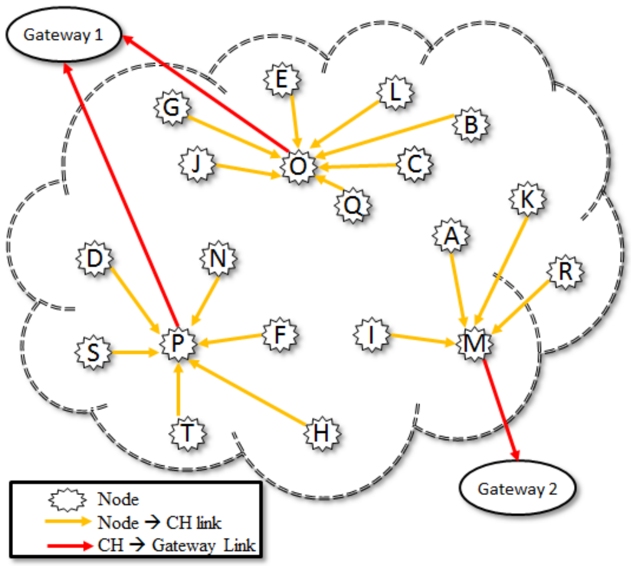


Figure 7. Clustering in a multi-gateway WSN with cluster heads: P, M, and O.

The foundation of a clustering algorithm is the method for choosing a CH and the method for nodes in the WSN to associate with their CH. One popular clustering mechanism is LEACH, originally proposed by [1]. This algorithm was originally presented in 2000 as a proposed solution for CH election criteria. Specifically, the high energy CH role is periodically rotated throughout the network so that energy consumption at each node is balanced throughout the network lifetime. There has been a significant amount of research on LEACH and variants of the algorithm that have been proposed. At the time of writing of this thesis, the IEEE Explore database indicated 1604 citations for [1], while GOOGLE citation tracking claimed it was cited 9600 times. The author republished LEACH in 2002, addressing several improvements of the algorithm [2]. We utilize LEACH in our research and explain the algorithm in detail later.

Another clustering mechanism is that of zone based clustering, in which nodes are assigned to a cluster according to a predefined spatial arrangement [31]. Cluster heads are then chosen based on some prioritization of the nodes. This prioritization was performed randomly in [31]. We employ this technique in our simulations. In addition, we implement our own zoning technique and introduce a novel energy-efficient process to elect CHs with the desire to minimize WSN energy depletion.

Network control message schemes are implemented at the network layer and provide an autonomous control mechanism for the network to manage communications and errors. The control message scheme that dominates the internet is called internet control management protocol (ICMP) in which events trigger the generation of control messages to be exchanged between network devices [32]. Version 4 of ICMP (ICMPv4) is based on the internet protocol version 4 (IPV4). The applicability of ICMP to WSN load balancing is that as control messages are generated, they must be processed by WSN nodes, which may cause unnecessary energy use. Thus, control message implementations for WSNs must specifically minimize the amount of messages generated to lower node overhead. For example, [33] introduced a multi-path routing algorithm with a reduced control message structure for WSNs that ensures messages are only sent to specific nodes instead of to everyone (flooding). In terms of clustering algorithms, control messaging is used for CH selection, CH announcement and during the phase that nodes are associating with their CH. We do not simulate control messages in this thesis. However, we adapt best practices from the literature to describe our proposed messaging scheme and offer further simulations as future work.

4. Transport Layer Load Balancing

The common area for transport layer load balancing is in the form of flow control techniques. A flow control technique is used to individually track the communications (packets) from start to completion of the entire message and provide a mechanism to control an orderly transmission of messages, which is also known as congestion control. If a packet is missing or in error, flow control is used to correct the deficiency.

In [34], the authors implemented a flow control load balancing scheme where each node kept track of the sequence of packets it received from every source node. A gap in sequence number indicates packet loss at which point the missed sequence number is added to a missing packet list. At the end of the transmission for all packets in a sequence, the missing sequence numbers are requested from the previous node, and the packets are obtained. In another work, [35], a flow control scheme on multiple paths without causing packet reordering is described, which minimized processing demand and packet latency. Li et al. developed a feedback congestion control scheme specific for WSNs that detects the onset of congestion using queue length [36]. They then implemented a newly proposed control scheme to throttle packet transmission to minimize energy consumption and increase throughput.

5. Application Layer Load balancing

Application layer load balancing requires that an application operating at a source node scales its generated traffic in a manner that is optimal for a particular network or scenario. For example, a video surveillance WSN may contain several redundant video feeds that are not valuable at a particular instance in time. If the application somehow knows that its traffic is redundant or unnecessary, it can be triggered to lower its generated traffic. This preserves the sensor's battery life and lowers downstream network traffic, allowing other data to have priority and subsequently reduced latency.

Context modeling is used in [37] to capture multiple context parameters at different layers of the network stack to balance runtime application demands of the node and other nodes in the network. Some strategies at the application layer throttle the program based on a demand signal at a lower layer. Application layer scheduling can also be implemented as a load balancing technique. Here, application layer programs are scheduled to perform functions at predetermined times to minimize the energy demand on the node.

Data aggregation, also known as data fusion is the process of combining data fusing techniques such as compression, suppression (eliminating redundant data), data reduction, and other signal processing techniques [8]. Data aggregation has been employed by several protocols such as [38], [39]. An example of data aggregation is shown in Figure 8 for a clustering mechanism, where the CH (Node P) receives six 2000-bit data packets from its child nodes and aggregates them along with its own 2000-bit packet into one 2000-bit packet that is transmitted to the gateway.

Data aggregation requires that each node have the processing capability to perform the data fusion, which brings along an energy cost to the node. Data aggregation can be employed by both MTE and cluster based routing approaches. With respect to MTE routing in Figure 1, Node D can collect several messages over time, perform data aggregation, and transmit the data to the gateway to minimize the number of transmissions required by Node D, subsequently extending its energy life. In Chapter V we simulate data aggregation for the clustering mechanisms developed in this thesis. We leave data aggregation in MTE routing as a topic of future work.

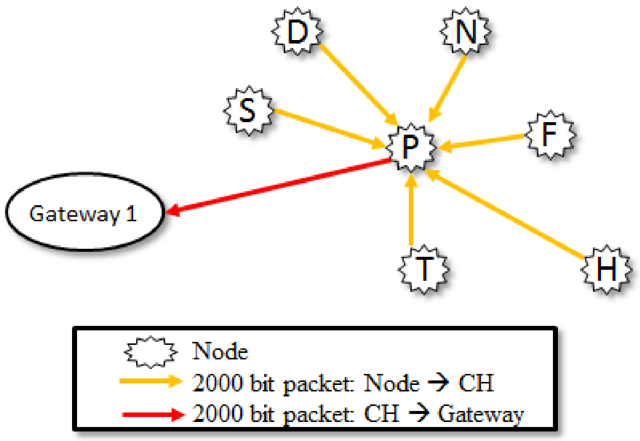


Figure 8. Data aggregation by node P (the CH), which receives a 2000-bit packet from each child node and compresses it into a 2000-bit packet, which is transmitted to the gateway.

E. DESIGN METHOLODY FOR LOAD BALANCING

We have summarized several load balancing techniques in this chapter, several of which we incorporate in our simulations. The load balancing techniques used in this thesis are summarized as follows:

- Physical layer: Variable transmit power with a sleep cycle such that a source node only transmits with the power required to reach the destination node with acceptable signal-to-noise ratio (SNR).
- MAC layer: Contention-free TDMA based MAC layer where time division is divided into rounds with each node having a timeslot to transmit information within each round.
- Networking layer: We implement the following cases, which are thoroughly discussed in Chapter IV:
    1. An algorithm in which each node transmits its payload directly to the gateway (direct).
    2. Each node transmits its payload using a MTE routing scheme using Dijkstra's classic shortest path algorithm.
    3. LEACH clustering algorithm.
    4. A zone-based network clustering algorithm (zone) that partitions the network into zones and hierarchically determines node to gateway routes using a CH assigned in each zone.
    5. Our energy efficient zone routing algorithm (EZone) that partitions the sensor field tactically and assigns a CH in each zone using an energy efficient strategy. Each CH serves as an intermediate relay for node-gateway routing in each zone.
- Transport layer and application layer: Modeled as constant bit rate payloads, where for every round, every node sends a similar sized packet to the gateway. Data aggregation is employed by the CH in the clustering mechanisms.

F. CHAPTER II SUMMARY

In this chapter we introduced the problem of load balancing in WSNs. We described the layering construct for wireless devices, which provides load balancing opportunities that can be employed at each layer of a WSN to extend its service life. Various load balancing techniques for each layer were presented based on our background research. We concluded the chapter by providing the framework for the load balancing strategies we implement later for single gateway and multi-gateway WSNs.

THIS PAGE INTENTIONALLY LEFT BLANK

III. PHYSICAL AND MEDIUM ACCESS LAYER MODELS

In this chapter we describe our implementation of the sensor grid, the physical layer, and the MAC layer. We describe how the sensors and gateways are distributed for both a single gateway scenario and a multi-gateway scenario. We then describe our wireless propagation model and perform an analysis to demonstrate the desirability of a clustering mechanism for the network layer based on energy analysis of the physical layer protocol. We complete the chapter by describing the implementation of the MAC layer and our associated assumptions.

A. SENSOR AND GATEWAY PLACEMENT

Sensor and gateway placement are inputs to the physical layer. Sensors and gateways are all placed on a Cartesian grid with axes *x* and *y*. Our analysis involves a grid of 100 sensors such that each sensor's *x* and *y* coordinate is modeled as a uniformly distributed random variable between 0 and 50 m:

$$(x, y) = (U[0,50], U[0,50]). \tag{1}$$

Our scenarios assume that the network is a tactical WSN that requires the gateways to be placed reasonably far from the sensor network. Following this assumption, the single gateway scenario employs the gateway at (x, y) = (25 m, −100 m), while the multi-gateway simulations have the gateways positioned at (x_1, y_1) = (25 m, −100 m) and (x_2, y_2) = (25 m, 150 m). The single gateway and multi-gateway WSNs are graphically shown in Figure 9 and Figure 10, respectively.

Our presentation of the WSN is consistent throughout this thesis. Gateways are always displayed as they are in Figure 9 and Figure 10 as a solid green circle. Live nodes are shown as a white circle with a blue outline. Although we only show live nodes in Figure 9 and Figure 10, as nodes eventually die out, they are then represented as a solid red circle at their same location. We show the perimeter and field zones (for zone routing algorithms) as sold red lines.

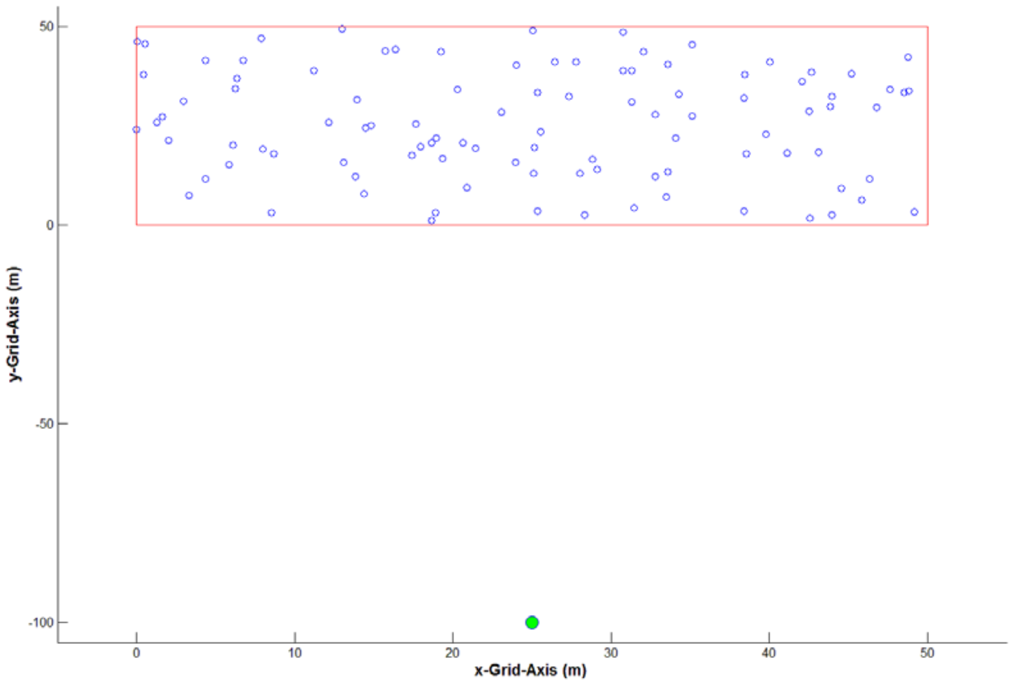


Figure 9. Single gateway WSN arrangement simulated in Chapter V.

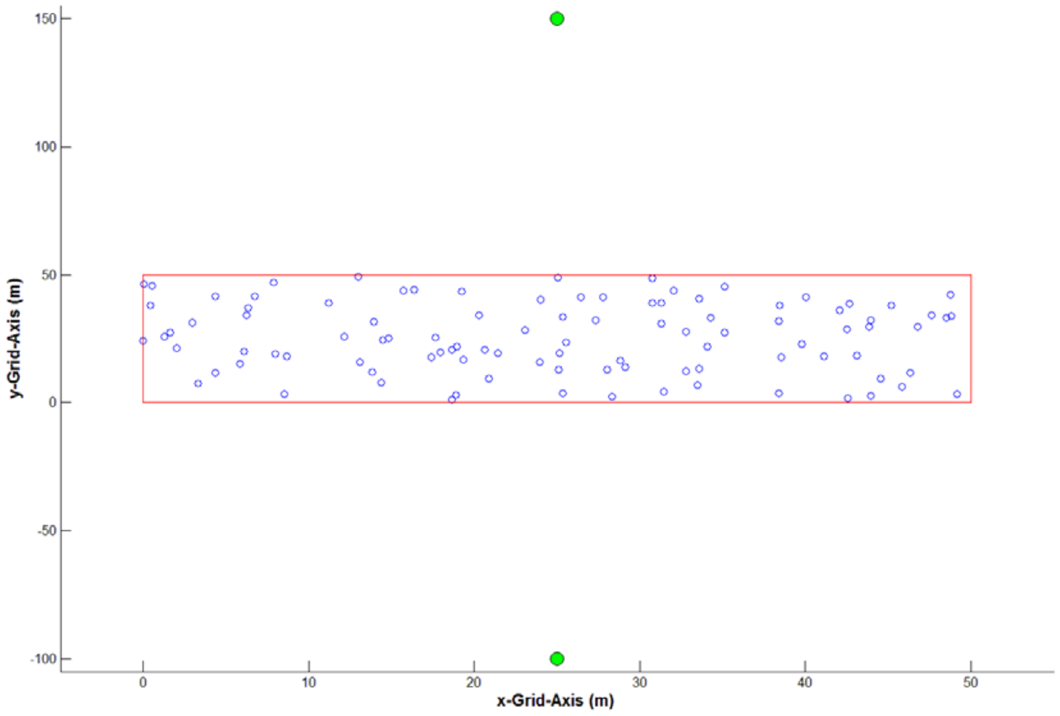


Figure 10. Multi-gateway WSN arrangement simulated in Chapter V.

B. NODE STARTING ENERGY LEVEL

All nodes in our simulations begin with a starting energy level of 0.5 Joules (*J*). This is a value commonly used in the literature because it provides small enough energy to quickly see the effects of the varying algorithms involved yet it provides enough energy to demonstrate the longevity by making algorithmic improvements. That being said, 0.5 J is a very small amount of energy. In comparison, a typical 1.5 V alkaline AA battery contains approximately 1,500 mAh of energy, which provides 1500 mA for one hour at 1.5 V. Ignoring load curves versus voltage for the battery, the equivalent energy in Joules is

$$(1500\times 10^{3}A)\left(1.5V\right)(1hr)\left(\frac{3600s}{1hr}\right)=8100J \tag{2}$$

which indicates that just a few AA batteries in a small sensor node can provide significant lifetime for a WSN node using nano-watts of energy at any given time.

C. PHYSICAL LAYER MODEL

The physical model relates the amount of energy a sensor node consumes during transmit and receive operations. As a result, principles of wireless propagation must be included. We utilize the first order radio energy model, which is common throughout the literature [1] [2] [40]. This model relates the energy expended to send and receive an *L*-bit message over a distance *d* when considering direct path and multi-path propagation. The first order radio energy dissipation model is shown in Figure 11.

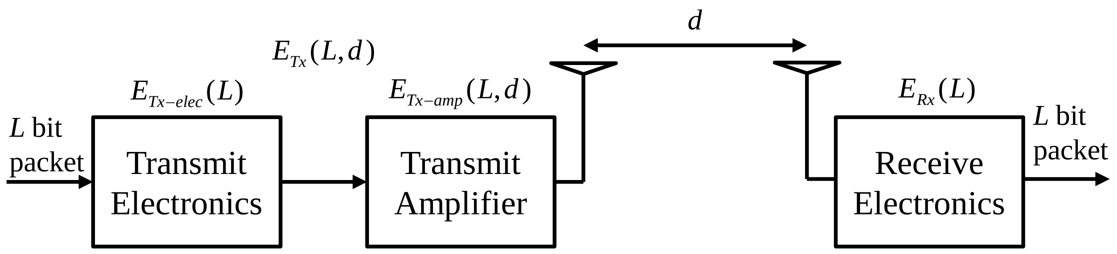


Figure 11. First order radio energy model for physical layer simulation.

The energy expended in the transmit electronics of Figure 11 for free space (direct path) propagation, $E_{Tx\text{-}fs}$, is described by

$$E_{Tx-fs}(L,d) = E_{Tx-elec}(L) + E_{Tx-amp}(L,d) = E_{elec}L + \varepsilon_{fs}Ld^2 \tag{3}$$

and for multipath propagation by

$$E_{Tx-mp}(L,d) = E_{Tx-elec}(L) + E_{Tx-amp}(L,d) = E_{elec}L + \varepsilon_{mp}Ld^4 \tag{4}$$

where E_{elec} corresponds to the energy per bit required in transmit and receive electronics to process the information, $E_{Tx\text{-}amp}$ is electrical energy required to transmit an L-bit message over a distance d, and ε_{fs} and, ε_{mp} are constants corresponding to the energy per bit required in the transmit amplifier to transmit an L-bit message with adequate SNR over a distance d^2 and d^4 for free space and multi-path propagation modes, respectively.

The energy expended to receive the *L-bit* message in the receive electronics of Figure 11 is described by

$$E_{Rx}(L) = E_{elec}L. \tag{5}$$

The corresponding values from equations (3), (4), and (5), for the amplifiers and electronics used in our subsequent simulations are described in Table 1.

Table 1. Radio energy dissipation parameters used during our simulations.

Constant	Value
Transmit Electronics, E_{elec} Receiver Electronics, E_{elec}	50 nJ/bit
Transmit Amplifier, free space propagation, ε_{fs}	10 pJ/bit/m^2
Transmit Amplifier, multi-path propagation, ε_{mp}	0.0013 pJ/bit/m^4

By equating Equations (3) and (4), we determine the distance d=d_0 when the propagation transitions from direct path to multi-path:

$$E_{Tx-fs} = E_{Tx-mp} \Rightarrow d_0 = \sqrt{\frac{\varepsilon_{fs}}{\varepsilon_{mp}}}. \tag{6}$$

Distance d_0 is solely a function of the transmit amplifier parameters as shown in Equation (6). Substituting the amplifier parameters of Table 1 into (6), d_0= 87.7 m. Specific to the sensor network shown in Figure 9 and Figure 10, communications between nodes are generally direct path propagation, and communications between nodes and the gateway are multi-path propagation.

D. PHYSICAL LAYER IMPACT TO THE NETWORK LAYER

Our implementation suggests that the physical layer has the largest impact on a sensor's energy level since this is where our model depletes energy based on the magnitude of the wireless propagation distance. This concept provides two options at the network layer. Each node sends their information to the gateway directly, or each node sends their information with MTE by utilizing the nearest neighbor in the optimum path toward the gateway. If we assume the propagation is solely direct-path propagation in which the energy is proportional to d^2, there exists an energy balance such that in some cases either direct transmission or MTE routing to the gateway is preferred. For example, a simple network is shown in Figure 12, in which *n* nodes are separated a distance *r* apart from each other. If direct routing is performed, each node transmits its packet a distance *nr* to the gateway. If employing MTE routing, each node (except the source node) receives a packet and retransmits the packet to the next node a distance *r* away.

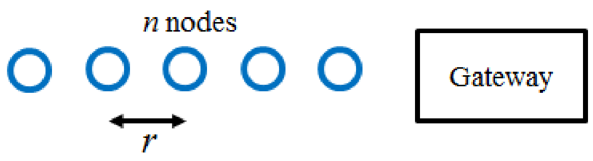


Figure 12. A simple network of *n* nodes and one gateway each separated by a distance *r*.

Performing the energy analysis of Equation (3) for direct and MTE routing, we get

$$E_{direct} = E_{Tx}(n, d = nr) = E_{elec}L + \varepsilon_{fs}L(nr)^2 = L(E_{elec} + \varepsilon_{fs}n^2r^2) \quad (7)$$

and

$$E_{MTE} = nE_{Tx}(L, d = r) + (n-1)E_{Rx}L = n(E_{elec}L + \varepsilon_{fs}Lr^2) + (n-1)E_{elec}L \\ = L((2n-1)E_{elec} + \varepsilon_{fs}nr^2), \tag{8}$$

respectively.

The preferred networking mode is the one that requires less energy. Direct communication with the gateway requires less energy than MTE routing provided:

$$E_{direct} < E_{MTE}. \tag{9}$$

Substituting Equation (7) and (8) into Equation (9), we get

$$L(E_{elec} + \varepsilon_{fs}n^2r^2) < L((2n-1)E_{elec} + \varepsilon_{fs}nr^2) \tag{10}$$

which after being simplified, provides

$$\frac{E_{elec}}{\varepsilon_{fs}} > \frac{r^2 n}{2}. \tag{11}$$

Equation (11) reveals that the optimal routing technique is a function of the amplifier parameters and network topology. A simulation using this analysis with a *n*=100 node network in which the network dimension was increased from 10 m × 10 m to 100 m × 100 m, each node had a starting energy of 0.5 *J*, *L* = 2000 bits, E_{elec} was increased from 10 to 100 nJ/bit, ε_{fs} held constant and the gateway was placed at (*x*=0, *y*=−100 m) was performed in [1]. From these parameters, Figure 13 was produced, which graphically demonstrates that the most energy efficient algorithm to use depends on the network topology and the radio parameters of the system.

A simulation was performed on one 100 node network arrangement with sensors uniformly distributed in a 50 m × 50 m grid with similar gateway placement (*x*=0, *y* = −100 m) and constant amplifier parameters. The results are shown in Figure 14. The tradeoff between MTE and direct routing from when the first node dies until the last node dies are revealed in Figure 14. The direct case provided the longer network lifetime with all nodes alive since MTE routing overwhelmed some preferred nodes with network traffic. These preferred nodes subsequently die out first. Conversely, MTE routing provided the longer network lifetime with at least some nodes alive. This indicates that nodes in a less preferred position relative to the gateway are underwhelmed with network traffic and subsequently remain in service longer.

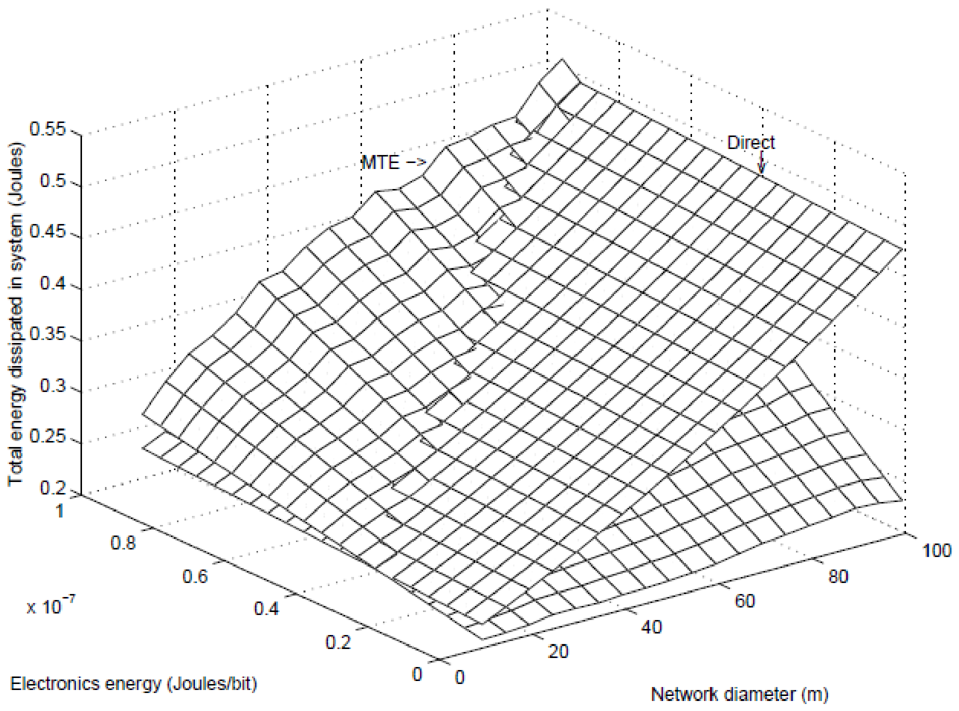


Figure 13. Total energy dissipation for Direct and MTE routing versus network dimension versus E_{elec} demonstrating the tradeoffs of each technique utilizing direct path propagation (from [1]).

The tradeoff occurs in the design methodology of the network. Does the network designer prefer to have all nodes operating the maximum amount of time, or is it preferred to have only some nodes operating the maximum amount of time? Our response to this question is somewhat a mix of both, as the topology of the network die out must also be considered to achieve a desired functionality. For example, if the last 10 to 15 nodes in the MTE algorithm of Figure 14 are tightly grouped together on the periphery of the WSN field, this coverage may offer little benefit to network functionality. However, if the last 10 to 15 nodes maintain a consistent uniform distribution throughout the sensor field, useful coverage still exists, and the WSN continues to serve its purpose.

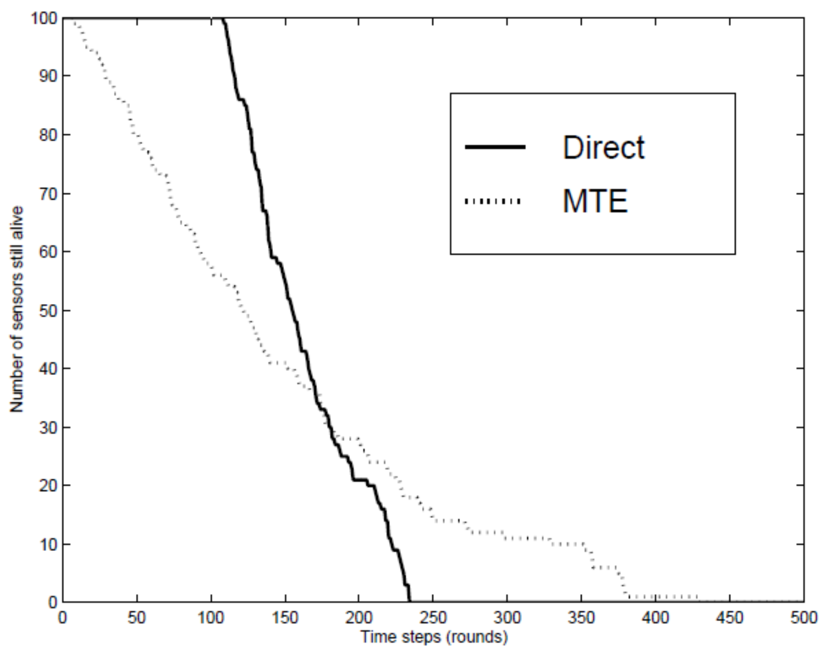


Figure 14. Tradeoff of direct versus MTE routing on sensor die out for direct path propagation (from [1]).

With respect to MTE routing, the node closest to the gateway is termed the hot-node. The hot-node is repeatedly chosen by other nodes to route information to the gateway. The first hot-node dies out quickly, and the next hot node is subsequently chosen by the MTE algorithm. This results in the node closest to the gateway dying out very quickly and the least preferred node (least preferred position for routing) dying out last because no other node utilizes it in the calculation of the preferred route. Thus, these farther nodes always transmit with minimum energy to reach the closest nearby node. This creates the MTE die out curve in Figure 14. Conversely for direct routing, nodes that are farthest away from the gateway expend the most energy transmitting and subsequently die out first while those nodes closest to the gateway remain in service much longer since their energy consumption is lower due to lower propagation distances.

Our purpose for going through this analysis is to briefly demonstrate that the physical layer has a significant impact on the network layer and the spatial die out distribution of the network. Since our WSNs are tactical and rely solely on battery power, we require a networking algorithm that maximizes the network lifetime when all the nodes contain energy and that, as nodes begin to die out, we still obtain service coverage from any particular area in the network. For direct and MTE routing, our qualitative discussion above reveals that as the overall WSN depletes energy, areas are left without service while other areas continue to provide service.

The previous discussion leads us to consider a cluster based network layer algorithm. We initially presented the basic framework for such an approach in Chapter II, Section D.3. The CH as chosen by the networking algorithm transmits all the data that was obtained from the nodes in the cluster. Provided the CH role is periodically rotated, the batteries of all nodes are expected to deplete in a more uniform fashion. This causes the die out of the network to occur in a more preferred fashion, preserving coverage areas during WSN die out. A cluster based approach also allows node data aggregation to take place to minimize the energy consumed by the CH when performing long haul transmission. This is revisited and further explained in Chapter IV.

E. MEDIUM ACCESS CONTROL LAYER

We simulate the MAC layer simply through the performance of transmission rounds. Each simulation begins at round one (zero for LEACH) and ends at some maximum number of rounds (or when the last node dies). During each round, each node in the WSN sends an *L*- bit packet from the application layer to the gateway. We assume a TDMA scheme is in place, in which each node is assigned a timeslot to transmit its packet. We are not concerned with how the TDMA assignment takes place, just that during each round, each node transmits its packets to the gateway. For clustering algorithms, there are MAC protocols in the literature that allow for nodes within a cluster to transmit to their CH at similar times as nodes in an adjacent cluster transmitting to their respective CH. This approach uses a coding scheme that eliminates the possibility of inter-cluster interference. We adopt these techniques for our MAC layer implementation.

F. CHAPTER III SUMMARY

In this chapter we described our implementation of the sensor grid, the physical layer, and the MAC layer. We described how the sensors and gateways are distributed for both a single gateway scenario and a multi-gateway scenario. We presented our methodologies for each node’s starting energy level and then described our wireless propagation model. We performed an analysis that demonstrated how our physical layer model motivated our research toward a clustering algorithm at the network layer. We completed the chapter by describing the implementation of the MAC layer in this thesis and our associated assumptions.

IV. NETWORKING LAYER MODEL

In this chapter, we describe the network layer algorithms implemented for simulation in Chapters V and VI. We utilize certain algorithms from the literature that were analyzed in single gateway WSNs and apply them to our single gateway and multi-gateway scenarios. The algorithms we utilized from the literature are: each node transmits its information directly to the gateway(s) (Direct), MTE routing utilizing Dijkstra's algorithm (MTE) Low Energy Adaptive Cluster head (LEACH) routing, and zone clustering with random CH assignment (Zone). We then describe an energy-efficient zone clustering algorithm (EZone) that we devised as a result of our research.

Our interest is to investigate the result of load balancing techniques in single gateway and multi-gateway WSNs by employing load balancing techniques at each layer while focusing on the impact of varying network layer routing algorithms on the WSN. Our goal is to show how WSN service life can be extended by implementation of load balancing techniques and the addition of another gateway while continuing to provide uniform service in all areas of the WSN as nodes die out.

A. SUMMARY OF PHYSICAL AND MAC LAYER PARAMETERS

The physical and MAC layers were described in Chapter III. For each simulation, sensor network parameters must be entered or generated. For the purpose of comparison, we utilized the same network for each network layer model simulated. The sensor network parameters, as well as description of each parameter, are provided in Table 2. The parameter descriptions and symbols are common to each network layer protocol.

The *x*-coordinates for each node are obtained from a uniform distribution between zero and *xm* meters and *y*-coordinates are similarly obtained from a uniform distribution between zero and *ym*. The amplifier constants were chosen to match those commonly used in the literature [1] [2] [40]. We explain our choices for the probability of being a CH (*p)* and the number of zones *z* in Sections E and F in this chapter.

Table 2. Summary and description of physical layer parameters.

Description	Parameter	Value
Maximum rounds for iteration	*rmax*	variable
Maximum Cartesian *x* range (m)	*xm*	50
Maximum Cartesian *y* range (m)	*ym*	50
x-coordinate of gateway 1 (m)	*sink.x1*	25
y-coordinate of gateway 1 (m)	*sink.y1*	−100
x-coordinate of gateway 2 (m)	*sink.x2*	25
y-coordinate of gateway 2 (m)	*sink.y2*	150
Packet size (bits)	*L*	2000
Number of nodes in the field	*n*	100
Initial node energy (Joules, J)	E_o	0.5
Energy to Transmit (nJ/bit)	*ETX*	50
Energy to Receive (nJ/bit)	*ERX*	50
Free space propagation (pJ/bit/m^2)	*Efs*	10
Multi-path propagation (pJ/bit/m^4)	*Emp*	0.0013
Probability of being a CH (LEACH)	*p*	0.05
Number of Zones (Zone clustering)	*z*	5

Our WSN simulations assume that every node is within communication range of the gateway. In actual practice, this may not be the case. Each of the network layer algorithms tested (except for the direct transmission to gateway scenario) can be extended to a scenario where all nodes are not within communication range. These are not simulated in this research. However, we explain later how the models can be extended and leave this task as an area of future work.

B. DIRECT TRANSMISSION TO THE GATEWAY

Direct transmission to the gateway involves each node sending a packet to the gateway directly without using any other nodes along the way. During each round, the Euclidean distance is calculated between the node and the gateway, the distance is compared to d_0 (Chapter III) in order to determine the propagation mechanism, and the node's energy is decremented in proportion to the required energy for packet transmission to the gateway. For the multi-gateway scenario, the node chooses the gateway that requires the smaller transmit energy (i.e., the closer gateway) and transmits the packet to that gateway. The pseudocode for the direct transmission case is provided in Figure 15. The complete code is provided in Appendix B. Results for this simulation are provided in Chapter V.

```
%  % denotes comment or to implement a task

%input sensor network parameters
%         or generate sensor network

%setup plotting formats

%Assign starting Energies
for node = 1:n
        S(node).E=Eo
end

%
for round = 1:rmax

    for i=1:n %check and assign alive nodes to a aliveNode array
        if(S(i).E>=0
            aliveNode(i)=1
        end
    end


    for i = aliveNode
        distance = Euclidean distance between node and gateway

        if(distance > do) %mulitpath propagation
            S(i).E = S(i).E - ( (ETX)*(L) + Emp*L*( distance^4 ))
        end

        if(distance <= d0) %directpath propagation
            S(i).E = S(i).E - ( (ETX)*(L) + Emp*L*( distance^2 ))
        end

        %obtain node energy at conclusion of each round for post processing
        for i=1:n
                energyBar(i)=S(i).E;
        end

        %determine dead nodes and alive nodes for post processing

        %plot and obtain desired statisticts each round

        %if all nodes are dead, break out of the loop
        if(aliveNode is empty)
                break
        end
end

%plot and save statisticts for round = 1:rmax
```

Figure 15. Pseudo code for direct transmission to gateway.

C. MINIMUM TRANSMISSION ENERGY WITH DIJKSTRA'S ALGORITHM

To perform the MTE scenario, an algorithm must be generated that produces a route from every node to the gateway using the node's closest peers without generating loops in a fashion that minimizes propagation distance to the gateway. We desire to minimize propagation distance to the gateway in order to produce a route that minimizes the overall sensor energy depletion rate. In this construct, we utilize propagation distance as our link cost parameter to input into the MTE algorithm. Previously, our d_0 parameter indicated that propagation inside our sensor network grid is mostly a free space propagation (direct path instead of multi-path propagation), which allows the Euclidean distance between any two nodes $(d)^2$ to be utilized as the link cost parameter for a link cost algorithm.

There have been several link cost MTE algorithms developed to generate routes, some of which are energy conservation algorithms for WSNs. For the purpose of this research, we utilize the classic networking algorithm known as Dijkstra's algorithm to generate our MTE routes. Dijkstra's algorithm is the foundation for most least-cost packet switching networks [30]. Utilizing d^2 as the link cost parameter represents an energy efficient routing technique because it is minimizes the d^2 propagation distance between a source node and the gateway. The implementation of Dijkstra's algorithm is described in [30] and is executed in three steps (steps two and three are performed until all nodes $T = N$ are incorporated by the algorithm) where N is the set of nodes in the network, S is the source node, T is the set of nodes incorporated by the algorithm, $w(i,j)$ is the link cost from node i to node j and $L(n)$ is the cost of the least cost path from node S to node n that is currently known to the algorithm.

If two nodes cannot communicate with each other, the link cost between them is infinity. Since our simulations assume the gateway is within transmit range of any node, there exists a total link cost from each node to the gateway.

1. Dijkstra's Algorithm, Step 1: Initialization

$T = \{S\}$, the set of nodes incorporated begins with the source node.

$L(n) = w(s,n)$ for $n \neq s$, source neighbors are incorporated by the algorithm with the initial path costs to neighboring nodes represented by the link costs.

2. Dijkstra's Algorithm, Step 2: Get Next Node

Find the neighboring node, *x*, not in *T* that has the least-cost path from node *S* and incorporate that node into *T*. Incorporate the edge of *x* into *N* and calculate updated routes for other nodes in *T*.

3. Dijkstra's Algorithm, Step3: Update Least Cost Paths

If an updated route from *S* to another node *x* is minimized, then that route becomes the updated route.

Utilizing Dijkstra's algorithm, we determine a path from the source node to the gateway during each round for each transmission. While this is computationally intensive, it must be done each round for each node so that, if a node dies mid-round, that node can no longer be utilized by other nodes and must be excluded from further route calculations. The output of the algorithm for each node is a least cost path that the packet travels through to reach the gateway. Using the distance between nodes along the path, we can use our physical layer implementation to deduct the energy to transmit and the energy to receive at each hop along the path. The pseudocode for implementing an MTE algorithm utilizing Dijkstra is shown in Figure 16.

To verify that our implementation of Dijkstra's three-step algorithm produces reasonable routes, we first tested just the algorithm solely to generate the path, which corresponds to the function dijkstra( ) from the pseudocode presented in Figure 16. Our function dijkstra( ) was constructed in MATLAB. The function dijkstra( ) using the three step process described above, utilizes the distance squared between nodes as the link cost routing metric. Two examples of our implementation are shown in Figure 17 and Figure 18, where each network has 50 nodes and one gateway (node 51). Figure 17 and Figure 18 are for example only and are not representative of networks used in later simulations. The function, dijkstra( ), was used to calculate the path from node 1 (the source node) to node 51 (the gateway). Nodes are indicated by a circle with a blue outline while nodes used for routing are indicated by a solid black circle inside the blue circle with the node number plotted alongside. The gateway (node 51) is indicated by a solid green circle at position (25 m, −100 m). The route in Figure 17 was determined to be 1→45→40 →6 → 43→4→51.

```
%  % denotes comment or to implement a task

%Input sensor network parameters or generate sensor network.
%Setup plotting formats.

%Assign starting Energies
for node = 1:n
        S(node).E=Eo
end

for round = 1:rmax
    for node = 1:n
        nodesAvailable = [ ] % initialize nodesAvailable to an empty array to preclude
                             %the route contribution of nodes that died mid round
        for i = 1:n
            if(S(i).E>0)
                aliveNode(i)=1
            end
        end
        numAlive = sum(aliveNode)

        NodesAvailable = find(aliveNode = 1) %available nodes for routing

        %output the path from the node to the gateway using NodesAvailable
        %output path provides sequential node numbers from source to gateway
        path = dijkstra(node, gateway, NodeAvailable)

        %Decrement the energy for each node in path
        for i=path
           if (i is not the first or last element in path)
                        %Decrement energy for each node to recieve the message.
                        %Do not include the first element since it is the source.
                        %Do not include the last element since it is the gateway.
                S(i).E = S(i).E-ERX*L
           end

           distance = Euclidean distance between i and the next node in path %TX distance

           if(distance > do) %mulitpath propagation
               S(i).E = S(i).E - ((ETX)*(L) + Emp*L*( distance^4 ))
           end

           if(distance <= d0) %directpath propagation
               S(i).E = S(i).E - ((ETX)*(L) + Emp*L*( distance^2 ))
           end
        end
     end

    %obtain node energy at conclusion of each round for post processing
    for i=1:n
        energyBar(i)=S(i).E;
    end

    %determine dead nodes and alive nodes for post processing
    %plot and obtain desired statistics each round

    %if all nodes are dead, break out of the loop
    if(aliveNode is empty)
        break
    end
end

%plot and save statistics for round = 1:rmax
```

Figure 16. Pseudocode for MTE routing using Dijkstra’s algorithm.

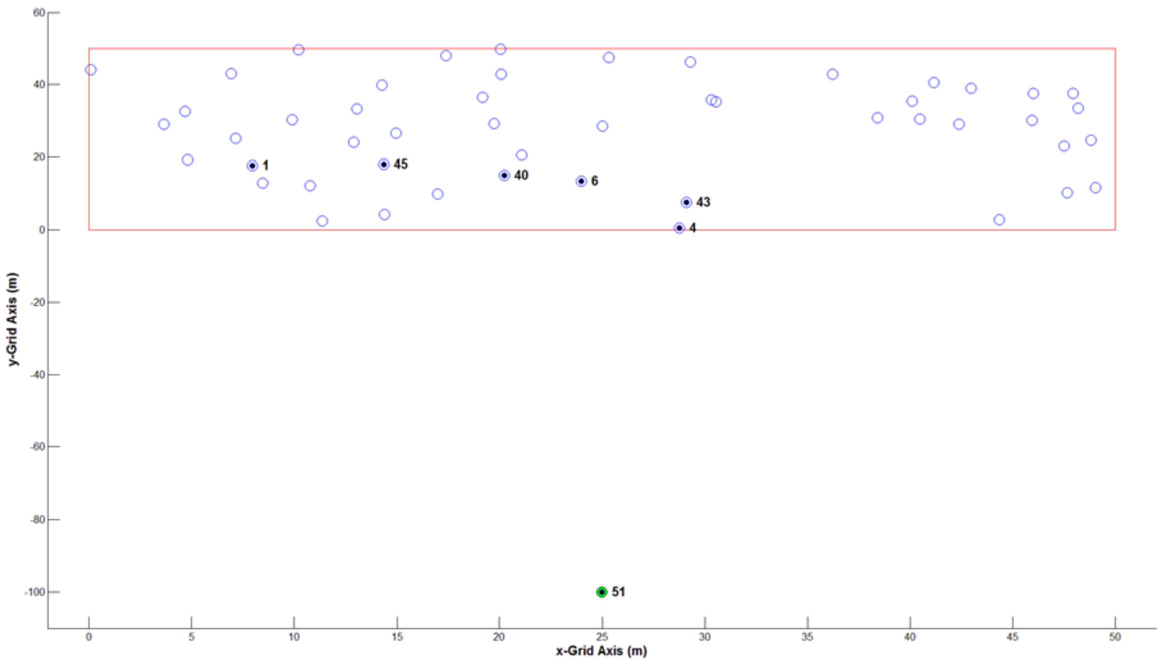


Figure 17. Dijkstra's algorithm from node 1 to node 51 (gateway) produced the path 1→45→40 →6 → 43→4→51.

The route chosen in Figure 17 minimizes the direct path energy transmission cost between the source and the gateway (1→51) and produces a visually expected result. The route chosen from the left region of the sensor field to the gateway moves toward the center before choosing the least cost node that performs the long-range wireless transmission to the gateway. Node 4 is also considered to be the aforementioned hot-node for this WSN in that it is in the most preferred position for transmission to the gateway. This is a position that guarantees its continued use by other nodes, causing it to die out quickly. Another randomly generated WSN is shown in Figure 18 in which dijkstra() is tested. The path from Node 1 to the gateway node 51 was chosen to be to be 1→28→9→44→51.

Even though we are implementing an energy efficient MTE algorithm, we allow the network to be greedy in selection of the routes. The way that the dijkstra( )is implemented allows nodes that are commonly in the least cost path to be significantly used during each round; thus, their battery energy levels deplete quickly, and subsequently, they die out first.

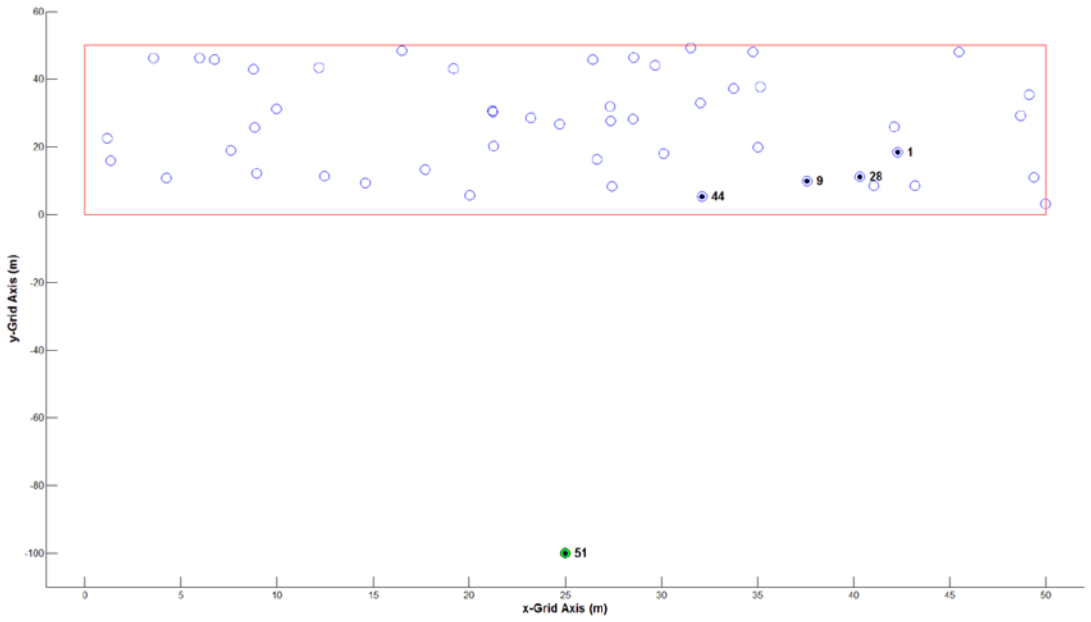


Figure 18. Dijkstra's algorithm from node 1 to 51 (gateway) produced the path 1→28→9→44→51.

Our MTE algorithm does not employ any data aggregation strategy since at each round every node is assumed to transmit its message in a TDMA scheme where there is only one message passing from source to gateway through the network at a time. This prevents any sort of efficient data aggregation technique to be employed. However, it remains useful to see the behavior of this type of MTE algorithm to observe how the network dies out over time as well as the distribution of battery level over time.

The multi-gateway scenario runs in a similar fashion as the single gateway except each node maintains a least cost route to both gateways. As each node transmits its message, the gateway with the smaller least cost route is chosen, and the battery energy levels for each node along the least cost path are decremented, respectively. Maintaining a least cost route to both gateways increases the processing required to generate the route but provides additional options for a node's payload to be transmitted to a gateway. Utilizing a multi-gateway WSN in this fashion offers an opportunity to utilize a different link-cost metric in our implementation of dijkstra ( ) (i.e., to tailor route identification in some specific fashion). For example, if one tailored the link cost parameter to link-SNR versus time, changes in weather patterns and atmospheric conditions can be incorporated.

The node would choose the optimal route to either of the gateways considering the less than optimal propagation environment. We continue using distance squared as the link cost parameter for this research. Analyzing the impact of varying the link cost parameter for WSNs is left for future work.

D. LOW ENERGY ADAPTIVE CLUSTER HEAD ROUTING

We now move into describing our first clustering technique. Our motivation for employing a clustering technique is aimed at rotating the energy intensive role of the node that performs the long-range wireless transmission to the gateway as well as providing the opportunity to perform data aggregation. In Figure 17, Node 4 is the hot-node because it consistently performs the long haul wireless transmission to the gateway. Coupled with a physical layer that expends only an amount of energy required for the transmission, this causes the hot-node to die out first. As a hot-node dies, the next most preferred hot-node is utilized and one can quickly conceptualize how the network will die out. Utilizing a clustering mechanism, we rotate the role of the CH to minimize the probability that any node is a hot-node in an effort to balance the energy depletion of all nodes and take into consideration the topology of the network as subsequent nodes die out.

The clustering techniques in the literature primarily investigate a single gateway WSN. We study clustering from a single gateway perspective but also identify WSN performance benefits from introducing another gateway.

In our overview of the network layer load balancing techniques, we briefly described the LEACH networking algorithm. LEACH periodically generates CHs through a random process. LEACH operates on a round-by-round basis in phases, which are summarized in [1]. A block diagram of the LEACH algorithm is shown in Figure 19.

The LEACH algorithm shown in Figure 19 starts with a WSN consisting of energized nodes starting at transmission round zero. The first phase elects the CHs using a random process. Once CHs are chosen, they are assigned a TDMA assignment by the gateway to communicate their payload to the gateway at each round. For intra-cluster communications, each CH is assigned a different coding scheme by the gateway to communicate with its child nodes (i.e., nodes contained within the cluster excluding the CH). Each cluster uses different clustering schemes to reduce the likelihood of cluster communications interfering with the communications from other clusters. Each CH then broadcasts a CH announcement message to the entire WSN. The message is broadcast at the same power-level by all CHs. Individual sensors receive the broadcast messages from the cluster head and choose the one that is received at the highest SNR to associate with. The individual sensors respond back to their preferred CH requesting cluster association for the round. The CH then assigns the node its coding scheme and a transmission time slot within the round. The time slot indicates when the node can transmit its data to the CH. At this point all clusters within the WSN are formed, each node is assigned a timeslot and a coding scheme to communicate with the CH, and the CH is assigned a timeslot to communicate all the information generated by the cluster to the gateway. The round then continues with the CH receiving all the packets from the associated nodes. The CH then aggregates the packets, performs signal compression to reduce the final packet size (data aggregation at the application layer), and transmits the final packet to the gateway. At this point, the round is complete; all sensor data packets have reached the gateway, and it is time to rotate the high-energy CH role to another node. As long as nodes are still alive (they have remaining energy), the process continues with subsequent rounds.

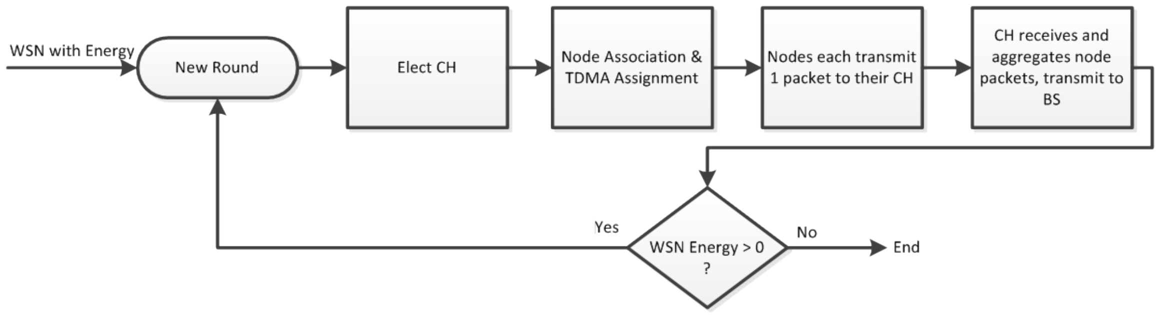

WSN with Energy
New Round
Elect CH
Node Association & TDMA Assignment
Nodes each transmit 1 packet to their CH
CH receives and aggregates node packets, transmit to BS
Yes
WSN Energy > 0 ?
No
End

Figure 19. LEACH algorithm block diagram.

LEACH CH selection utilizes a random process. The network designer chooses a desired percentage of nodes to serve as CHs (*p*). This value is known apriori by the network. The desired percentage of CHs does not change during the network's service life. For example in a 100-node WSN, if the designer desires 10 CHs, *p*=0.10. At the beginning of each round, every node computes a uniform random number, *temp_rand*, between zero and one. Each node then individually computes or performs a table lookup to determine its threshold number *Tn* for the round. The threshold number is a function of *p*, the round *r*, and whether or not the node has been a CH in the last 1/*p* rounds (i.e. the node is contained in *G*, where *G* represents the subset of nodes that have not been a CH in the last 1/*p* rounds*)*. The threshold number for each round is represented as:

$$Tn(r) = \left\{ \begin{array}{ll} \dfrac{p}{1-p\left[\mathrm{mod}(r, 1/p)\right]} & if\ node \in G \\ 0 & Otherwise \end{array} \right\} \qquad (12)$$

where mod() is the modulus function. In order to identify if the node is elected as a CH for the round, the node compares its random number generated for the round (*temp_rand*) with the threshold number that it just calculated. If the node's random number is less than the threshold number, the node is elected as a CH for the round, and a flag is flipped to indicate the node has been a CH in the last *1/p* rounds (which includes it in *G* for subsequent calculations):

$$if\ temp\_rand(node) < Tn \rightarrow \left\{ \begin{array}{l} true:\ node\, is\, CH\ for\, round \\ false: normal\, node\ for\, round \end{array} \right\}. \qquad (13)$$

This scheme ensures that all nodes are elected a CH within *1/p* rounds. This is ensured based on the calculation of the threshold number since *Tn* increases after each round. At (1/*p)* −1 rounds, *Tn* = 1, and any remaining nodes that have not been CHs in the previous 1/*p* rounds are elected as CHs. A plot of *Tn* is shown in Figure 20 for *p* = 0.01, 0.02, 0.05, 0.10 and 0.20 (representing 1 percent, 2 percent, 5 percent, 10 percent and 20 percent of nodes desired to act as CHs, respectively) to show how the threshold number increases each round during 1/*p* round increments. If *p*=0.01 (only one node in 100 nodes is desired to be a CH), all nodes are guaranteed to have been chosen as CHs every 100 rounds. If *p*=0.02 (only two nodes in 100 are desired to be a CHs), all nodes are guaranteed to be CHs every 50 rounds. As *p* increases, nodes are guaranteed to become CHs at faster 1/*p* intervals, which intuitively make sense since there are more nodes serving in the CH role, thus increasing the probability that any node is serving as a CH at any time.

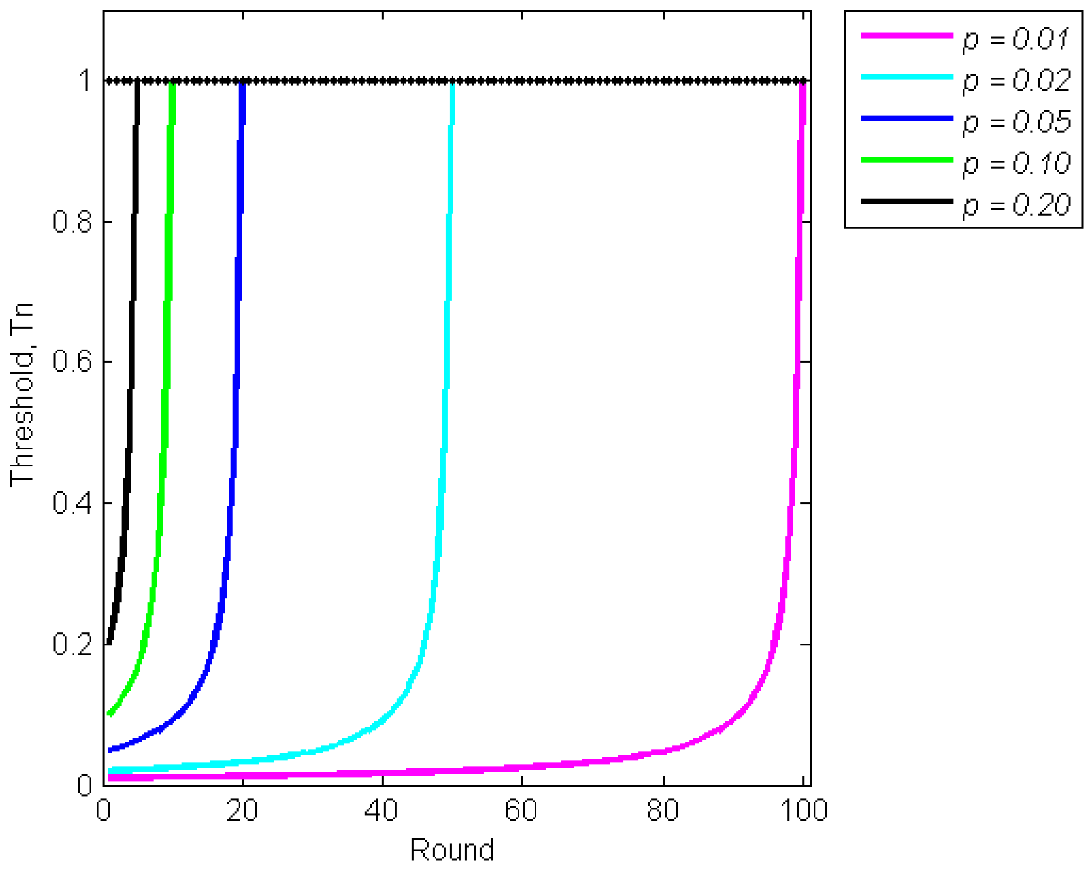


Figure 20. Threshold number *Tn* versus transmission round when varying the probability of a node being elected as a cluster head.

The threshold number increases in subsequent rounds for a given *p* value to one as illustrated in Figure 20, which guarantees that all nodes become a CH in 1/*p* rounds. However, since CHs are chosen in a random fashion, there is a possibility that all nodes will have been chosen in the last 1/*p* rounds, which results in an empty set of nodes available to be chosen as CHs (i.e., *G* is a null set). This is noted in later rounds, when many of the nodes have died and it takes substantially less than 1/*p* rounds for all nodes to be chosen as CHs. This aspect of LEACH is not addressed in the original documentation of LEACH [1]. In these circumstances there are two options: 1) No CHs are chosen and each node sends its information directly to the gateway (assuming each node is within communication range of the gateway), or 2) the set of nodes *G* that have served as CHs is reset so they can be used as a CH in the remaining 1/*p* rounds. Since we utilize the assumption that all nodes are within communication range of the gateway, we implement option 1 in our LEACH simulations.

Pseudocode for the LEACH algorithm is provided in Figure 21, and the full code used in our simulations is provided in Appendix B. The multi-gateway case requires an extra step in that each CH identifies the preferred gateway to transmit its aggregated packet. In our research, the preferred gateway is the closer gateway to minimize energy from the wireless transmission.

The desired probability for a node to be chosen as a CH is an input to the algorithm and must be specified. The original author of LEACH performed analysis to determine the optimum value for *p*, *p = 0.5* [1]. This value was determined by running simulations increasing *p* and plotting the rate of energy depletion normalized against the equivalent rate if the network algorithm used was each node communicating directly with the gateway. Their results show that there is an initial steep decline in energy depletion rate of LEACH as *p* increases from zero to approximately five percent. The normalized energy depletion rate of LEACH starts at one because the condition where there are no CHs corresponds to each node communicating its payload directly to the gateway, which is the same as the direct routing case. Similarly, 100 percent of nodes as CHs (*p*=1.0) also corresponds to the same routing process as the Direct case because each node is serving as a CH but contains no serviced nodes, thus both have a normalized energy dissipation of 1.0 in Figure 22. In addition, there is a distinct point where LEACH's normalized energy dissipation is minimized Figure 22, which was determined to be 0.05. As a result, we set our probability for any node to be chosen as a CH as 0.05 (*p*=0.05) in single and multi-gateway simulations.

```
%input sensor network parameters or generate the WSN
%setup plotting formats

for node = 1:n
        S(node).E=Eo %Assign starting Energies
end

for round = 0:rmax
    for i=1:n %check and assign alive nodes to a aliveNode array
        if(S(i).E>=0
            aliveNode(i)=1
        end
    end

    for i = 1:n %do for each node
        temp_rand = rand
        if temp_rand <= (p/(1-p*mod(r,round(1/p))))
            S(i).CH = 1 % the node is chosen as a CH
            S(i).G = round(1/p)-1
        else
            S(i).CH = 0 % the node is not chosen as a CH
        end
    end

    for i = 1:n
        if S(i).G ==0 %for each node not a CH
            %Identify the closest CH to associate with
            %Append the CH array with i
            %Decrement energy for i to transmit to that CH.
        end
    end

    for i = 1:n
        if S(i).G==1 %for each CH
            %decrement energy to recieve all nodes transmission in the cluster
            %decrement energy for the CH to aggregate the packet
            %decrement energy for the CH to transmit aggregated packet
        end
    end

    if mod(r, round(1/p) )==0 %identify the rounds to reset G
        for i=1:n
            S(i).G = 0 % reset the set of nodes been CHs in last 1/p rounds
        end
    end

    % obtain desired statistics for the round
    % Update Plots
end
%plot and save statistitcts for round = 1:rmax
```

Figure 21. Pseudocode for our implementation of LEACH. Complete code is provided in Appendix B.

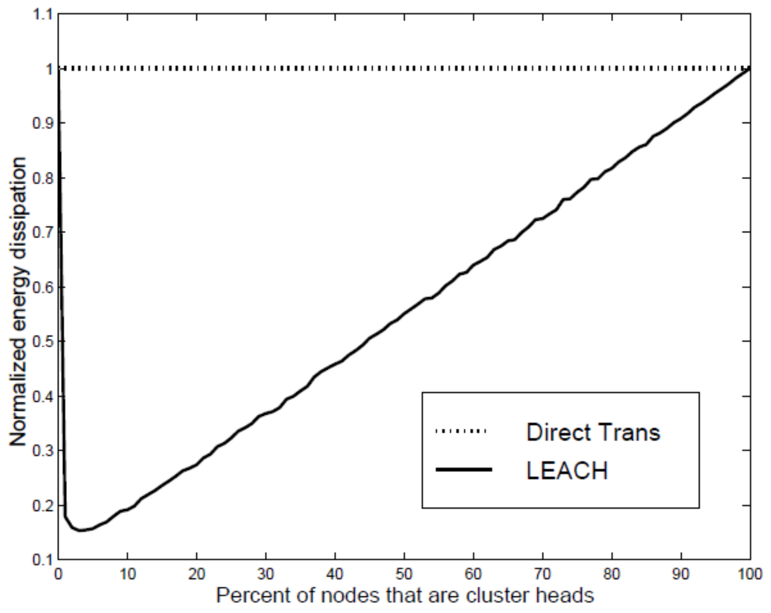


Figure 22. Percentage of CHs in LEACH versus normalized energy dissipation rate with $p = 0.05$; therefore, energy dissipation for LEACH is minimized (from [1]).

E. ZONE CLUSTERING WITH RANDOM CLUSTERHEAD SELECTION

Zone clustering appears less frequently in the literature as compared to LEACH. However, for a tactical network, it may be a preferred networking algorithm because the user can specify how zones are characterized for the network. This yields another aspect of network layer control as compared to LEACH.

The general methods that we use for our zone routing algorithm are based on techniques described in [31]. In [31], the authors utilize a sensor filed comprised of homogenous zones. A sensor in each zone has a probability *p* of becoming a CH during each round. The probability *p* is determined relative to the number of nodes in the zone: $p = 1/(\textit{number of nodes in zone})$. Their methodology is useful for a tactical network in that the objective of zoning in a WSN is to ensure that CHs are uniformly selected through the network.

Our zone clustering algorithm divides the sensor field into *z* equal zones. Equal zones were chosen because the distribution of nodes is uniform in the field. Equal zones span along the Cartesian *x-axis* to create *z* vertical rectangular zones. As shown in Figure 23, our simulation uses *z* = five zones. Five zones were chosen to provide a comparison with the LEACH algorithm. Recall that in our LEACH algorithm, the probability of any node being chosen a CH is p=0.05. Thus, in a 100 node network, we would on average to have five CHs. To ensure there are five CHs for our zone clustering algorithm, we must have five zones and each zone is only allowed to have only one CH.

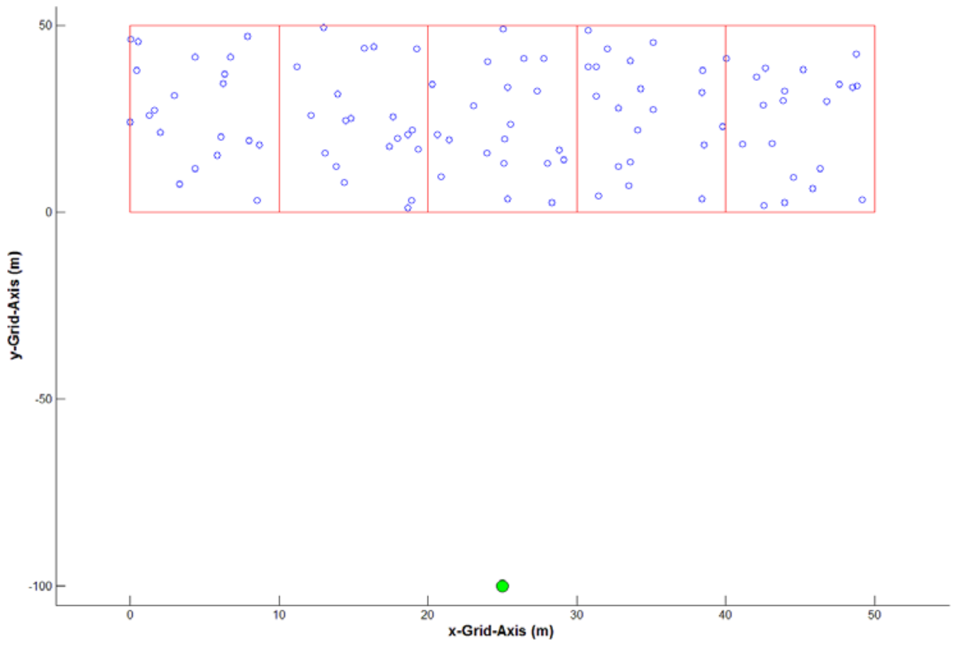
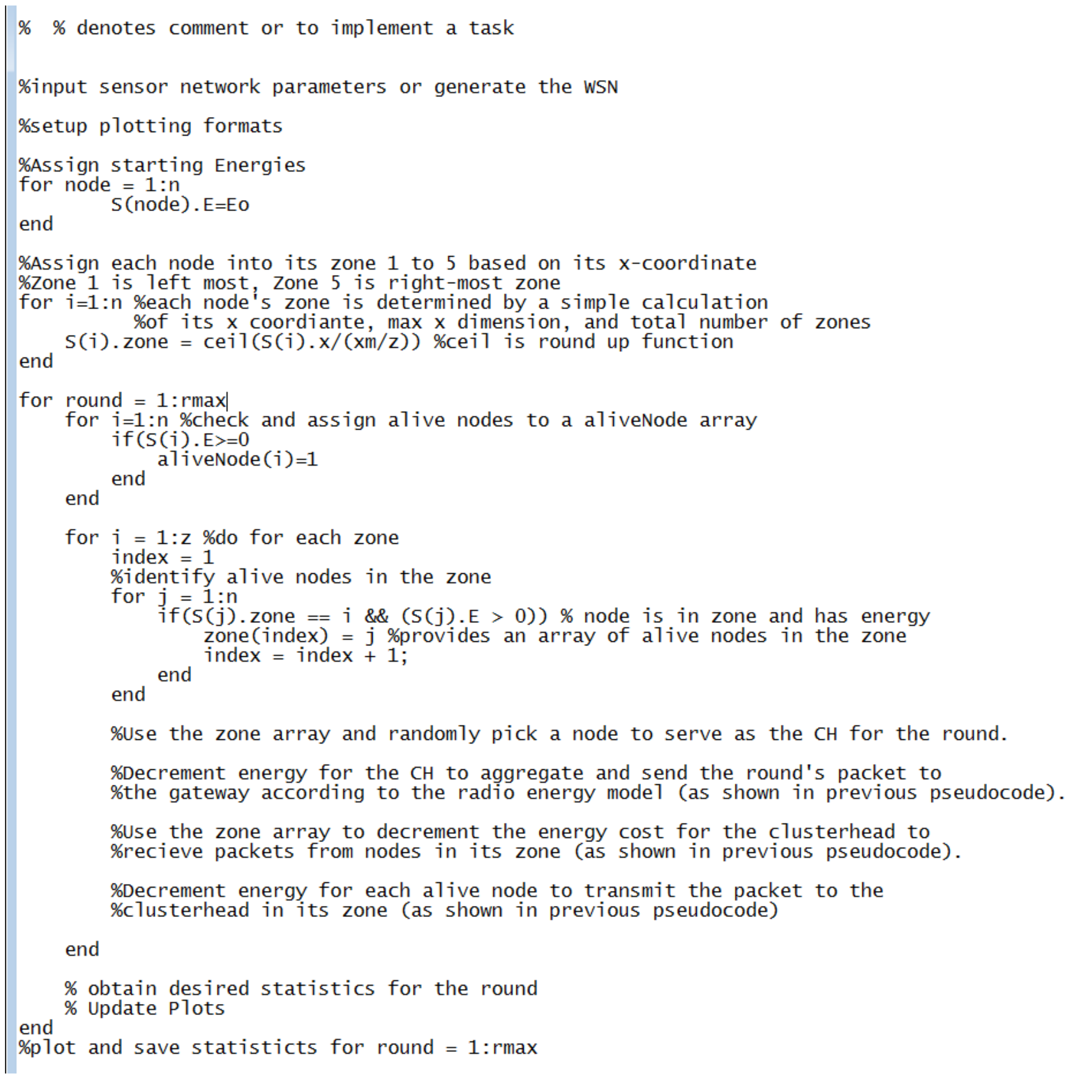


Figure 23. Our single gateway (green circle), 100 node (blue outlined circles) partitioned into five zones along the *x*-grid axis.

The pseudocode for our zone based simulation is shown in Figure 24. The algorithm is executed in three phases: 1) setup, 2) CH election for each zone, and 3) communications between nodes and the gateway. Network layer functions take place in phases 2 and 3. The setup phase utilizes the user's inputs for the WSN (or creates the WSN) and partitions the network into the required number of zones. Partitioning the network in zones essentially creates several smaller WSNs that all utilize the same gateway. The zone assigned to any node is described by the node's *x*-coordinate in the field, which is shown in Figure 24.



```
%  % denotes comment or to implement a task


%input sensor network parameters or generate the WSN

%setup plotting formats

%Assign starting Energies
for node = 1:n
        S(node).E=Eo
end

%Assign each node into its zone 1 to 5 based on its x-coordinate
%Zone 1 is left most, Zone 5 is right-most zone
for i=1:n %each node's zone is determined by a simple calculation
          %of its x coordiante, max x dimension, and total number of zones
    S(i).zone = ceil(S(i).x/(xm/z)) %ceil is round up function
end

for round = 1:rmax
    for i=1:n %check and assign alive nodes to a aliveNode array
        if(S(i).E>=0
            aliveNode(i)=1
        end
    end

    for i = 1:z %do for each zone
        index = 1
        %identify alive nodes in the zone
        for j = 1:n
            if(S(j).zone == i && (S(j).E > 0)) % node is in zone and has energy
                zone(index) = j %provides an array of alive nodes in the zone
                index = index + 1;
            end
        end

        %Use the zone array and randomly pick a node to serve as the CH for the round.

        %Decrement energy for the CH to aggregate and send the round's packet to
        %the gateway according to the radio energy model (as shown in previous pseudocode).

        %Use the zone array to decrement the energy cost for the clusterhead to
        %recieve packets from nodes in its zone (as shown in previous pseudocode).

        %Decrement energy for each alive node to transmit the packet to the
        %clusterhead in its zone (as shown in previous pseudocode)

    end

    % obtain desired statistics for the round
    % Update Plots
end
%plot and save statistics for round = 1:rmax
```

Figure 24. Pseudocode for zone routing algorithm simulation with random CH election in each zone.

Once all nodes are assigned to a zone, we begin the simulation at round one and complete the simulation at the maximum desired round. In each round, the set of live nodes for each zone is identified, and the CH is chosen based on a random assignment from this set. Each node in the zone then transmits its *L*-bit packet to the zone's CH and its energy is decremented according to our radio energy model. The CH for the zone then aggregates all the messages from the nodes in the zone and transmits the aggregated message to the gateway. The process is repeated for each zone at each round. The pseudocode for the multi-gateway configuration is similar to Figure 24, except for each zone at each round, the CH chooses the optimal gateway to which the aggregated data is sent. In our simulations, the optimal gateway corresponds to the closest gateway to the CH, which minimizes the energy required for the transmission.

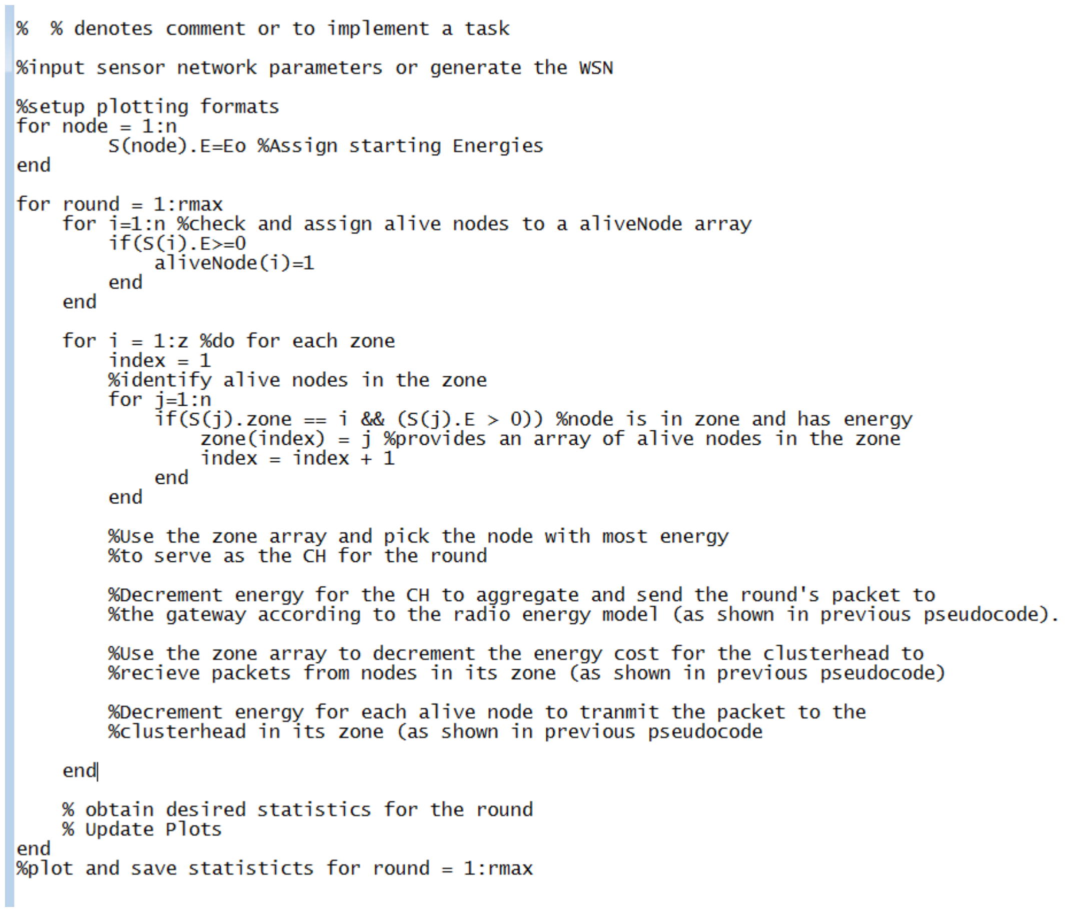
For ease of simulation, our MAC based approach uses rounds where each node sends a packet to the gateway sometime within each round. All the packets must be received by the CH in each round, at which point the CH aggregates all *L*-bit packets into one collective *L*-bit packet and transmits the aggregated packet to the gateway at a later point in the round. We assume the MAC process is similar to that described for LEACH, in which CHs are assigned a TDMA timeslot for transmission to the gateway and CHs are assigned code-division multiple access (CDMA) schemes for intra-cluster communications to prevent interference with other zones. Each zone is assigned a CDMA scheme for the life of the network. Another possible strategy is to develop an aggregation scheme where CHs transmit the aggregated packet to the gateway using a more energy efficient strategy. This aspect is not simulated but is an opportunity for future work.

F. ZONE CLUSTERING WITH ENERGY EFFICIENT CLUSTER HEAD SELECTION

The zone clustering case described in Section E chooses the CH for each zone randomly. A clustering algorithm that partitions nodes into specific zones is an energy saving technique when compared to the previous LEACH algorithm because there is a lower maximum distance that any node must transmit to reach its CH. Our implementation of the zones guarantees a nearby CH in the zone as compared to that of LEACH. In LEACH the nearest CH may be on the other side of the network since the criteria for a node to be elected as a CH may have only been met randomly on the other side of the field (we show this effect later). After observing the results that we present later in Chapter V, we noted significant differences in the energy distribution of the nodes in the network. The differences in energy levels across the WSN caused some nodes to die out earlier and some nodes to die out later.

As a result of our observation of how nodes’ energy levels are depleted (visual observation based on energy distributions plotted later in Chapter V), we modified our zone based CH election criteria based on a cross-layer implementation between the networking layer and the physical layer. Specifically, in any given round, if the highest energy node is chosen to be the CH, individual node energy depletion rates are minimized with the battery levels in any zone depleting at a uniform rate.

To accomplish this strategy, we modify our zone routing pseudocode in Figure 24 to revise the process of electing the CH in each zone at each round and is shown in Figure 25. Instead of randomly choosing the CH from the live nodes in the zone, we choose the CH that has the maximum energy level in the zone. Based on this election criteria, nodes that are in a more preferred location (a location that minimizes energy depletion rate such as locations closer to the gateway) are chosen to be the CH for the zone more than those in a less preferred location (a location farther away from the gateway).



```
%  % denotes comment or to implement a task

%input sensor network parameters or generate the WSN

%setup plotting formats
for node = 1:n
        S(node).E=Eo %Assign starting Energies
end

for round = 1:rmax
    for i=1:n %check and assign alive nodes to a aliveNode array
        if(S(i).E>=0
            aliveNode(i)=1
        end
    end

    for i = 1:z %do for each zone
        index = 1
        %identify alive nodes in the zone
        for j=1:n
            if(S(j).zone == i && (S(j).E > 0)) %node is in zone and has energy
                zone(index) = j %provides an array of alive nodes in the zone
                index = index + 1
            end
        end

        %Use the zone array and pick the node with most energy
        %to serve as the CH for the round

        %Decrement energy for the CH to aggregate and send the round's packet to
        %the gateway according to the radio energy model (as shown in previous pseudocode).

        %Use the zone array to decrement the energy cost for the clusterhead to
        %recieve packets from nodes in its zone (as shown in previous pseudocode)

        %Decrement energy for each alive node to tranmit the packet to the
        %clusterhead in its zone (as shown in previous pseudocode

    end

    % obtain desired statistics for the round
    % Update Plots
end
%plot and save statistics for round = 1:rmax
```

Figure 25. Pseudocode for zone routing protocol with energy efficient CH election.

In practice, electing the highest energy node to be the CH during each round in each zone requires additional processing by the gateway to perform CH election. Our simulations perform this aspect automatically with the assumption that it is normally performed by the gateway. One possible implementation of this strategy in practice is that the aggregated packet sent to the gateway includes updated node energies for each node in the zone in the packet header. The gateway, being unconstrained by energy, can then estimate the amount of energy consumed by the CH to transmit the aggregated packet. The gateway can then decide which node in each zone should be assigned the CH for the next round and broadcast this information back to the WSN. In our case, since each node is within communication range of the gateway, every node in every zone will know who its CH is for the next round.

G. APPLICATION LAYER

We employ two application layer strategies, 1) a constant bit rate (CBR) generator and 2) a data aggregation application. The CBR application allows each node to send an L=2000 bit message to the gateway at each round. We are not concerned with the contents of each message. Instead, we only care that messages are produced so that we may observer how energy is depleted throughout the network due to the network routing algorithm in use. Our CBR technique is most analogous to that of the user datagram protocol (UDP) used in the modern day Internet. In UDP, a connection-less link is used in which the source node does not obtain any acknowledgement that its packets were successfully delivered to the gateway. Eliminating the acknowledgement precludes additional transmissions providing a further load balancing strategy.

Only clustering mechanisms use data aggregators in our scenarios. Data aggregation requires energy to perform the signal compression, which must be accounted for. We adopt a similar technique, used in the literature, which applies an energy cost to the data aggregator for the task of aggregating all the data during a round [1], [2], [38]–[40]. The node performing data aggregation is always the CH, and a data aggregation constant *EDA* is used to account for the energy to compress messages into one final

L=2000 bit message. The data aggregation constant used in our scenarios is consistent with the literature (*EDA* = 5 nJ/bit) and results in an aggregation cost of *EDA* × *L* [1], [2], [38]–[40].

H. CHAPTER IV SUMMARY

Our network layer algorithms for simulation in Chapter V and VI were described in Chapter IV. We utilized certain algorithms from the literature that were analyzed in single gateway WSNs. We apply them to our single gateway and multi-gateway scenarios. Algorithms we utilized from the literature are where each node transmits its information directly to the gateway(s) (direct), MTE routing utilizing Dijkstra’s algorithm (MTE), low energy adaptive cluster head (LEACH) routing, and zone clustering with random CH assignment (zone). We then described an energy-efficient zone clustering algorithm (EZone) that we devised as a result of our research.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SIMULATIONS AND RESULTS

In this chapter, we provide the results for the algorithms described in Chapter IV. All simulations were executed using MATLAB. We start by describing metrics of interest for our simulations. We then provide results for each algorithm individually for single and multi-gateway simulations and then show and discuss how all results compare to each other. Each algorithm was executed on the same WSN that was shown previously (Figure 9 and Figure 10) for the single and multi-gateway cases, which provide a means to compare the networking layer performance of each case with one another. We then let each algorithm run separately thousands of times, each time regenerating a WSN with 100 uniformly distributed nodes. We capture die out metrics and model them as random variables. The purpose of the random variable testing is to draw further conclusions on our results from many network simulations instead of the single WSNs scenario we present in this chapter. Random variable modeling is presented in Chapter VI.

A. SIMULATION METRICS

Our simulation metrics are based on the motivations of this thesis: to identify the performance advantages of load balancing techniques of various network layer algorithms on the lifetime of a WSN in single gateway and multi-gateway configurations. To accomplish this, we model the WSN for its total useful life, observing how and why the network dies out over time. For the clustering mechanisms, we are interested in the number of clusters at each round, the distribution of clusters across the sensor field, and how CH election criteria impacts the distribution of the network dying out over time. More specifically, we track the energy level of each node at each round to produce the metrics displayed in Table 3. Tracking the energy level of each node allows the calculation of a discrete energy variance at each round, which provides realization of the magnitude of the energy difference between all the nodes each round. The energy variance is given by

$$Var(E) = \frac{1}{n}\sum_{i=1}^{n}(e_i - \mu)^2 . \tag{14}$$

In Equation. 14, E is the random variable for energy, n represents the number of live nodes in the round, e_i is the energy of the i^{th} live node in the round, and μ is the mean energy for the round.

Table 3. Simulation metrics for WSN network simulations.

Summary of Simulation Metrics
1. Energy level of each node at each round
2. Number of alive nodes at each round
3. Variance of the WSN energy distribution
4. Total WSN energy level at each round
5. The round and location of the first dead node
6. The round and distribution of nodes when 10% of the nodes are dead
7. The round and distribution of nodes when 50% of the nodes are dead
8. The round and distribution of nodes when 80% of the nodes are dead
9. Number and location of CHs at each round (clustering algorithms only)

The service life of a WSN is subject to opinion. Some would contest that the WSN is completely in service as long as *all* nodes in the network are live while some pick a percentage of nodes that must be alive during each round for the network to be considered at full service. The metrics in Table 3 consider both perspectives since we account for the round and location of the first dead node and the round and locations of 10 percent, 50 percent and 80 percent of nodes being dead.

B. DESCRIPTION OF PLOT RESULTS

Every round we generate several plots to characterize energy consumption and the distribution of live and dead nodes in the network. We update three plots each round dynamically. The first plot is a bar plot that provides the energy of each node from 1 to 100 where node 1 is the closest node to x=0 (the *y-axis*) and node 100 is on the other side of the sensor field closest to the line x=50 m. The second plot is a three-dimensional energy stem plot where each stem is located in the position of the node in the field, and the height of the stem represents the amount of residual battery energy available. The final plot is an overhead of the sensor field including the gateway. We refer to the first plot as the energy bar plot, the second plot as the energy stem plot and the third plot as the node distribution plot. These plots are interactive and contain information that updates each round. For example, the energy stem plot is green, and the elevation (energy level) decreases each round corresponding to energy consumption. When the stem reaches zero energy (the floor), the green bubble changes to red to indicate the node has died. The node distribution plot shows live nodes as a circle with a blue outline and dead nodes as solid red bubbles. The node distribution plot also contains the round from which all three plots are drawn. The energy bar and stem plots are stacked on top of each other on the left hand side of the subplot, and the node distribution plot is on the right side of the subplot. For each simulation, this is plotted four times corresponding to the round the first node dies and the round that 10 percent, 50 percent, and 80 percent of nodes have died.

The node distribution plots for our clustering simulations indicate which nodes are the CHs for the round the plot is drawn from. The CHs are indicated by a blue asterisk that fills the nodes that are outlined in blue on the plots. For LEACH, this allows the reader to see how LEACH inefficiently partitions the sensor field with CHs without regard to any spatial arrangement. For the zone routing algorithms (Zone and EZone), there exists only one node in each zone during each round that serves as the CH. As a result, the reader can observe one blue asterisk in each zone during each round. Our display of CHs involves one caveat for LEACH and zone routing algorithms. In some cases, the CH asterisk indicator is plotted over with a solid red circle because its energy was fully depleted in its last round as the CH. Our plots are drawn at the end of each round; thus, if a node is dead and it was the CH during the round, it is depicted as a dead node.

At the conclusion of each simulation, we obtain three additional plots that provide information from the start of the simulation to the end of the simulation. The simulation is over after the round in which the last node died. We plot the total WSN energy level during each round where the total energy is the sum of individual node energies. There is a distinct linear region of this plot that allows extraction of the WSN energy depletion rate when all nodes are alive. We then plot the energy variance versus round for the WSN throughout the simulation, and the final plot shows the number of live nodes versus transmission round. After all individual results are plotted, we conclude the chapter by plotting the total WSN energy, energy variance, and distribution of alive nodes for each algorithm on common plots to compare network layer performance.

C. DIRECT TRANSMISSION TO THE GATEWAY

1. Single Gateway

Direct transmission to the gateway routing initiates each node to send an L=2000 bit packet to the gateway during each round. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 26 through Figure 29, respectively, for the single gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 30 through Figure 32, respectively. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 356, 410, 652, and 939, respectively. The energy stem plot and node distribution plots demonstrate that nodes farthest from the gateway die out first, quickly eliminating service coverage in areas farthest from the gateway. This die out topology is expected because our physical layer depletes energy proportional to transmission distance.

The energy stem plot demonstrates that nodes closest to the gateway remain in service longer than nodes farther from the gateway because our physical layer depletes energy proportional to distance squared for free space (direct) propagation and d^4 for multi-path propagation. Since our d_0 parameter is approximately 87 m and the gateway is at least 100 m below the closest node, all propagation in this simulation is multi-path propagation. The same is also true for the multi-gateway simulation since the additional gateway is at least 100 m above the closest node. The energy depletion rate of the network during the linear region of Figure 30 is 0.0798 J/round.

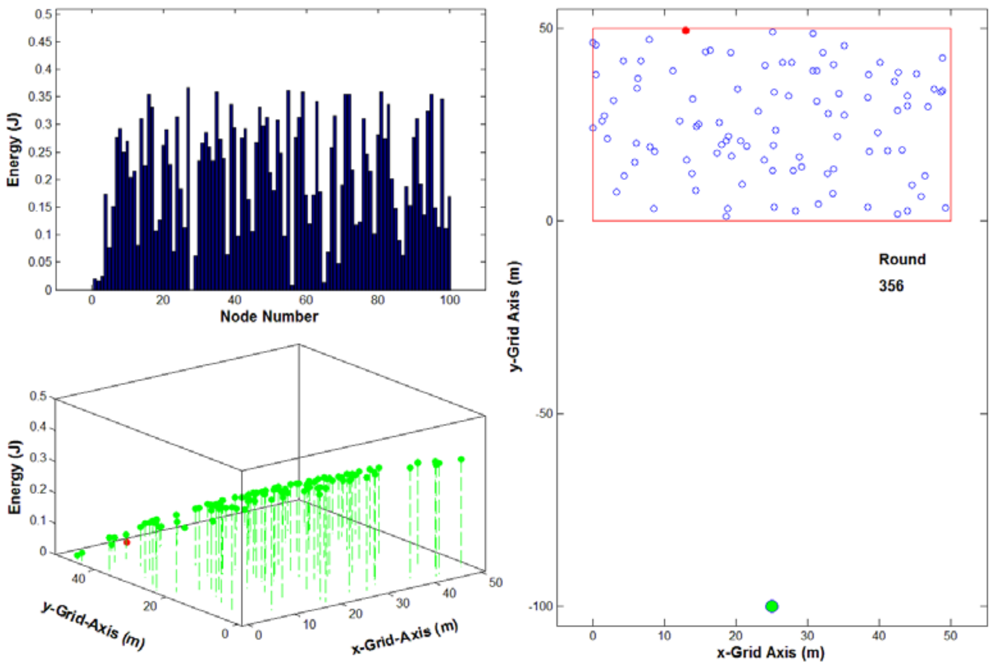


Figure 26. Direct routing in a single gateway WSN illustrating first node dead die out topology versus transmission round versus round and energy distributions.

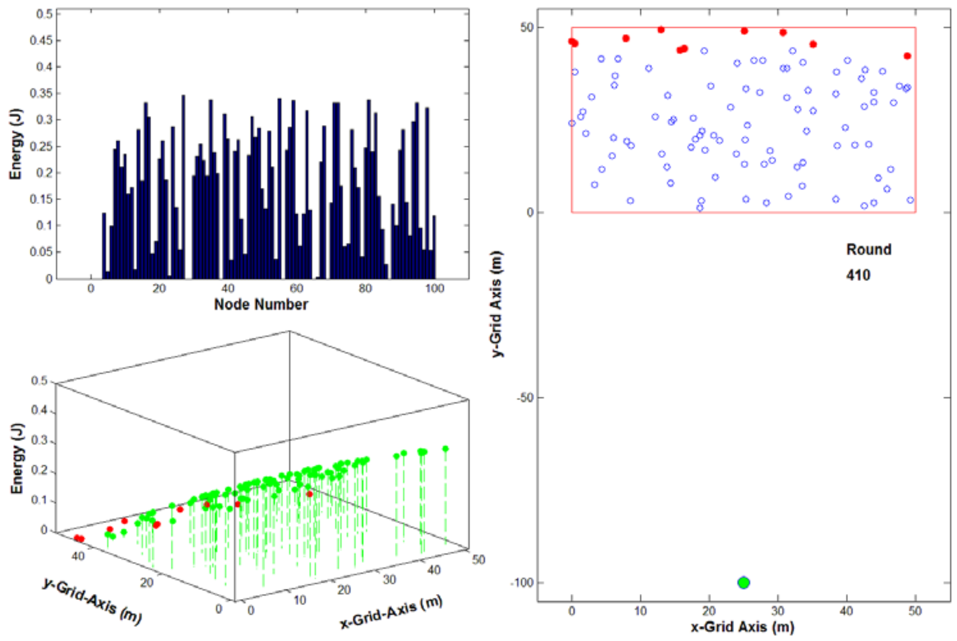


Figure 27. Direct routing in a single gateway WSN illustrating 10 percent nodes dead die out topology versus transmission round and energy distributions.

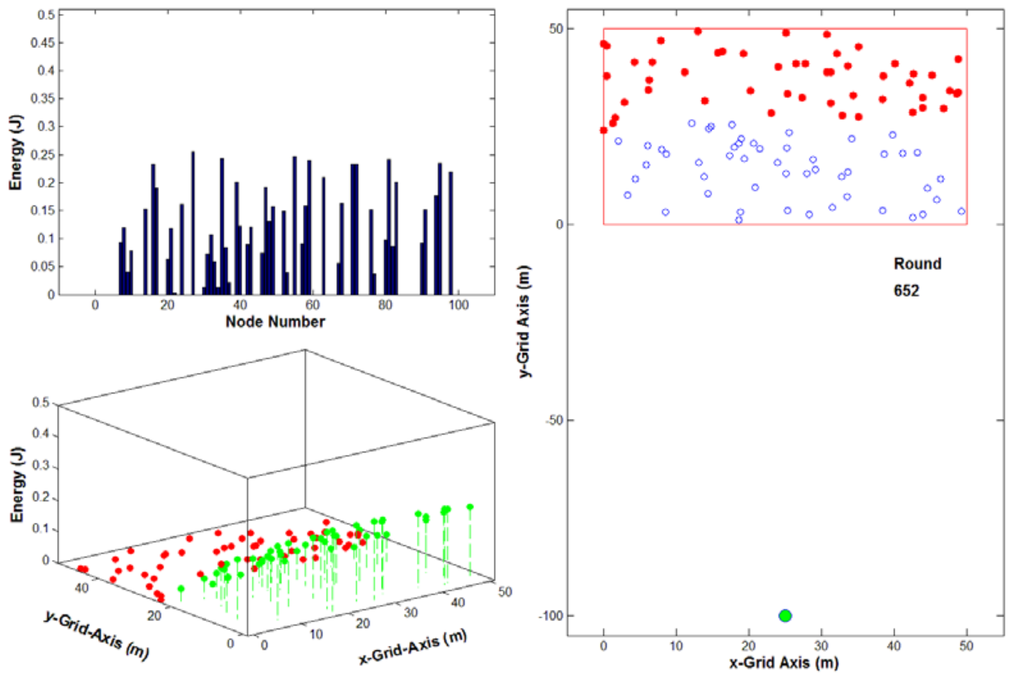


Figure 28. Direct routing in a single gateway WSN illustrating 50 percent nodes dead die out topology versus transmission roundand energy distributions.

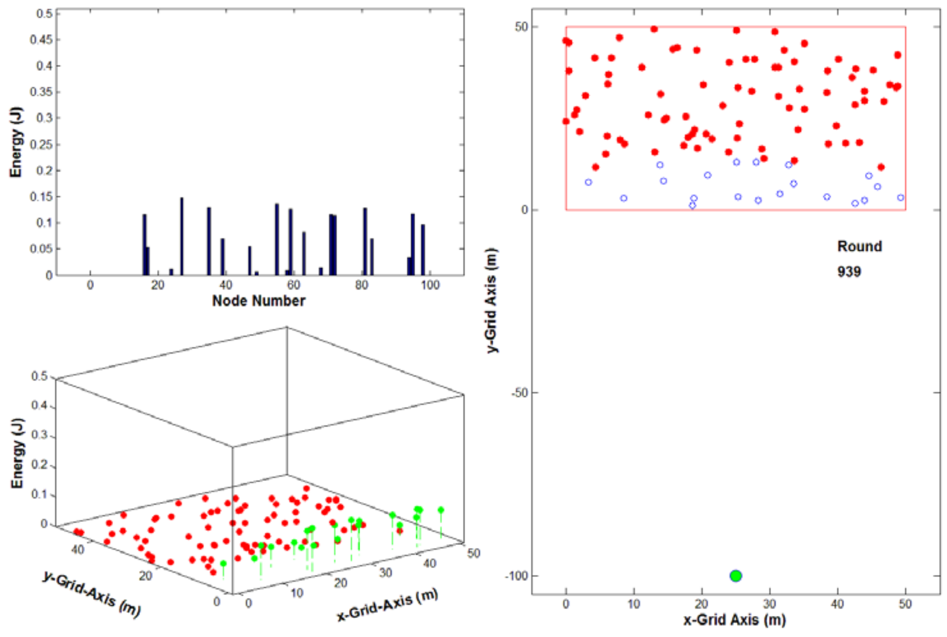


Figure 29. Direct routing in a single gateway WSN illustrating 80 percent nodes dead die out topology versus transmission roundand energy distributions.

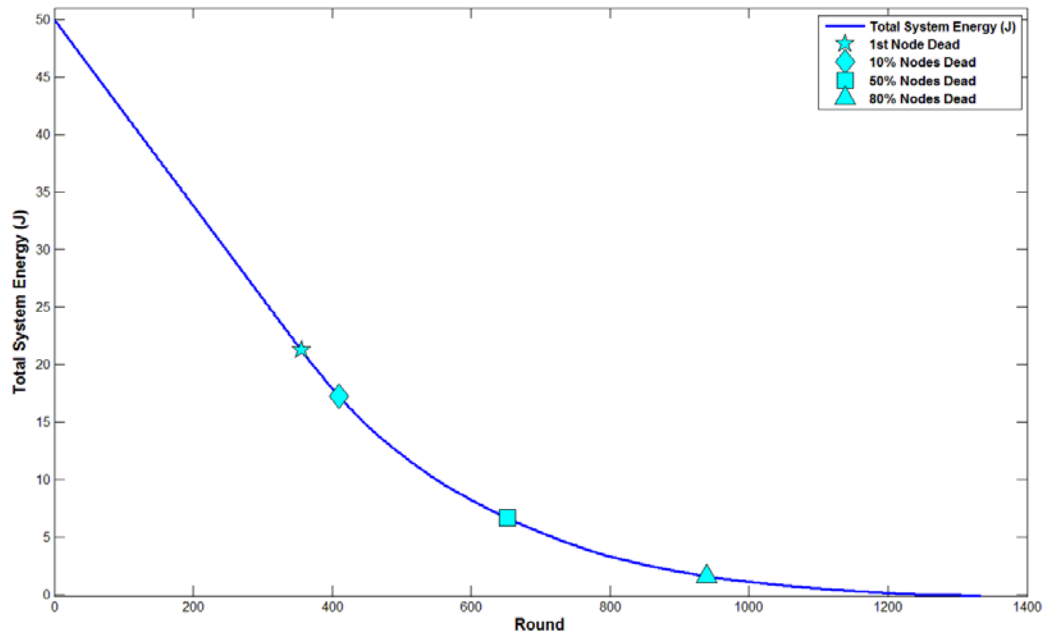


Figure 30. Direct routing in a single gateway WSN. The total WSN energy versus transmission round is illustrated.

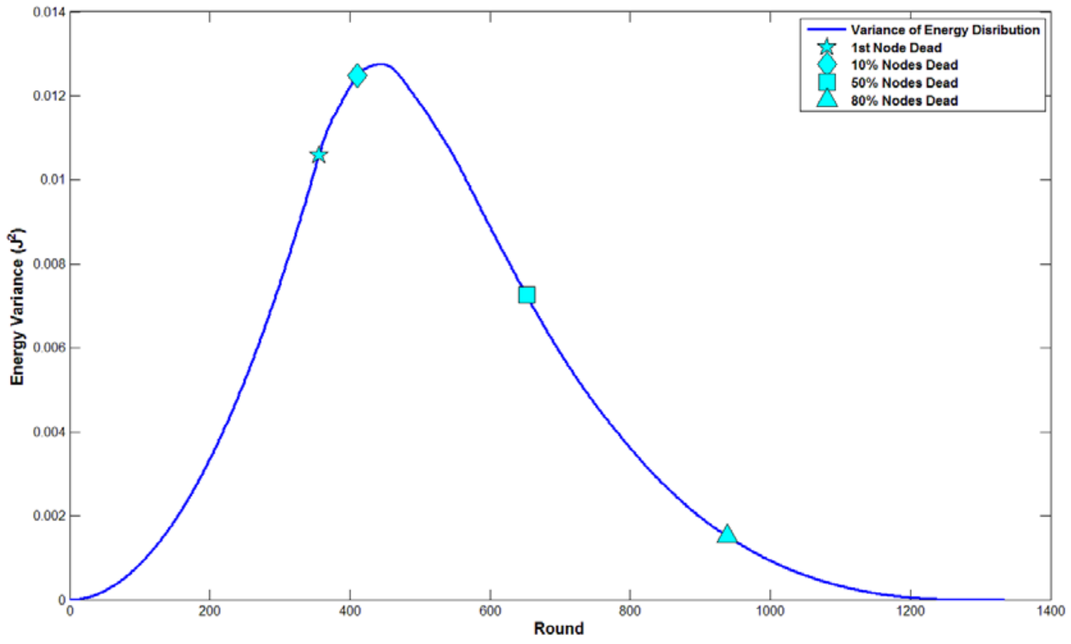


Figure 31. Direct routing in a single gateway WSN. The WSN energy variance versus transmission round is illustrated.

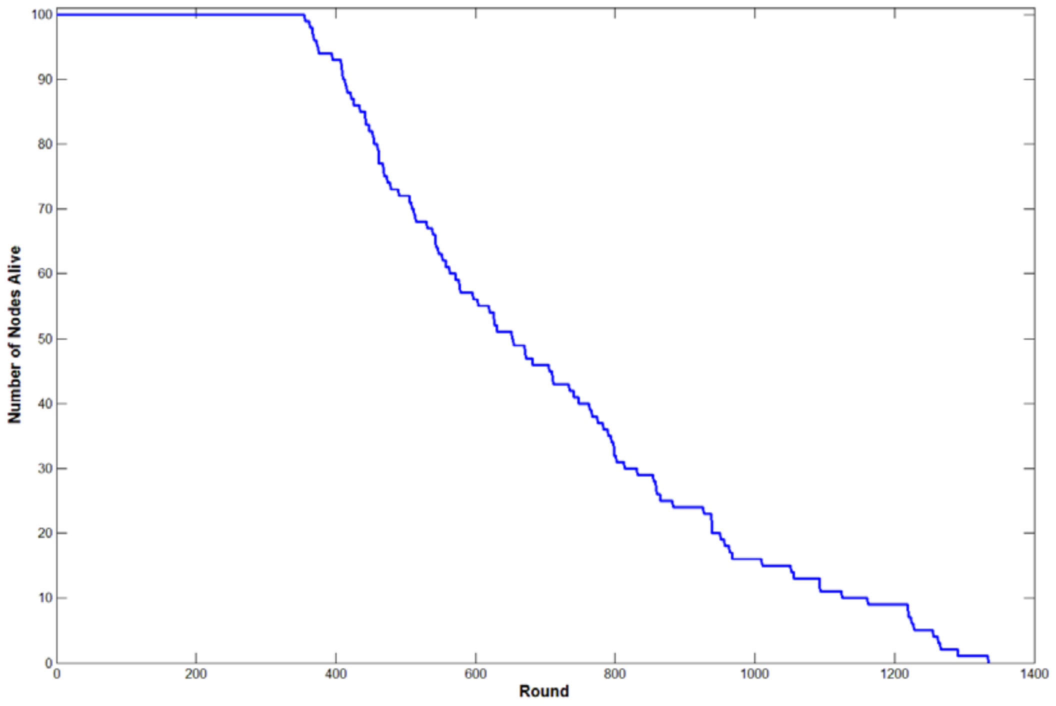


Figure 32. Direct routing in a single gateway WSN. The number of nodes alive versus transmission round is illustrated.

2. Multi-gateway

Direct transmission to multi-gateway routing initiates each node to send an *L*=2000 bit packet to the closest gateway during each round. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 33 through Figure 36, respectively, for the multi-gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 37 through Figure 39, respectively. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 652, 712, 911, and 1128, respectively. This corresponds to a percent increase of 83 percent, 74 percent, 40 percent, and 20 percent, respectively, when compared to the single gateway die out statistics for the same algorithm. The energy stem plot and node distribution plots demonstrate that nodes farthest from the closer gateway die out first, quickly eliminating service coverage in the center region of the field along the line *y*=25 m. This is expected since the physical layer depletes energy based on required transmission distance and nodes along the line *y*=25 m are farthest away from the gateway in the multi-gateway case. The energy depletion rate of the network during the linear region of Figure 37 is 0.0554 J/round, which corresponds to a 31 percent reduction in network energy depletion rate compared to the single gateway case.

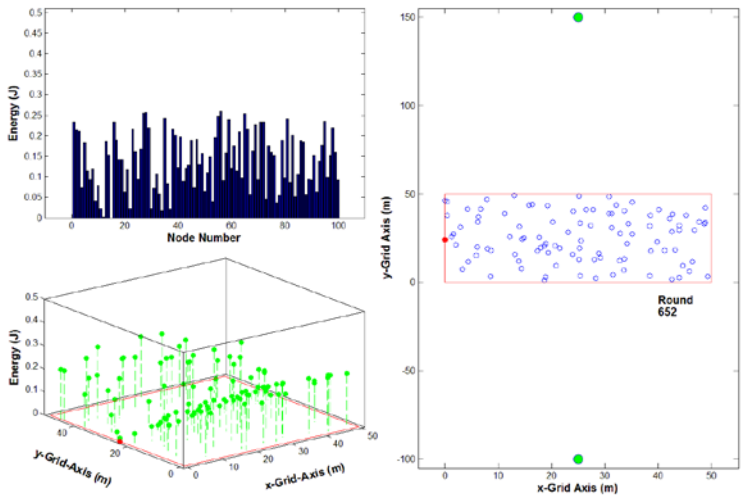


Figure 33. Direct routing in a multi-gateway WSN illustrating first node dead die out topology versus transmission round and energy distributions.

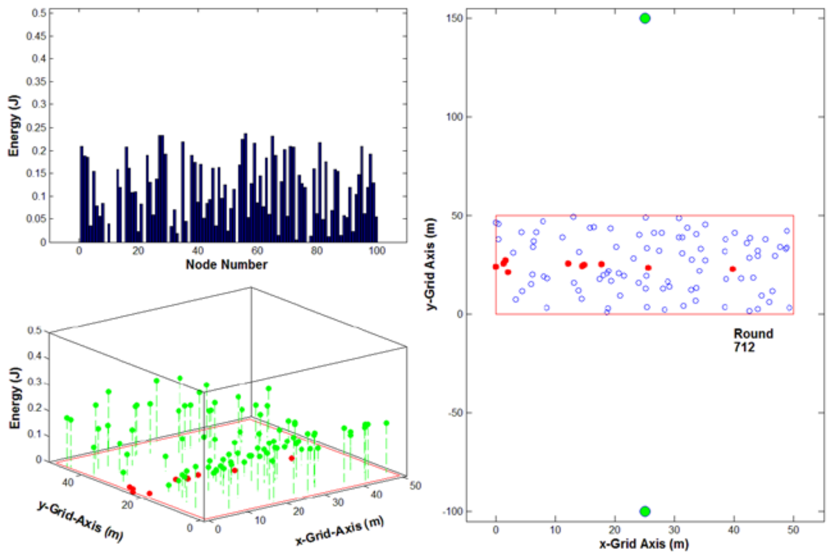


Figure 34. Direct routing in a multi-gateway WSN illustrating 10 percent nodes dead die out topology versus transmission round and energy distribution.

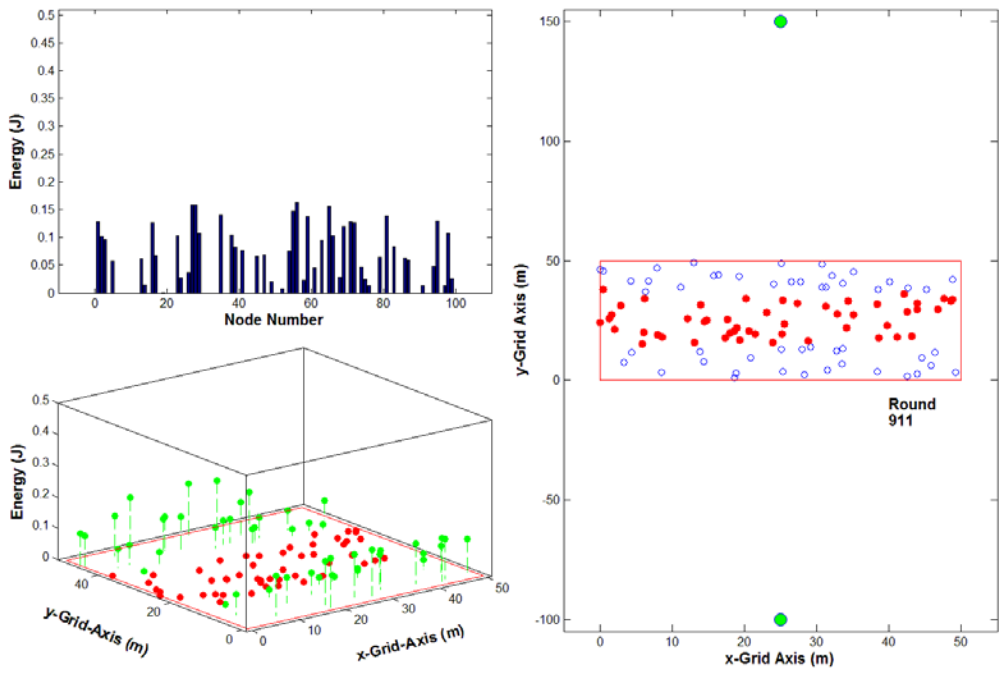


Figure 35. Direct routing in a multi-gateway WSN illustrating 50 percent nodes dead die out topology versus transmission round and energy distribution.

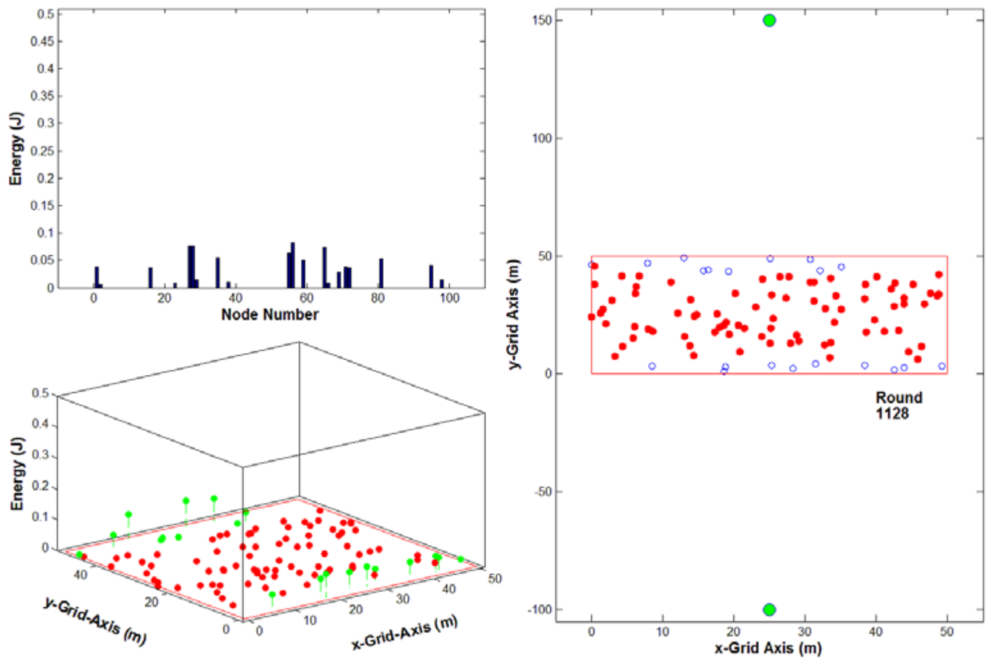


Figure 36. Direct routing in a multi-gateway WSN illustrating 80 percent nodes dead die out topology versus transmission round and energy distribution.

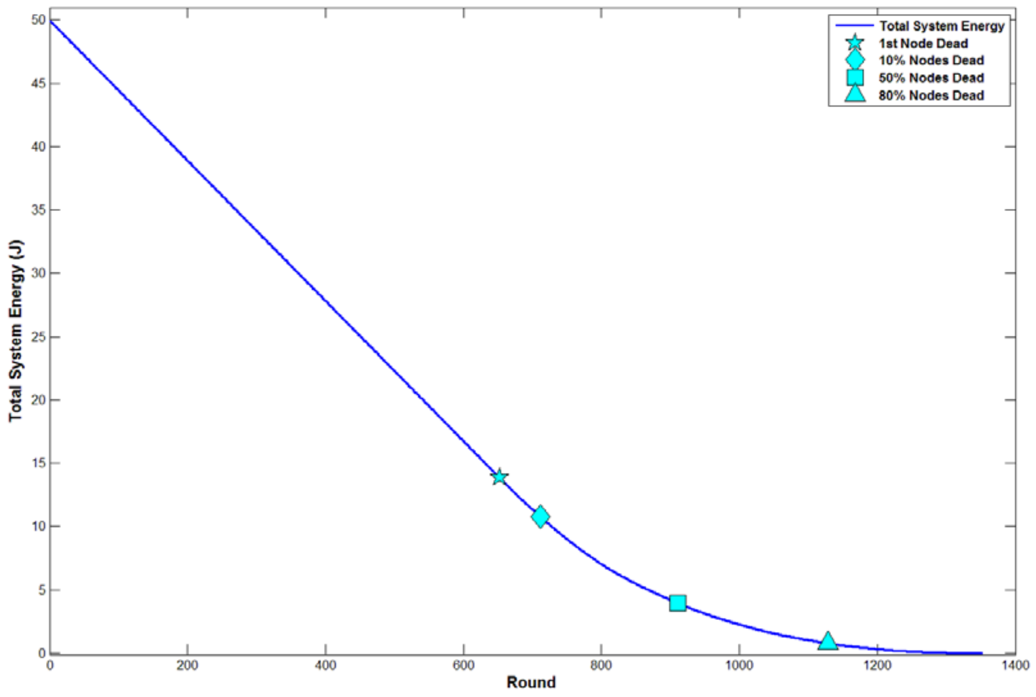


Figure 37. Direct routing in a multi-gateway WSN. The total WSN energy versus transmission round is illustrated.

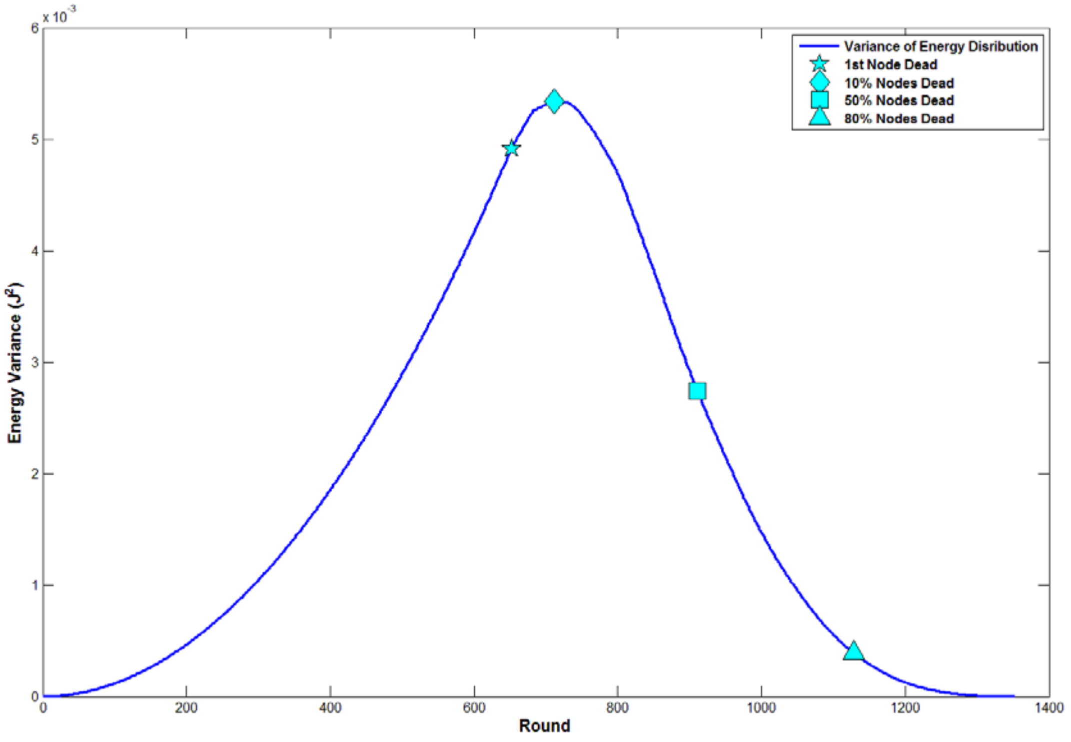


Figure 38. Direct routing in a multi-gateway WSN. The WSN energy variance versus transmission round is illustrated.

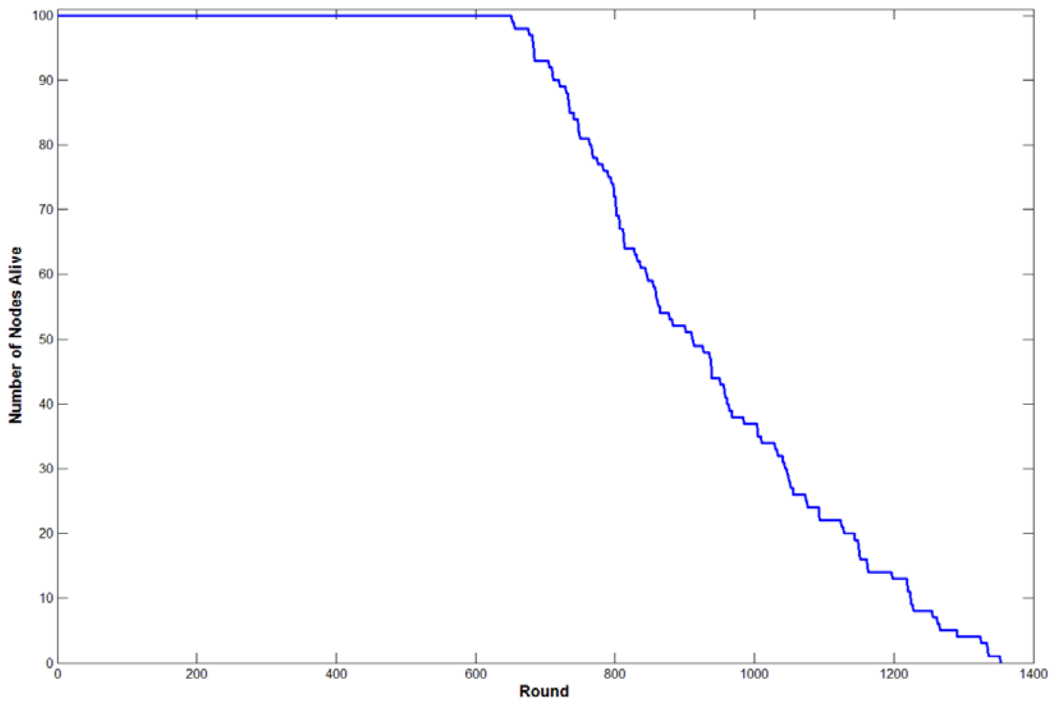


Figure 39. Direct routing in a multi-gateway WSN. The number of nodes alive versus transmission round is illustrated.

D. MTE WITH DIJKSTRA (MTE)

1. Single Gateway

Minimum transmission energy with Dijkstra routing initiates each node to send an L=2000 bit packet to the gateway during each round utilizing a route through neighboring nodes that minimizes the d^2 propagation link cost to the gateway. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 40 through Figure 43, respectively, for the single gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 44 through Figure 46, respectively. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 11, 77, 199, and 354, respectively. The energy stem plot and node distribution plots demonstrate that nodes closest to the gateway die out first and then fan out as subsequent live nodes closest to the gateway become the hot-nodes.

This quickly eliminates service coverage in those areas as expected but is a negative aspect of MTE routing in WSNs. The energy depletion rate of the network during the linear region of Figure 44 is 0.2140 J/round.

The energy bar and stem plots in Figure 40 through Figure 43 reveal a large variation in energy across all nodes. Nodes that are farthest away from the gateway are not used by their peers as frequently for routing, thus their energy is preserved. This creates a large energy variance and the quickest die out for all results collected in this thesis. The first node dies out quickest using this algorithm in part because there is no data aggregation strategy in place. This increases the transmission energy required of the hot-node at each round, thus fully depleting its battery power quickly. Since the first node dies out quickly, the timeframe for a linear energy depletion rate is also low (Figure 44). Also, since nodes farthest from the gateway are not utilized compared to nodes closer to the gateway, they remain in service the longest (Figure 46).

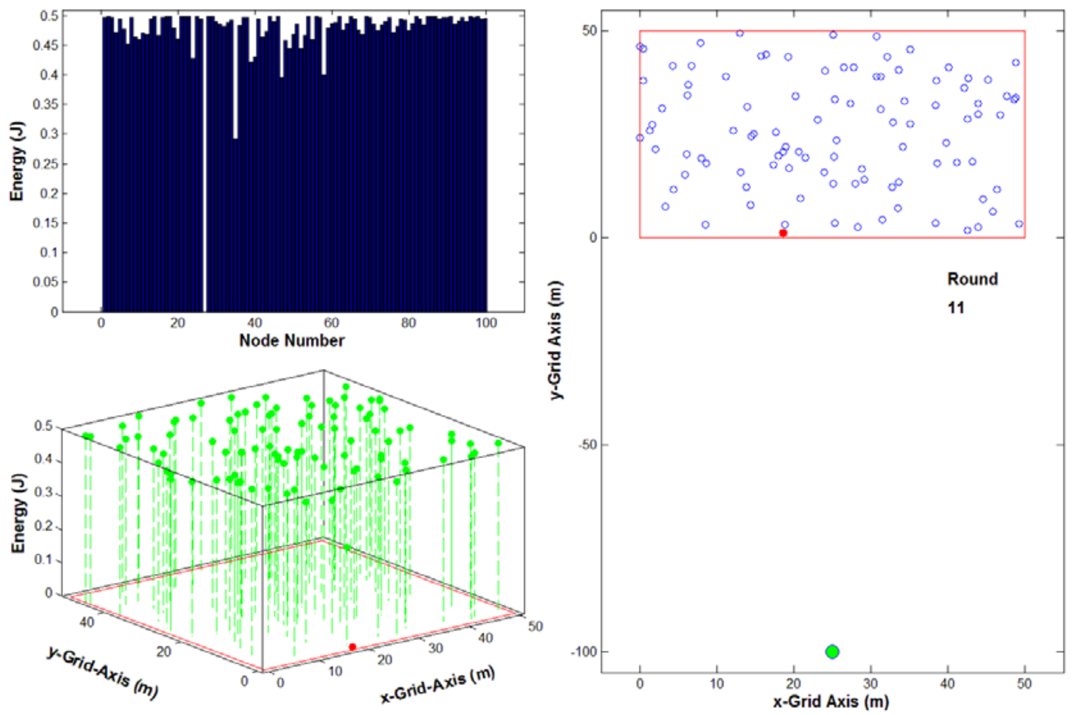


Figure 40. MTE routing in a single gateway WSN illustrating first node dead die out topology versus transmission roundand energy distribution.

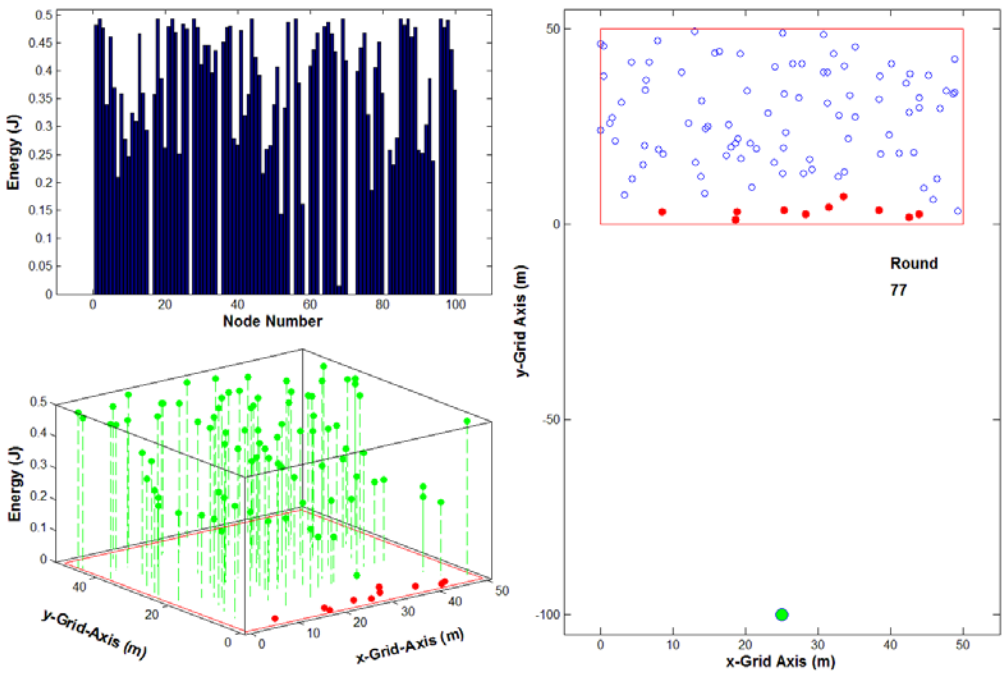


Figure 41. MTE routing in a single gateway WSN illustrating 10 percent nodes dead die out topology versus transmission roundand energy distribution.

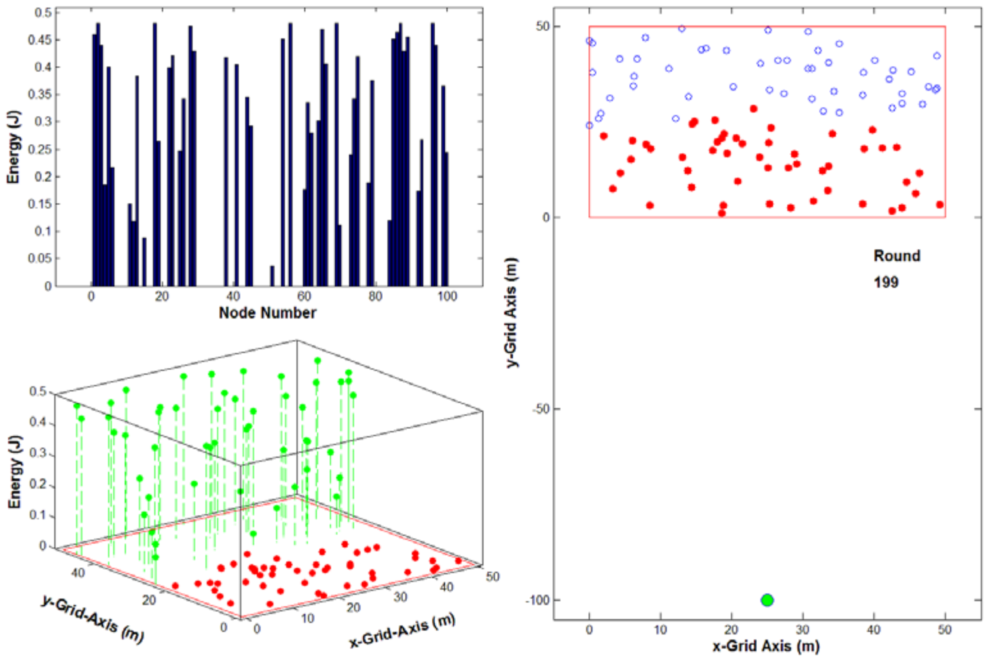


Figure 42. MTE routing in a single gateway WSN illustrating 50 percent nodes dead die out topology versus transmission round and energy distribution.

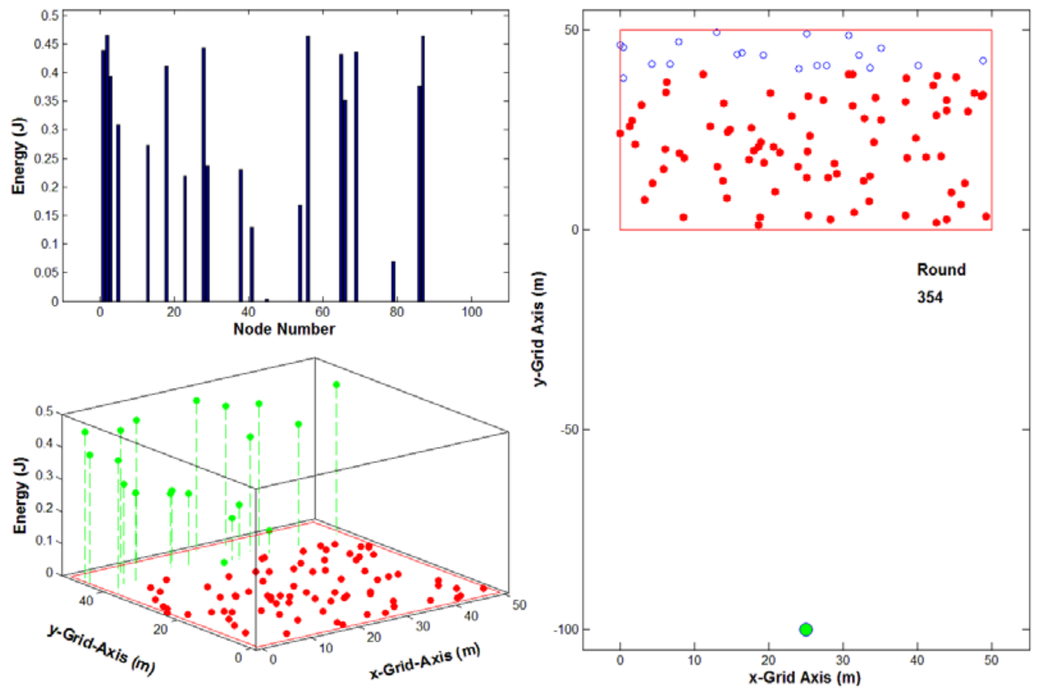


Figure 43. MTE routing in a single gateway WSN illustrating 80 percent nodes dead die out topology versus transmission round and energy distribution.

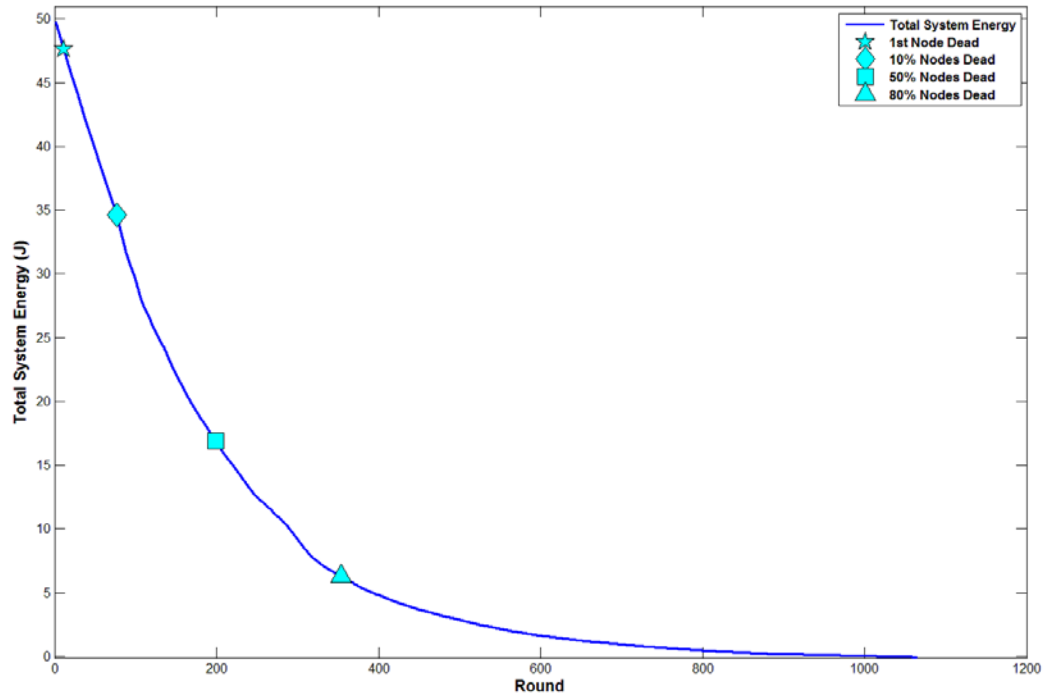


Figure 44. MTE routing in a single gateway WSN. The total WSN energy versus transmission round is illustrated.

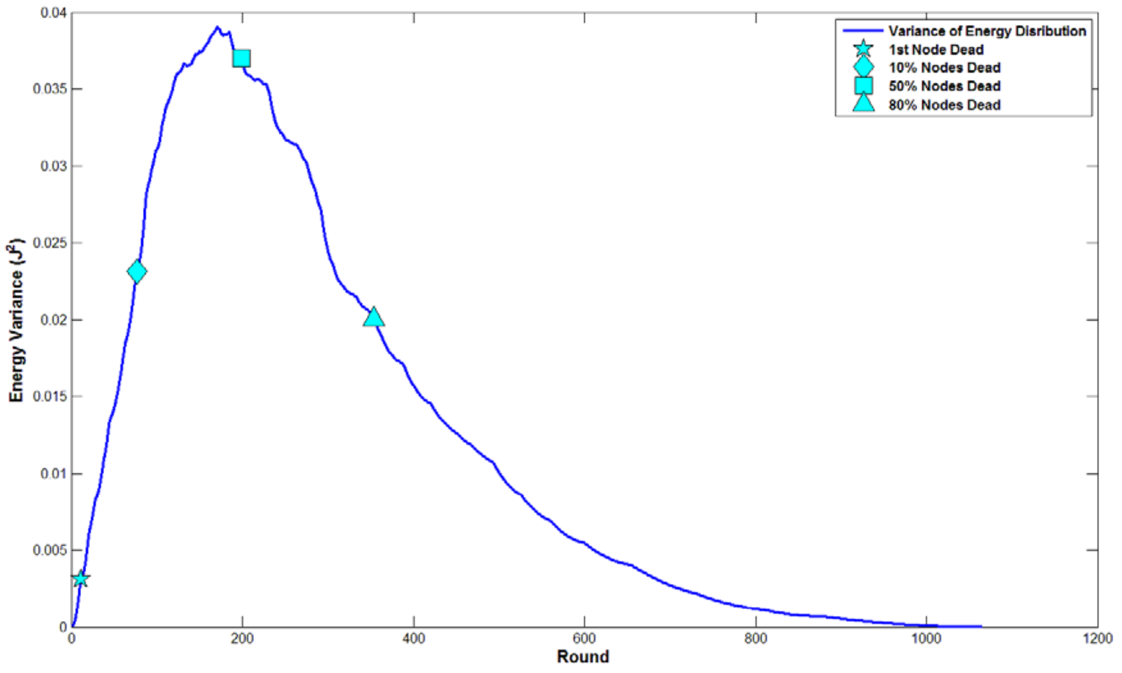


Figure 45. MTE routing in a single gateway WSN. The WSN energy variance versus transmission round is illustrated.

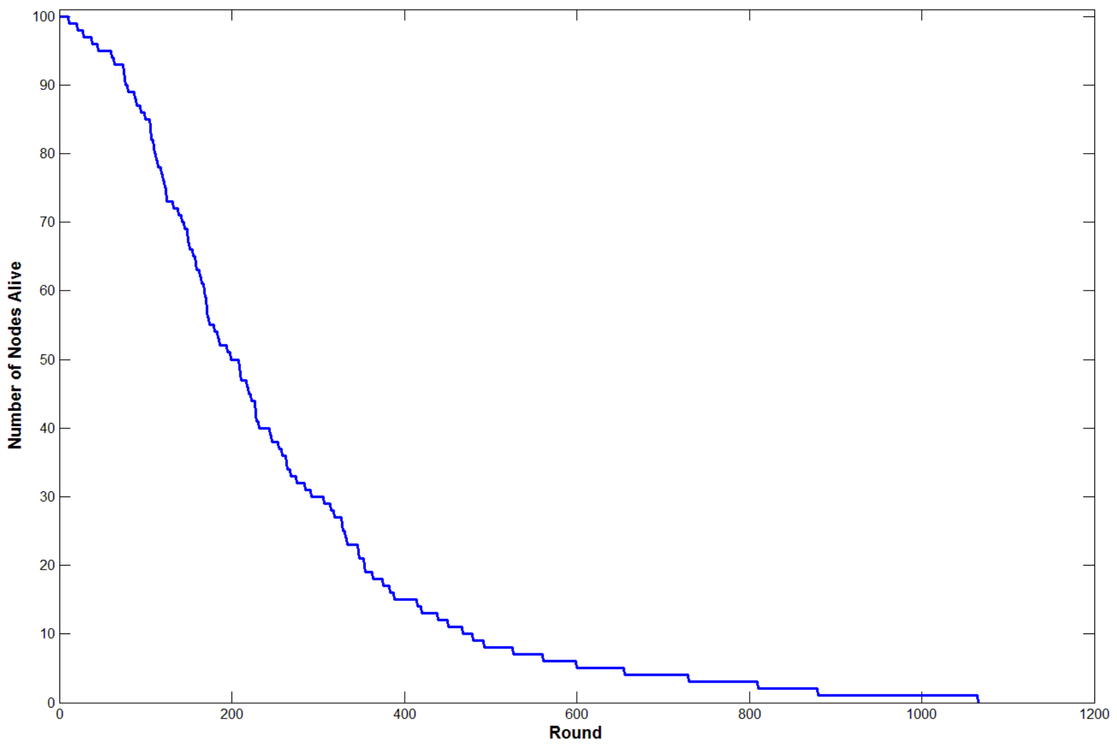


Figure 46. MTE routing in a single gateway WSN. The number of nodes alive versus transmission round is illustrated.

2. Multi-gateway

Minimum transmission energy with Dijkstra routing initiates each node to send an L=2000 bit packet to the gateway during each round utilizing a route through neighboring nodes that minimizes the d^2 propagation link cost to the closer gateway. The gateway with the lower total link cost route metric is used. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 47 through Figure 50, respectively, for the multi-gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 51 through Figure 53, respectively. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 17, 100, 293, and 453, respectively, corresponding to a percent increase of 55 percent, 30 percent, 47 percent, and 28 percent, respectively, when compared to single gateway die out statistics for the same algorithm. The energy stem plot and node distribution plots demonstrate nodes closest to each gateway die first quickly, eliminating service coverage in those areas. The energy depletion rate of the network during the linear region of Figure 51 is 0.1418 J/round, which corresponds to a 34 percent reduction in network energy depletion rate compared to the single gateway case.

The previous single gateway case depleted nodes farthest from the gateway slowly. In the multi-gateway scenario, nodes in the middle of the sensor field are preserved longest since they are not used for routing as much as hot-nodes that are closest to the gateway. There is an opposite node die out reaction when using Dijkstra's algorithm in MTE routing compared to the direct routing simulation. Specifically, the direct packet transmission algorithm depleted nodes farthest from the gateway, while MTE transmission depleted areas needed for packet routing first.

Our comments regarding energy variance are similar in the single gateway and multi-gateway MTE scenarios except the multi-gateway yields a smaller energy variance during each round. The addition of the second gateway lowers the total number of transmissions during each round, which is an improvement and, subsequently, lowers the energy variance as compared to the single gateway case. However, the lack of data aggregation and the fact that nodes use their neighbors excessively for routing causes a large energy variance and, therefore, nodes die out quickly.

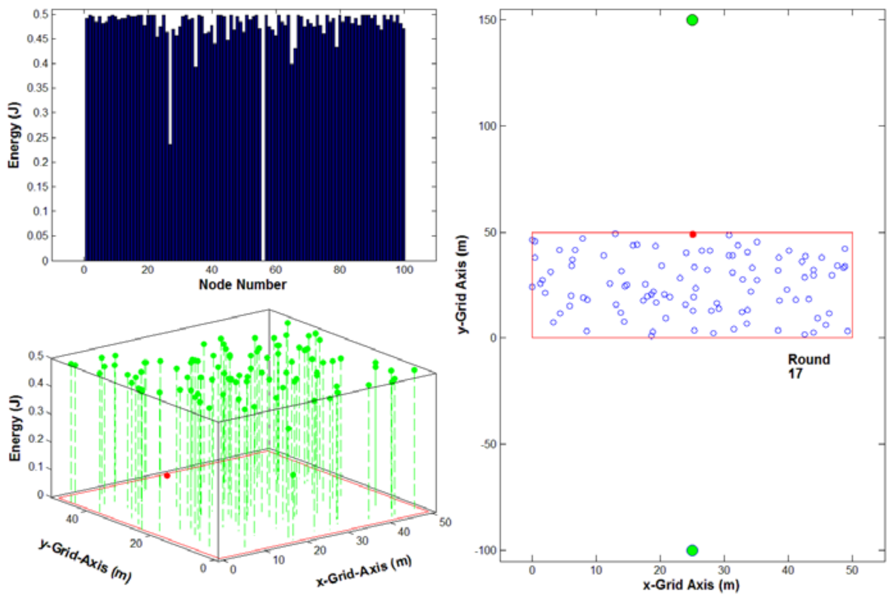


Figure 47. MTE routing in a multi-gateway WSN illustrating first node dead die out topology versus transmission round and energy distribution.

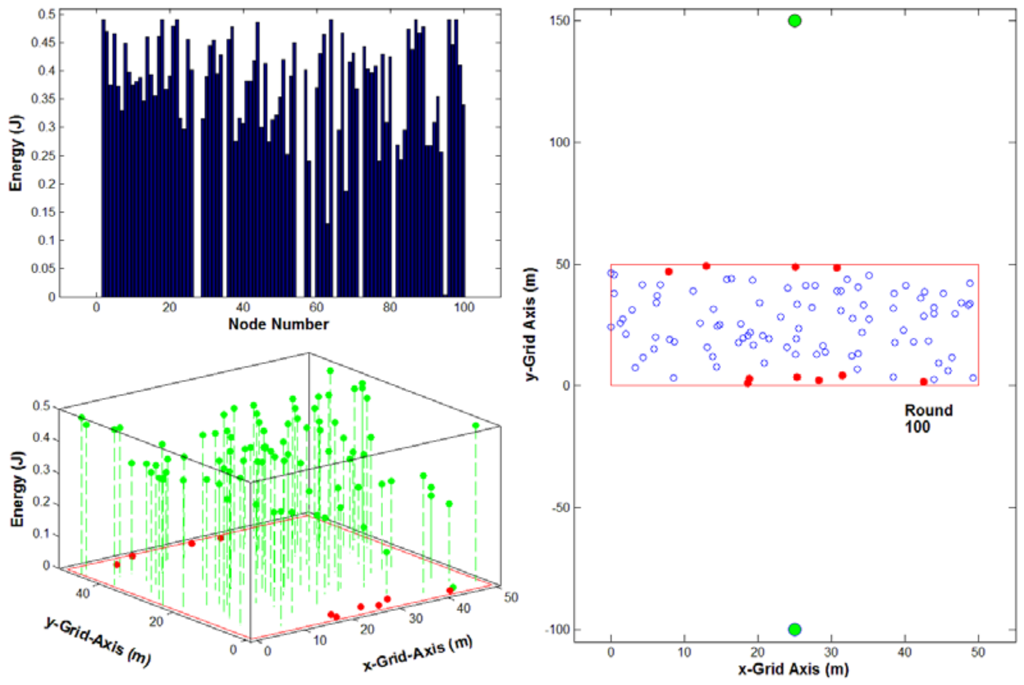


Figure 48. MTE routing in a multi-gateway WSN illustrating 10 percent nodes dead die out topology versus transmission round and energy distribution.

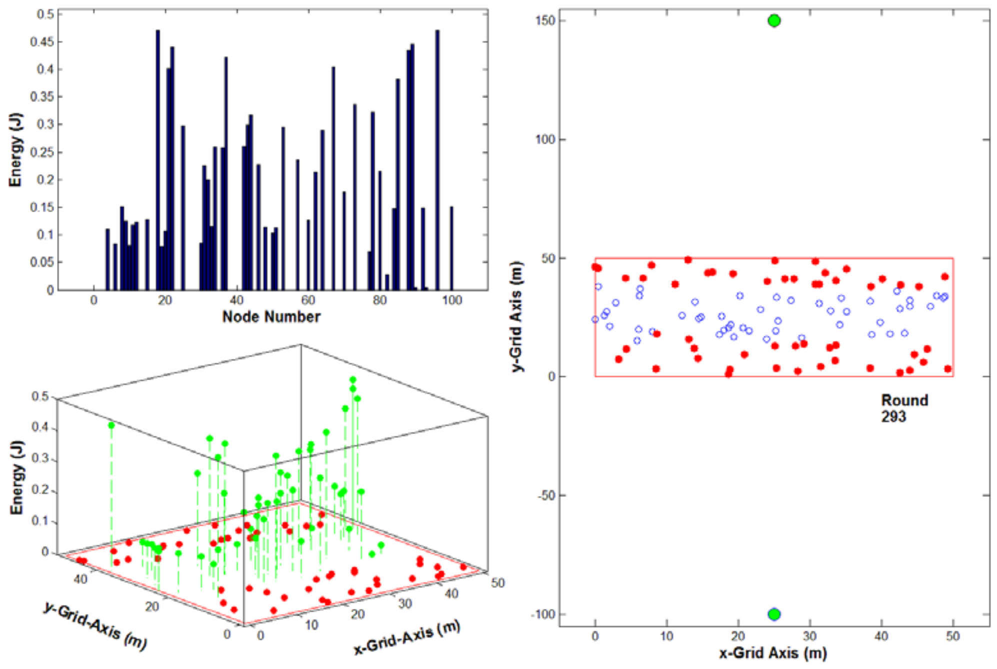


Figure 49. MTE routing in a multi-gateway WSN illustrating 50 percent nodes dead die out topology versus transmission round and energy distribution.

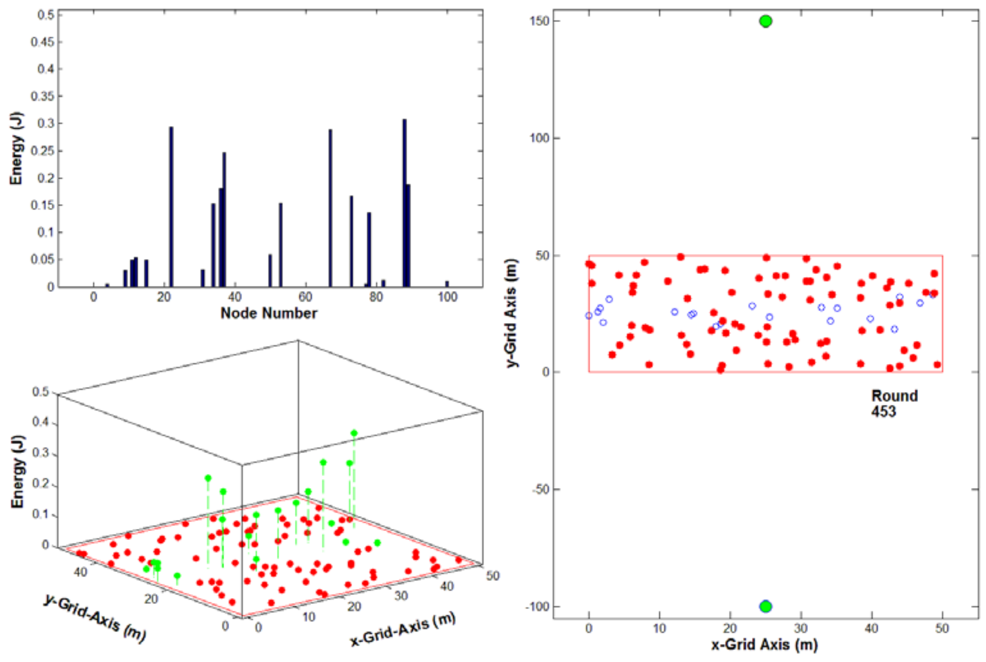


Figure 50. MTE routing in a multi-gateway WSN illustrating 80 percent nodes dead die out topology versus transmission round and energy distribution.

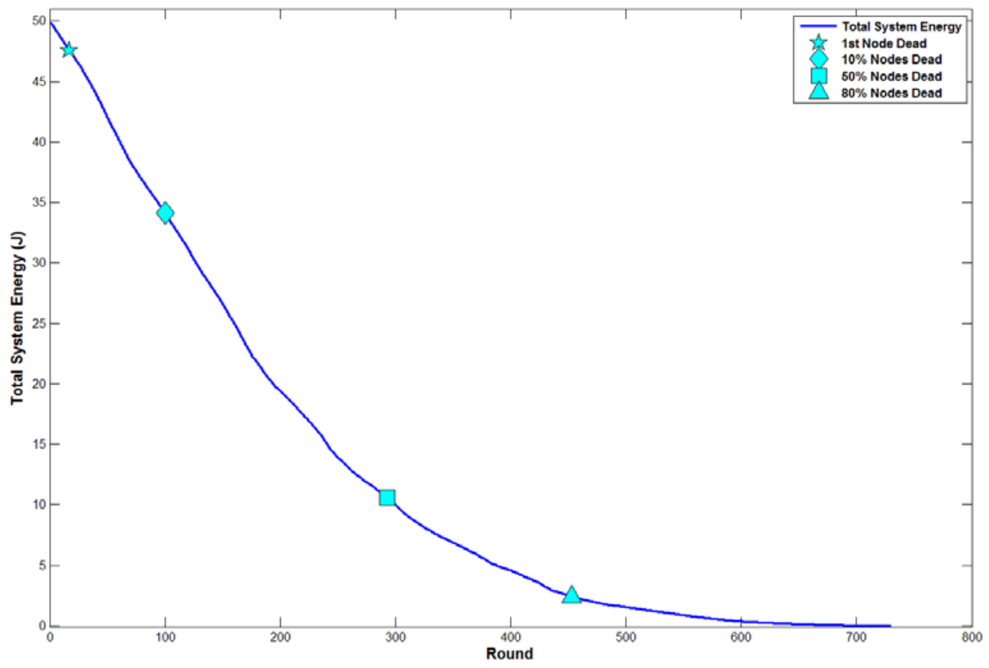


Figure 51. MTE routing in a multi-gateway WSN. The total WSN energy versus transmission round is illustrated.

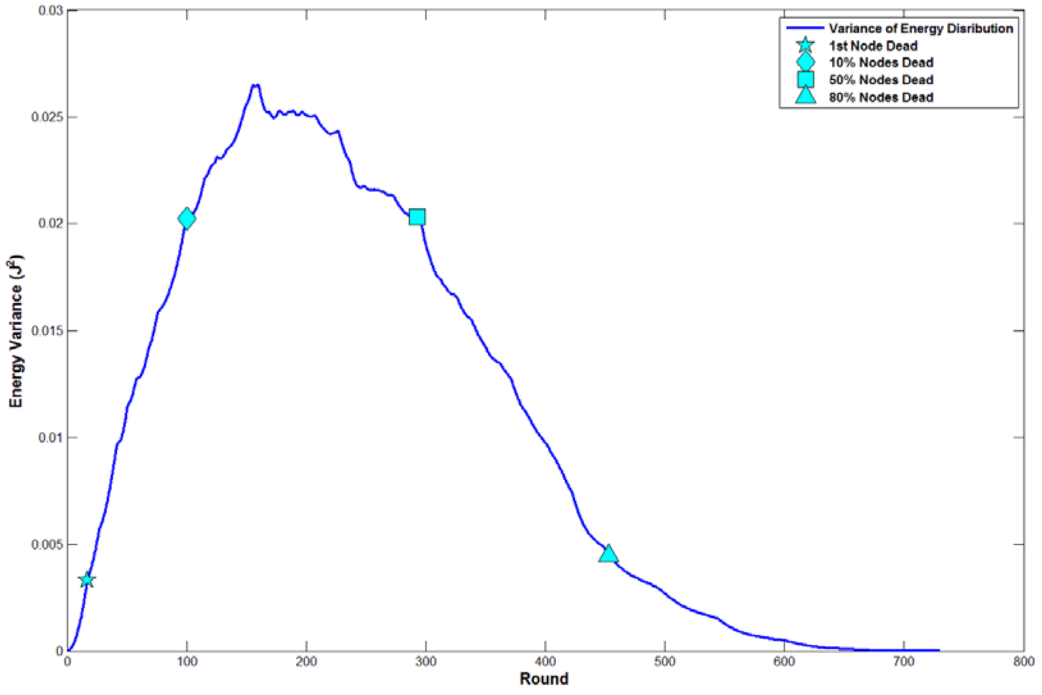


Figure 52. MTE routing in a multi-gateway WSN. The WSN energy variance versus transmission round is illustrated.

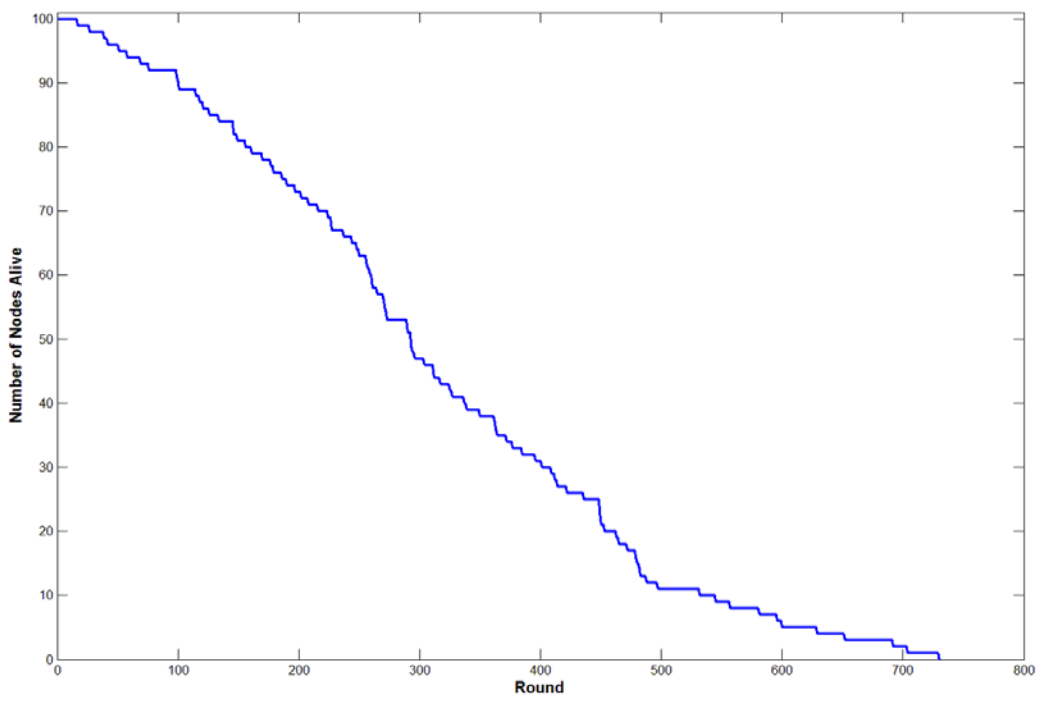


Figure 53. MTE routing in a multi-gateway WSN. The number of nodes alive versus transmission round is illustrated.

E. LOW ENERGY ADAPTIVE CLUSTERHEAD (LEACH) ROUTING

1. Single Gateway

LEACH routing initiates LEACH clustering, and each node sends an L=2000 bit packet to the gateway through the CH during each round. The CH aggregates all packets into a single 2000 bit packet for the round and performs the transmission to the gateway. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 54 through Figure 57, respectively, for the single gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 58 through Figure 60, respectively. First node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 1642, 1760, 1990, and 2182, respectively. The energy stem plot and node distribution plots demonstrate that nodes die out starting in the middle of the network and progress out. From this outward progression, nodes toward the top of the network die out more quickly than nodes at the bottom of the sensor field because nodes at the top of the gateway use more energy to transmit a cluster's payload to the gateway during the random times they are selected as the CH. Nodes at the center of the field start to die out first as a result of LEACH's mechanism for determining CHs and cluster assignments at each round. We examine this again later in the chapter when we compare the dynamics of our simulated clustering algorithms. The energy depletion rate of the network during the linear region of Figure 58 is 0.0245 *J*/round.

This simulation presents a first look at the performance advantage of using a clustering algorithm with data aggregation as we realize a significant increase in our service life die out parameters. As a result, our energy variance versus transmission round and energy depletion rate versus transmission round is significantly lower compared to the other network algorithm simulations presented (i.e., MTE and direct). Also, the range of all nodes live compared to the range of nodes dying out is much greater than the previous algorithms, indicating that all nodes are live in the network for a much longer period of time relative to the other algorithms. This indicates a longer relative time of complete service coverage.

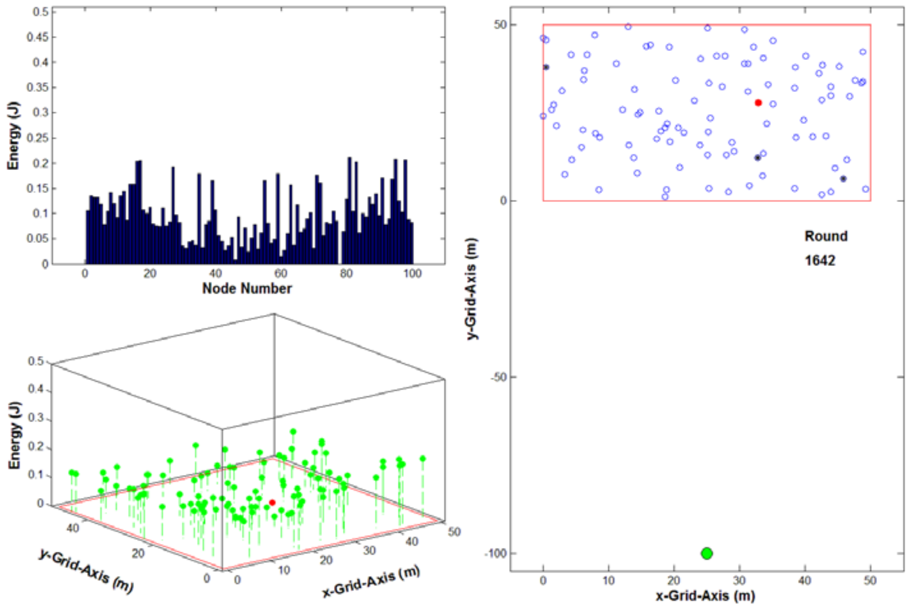


Figure 54. LEACH routing in a single gateway WSN with first node dead die out topology versus transmission round and energy distribution. The node distribution plot indicates three CHs chosen during round 1642.

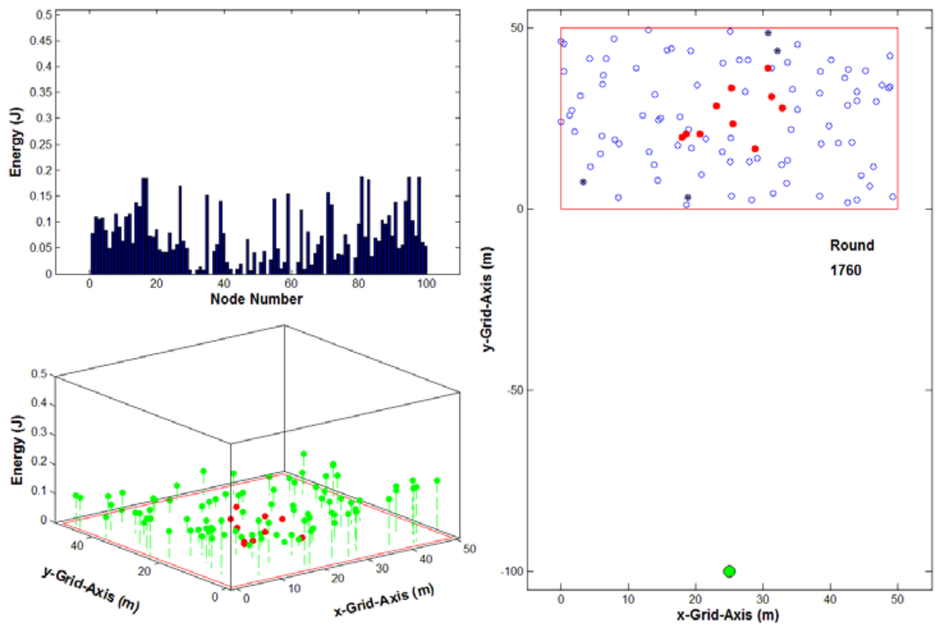


Figure 55. LEACH routing in a single gateway WSN with 10 percent nodes dead die out topology versus transmission round and energy distribution. The node distribution plot indicates at least four CHs however a dead node may have masked other CHs.

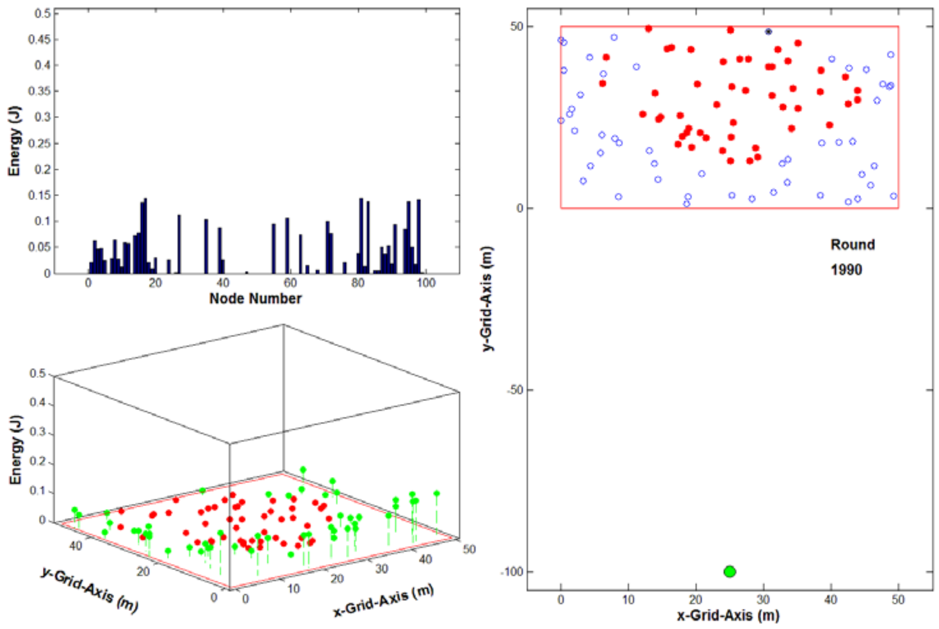


Figure 56. LEACH routing in a single gateway WSN with 50 percent node dead die out topology versus transmission round and energy distribution. The node distribution plot indicates at least one CH however a dead node may have masked other CHs.

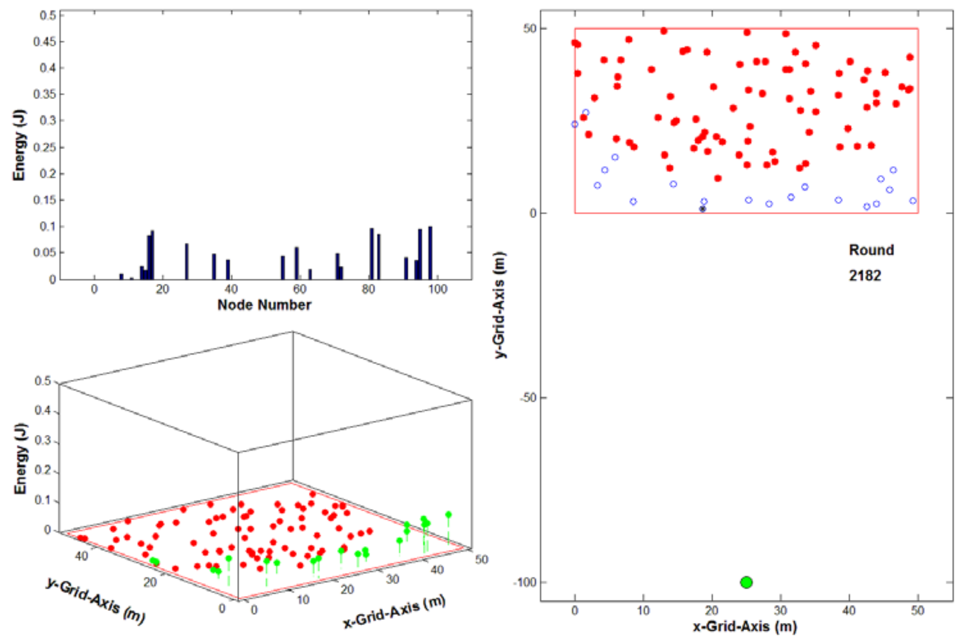


Figure 57. LEACH routing in a single gateway WSN with 80 percent node dead die out topology versus transmission round and energy distribution. The node distribution plot indicates at least one CH however a dead node may have masked other CHs.

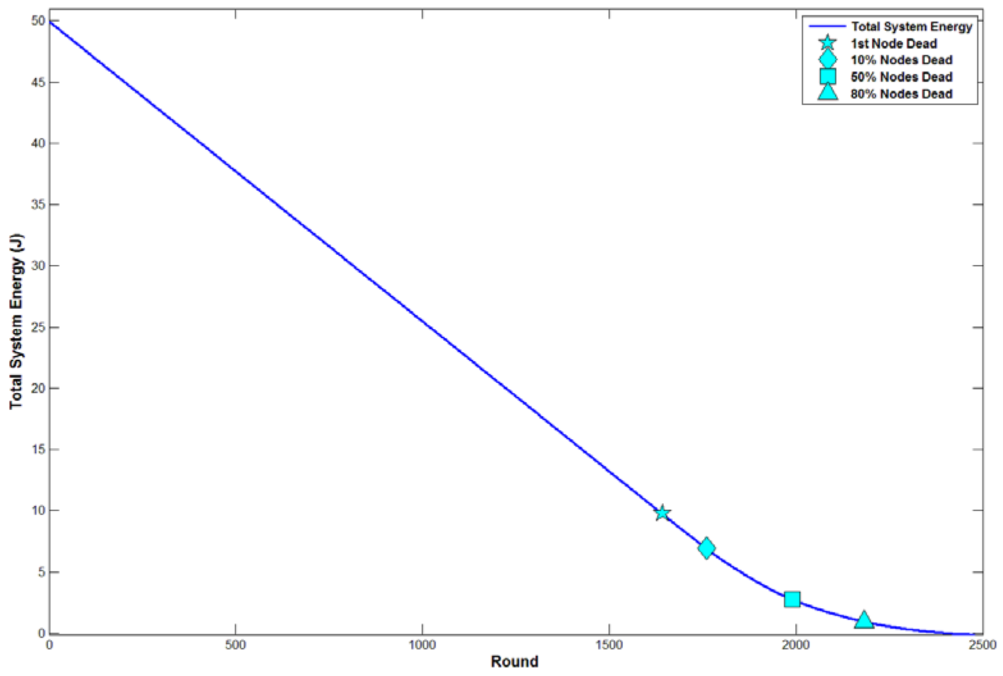


Figure 58. LEACH routing in a single gateway WSN. The total WSN energy versus transmission round is illustrated.

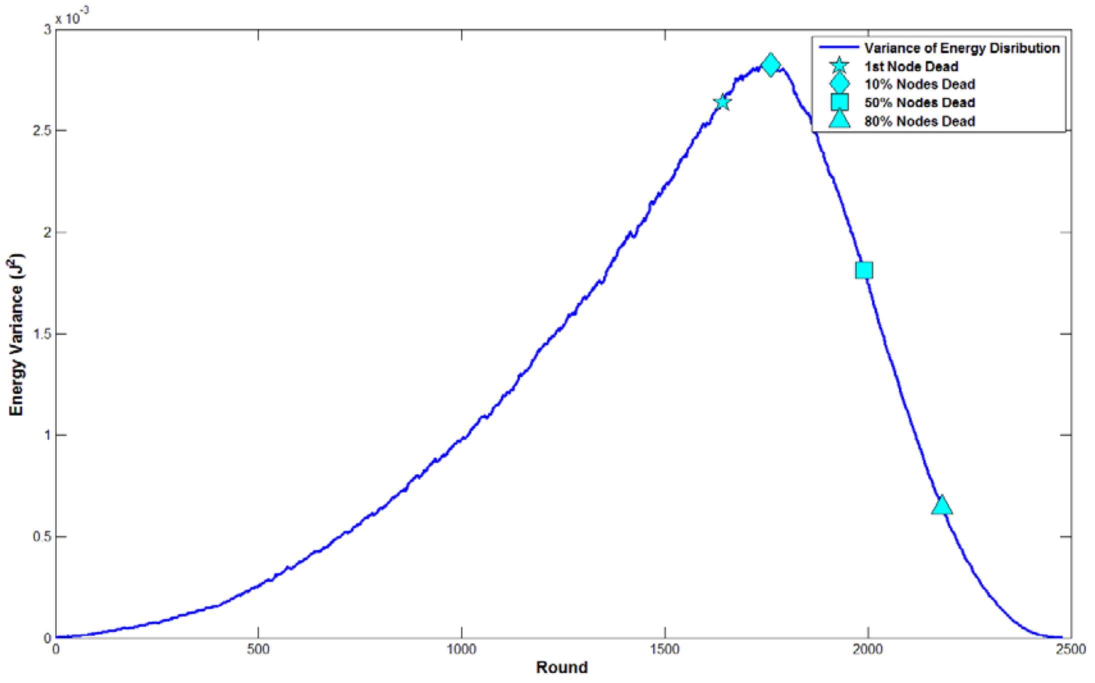


Figure 59. LEACH routing in a single gateway WSN. The WSN energy variance versus transmission round is illustrated.

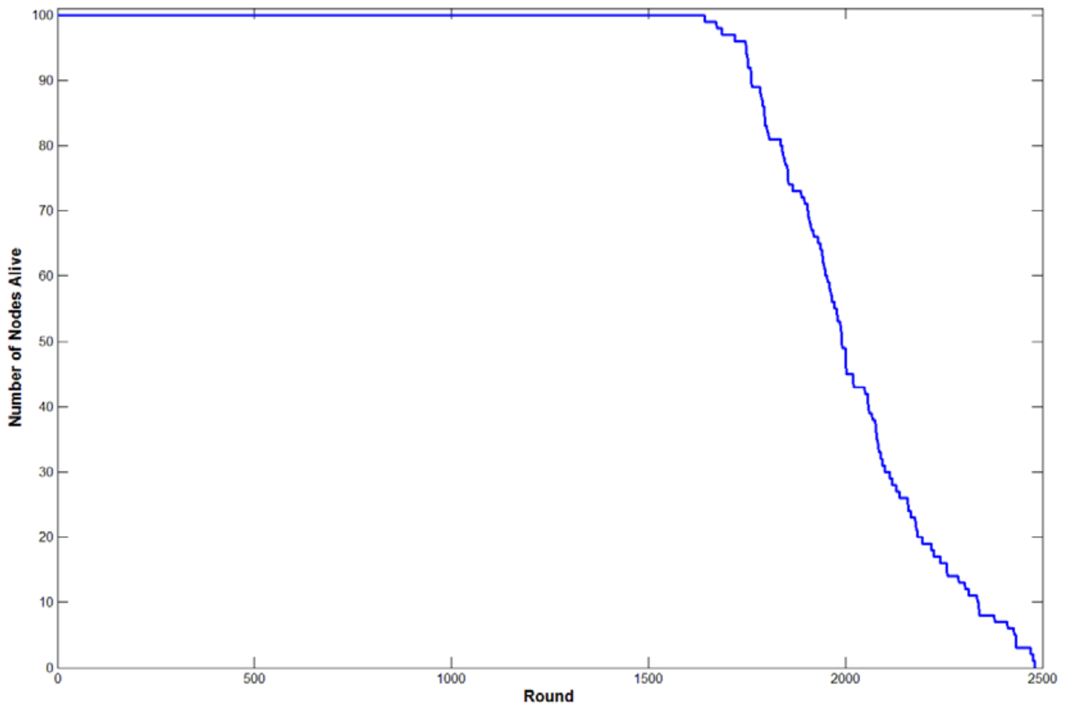


Figure 60. LEACH routing in a single gateway WSN. The number of nodes alive versus transmission round is illustrated.

2. Multi-gateway

Like the single gateway case, LEACH routing initiates LEACH clustering, and each node sends an L=2000 bit packet to the gateway through the CH during each round. The CH aggregates all packets into a single 2000 bit packet for the round and performs the transmission to the closer gateway. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 61 through Figure 64, respectively, for the multi-gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 65 through Figure 67, respectively. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 1633, 1805, 2112, and 2327, respectfully, corresponding to a percent increase of −1 percent, three percent, six percent, and seven percent, respectively, when compared to the single gateway die out statistics for the same algorithm. The energy stem plot and node distribution plots demonstrate that nodes die out starting in the middle of the network and progressing outwards. In comparison to the single gateway LEACH case, the presence of an additional gateway causes nodes to die out in a consistently radial fashion from the center of the field. The energy depletion rate of the network during the linear region of Figure 65 is 0.0232 J/round, which corresponds to a five percent reduction in network energy depletion rate compared to the single gateway case.

The addition of another gateway does not significantly extend the life of the WSN for the LEACH algorithm due to the network topology during die out. During die out for both the single and multi-gateway simulations, nodes die out radially from the center of the sensor field. The additional gateway causes the algorithm to extend to a few extra rounds during the die out because nodes are not predominantly dying out toward the top of the network as occurred during the LEACH single gateway scenario. We note that the round the first node dies for the multi-gateway case is nine rounds earlier than the single gateway case. We would expect the die out occur during a similar round because LEACH die out starts in the center of the field for both cases. However, because LEACH CHs are chosen through a random process, running the same algorithm on the same sensor field arrangement causes slightly different results. This is because CHs are chosen dynamically, and the time and place of these CHs results in a slightly different value for the round when the first node dies.

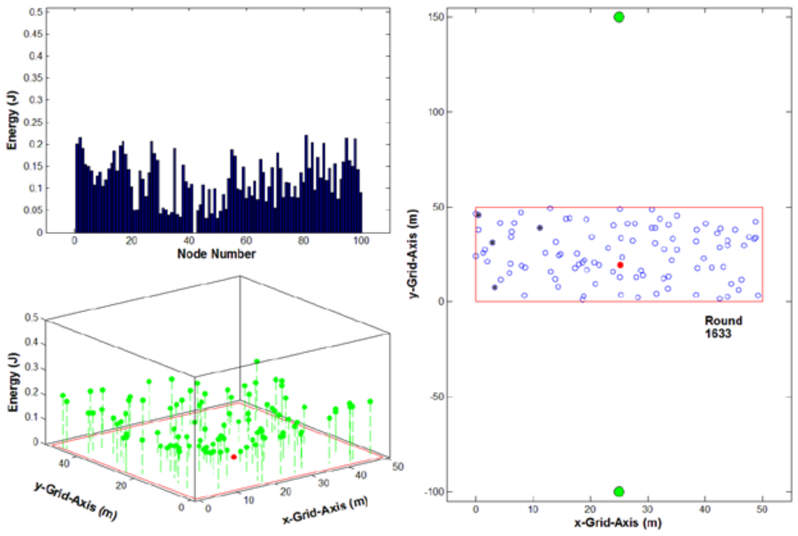


Figure 61. LEACH routing in a multi-gateway WSN. The first node dead die out topology versus transmission round and energy distribution is illustrated. Four CHs are inefficiently and tightly grouped together are shown on the node distribution plot.

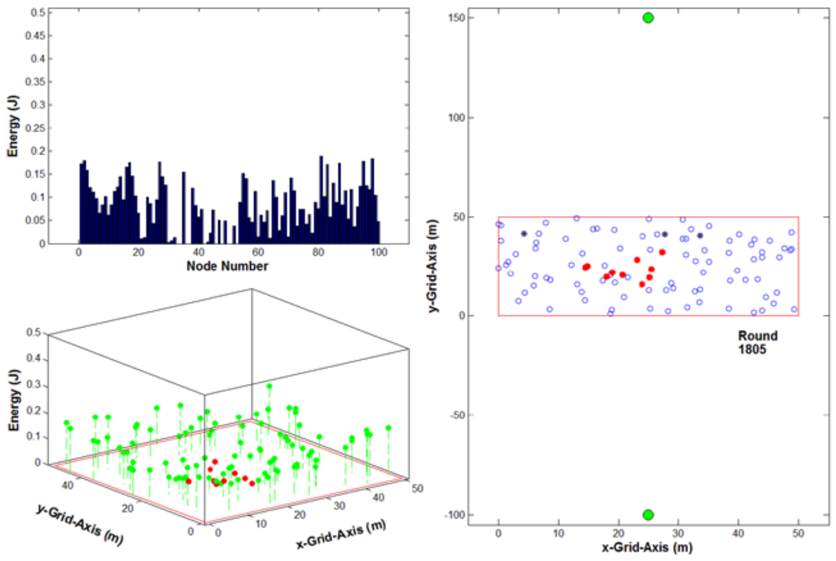


Figure 62. LEACH routing in a multi-gateway WSN. The 10 percent nodes dead die out topology versus transmission round and energy distribution is illustrated. At least three CHs are plotted on the node distribution plot however a dead node may have masked other CHs.

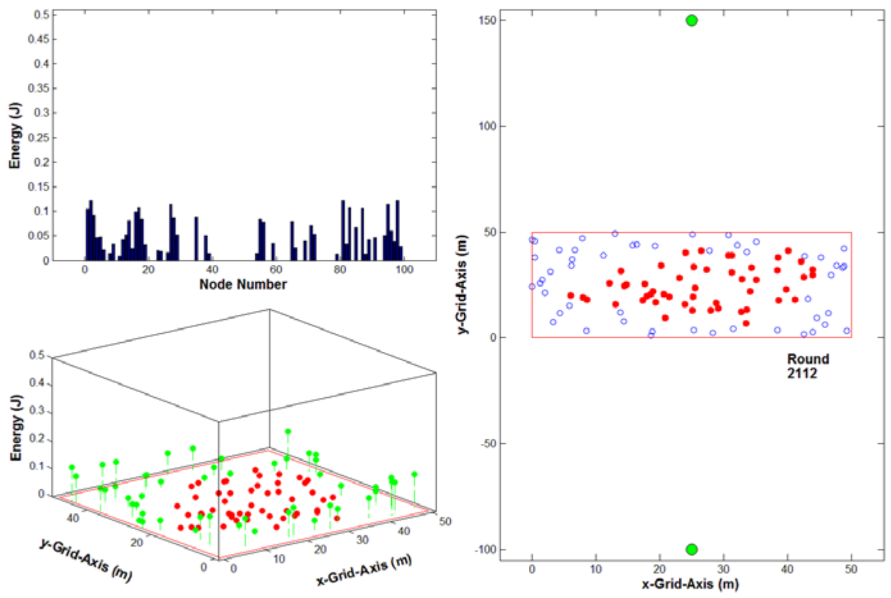


Figure 63. LEACH routing in a multi-gateway WSN. The 50 percent nodes dead die out topology versus transmission round and energy distribution is illustrated. No CHs are shown on the node distribution plot however a dead node may have masked other CHs.

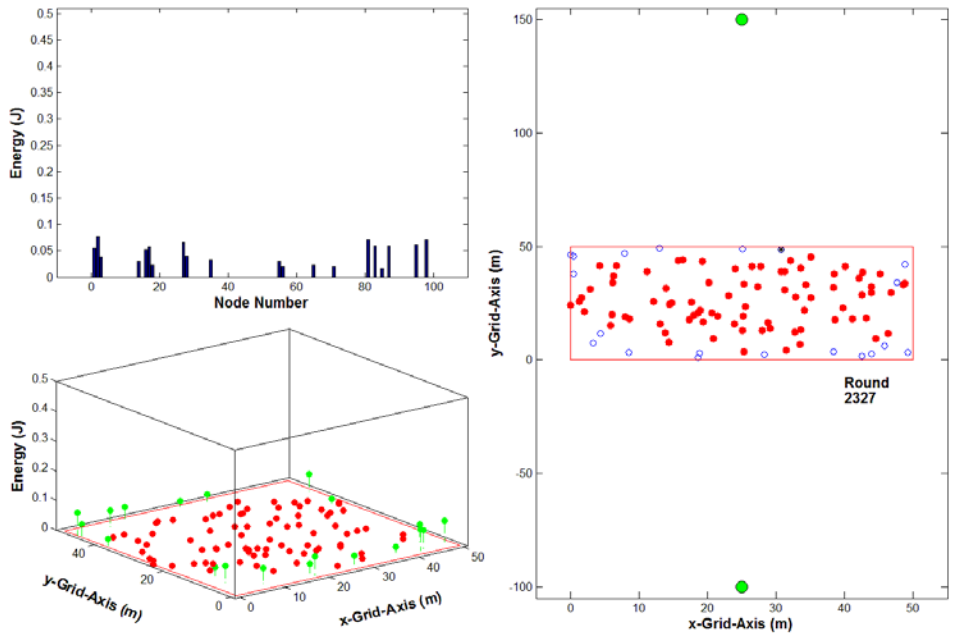


Figure 64. LEACH routing in a multi-gateway WSN. The 80 percent nodes dead die out topology versus transmission round and energy distribution is illustrated.

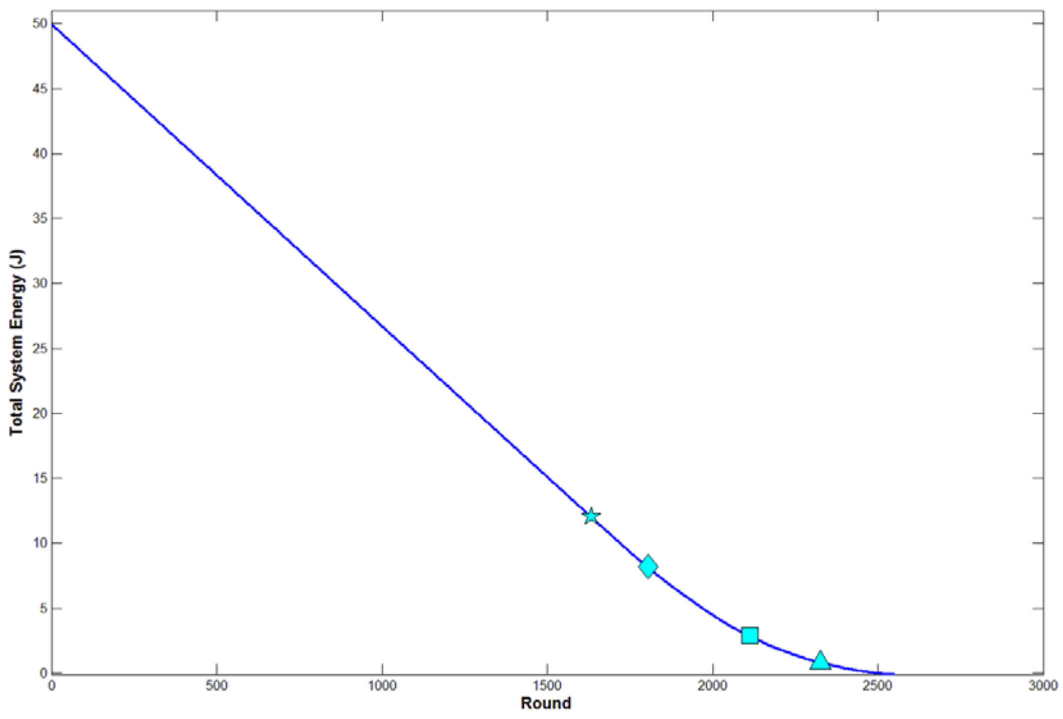


Figure 65. LEACH routing in a multi-gateway WSN. The total WSN energy versus transmission round is illustrated.

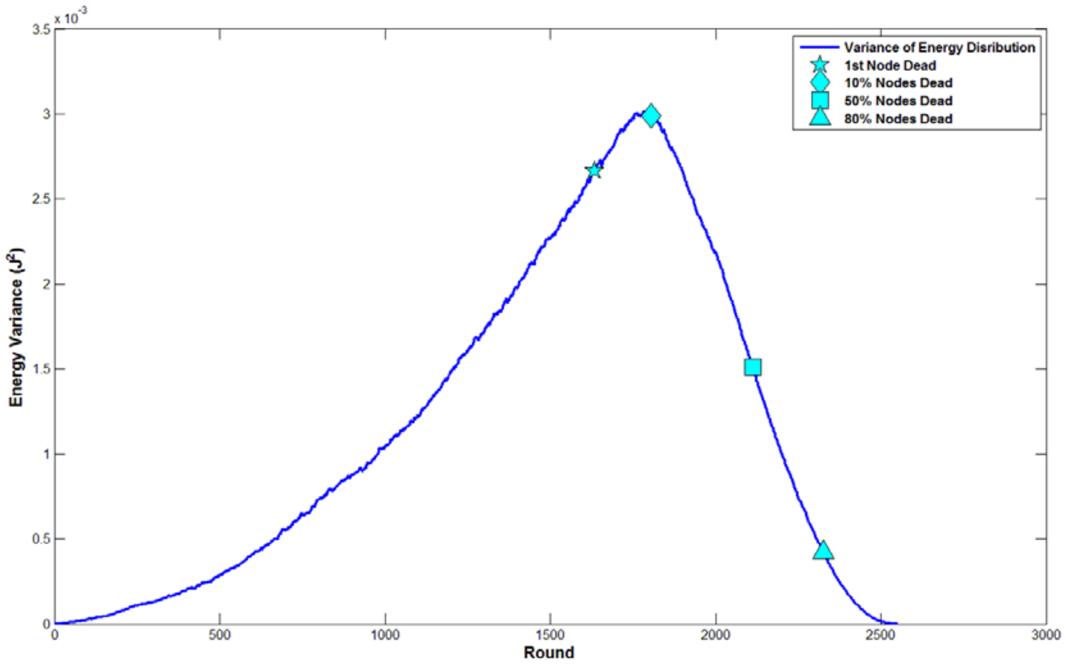


Figure 66. LEACH routing in a multi-gateway WSN. The WSN energy variance versus transmission round is illustrated.

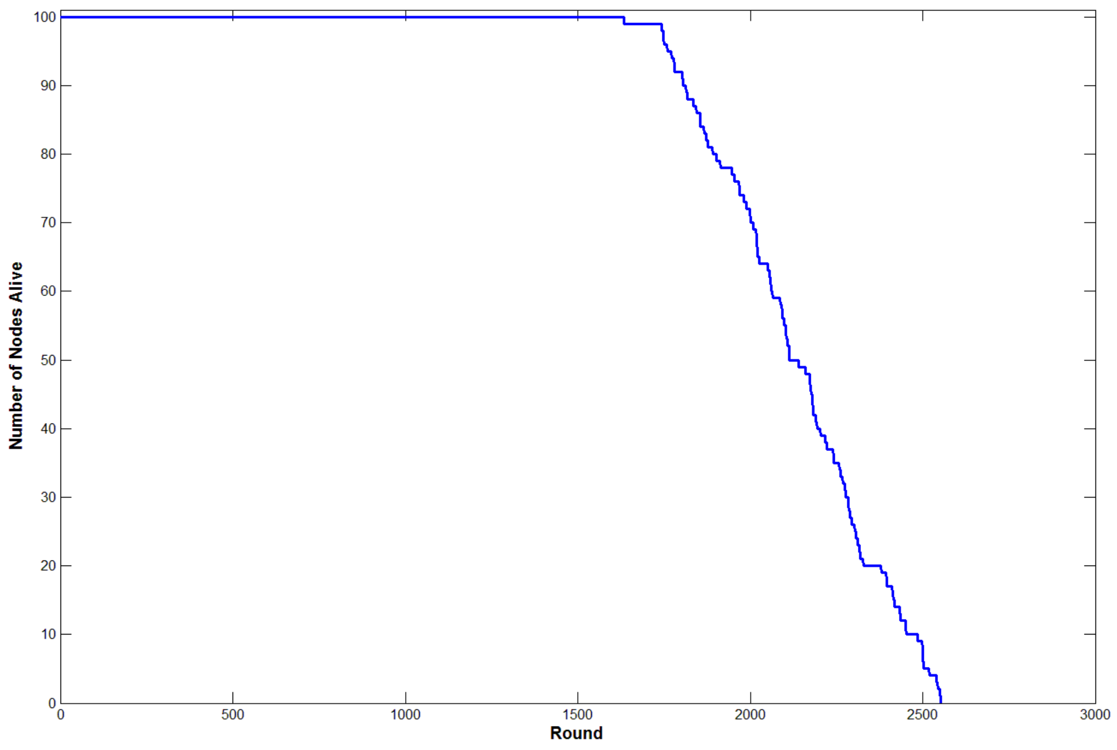


Figure 67. LEACH routing in a multi-gateway WSN. The number of nodes alive versus transmission round is illustrated.

F. ZONE CLUSTERING WITH RANDOM CLUSTER HEAD ELECTION

1. Single Gateway

Zone clustering with random CH election splits the field into five zones, randomly elects a CH in each zone to receive all L = 2000 bit packets from supported nodes and then aggregates the packet into a 2000 bit packet for transmission to the gateway. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 68 through Figure 71, respectively, for the single gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 72 through Figure 74, respectively. First node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 1649, 1821, 2022, and 2140, respectively.

The energy stem plot and node distribution plots demonstrate much more uniform energy depletion as compared to all the other algorithms tested thus far. The location of the first node dead appears to be random but favors the upper region of the plot since nodes in that region require the most energy for transmission to the gateway when they are chosen as the CH. The location of the first node that dies is a random location because it is a function of how many times it had previously been selected to be a CH combined with how far away it resides from the gateway. Since any node can be randomly selected to be a CH more than any other node in the network, this creates a random mode for nodes to die out. Observing the node distribution plots in Figure 68 and Figure 69, as nodes begin to die out, many of their peers remain alive, preserving sensor coverage in those regions. There is a slightly higher concentration of dead nodes at the top of the zones since those nodes require more energy to transmit over a larger distance. However, since CH election is random, we do see some nodes have died at the bottom of the zones in Figure 70 and Figure 71 as they were randomly chosen to be a CH for the zone more frequently. Zones in the node distribution plots die out consistently with no one zone dying out earlier than another zone. This algorithm with the first node dead at round 1649 provides the longest service life of 100 percent of nodes alive of all algorithms tested thus far. The energy depletion rate of the network during the linear region of Figure 72 is 0.0248 J/round.

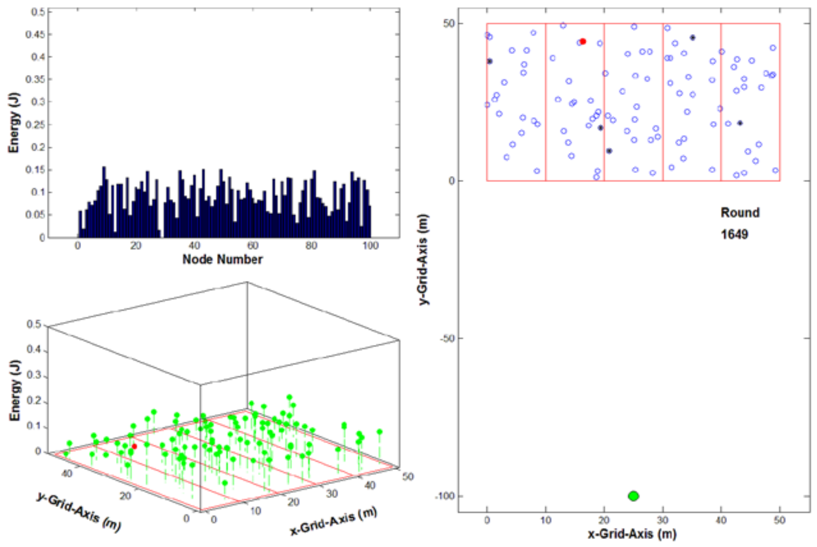


Figure 68. Zone clustering algorithm with random CH election routing. The first node dead die out topology versus transmission round and energy distributions is illustrated.

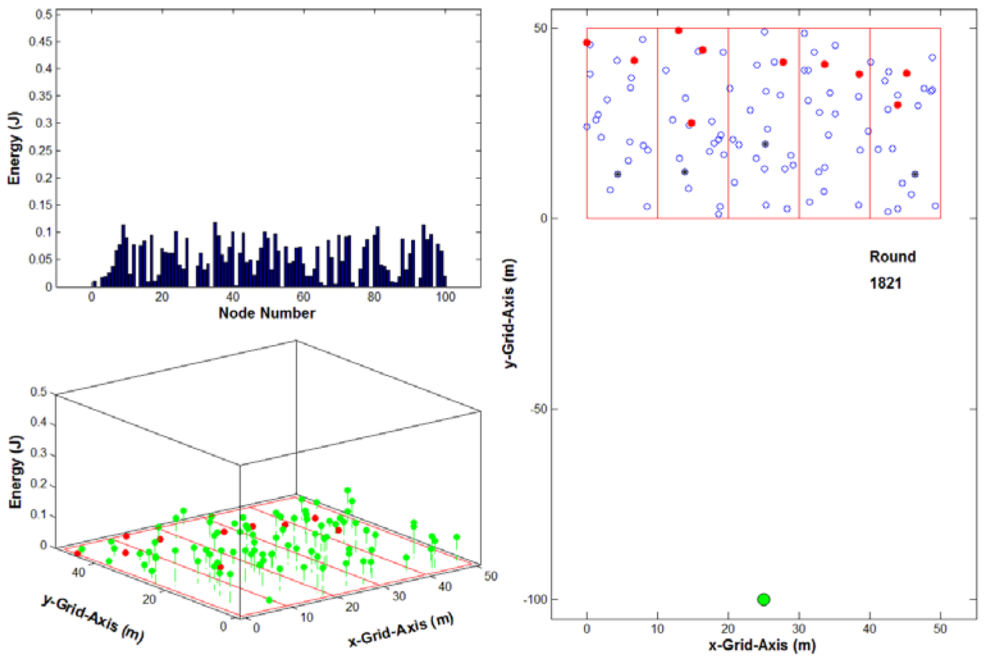


Figure 69. Zone clustering algorithm with random CH election routing. The first node dead die out topology versus transmission round and energy distributions is illustrated.

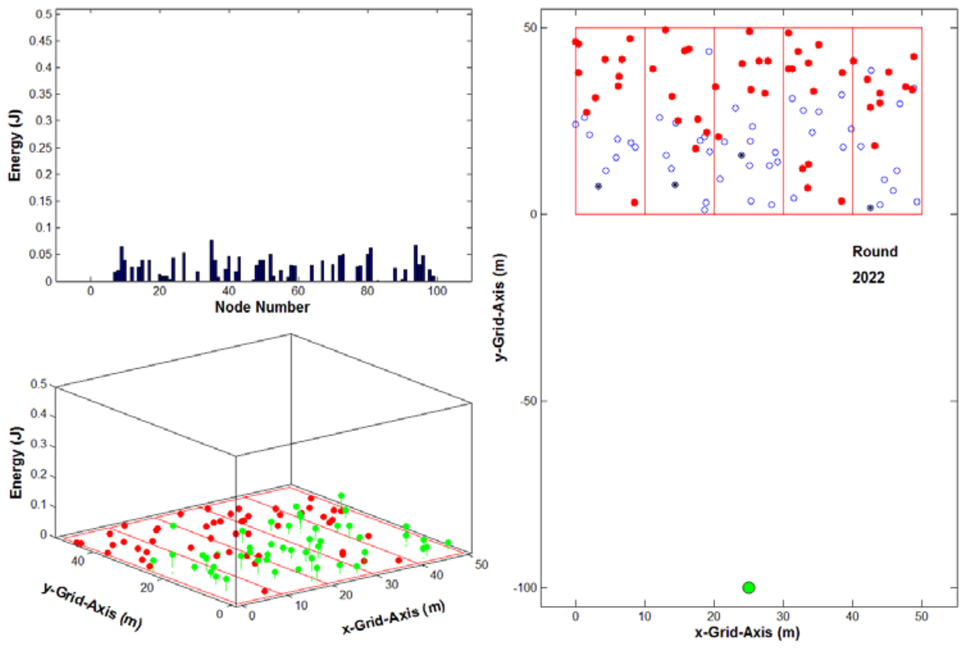


Figure 70. Zone clustering algorithm with random CH election routing. The 50 percent nodes dead die out topology versus transmission round and energy distributions is illustrated.

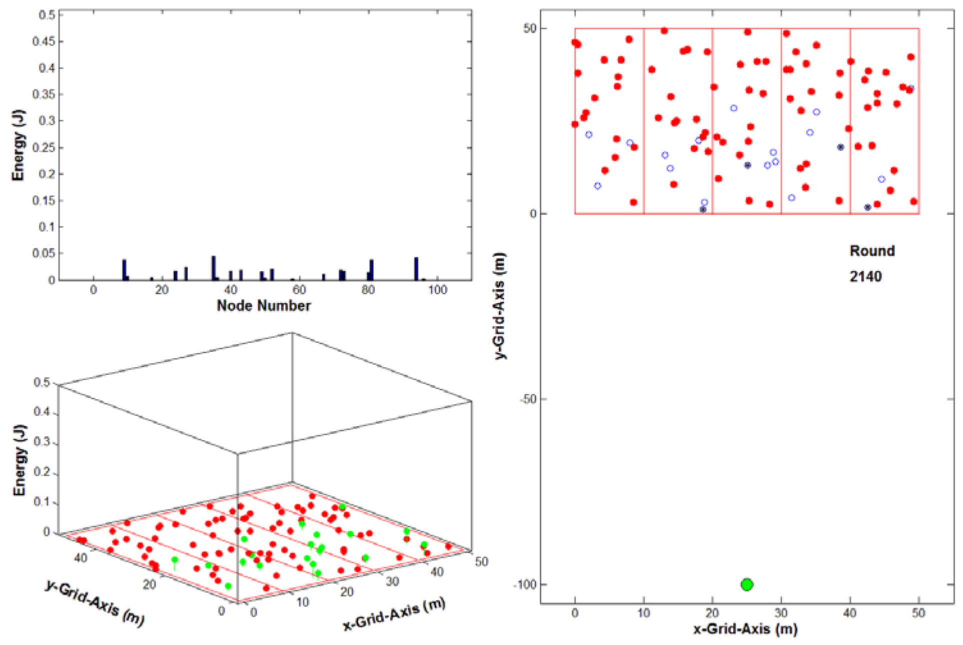


Figure 71. Zone clustering algorithm with random CH election routing. The 80 percent nodes dead die out topology versus transmission round and energy distributions is illustrated.

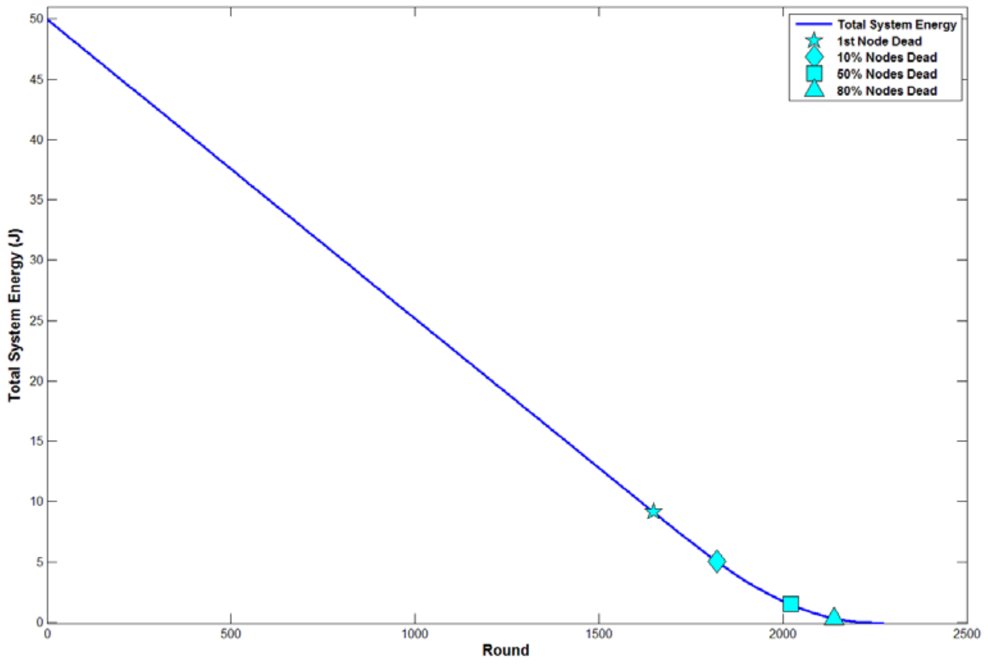


Figure 72. Zone clustering algorithm with random CH election routing in a single gateway WSN. The total WSN energy versus transmission round is illustrated.

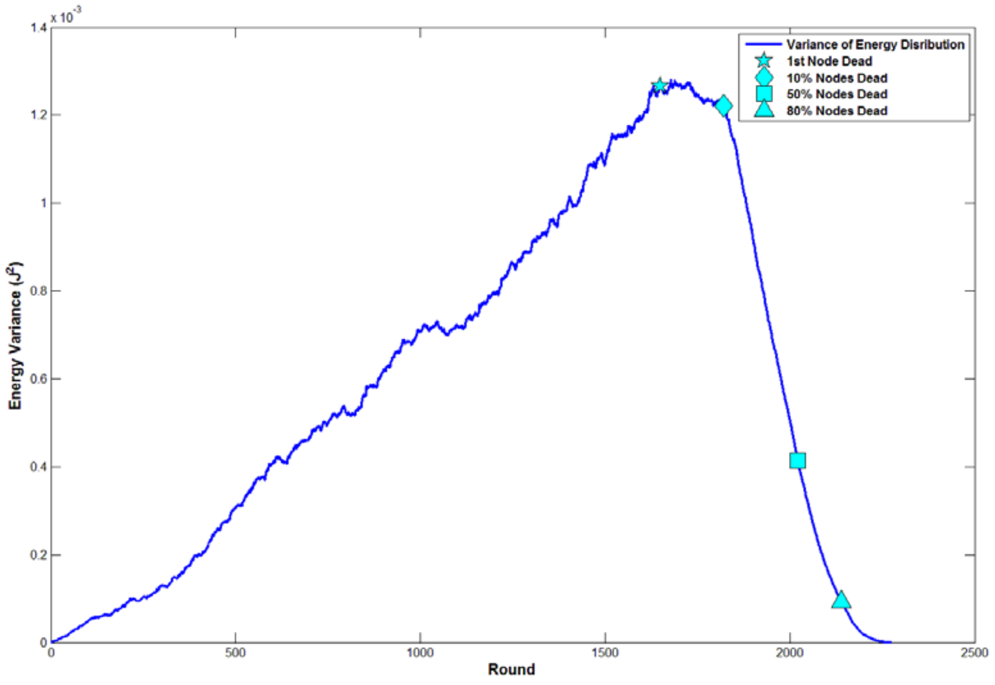


Figure 73. Zone clustering algorithm with random CH election in a single gateway WSN. The WSN energy variance versus transmission round is illustrated.

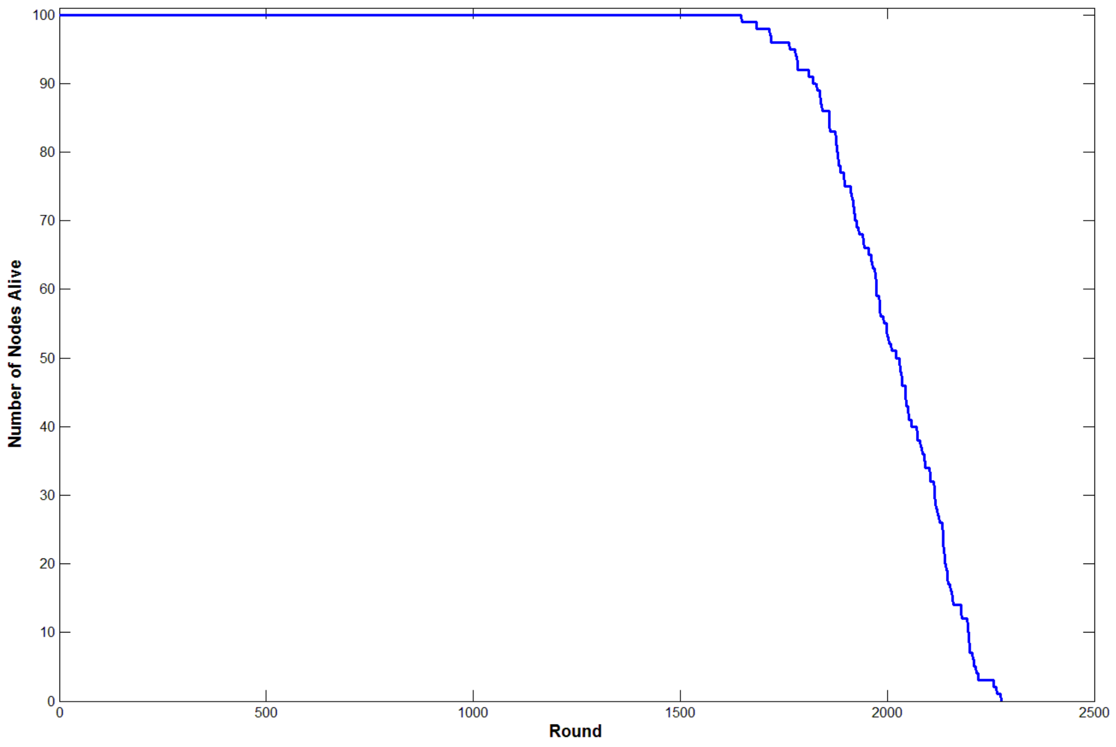


Figure 74. Zone clustering algorithm with random CH election routing in a single gateway WSN. The nodes alive versus transmission round is illustrated.

2. Multi-gateway

Like Zone clustering for the single gateway case, Zone clustering with random CH election for the multi-gateway scenario splits the field into five zones, randomly elects a CH in each zone to receive all L = 2000 bit packets from supported nodes and then aggregates the packet into a 2000 bit packet for transmission to the gateway. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 75 through Figure 78, respectively, for the multi-gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 79 through Figure 81, respectively. First node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 1862, 1964, 2117, and 2215, respectively. This corresponds to a percent increase of 13 percent, eight percent, five percent, and four percent, respectively, when compared to the single gateway die out statistics for the same algorithm. The energy depletion rate of the network during the linear region of Figure 79 is 0.0235 J/round, which corresponds to a five percent reduction in network energy depletion rate compared to the single gateway case.

The addition of another gateway in a zone routing algorithm with random CH election extends the time when 100 percent of nodes are alive and induces better uniformity of individual node energy depletion. This can be observed in the energy plots and the energy variance versus transmission round plot shown in Figure 80. We note that the maximum y-axis of Figure 80 is 8×10^{-4} J^2, which is the smallest (maximum) energy variance of any algorithm tested thus far. The number of rounds between the first node dead and 80 percent of nodes dead is 353. This results in the die out time versus 100 percent nodes alive time being 353/1862 = 0.19. This is the smallest fraction for the die out period presented thus far.

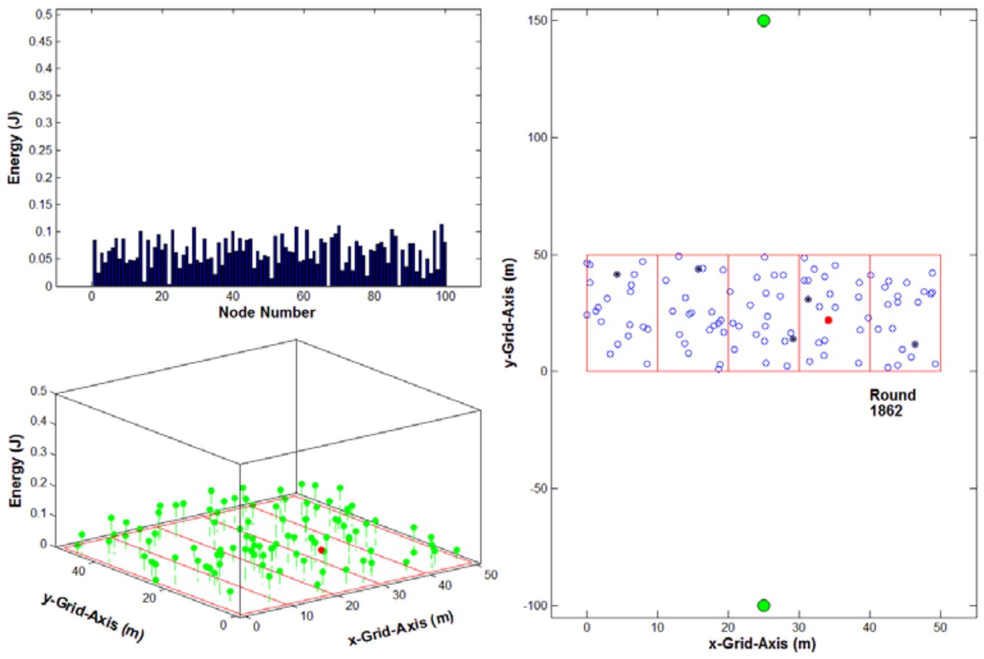


Figure 75. Zone clustering algorithm with random CH election routing in a multi-gateway WSN. The first node dead die out topology versus transmission round and energy distribution is illustrated.

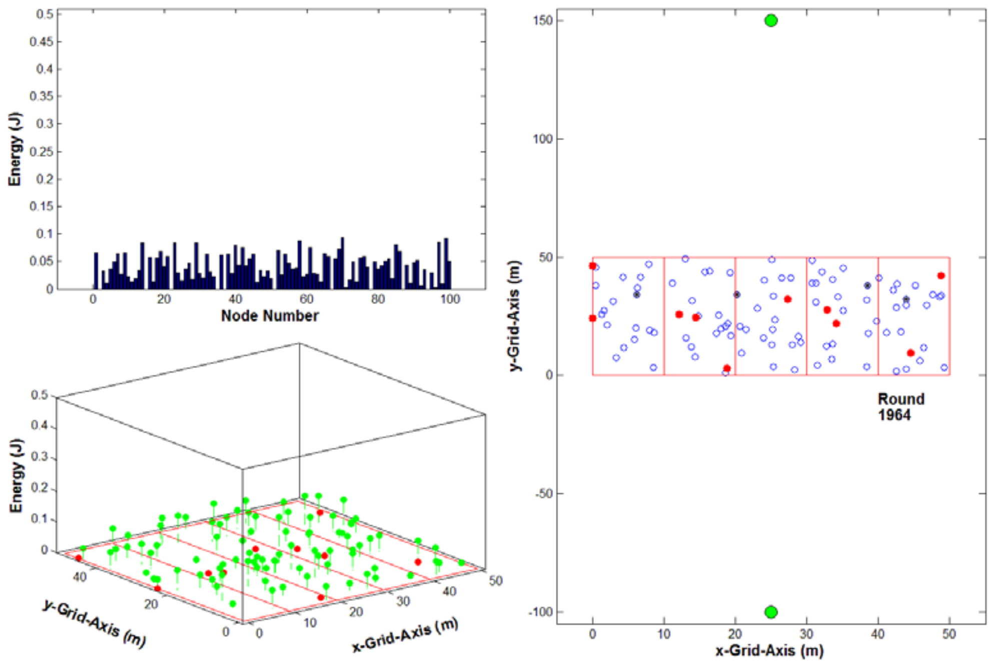


Figure 76. Zone clustering algorithm with random CH election routing in a multi-gateway WSN. The percent nodes dead die out topology versus transmission round and energy distribution is illustrated.

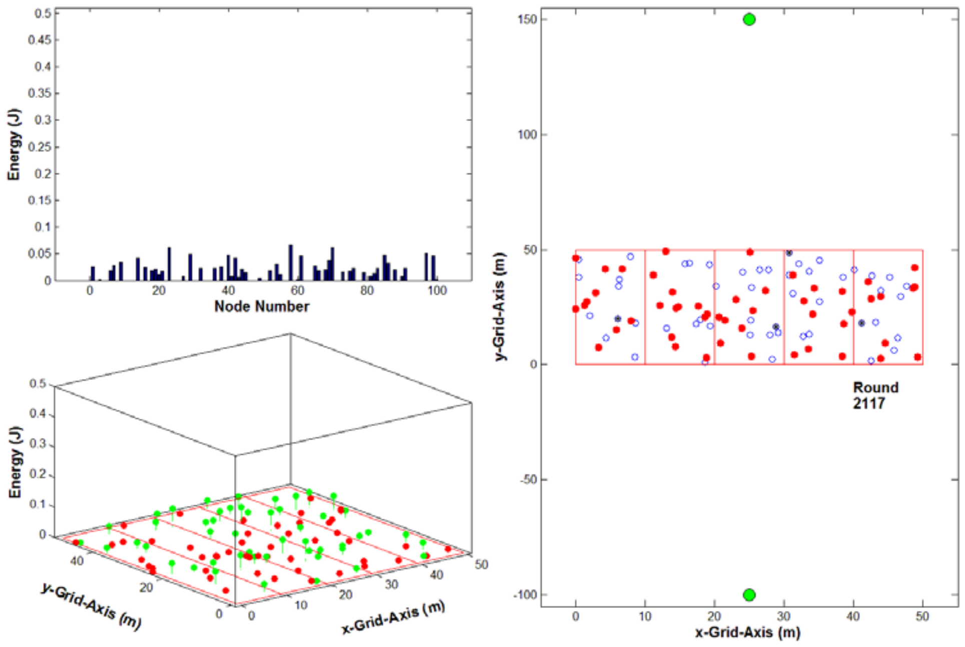


Figure 77. Zone clustering algorithm with random CH election routing in a multi-gateway WSN. The 50 percent nodes dead die out topology versus transmission round and energy distribution is illustrated.

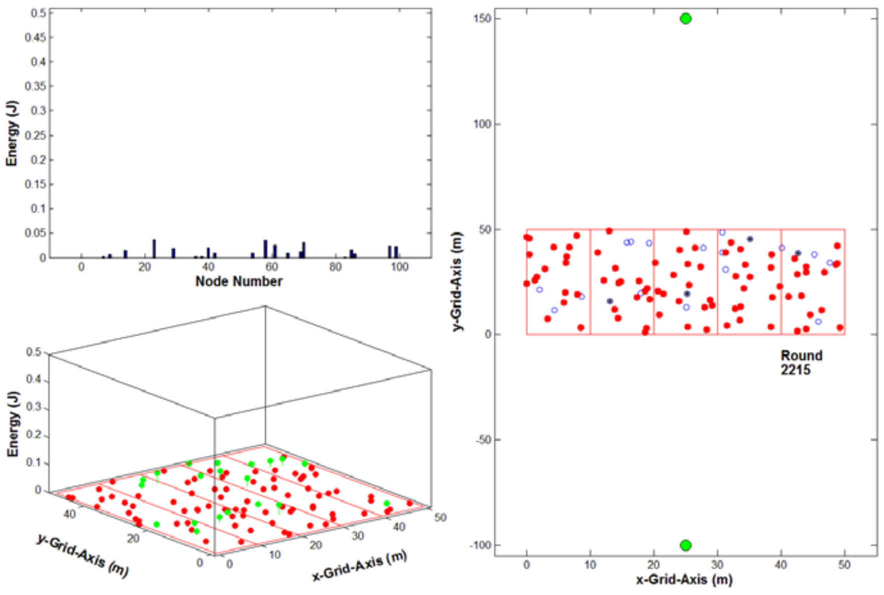


Figure 78. Zone clustering algorithm with random CH election routing in a multi-gateway WSN. The 80 percent nodes dead die out topology versus transmission round and energy distribution is illustrated.

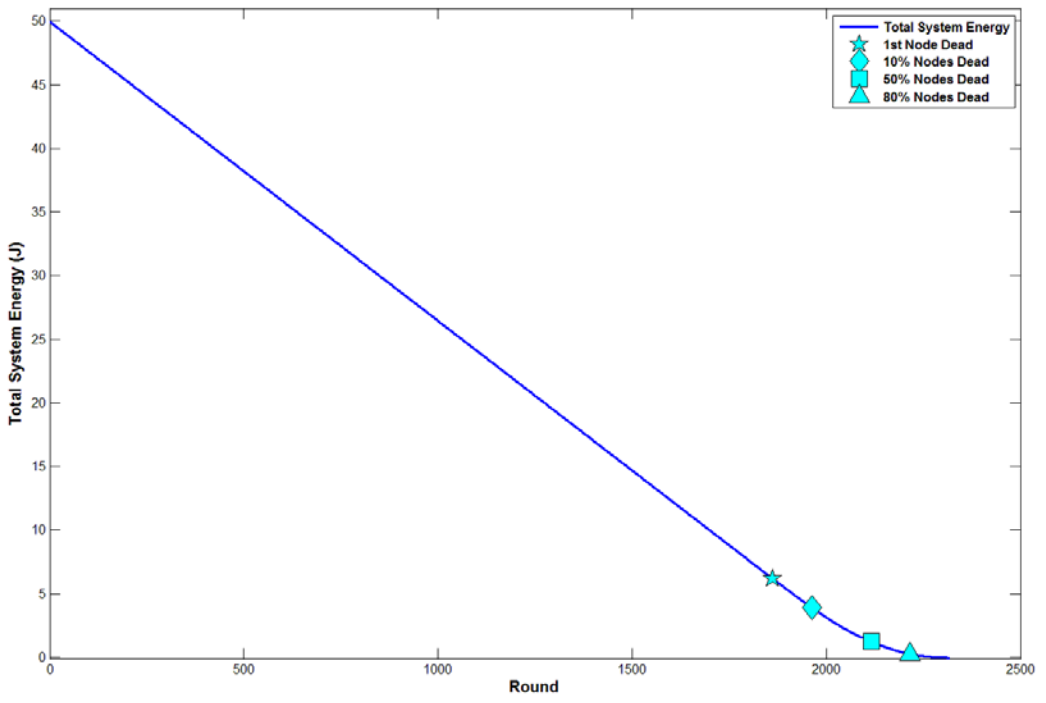


Figure 79. Zone clustering algorithm with random CH election routing in a multi-gateway WSN. The total WSN energy versus transmission round is illustrated.

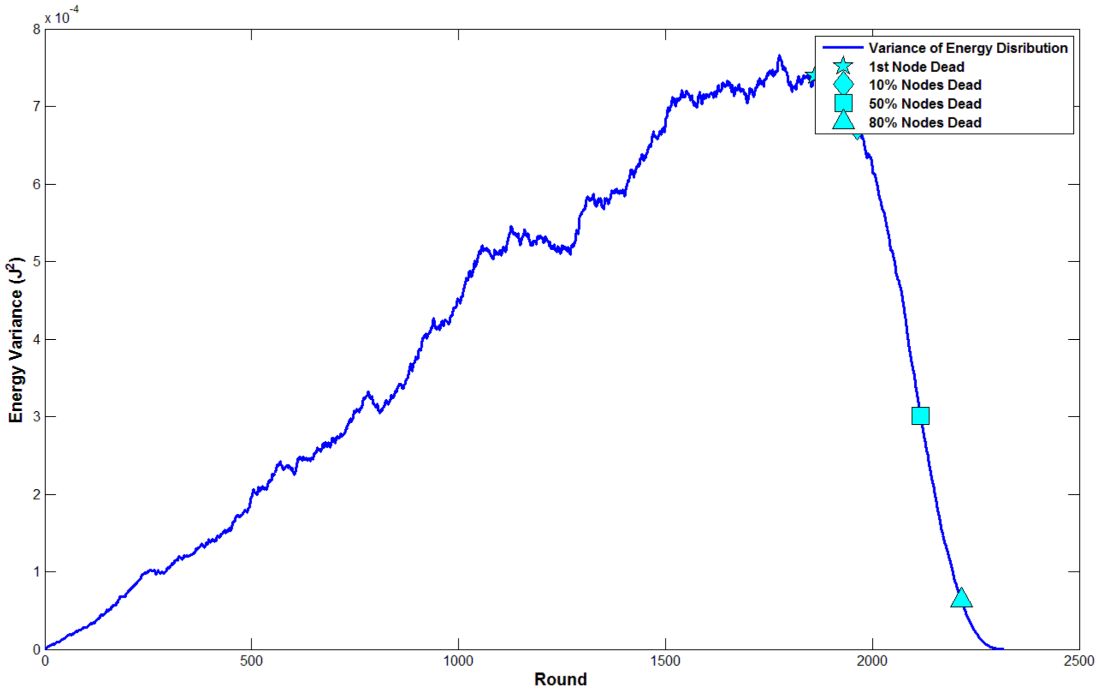


Figure 80. Zone clustering algorithm with random CH election routing in a multi-gateway WSN. The WSN energy variance versus transmission round is illustrated.

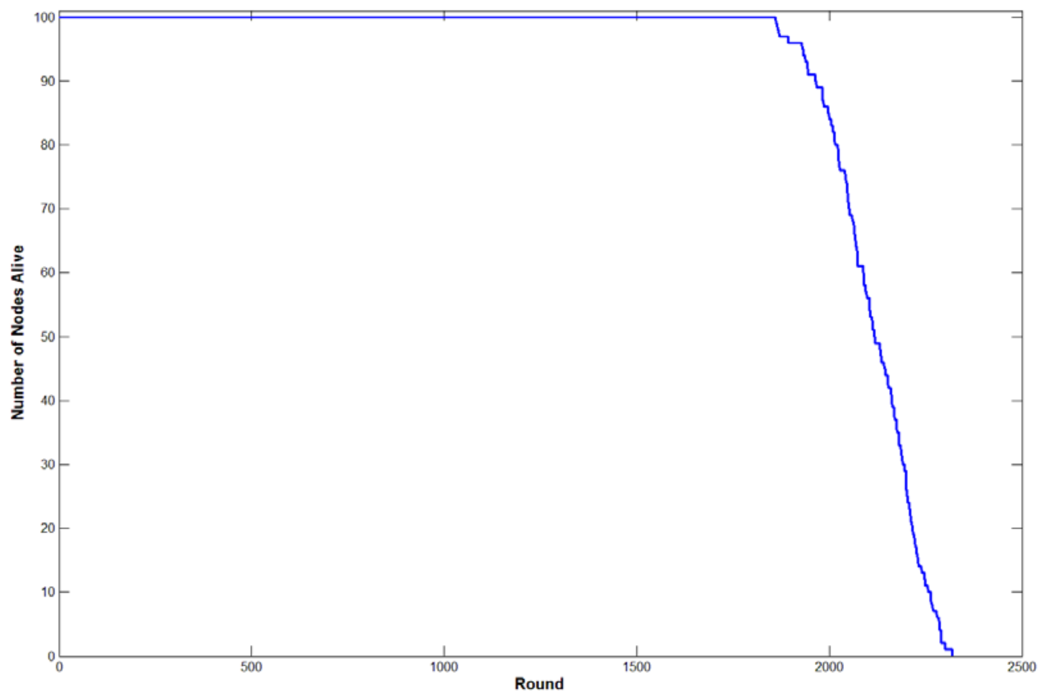


Figure 81. Zone clustering algorithm with random CH election in a multi-gateway WSN. The nodes alive versus transmission round is illustrated.

G. ZONE CLUSTERING WITH ENERGY EFFICIENT CLUSTER HEAD ELECTIONS

1. Single Gateway

Zone clustering with energy efficient CH election (EZone) splits the field into five zones, elects a CH in each zone to receive all L = 2000 bit packets from supported nodes according to the node that contains the highest energy, and then aggregates the packet into a 2000 bit packet for transmission to the gateway. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 82 through Figure 85, respectively, for the single gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 86 through Figure 88, respectively. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 2003, 2007, 2026, and 2051, respectively.

The energy stem plot and node distribution plots demonstrate that zones die out from the outer zones in the sensor network progressing toward the center. By restricting CH election criteria to choose the highest energy node in a zone, the energy level of all nodes in a common zone are uniformly preserved throughout the simulation. Nodes die out evenly such that there can potentially be several nodes dying out in any zone. This is noted in Figure 82 in that during the round where the first node died, there were actually two nodes that had died. Since the expected value of CH transmission distance is largest in outer zones, we expect them to die out first. The nodes alive versus transmission round plot shown in Figure 88 illustrates a very sharp knee with the network, going from 100 percent nodes alive to zero very quickly (approximately 50 rounds). This extends and preserves the timeframe that 100 percent of nodes are alive. The energy depletion rate of the network during the linear region of Figure 86 is 0.0246 J/round, which is a similar energy depletion rate for the other clustering mechanisms tested. By using an energy efficient clustering mechanism, we have extended the timeframe where 100 percent of nodes are alive, offering the greatest timeframe for total WSN service coverage.

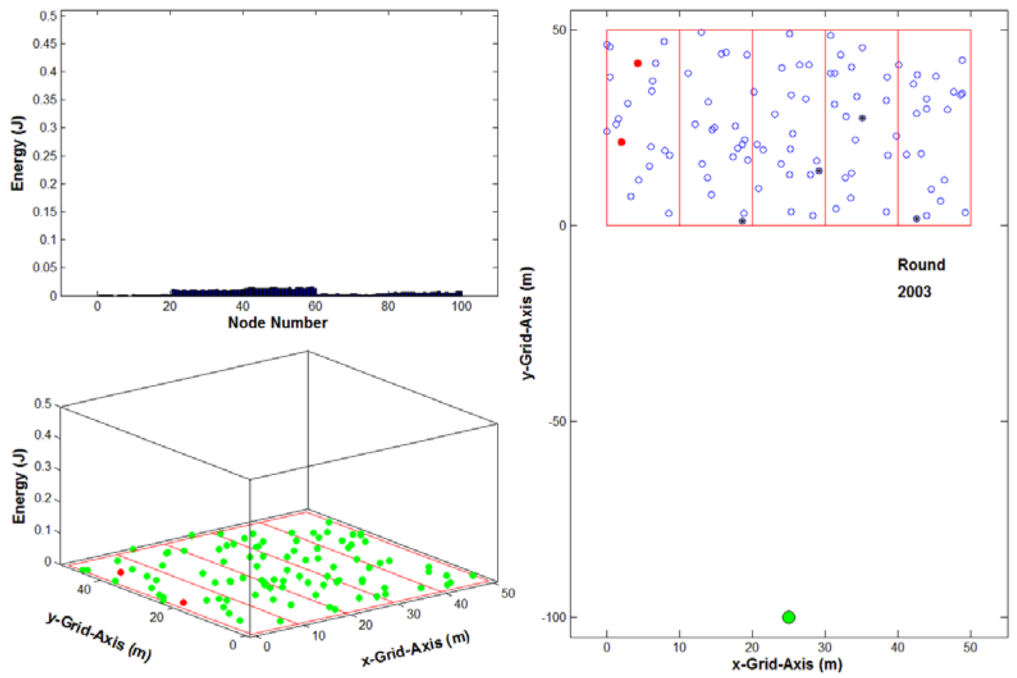


Figure 82. EZone cluster routing algorithm in a single gateway WSN. The first node dead die out topology versus transmission round and energy distributions is illustrated.

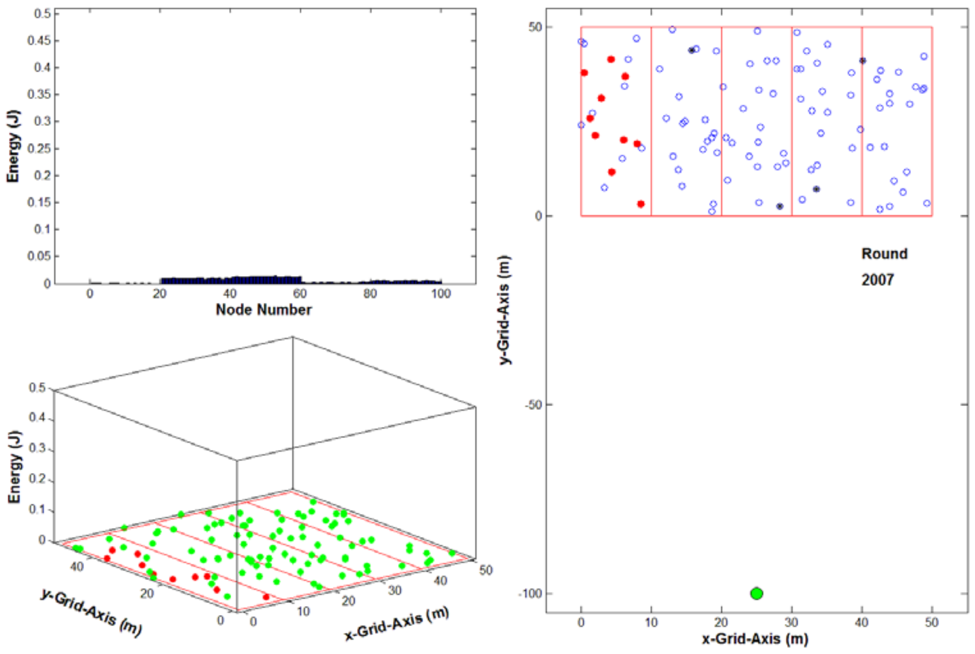


Figure 83. EZone cluster routing algorithm in a single gateway WSN. The percent nodes dead die out topology versus transmission round and energy distributions is illustrated.

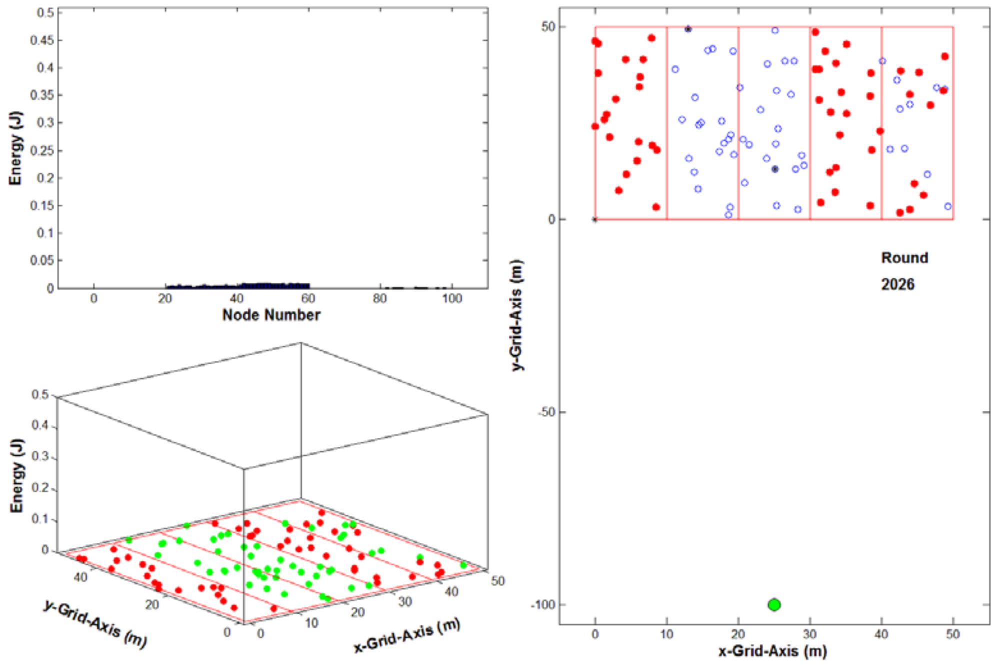


Figure 84. EZone cluster routing algorithm in a single gateway WSN. The 50 percent nodes dead die out topology versus transmission round and energy distributions is illustrated.

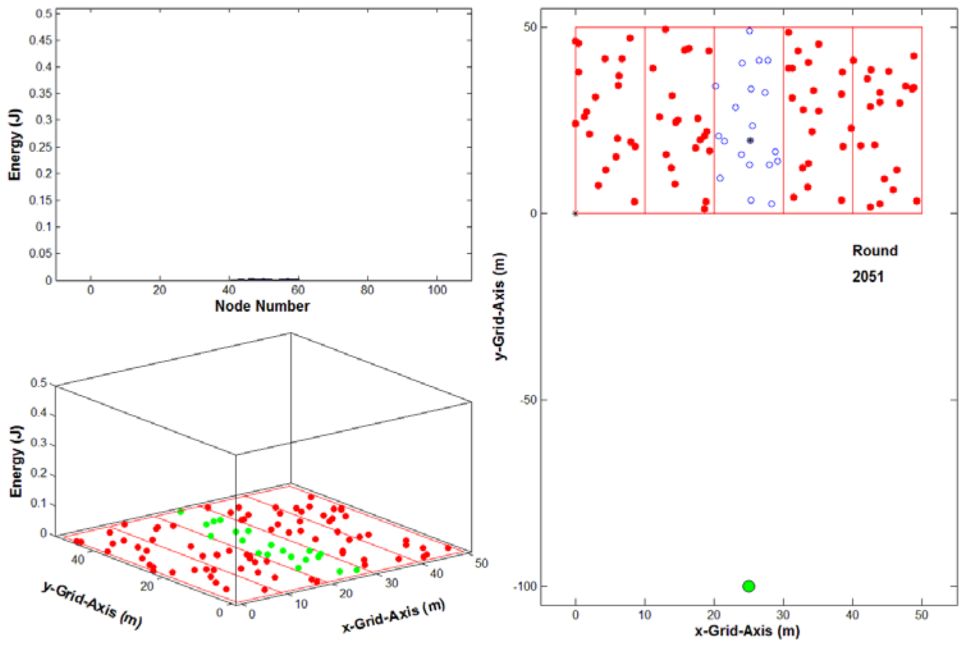


Figure 85. EZone cluster routing algorithm in a single gateway WSN. The the 80 percent nodes dead die out topology versus transmission round and energy distributions is illustrated.

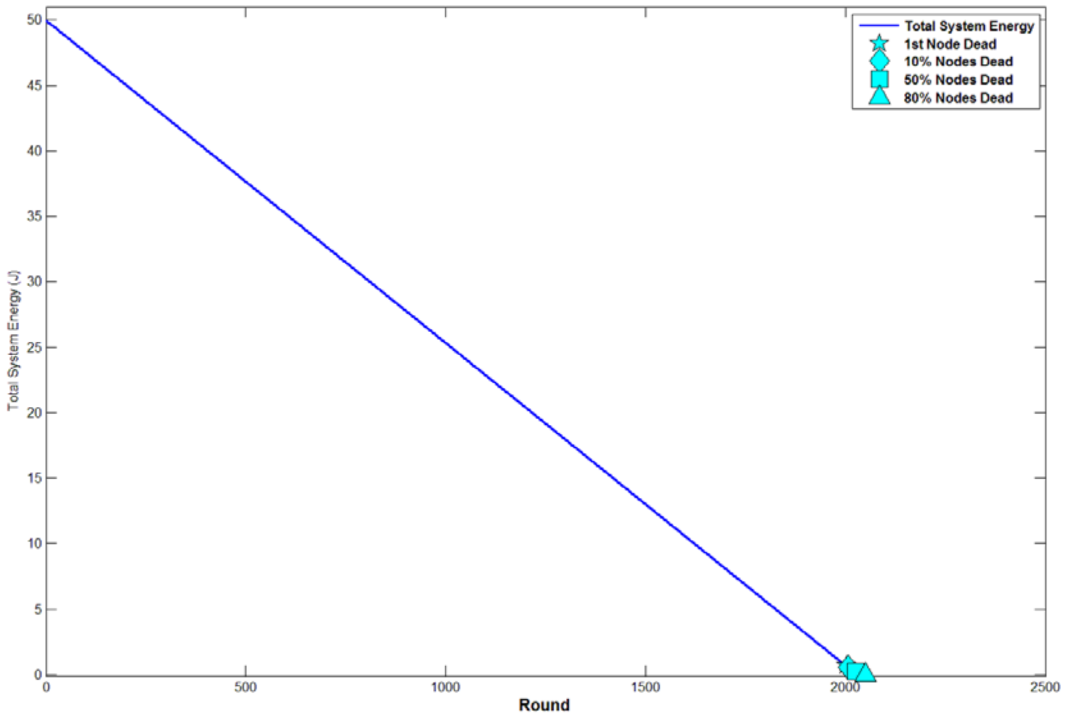


Figure 86. EZone cluster routing algorithm in a single gateway WSN. The total WSN energy versus transmission round is illustrated.

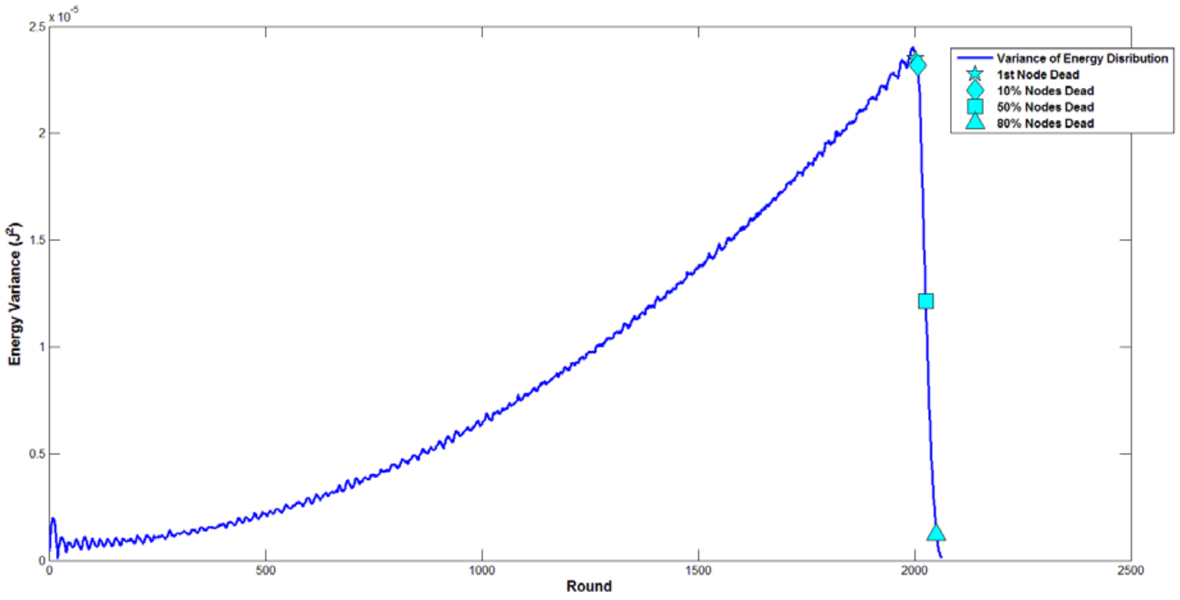


Figure 87. EZone cluster routing algorithm in a single gateway WSN. The WSN energy variance versus transmission round is illustrated.

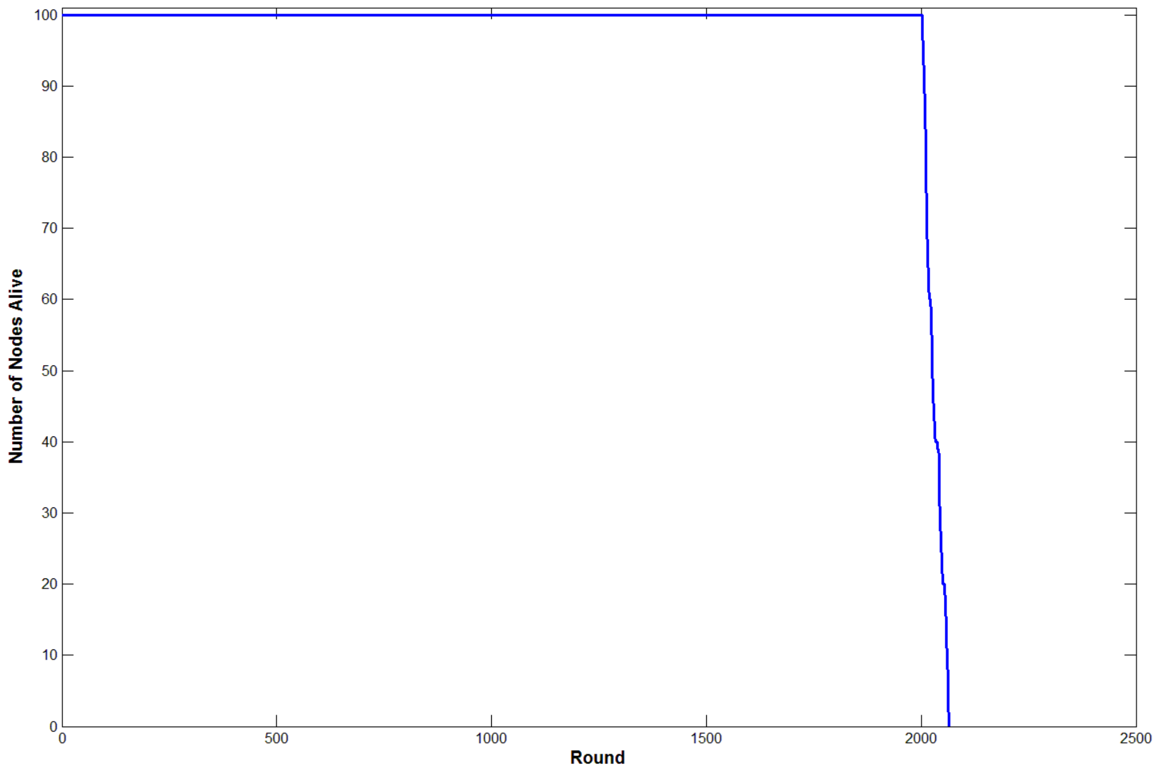


Figure 88. Zone EZone cluster routing algorithm in a single gateway. The nodes alive versus transmission round is illustrated.

Figure 82 through Figure 85 make it difficult to visualize the distribution of node energy since once the first node dies out, all nodes subsequently die out. To capture our claim that our energy efficient zone routing algorithm improves the energy distribution of all nodes, we show our energy bar plot and stem plots at round 500, and energy bar plots for rounds 1000 and 1500 in Figure 89 through Figure 91, respectively. A near uniform energy distribution is displayed in Figure 89 through Figure 91, which thus far has not been observed in previous simulations discussed in this thesis. As the round approaches 1500, the energy remains uniform in each zone, and the energy in each zone differs slightly from adjacent zones.

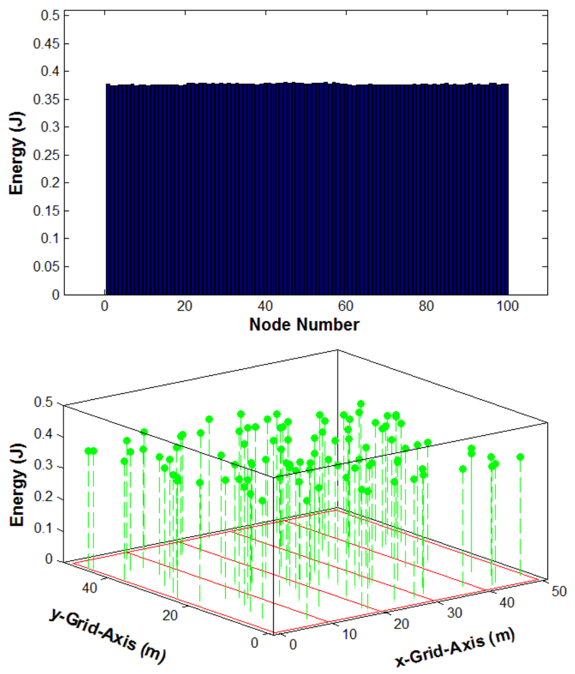


Figure 89. EZone cluster routing algorithm in a single gateway WSN. Round 500 is displayed with a uniform energy distribution.

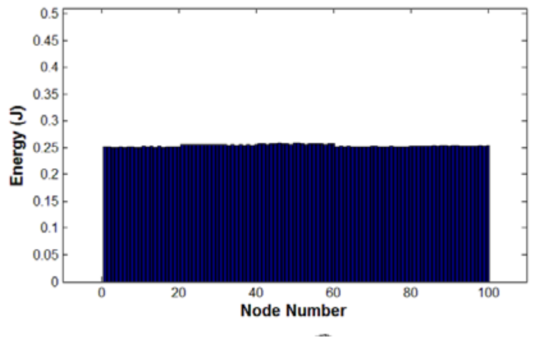


Figure 90. Similar to Figure 89 except the simulation is for round 1000.

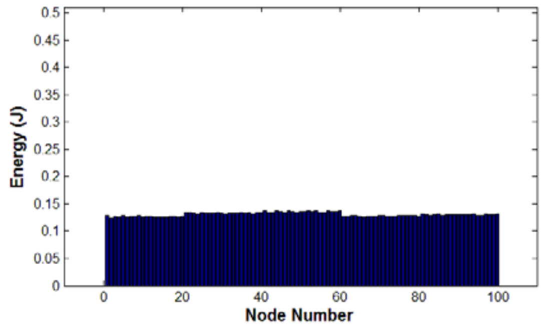


Figure 91. Similar to Figure 89 except the simulation is for round 1500.

2. Multi-gateway

EZone cluster routing algorithm for a multi-gateway scenario splits the field into five zones, elects a CH in each zone to receive all L = 2000 bit packets from supported nodes according to the node that contains the highest energy, and then aggregates the packet into a 2000 bit packet for transmission to the closest gateway. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead subplots are provided in Figure 92 through Figure 95, respectively, for the multi-gateway case. The total system energy, energy variance, and number of nodes versus transmission round are shown in Figure 96 through Figure 98, respectively. The first node dead, 10 percent, 50 percent, and 80 percent nodes dead occur at round 2116, 2119, 2126, and 2134, respectively. This corresponds to a percent increase of six percent, six percent, five percent, and four percent, respectively, when compared to single gateway die out statistics for the same algorithm. Nodes die out in a similar fashion as compared to the single gateway case except the knee on Figure 98 is sharper, causing just an 18 round period between the first node dead and 80 percent of nodes dead. Once the first node dies, all nodes quickly die. This is an ideal die out topology because it preserves the time of 100 percent network service. Maximizing the timeframe for 100 percent coverage is ideal to best provide maximum coverage area from all nodes. The energy depletion rate of the network during the linear region of Figure 96 is 0.0240 J/round, which corresponds to a four percent reduction in network energy depletion rate compared to the single gateway case.

The energy variance plot of Figure 95 shows a very small energy variance with a scale of 10^{-4} J^2. Rotating CHs according to highest energy in each zone causes an oscillatory compensating effect at the start of the simulation. This slowly increases toward the end of the simulation as the last zone to die out is the center zone because transmission energy required by the CH is less than the energy required by CHs in side zones.

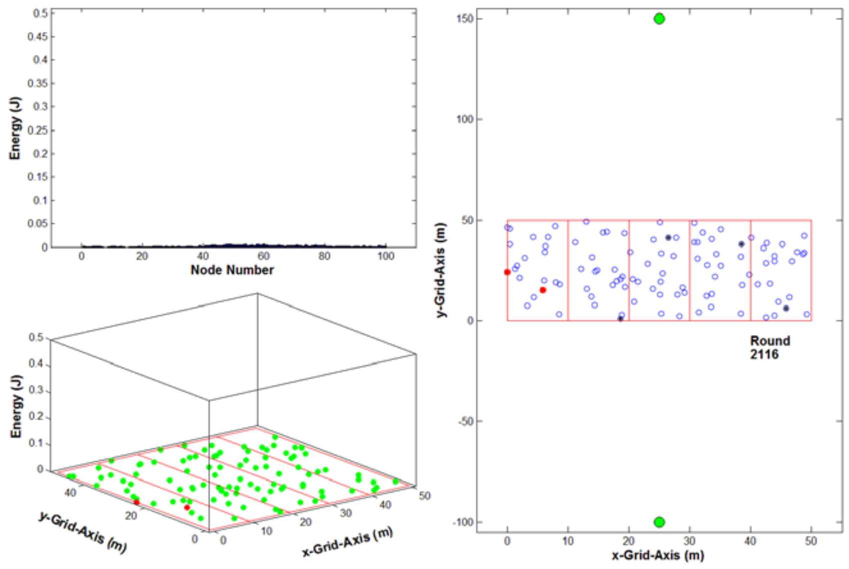


Figure 92. Multi-gateway EZone cluster routing algorithm. The first node dead die out topology versus transmission round and energy distributions is illustrated.

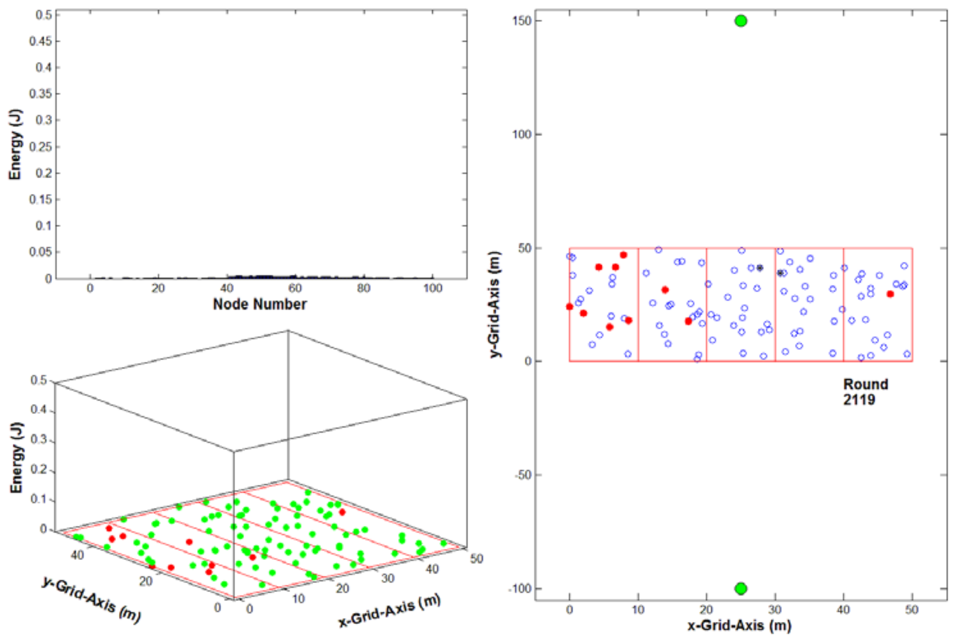


Figure 93. Multi-gateway EZone cluster routing algorithm. The 10 percent node dead die out topology versus transmission round and energy distributions is illustrated.

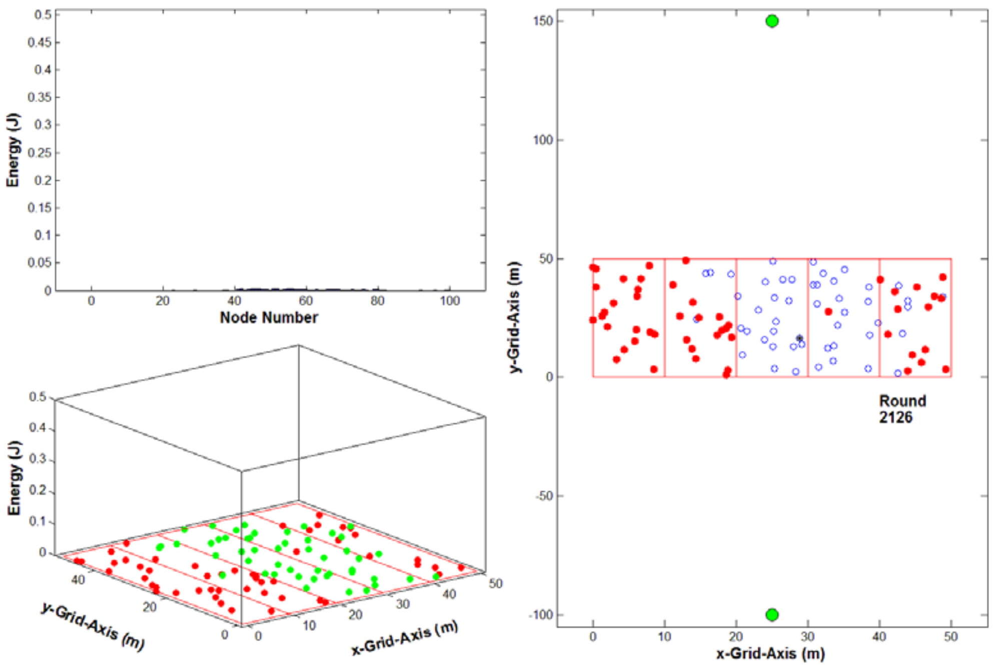


Figure 94. Multi-gateway EZone cluster routing algorithm. The 50 percent nodes dead die out topology versus transmission round and energy distributions is illustrated.

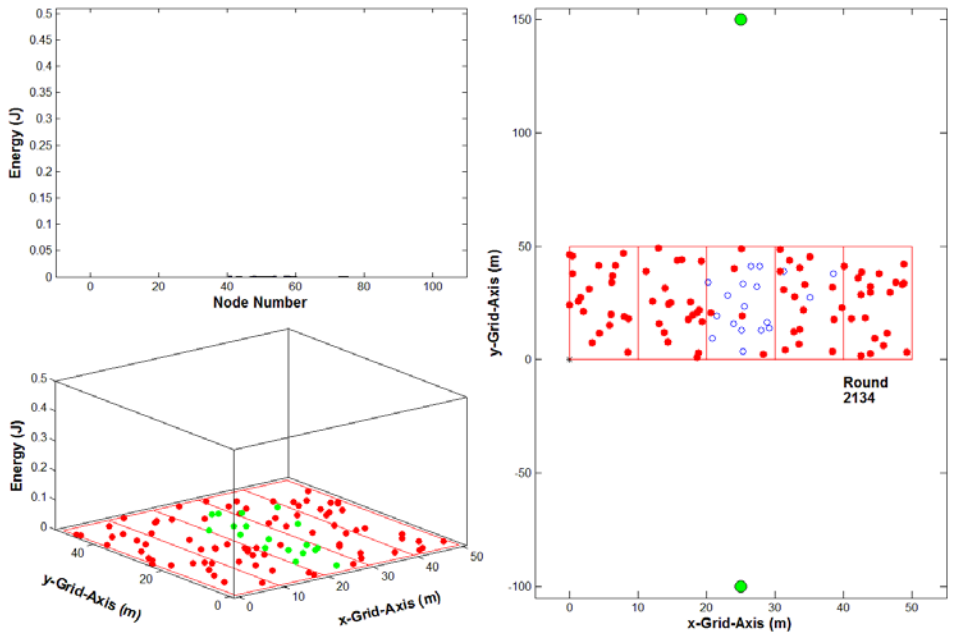


Figure 95. Multi-gateway EZone cluster routing algorithm. The 80 percent nodes dead die out topology versus transmission round and energy distributions is illustrated.

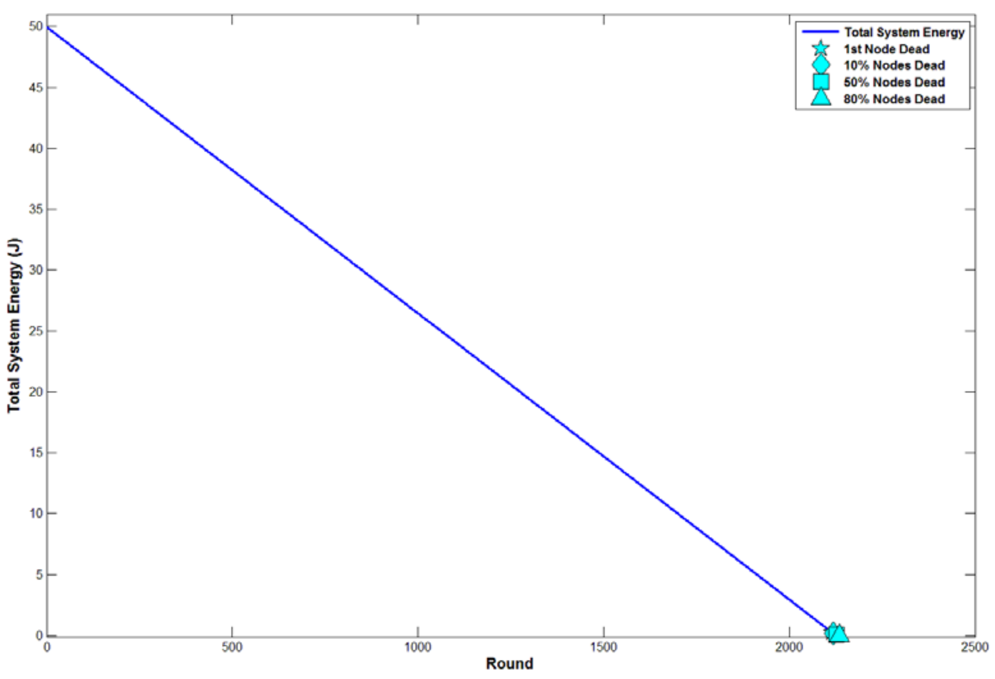


Figure 96. EZone cluster routing algorithm in a multi-gateway WSN. The total WSN energy versus transmission round is illustrated.

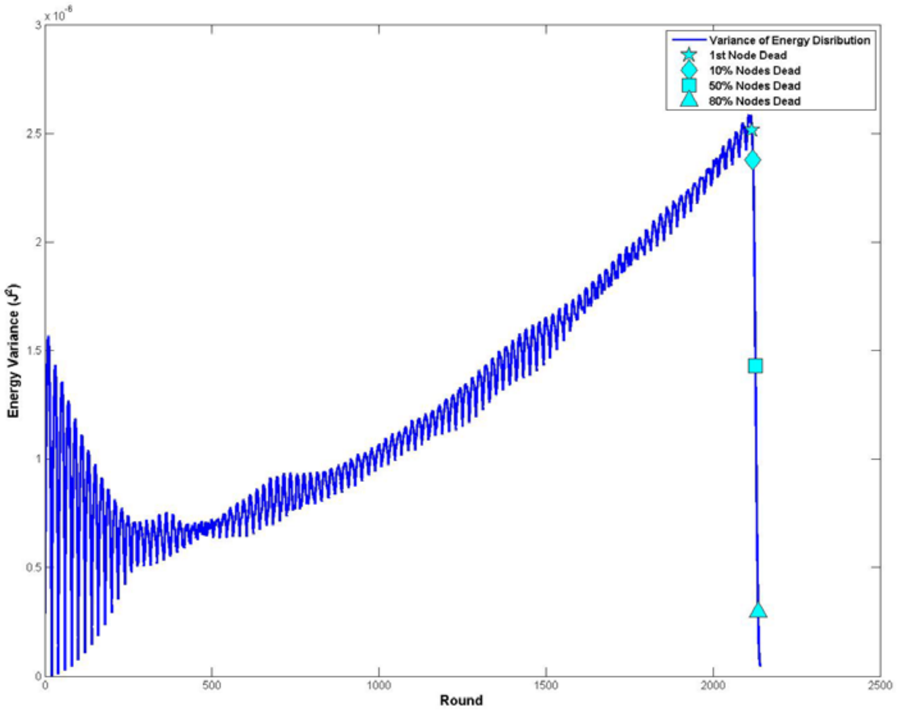


Figure 97. EZone cluster routing algorithm in a multi-gateway WSN. The WSN energy variance versus transmission round is illustrated.

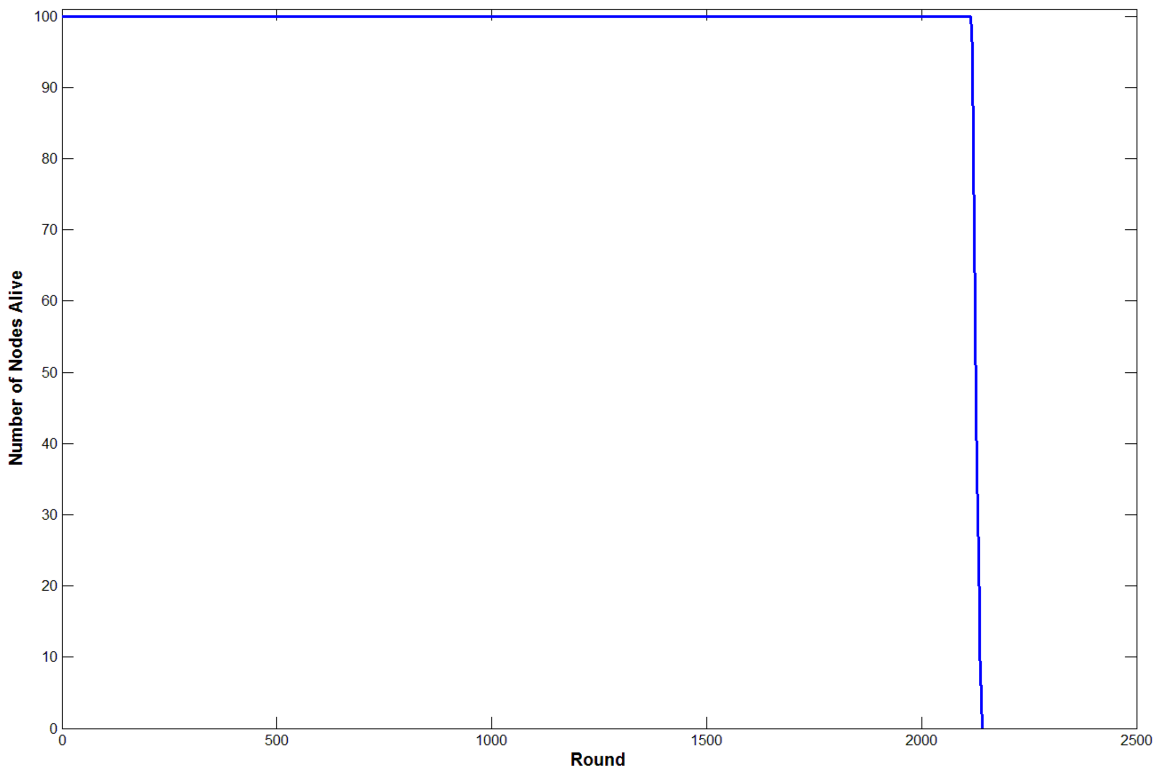


Figure 98. EZone cluster routing algorithm in a multi-gateway WSN. The number of nodes alive versus transmission round is illustrated.

Figure 92 through Figure 95 make it difficult to visualize the distribution of node energy since once the first node dies out, all node subsequently die out. To capture our claim that our energy efficient zone routing algorithm improves the energy distribution of all nodes, we show our energy bar plot and stem plots at round 500, and energy bar plots for rounds 1000 and 1500 in Figure 99 through Figure 101 for the multi-gateway scenario. A near uniform energy distribution is shown in Figure 99 through Figure 101, which is only noted as a result of the energy efficient zone routing approach taken in this thesis. As the round approaches 1500, energy variance is indistinguishable between zones. The additional gateway improves the uniformity of node depletion in comparison with the single gateway simulation.

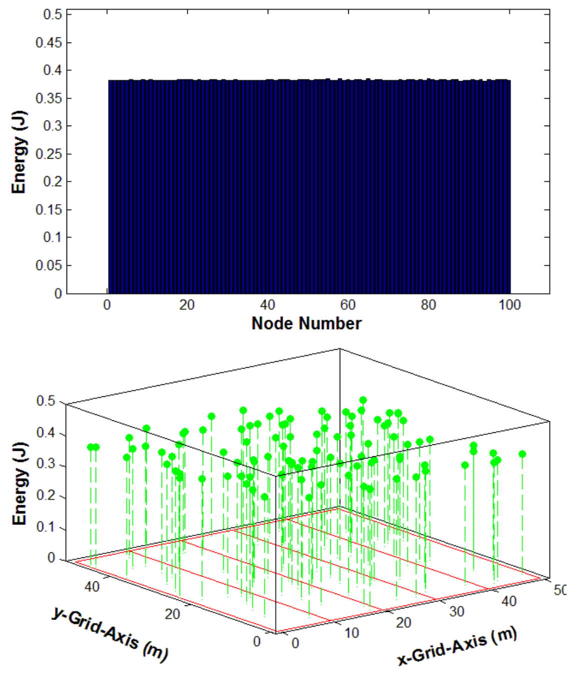


Figure 99. EZone cluster routing algorithm in multi-gateway WSN. Simulation for round 500 is shown with uniform energy depletion.

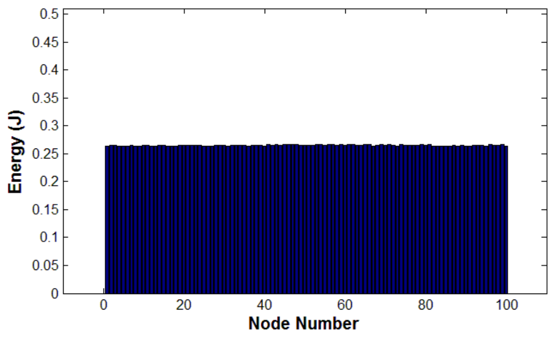


Figure 100. Similar to Figure 99 except the simulation is for round 1000.

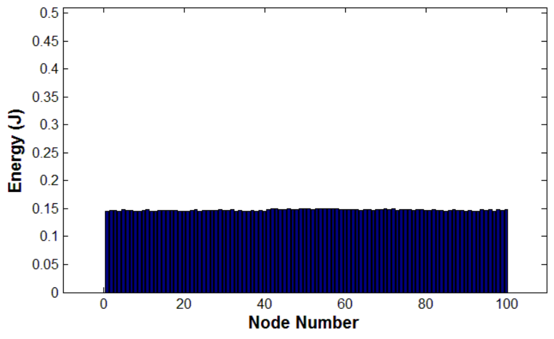


Figure 101. Similar to Figure 99 except the simulation is for round 1500.

H. ALGORITHM DATA COMPARISONS

1. WSN Die out Statistics and Energy Consumption Comparisons

An overall comparison of statistics for our WSN is provided in Table 4 and Figure 102 through Figure 104. The networking protocol for each simulation, several metrics of interest, and the transmission round in which the metric occurred for our single and multi-gateway simulations is described in Table 4. The percent increase of adding an additional gateway is calculated in Table 4 for each metric of each protocol. The energy depletion rates of the direct and MTE routing algorithms are much greater than those for clustering algorithms due to the energy balance at the physical layer imposed by the networking layer and data aggregation at the application layer of the CH. Zone routing with energy efficient CH election (EZone) provided the longest timeframe of full WSN service (all nodes alive). The single gateway case performed better than the multi-gateway cases for LEACH and zone clustering with random CH election. This demonstrates the impact of optimizing an energy efficient network layer strategy. As efficiency is gained at the network and application layer, the impact of the additional gateway is lowered when looking at the energy depletion rate (without consideration for topology of WSN die out). The clustering algorithms all demonstrated approximately similar energy depletion rates for respective single and multi-gateway configurations. This value was obtained in the linear region of the plots with all nodes alive and five CHs elected for each round.

The 80 percent die out range in each round is simply the round when 80 percent of nodes are dead subtracted from the round the first node died. This range, divided by the round the first node died, provides a metric for the timeframe during which network die out occurs compared to 100 percent WSN service life. The MTE algorithms (single gateway and multi-gateway) clearly behaved the worst due to the hot-node phenomena. Zone routing with energy efficient CH election (EZone) performed the best and minimized the die out range to the smallest possible value as a result of the physical layer amplifier used, the location of the gateway(s), and sensor field parameters.

LEACH provided the longest timeframe preserving some WSN service (80 percent nodes dead, or 20 nodes alive) due to the dynamic nature of choosing the CH. However, LEACH did not control the topology of WSN die out in any stable fashion. We provide further comments later in the chapter by comparing the clustering techniques utilized.

A comparison of the total system energy is illustrated Figure 102, a comparison of energy variance is illustrated in Figure 103, and a comparison of nodes alive versus transmission round is illustrated in Figure 104. The addition of another gateway was most significant in the direct and MTE algorithms, as shown in Figure 103, as the energy variance is lowered by approximately 50 percent. Energy variance of the zone routing algorithms were both lower than LEACH, with the single gateway scenarios performing better than LEACH in a multi-gateway configuration. The direct and MTE number of nodes alive do not cross in Figure 104 as they do in [1] because the authors only used a direct path propagation model, while our research uses both direct path and multi-path propagation models. Zone routing with energy efficient CH election (EZone) offered the most time with all nodes alive; however, LEACH offered the most time with at least one node alive.

Table 4. Overall algorithm die out statistics with a comparison of single and multi-gateway.

Protocol	Metric	Single Gateway	Multi Gateway	% Increase
Direct	Round First Dead	356	652	83
	10% Nodes Dead	410	712	74
	50% Nodes Dead	652	911	40
	80% Nodes Dead	939	1128	20
	Energy Depletion Rate (J^2)	0.0798	0.0554	−31
	80% Dieout Range (rounds)	583	476	−18
	80% Dieout range/Round First Dead	1.6376	0.7301	−55
MTE	Round First Dead	11	17	55
	10% Nodes Dead	77	100	30
	50% Nodes Dead	199	293	47
	80% Nodes Dead	354	453	28
	Energy Depletion Rate (J^2)	0.2140	0.1418	−34
	80% Dieout Range (rounds)	343	436	27
	80% Dieout range/Round First Dead	31.1818	25.6471	−18
LEACH	Round First Dead	1642	1633	-1
	10% Nodes Dead	1760	1805	3
	50% Nodes Dead	1990	2112	6
	80% Nodes Dead	2182	2327	7
	Energy Depletion Rate (J^2)	0.0245	0.0232	−5
	80% Dieout Range (rounds)	540	694	29
	80% Dieout range/Round First Dead	0.3289	0.4250	29
Protocol	Metric	Single Gateway	Multi Gateway	% Increase
Zone	Round First Dead	1649	1862	13
	10% Nodes Dead	1821	1964	8
	50% Nodes Dead	2022	2117	5
	80% Nodes Dead	2140	2215	4
	Energy Depletion Rate (J^2)	0.0248	0.0235	−5
	80% Dieout Range (rounds)	491	353	−28
	80% Dieout range/Round First Dead	0.2978	0.1896	−36
EZone	Round First Dead	2003	2116	6
	10% Nodes Dead	2007	2119	6
	50% Nodes Dead	2026	2126	5
	80% Nodes Dead	2051	2134	4
	Energy Depletion Rate (J^2)	0.0246	0.0235	−4
	80% Dieout Range (rounds)	48	18	−63
	80% Dieout range/Round First Dead	0.0240	0.0085	−65

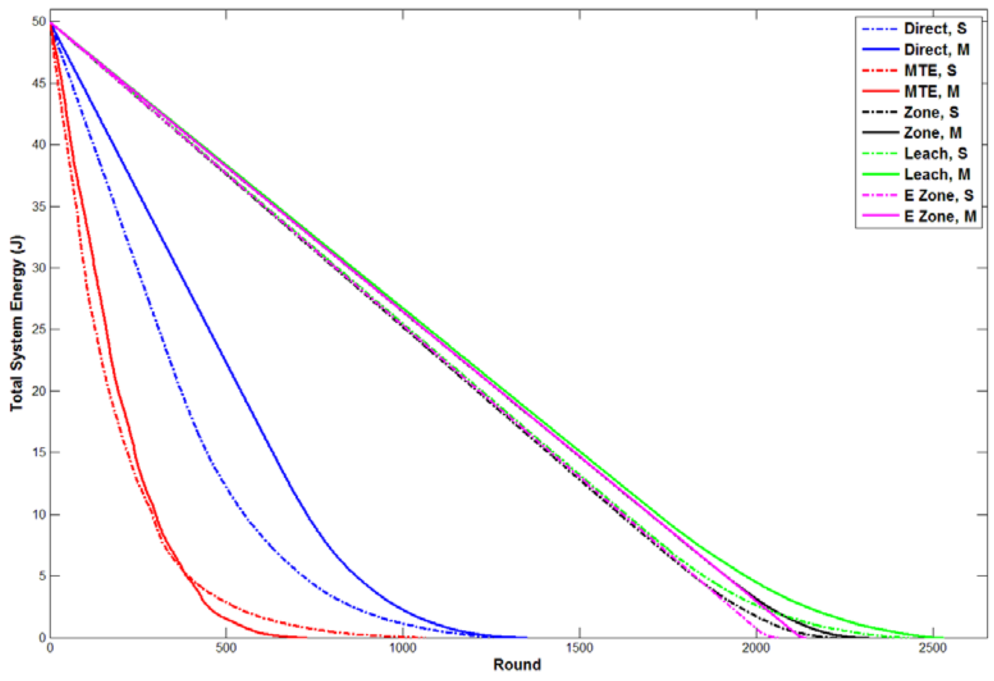


Figure 102. Comparison of total system energy versus transmission round for all algorithms in single and multi-gateway simulations.

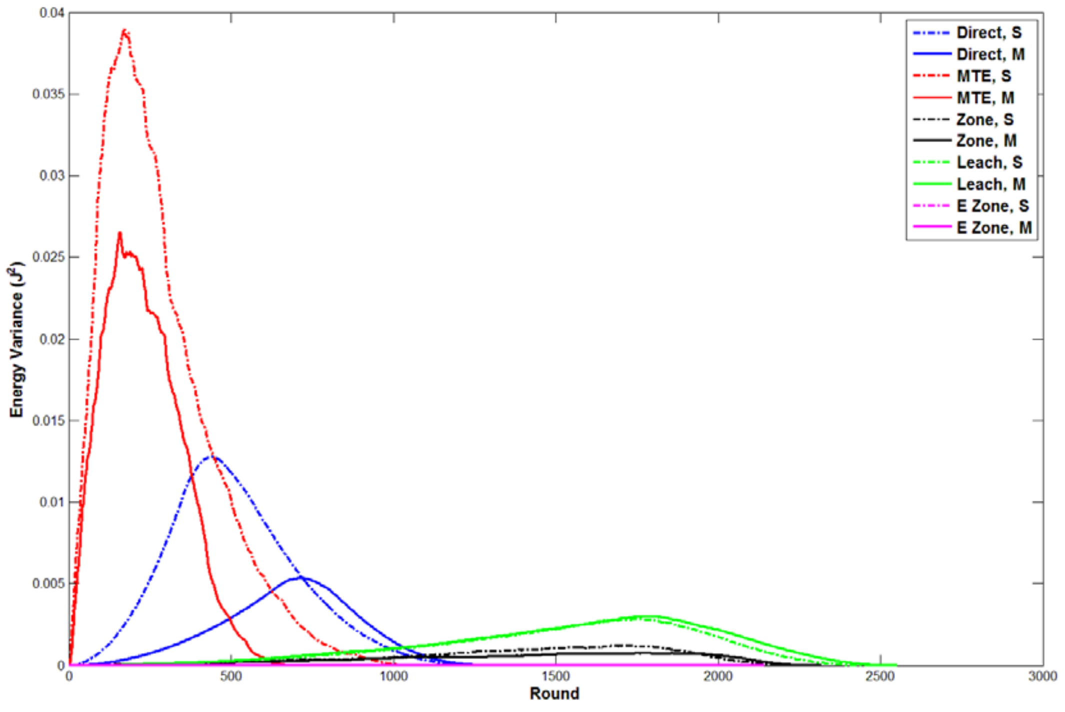


Figure 103. Comparison of Energy Variance versus transmission round for all algorithms in single and multi-gateway simulations.

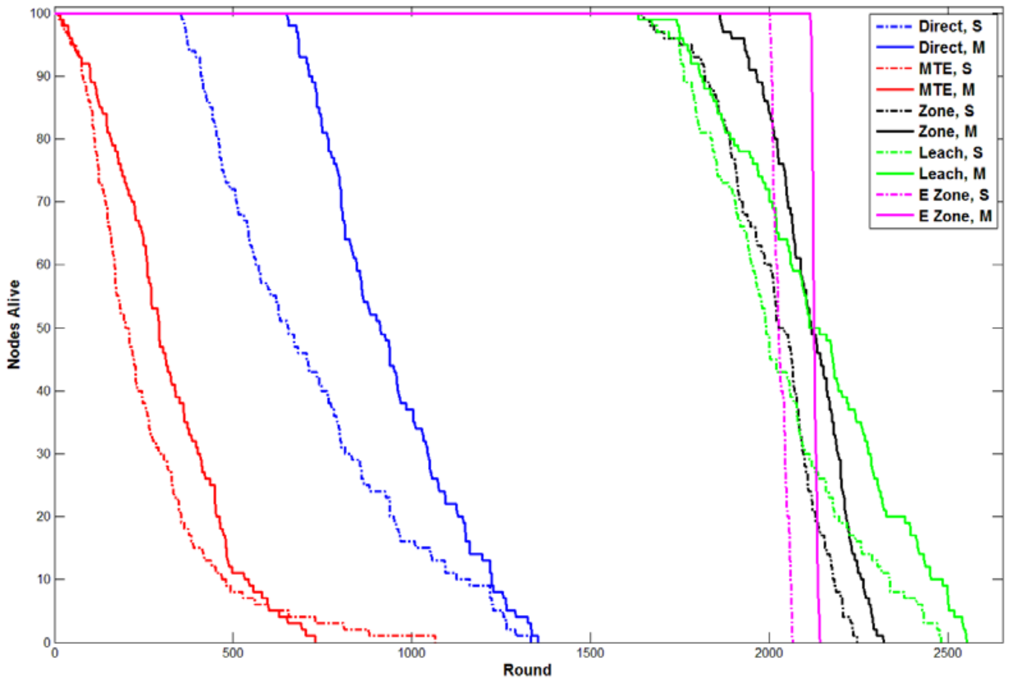


Figure 104. Comparison of Nodes Alive versus transmission round for all algorithms in single and multi-gateway simulations.

2. Comparison of Clustering Mechanisms

The difference in the WSN topology during die out is attributed to the CH election mechanism in each algorithm. Since LEACH employs a random method of CH election, the number of CHs is dynamic at each round yet maintains an average number of CHs. The number of CHs during each round in LEACH (blue) and a 50-point moving average (red) along with die out parameters for the single and multi-gateway scenarios are plotted in Figure 105 and Figure 106, respectively. The 50-point moving average remains at approximately five until the first node dies, corresponding to the input for the desired percentage of nodes to act as CHs (p=0.05). We note that after the first node dies, the average number of CHs is reduced to zero during the remaining rounds. This signifies that the LEACH process is stable only when all nodes are alive. As the number of CHs increases in the stable region, there is less demand on the CH performing data aggregation since each CH is aggregating fewer messages. This dynamic does not have an impact on the energy depletion rates because the algorithms only decrement energy to aggregate a final L-bit message without accounting for how many messages the CH is aggregating. This assumption is somewhat limiting. However, it is consistent with data aggregation techniques employed in the literature. Additional data aggregation strategies and their impacts are proposed as future work.

At approximately round 1750 and 2000 for the single and multi-gateway scenarios, respectively, there are no CHs chosen. Many more such occurrences are seen in later rounds. In this circumstance, each node transmits its payload directly to the closest gateway, eliminating any data aggregation opportunity. The result is that no CHs are chosen. This is a disadvantage of LEACH that can occur regularly as discussed in Chapter IV.

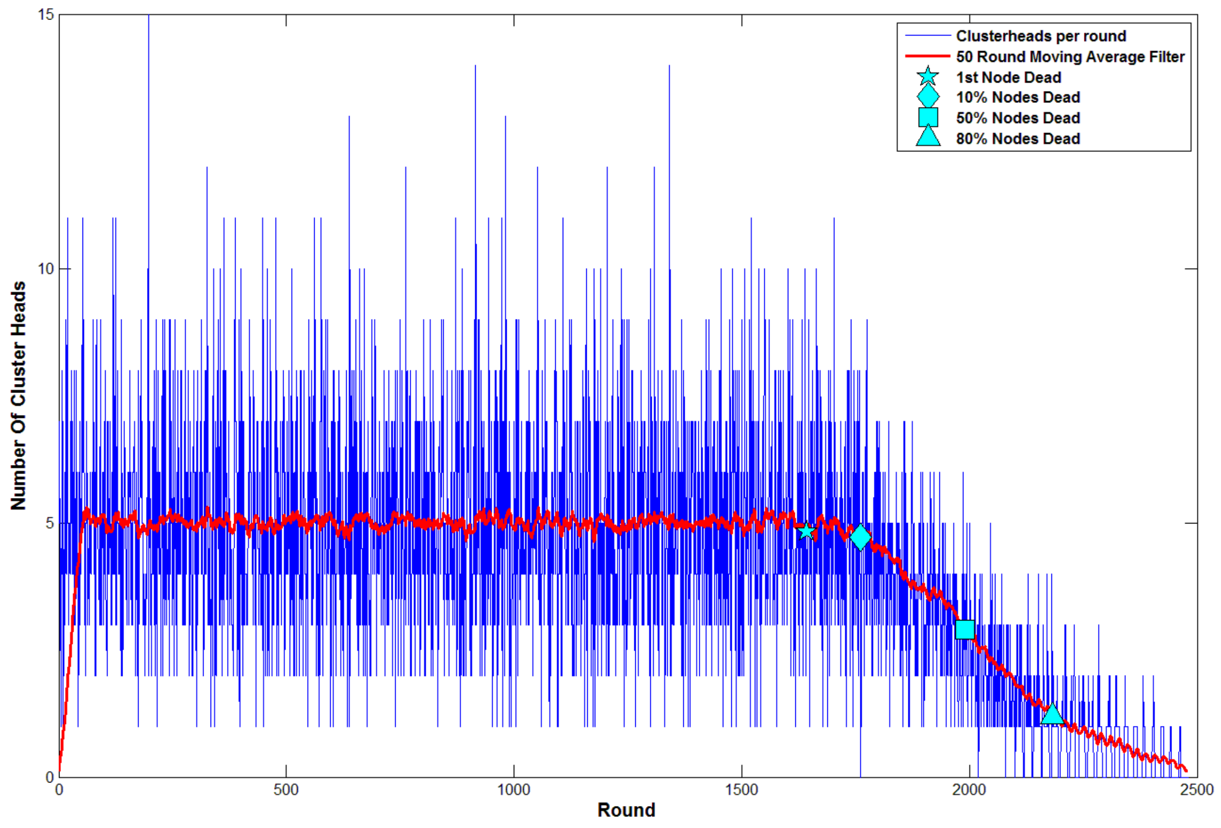


Figure 105. LEACH routing in a single-gateway WSN. The number of CHs chosen during each round (blue) along with a smoothed 50-point moving average filter (red) is illustrated.

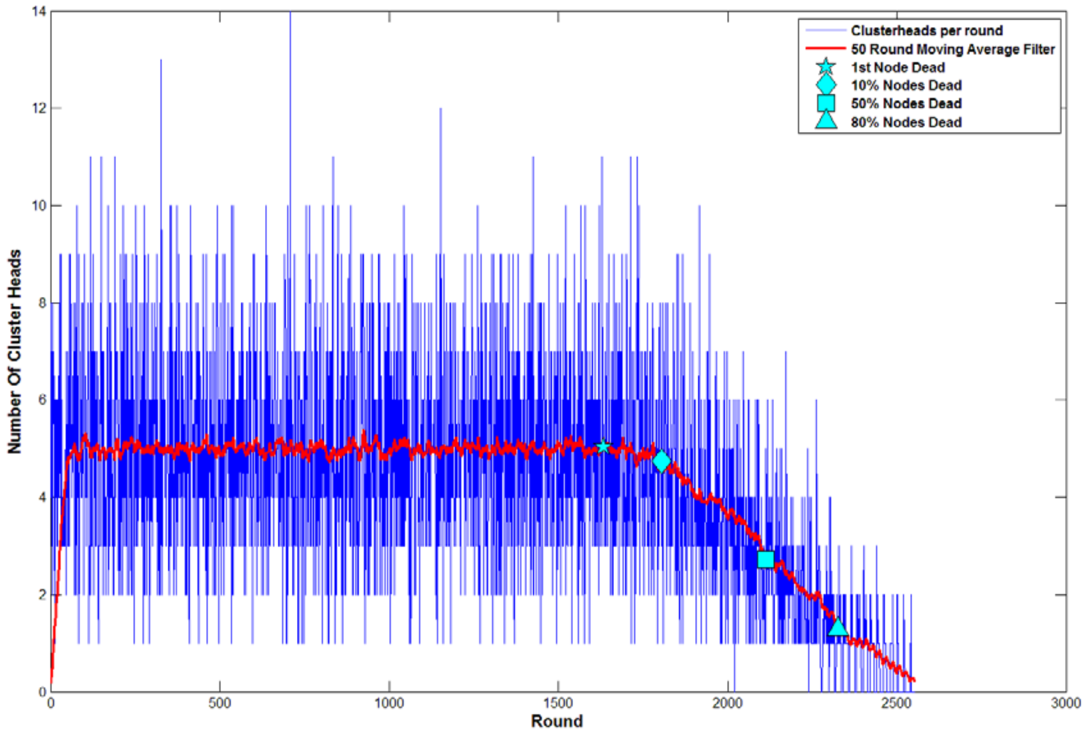


Figure 106. LEACH routing in a multi-gateway WSN. The number of CHs chosen during each round (blue) along with a smoothed 50-point moving average filter (red) is illustrated.

The moving averages plotted in Figure 105 and Figure 106 along with the number of CHs for the zone algorithms (Zone and EZone) versus each round are plotted in Figure 107. The zone routing algorithm reveals a stair-step pattern because each zone guarantees one CH at each round until all nodes in the zone are dead. Once all nodes in a zone dies, there is one less zone in the simulation, and subsequently, one less CH at each round is elected. Maintaining a more consistent number of CHs in each round over the lifetime of the WSN causes the energy variance of the zone routing algorithms to be lower than that of the LEACH algorithm (Figure 103).

To further investigate the topology of clustering in LEACH, we plot a Voronoi diagram for a few rounds. A Voronoi diagram partitions a space into areas according to a specific parameter. This relates nicely to our clustering algorithms because we can view the arrangement of the clusters in any round. A Voronoi diagram is shown for a few rounds during the performance of the LEACH simulations in Figure 108 through Figure 113. These figures illustrate the dynamic nature of LEACH CH election as it relates to

Figure 105 and Figure 106. In each Voronoi diagram, all nodes of the WSN are plotted as outlined blue circles, while the CH chosen for each round is a sold blue circle. Lines are plotted in the diagram to partition the space into clustered areas where each line divides the space in areas closest to their CH. All nodes in a particular area are served by the CH in the corresponding area. We note that there is a significant dynamic in the number of CHs chosen (ranging from three to nine) and the arrangement of each cluster spatially in the sensor field.

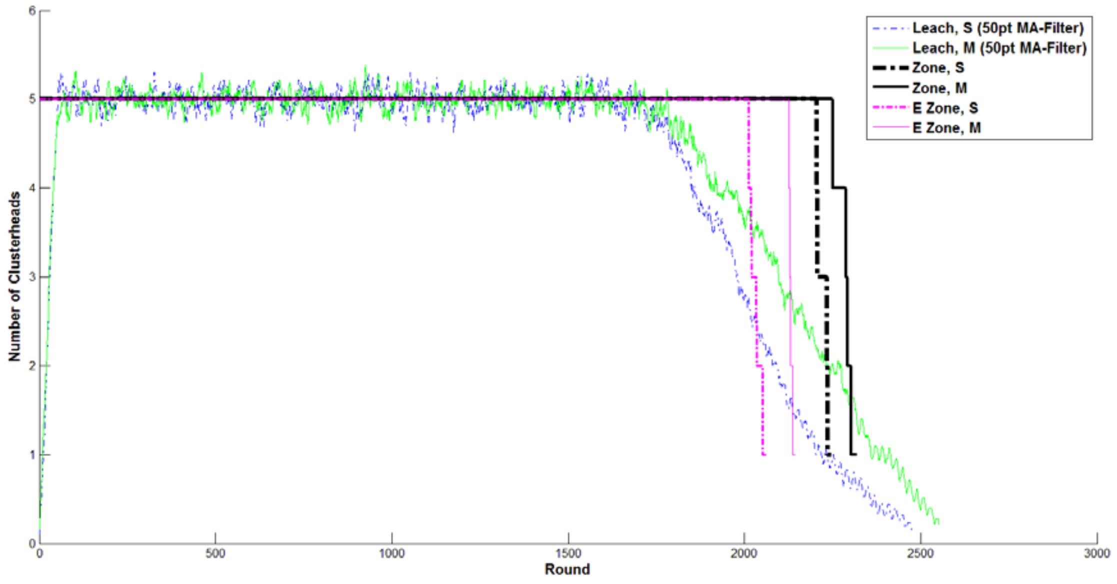


Figure 107. Comparison of the number of CHs versus transmission round for clustering and zone algorithms. The LEACH plot is the 50-point moving average contained in Figure 105 and Figure 106 (S~Single Gateway, M~Multi-gateway).

LEACH does not partition the sensor field into even nor tactically motivated clusters. For example, two CHs side by side with the third CH on the opposite side of the network are depicted in Figure 108. As a result, some nodes in an area have to transmit their payloads over far distances, contributing to them dying out earlier. The number of nodes in each LEACH cluster clearly is not balanced since nodes associate with their closest CH. Since LEACH is a dynamic zoning mechanism, it does not consider WSN topology in the decision of how and where to form clusters.

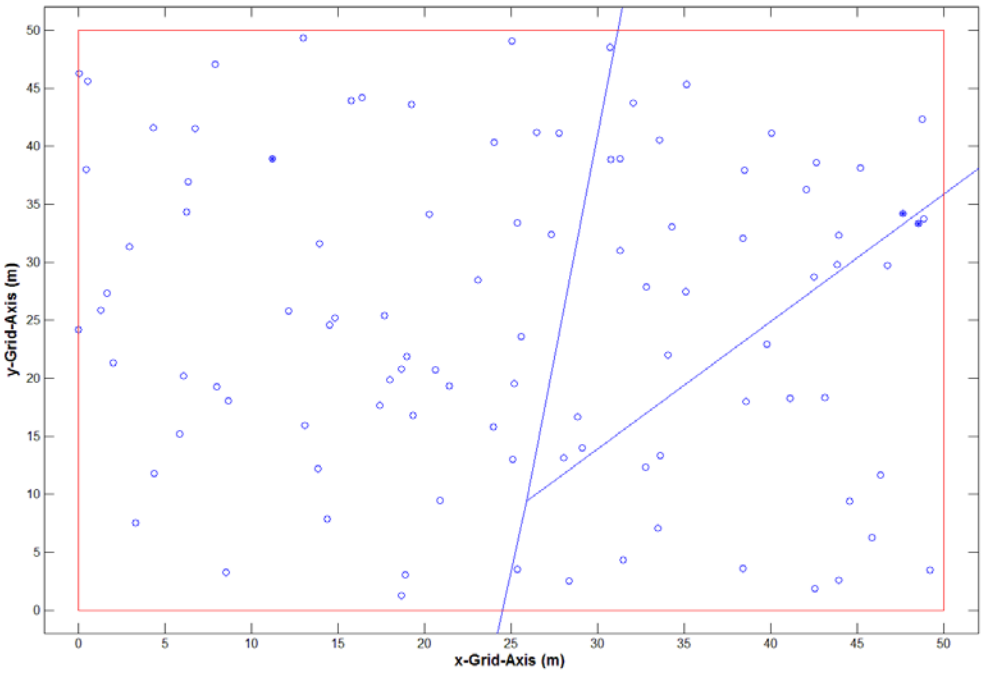


Figure 108. A Voronoi diagram from a round in the single gateway LEACH simulation displaying three clusters.

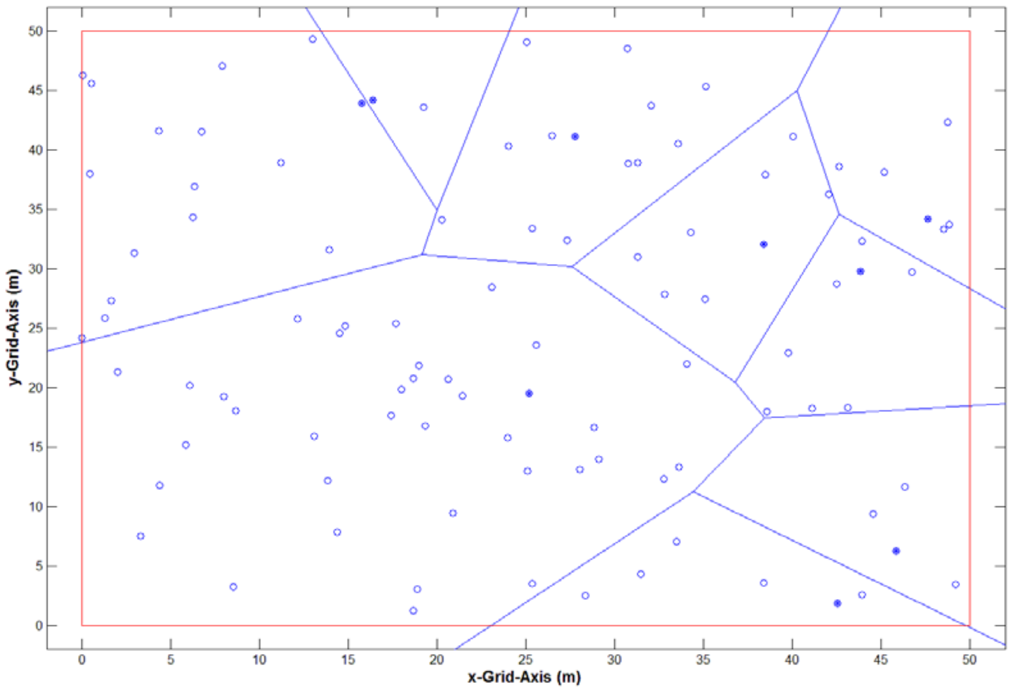


Figure 109. A Voronoi diagram from a round in the single gateway LEACH simulation displaying nine clusters.

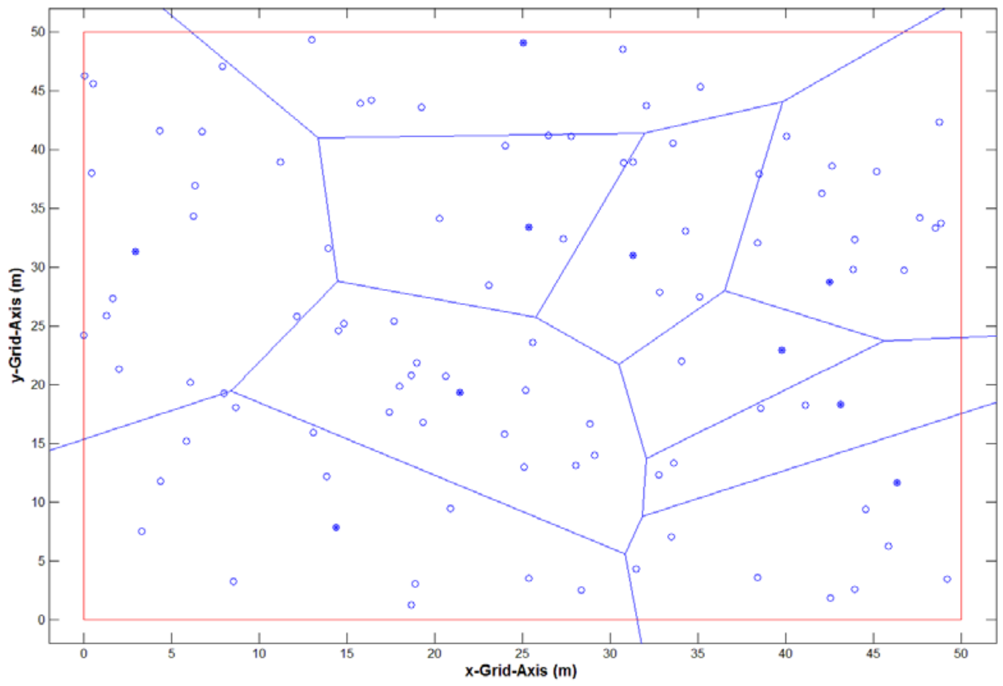


Figure 110. A Voronoi diagram from a round in the single gateway LEACH simulation displaying 10 clusters.

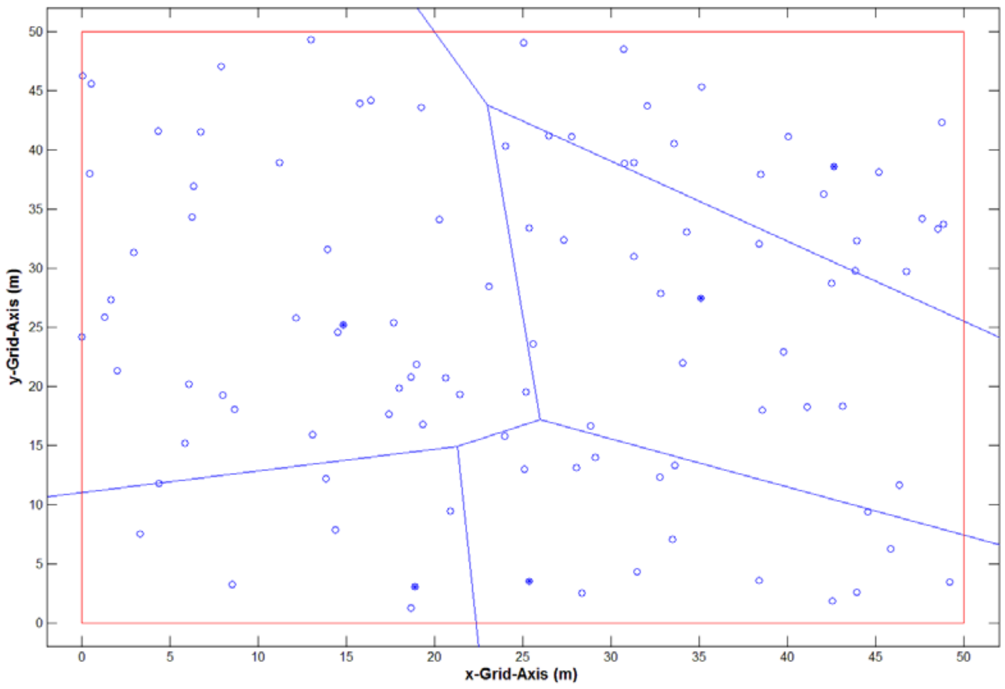


Figure 111. A Voronoi diagram from a round in the single gateway LEACH simulation displaying five clusters.

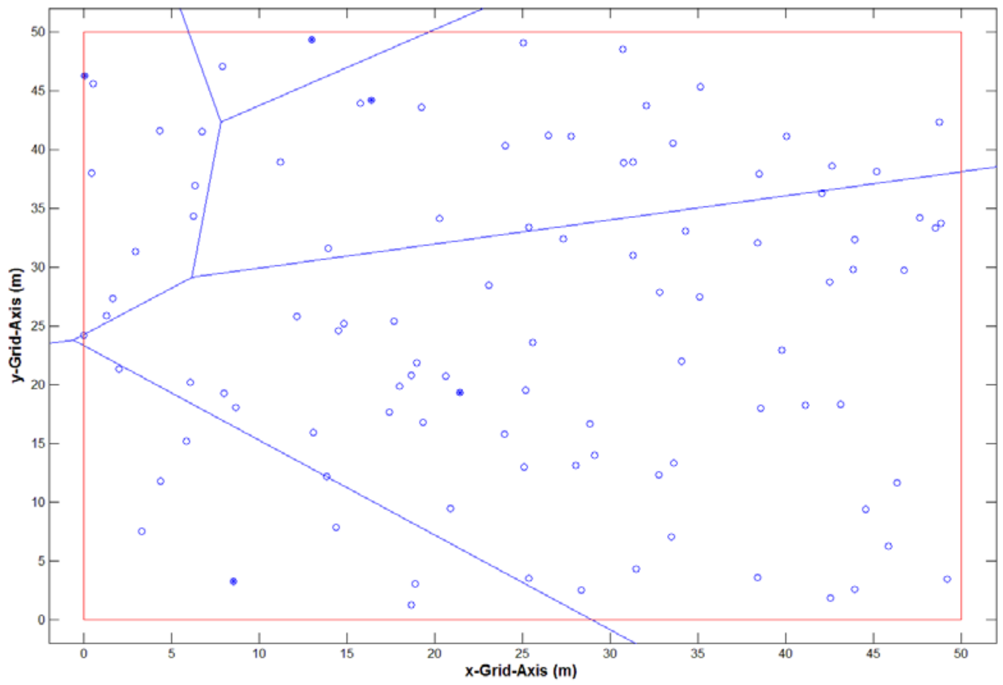


Figure 112. A Voronoi diagram from a round in the single gateway LEACH simulation displaying five clusters.

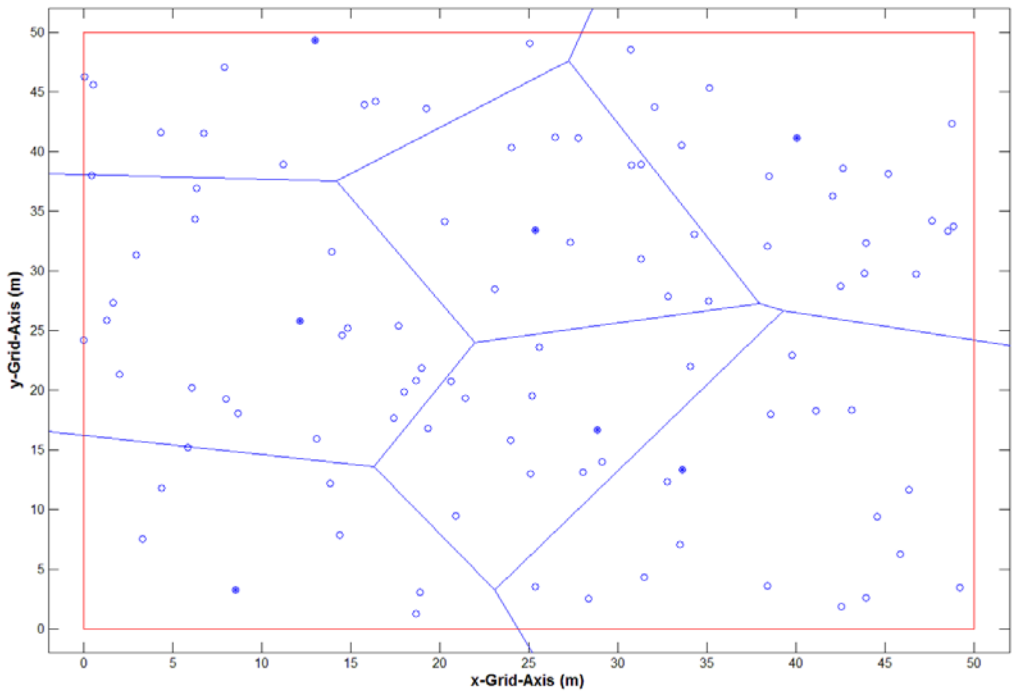


Figure 113. A Voronoi diagram from a round in the single gateway LEACH simulation displaying seven clusters.

Compare the clustering mechanism of LEACH to that imposed by our zone clustering algorithms. Zones fully describe clusters, and each zone has an equal number of sensors. We chose this mechanism for clustering since sensors are uniformly distributed across the network and application layer loading is CBR for each round. A Voronoi diagram for zone clustering is simply the node distribution plots of the results discussed earlier in this chapter. In those plots, the CH is represented by an asterisk in the corresponding blue circle as compared to the solid blue circle in the LEACH Voronoi diagrams. The zone clustering Voronoi diagram differs from the LEACH Voronoi diagrams in that LEACH zones separate nodes corresponding to their closest CH. In the event that the application layer loading is not uniform or nodes are not uniformly distributed throughout the field, zoning could be performed in some other tactical fashion to impact WSN service life and network die out characteristics. This is left as an opportunity for future work.

I. CHAPTER V SUMMARY

In this chapter, we provided the results for the algorithms described in Chapter IV. We described metrics of interest for our simulations and the specific metrics that we obtained from each algorithm. We provided results for each algorithm individually for single and multi-gateway simulations and then showed and discussed how all results compare to each other. We specifically focused on how the network layer routing algorithm affects when and how nodes die and the energy distribution of the WSN during the entire simulation. Ultimately, our energy efficient zone routing algorithm (EZone) provided the longest timeframe of 100 percent WSN service in a tactically motivated fashion, but LEACH provided the longest timeframe with at least one node alive. Extending the timeframe of 100 percent service coverage is desired to provide the greatest lasting performance of the entire WSN. The results presented in this chapter only focused on one WSN sensor field arrangement. To investigate other arrangements, we modeled network die out parameters as random variables and obtained the distribution of network die out. This is presented in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. WSN DIEOUT RANDOM VARIABLE MODELING

The results presented in Chapter V only focused on one WSN sensor field arrangement. To investigate other arrangements, we modeled network die out parameters as random variables and obtained the distribution of network die out. In this chapter, we describe our random variable (RV) WSN die out testing. We simulated each algorithm many times to obtain die out distributions and to obtain the mean and standard deviation of the distributions. We performed these simulations to extend our conclusions beyond the one WSN configuration simulated in Chapter V.

A. MODELING WSN DIE OUT AS RANDOM VARIABLES

All pseudocode presented in Chapter IV and their actual code in Appendix B is easily modified to model the round the first node dies and the rounds when, 10 percent, 50 percent and 80 percent of nodes die. This is accomplished by including another loop in the code. Every iteration, a new random WSN with uniform node distribution is created and run using similar parameters as before with die out parameters being appended to each RV array. We utilized similar parameters for the number of nodes in the field, field dimensions, gateway locations, and physical and networking parameters. The only difference is that during each iteration of the algorithm, nodes are placed in different uniform locations in the grid.

All algorithms were executed for 5,000 iterations except for the MTE algorithms that were executed for 1,000 iterations. These numbers were chosen to offer a large sample size to obtain a representative distribution yet small enough to limit total processing time. Each 5,000 iteration run required about one day of dedicated processing time on a modern Windows personal computer while the MTE algorithms required four and seven days for single and multi-gateway configurations, respectively. The MTE algorithms required significantly more time because of the computational complexity in calculating the MTE path for each node, each round, and each iteration using Dijkstra’s algorithm. As we said previously for each node, and each round, Dijkstra’s algorithm calculates the updated path, so that nodes that die mid round are not used by remaining live nodes. This technique can be optimized to update individual node routes to the gateway only after changes in WSN topology (nodes dying). However, we left the algorithm as is to ensure robust accounting for all routes all the time. To minimize required processing time, each iteration was completed when 80 percent of the nodes were dead nodes.

Results for our random variable testing are contained in Table 5. Individual algorithm die out distributions are plotted in Figure 115 through Figure 124. A graphical bar plot of our mean value results of Table 5 is shown in Figure 114. The standard deviation of network die out statistics is given in Table 6. Histograms of the data (blue) along with a Gaussian curve fits for mean and standard deviation values are shown in Figure 115 through Figure 124. The arithmetic mean and standard deviation values in Table 5 and Table 6 are calculated using first and second moment principles of the discrete sampled data set:

$$Mean = \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{15}$$

$$std = \sigma = \left(\frac{1}{n-1}\sum_{i=1}^{n} (x_i - \overline{x})^2 \right)^{1/2} \tag{16}$$

where the Gaussian pdf is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\overline{x})}{2\sigma^2}} . \tag{17}$$

The performance improvement of WSN clustering algorithms with data aggregation (LEACH, Zone, and EZone) and the improvement of WSN lifetime by the addition of an additional gateway compared to MTE and Direct Routing is illustrated in Figure 114. We noted similar results in Chapter V for the one uniform WSN arrangement tested. The energy efficient zone routing algorithm (EZone) maximized the service life when all nodes are alive by rotating the high energy CH role to the node in each zone with the most energy. LEACH provided the most time through 80 percent of network die out because of the random approach of CH election and the instability of the network to maintain a uniform number of CHs during each round after the first node dies (see Figure 105 and Figure 106).

Table 5. Mean value network die out statistics in single and multi-gateway simulations for all algorithms simulated.

Protocol	Metric	Single Gateway	Multi Gateway	% Increase
Direct	Round First Dead	350	660	89
	10% Nodes Dead	395	714	81
	50% Nodes Dead	664	942	42
	80% Nodes Dead	1004	1163	16
	80% Dieout Range (rounds)	654	503	−23
	80% Dieout range/Round First Dead	1.87	0.76	−59
MTE	Round First Dead	12	16	33
	10% Nodes Dead	73	97	33
	50% Nodes Dead	202	289	43
	80% Nodes Dead	351	472	34
	80% Dieout Range (rounds)	339	456	35
	80% Dieout range/Round First Dead	28.25	28.50	1
LEACH	Round First Dead	1840	1844	0
	10% Nodes Dead	1987	1995	0
	50% Nodes Dead	2294	2319	1
	80% Nodes Dead	2523	2565	2
	80% Dieout Range (rounds)	683	721	6
	80% Dieout range/Round First Dead	0.37	0.39	5
Zone	Round First Dead	1566	1841	18
	10% Nodes Dead	1777	1976	11
	50% Nodes Dead	2031	2122	4
	80% Nodes Dead	2151	2210	3
	80% Dieout Range (rounds)	585	369	−37
	80% Dieout range/Round First Dead	0.37	0.20	−46
Ezone	Round First Dead	1936	2070	7
	10% Nodes Dead	1944	2076	7
	50% Nodes Dead	2035	2132	5
	80% Nodes Dead	2083	2157	4
	80% Dieout Range (rounds)	147	87	−41
	80% Dieout range/Round First Dead	0.08	0.04	−45

Table 6. Standard deviation of network die out statistics in single and multi-gateway for all algorithms simulated.

Protocol	Metric	Single Gateway	Multi Gateway	% Increase
Direct	Round First Dead	7	11	57
	10% Nodes Dead	15	15	0
	50% Nodes Dead	45	33	−27
	80% Nodes Dead	56	33	−41
MTE	Round First Dead	3	5	67
	10% Nodes Dead	9	9	0
	50% Nodes Dead	12	20	67
	80% Nodes Dead	19	31	63
LEACH	Round First Dead	53	53	0
	10% Nodes Dead	34	35	3
	50% Nodes Dead	22	23	5
	80% Nodes Dead	29	30	3
Zone	Round First Dead	69	47	−32
	10% Nodes Dead	32	17	−47
	50% Nodes Dead	20	9	−55
	80% Nodes Dead	20	12	−40
EZone	Round First Dead	53	30	−43
	10% Nodes Dead	49	27	−45
	50% Nodes Dead	24	11	−54
	80% Nodes Dead	26	12	−54

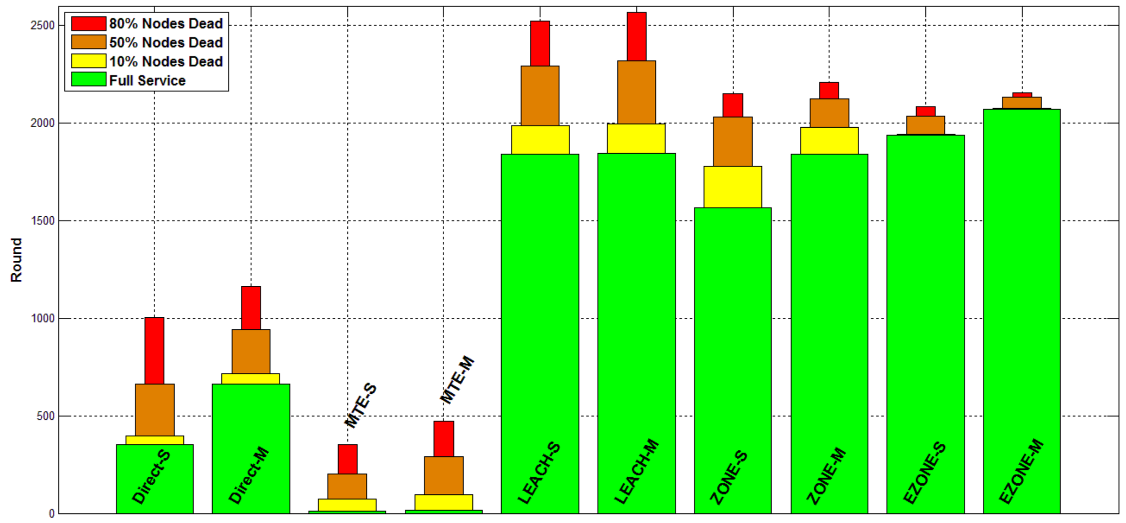


Figure 114. Summary of WSN availability for each algorithm (S~single gateway, M~Multi-gateway).

An interesting result is that there was effectively little to no performance gain when adding an additional gateway to the LEACH algorithm. The single gateway round when the first node died and subsequent metrics (10 percent, 50 percent, 80 percent) were so close to the multi-gateway parameters that the addition of another gateway is insignificant when only considering service life.

Plots for all algorithms examined in this thesis generally demonstrate that our die out distributions follow a normal distribution as illustrated in Figure 115 through Figure 124 except for the round the first node died, which displays a well-defined initial spike. A positive relative skew as compared to the corresponding standard normal for the round first dead and 10 percent of nodes dead is depicted in Figure 115 and Figure 116, and a negative skew for the same parameters is shown in Figure 121 and Figure 122. The significance of the plots aligning with the normal distribution is that WSN die out approximately follows the most common distribution seen in natural phenomena (i.e., the normal distribution [41]).

When considering standard deviation, the addition of another gateway generally decreases the distribution spread except for the MTE algorithm, in which the addition of another gateway increased the spread. The addition of another gateway did little to impact the distributions in LEACH as illustrated Figure 117 and Figure 118. The single and multi-gateways for LEACH displayed similar spread, which continues to follow our claim that an additional gateway did little to improve the characteristics of LEACH (Figure 119 and Figure 120).

Since the die out of our energy efficient zone routing algorithms occurred over a small number of rounds, the histograms have significant overlap (Figure 123 and Figure 124). We also note negative skew of the energy efficient zone routing algorithms, which causes the histograms to shift slightly to the right as compared to their Gaussian overlay. The first node dead and 10 percent node dead rounds see a bit of activity at the tails, which leads to a kurtosis effect in comparison to the standard normal.

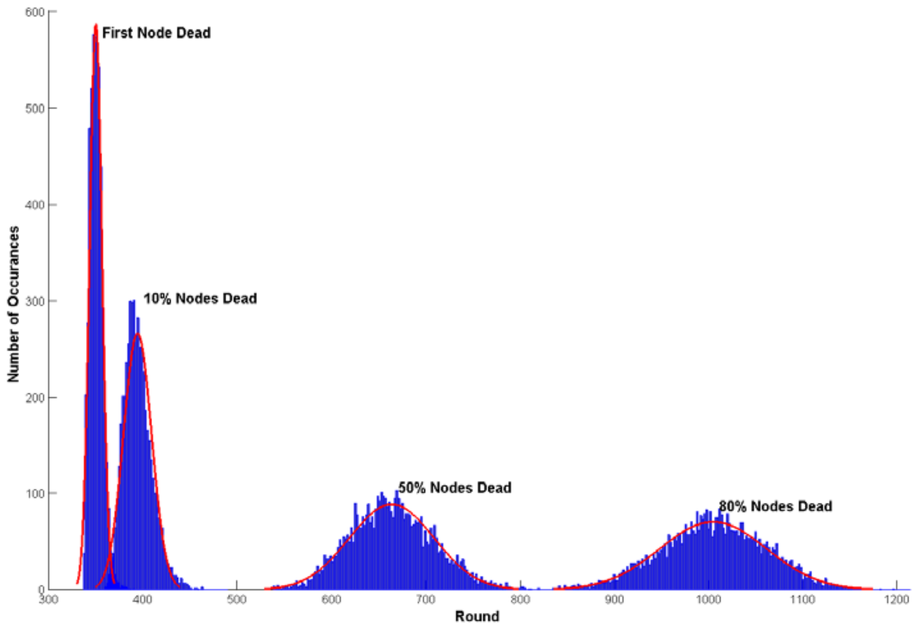


Figure 115. Direct routing in a single gateway WSN. The die out statistics with 5,000 trials are illustrated.

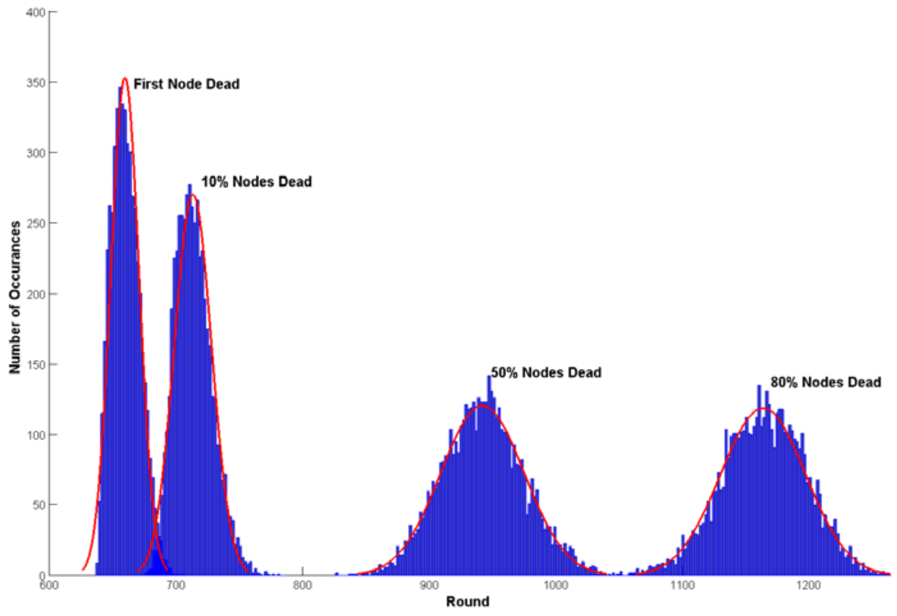


Figure 116. Direct routing in a multi-gateway WSN. The die out statistics with 5,000 trials are illustrated.

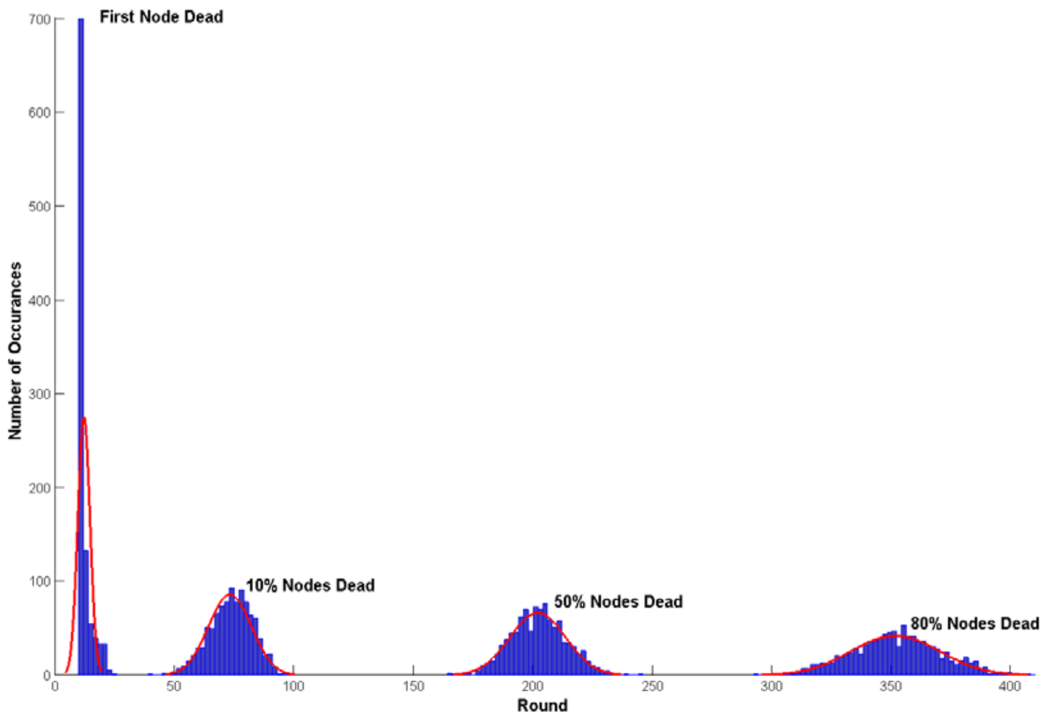


Figure 117. MTE routing in a single gateway WSN. The die out statistics with 1,000 trials are illustrated.

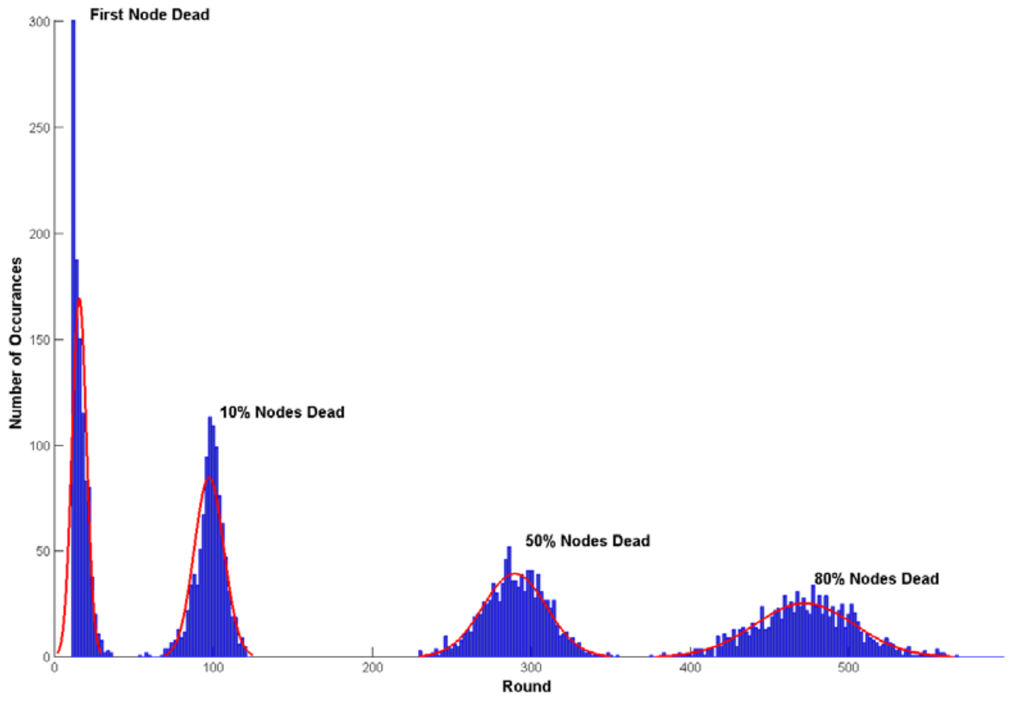


Figure 118. MTE routing in a multi-gateway WSN. The die out statistics with 1,000 trials are illustrated.

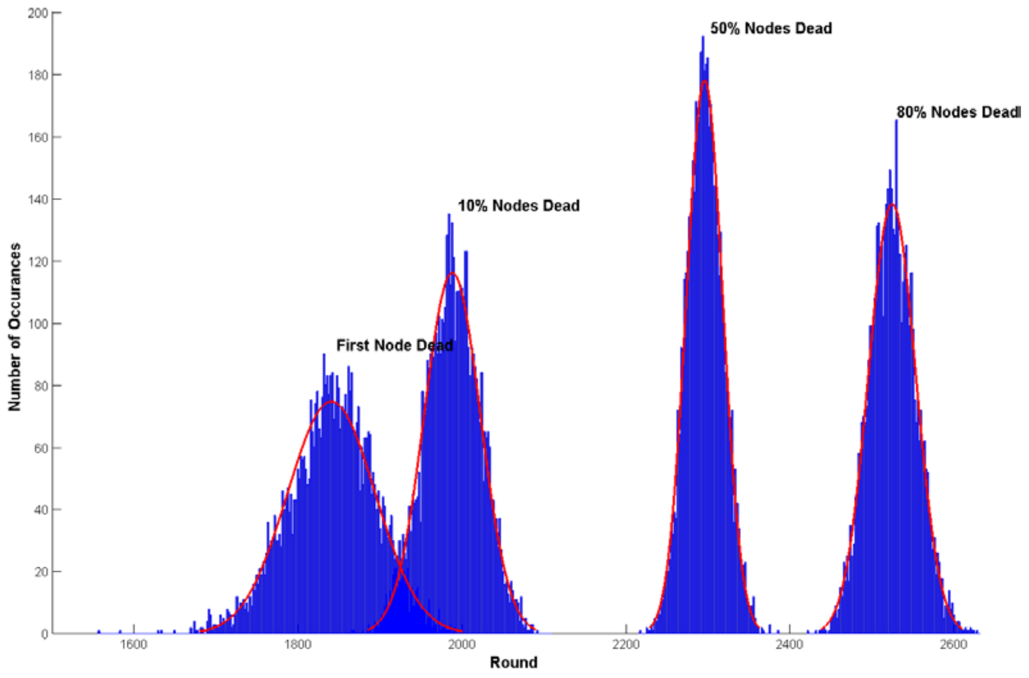


Figure 119. LEACH routing in a single gateway WSN. The die out statistics with 5,000 trials are illustrated.

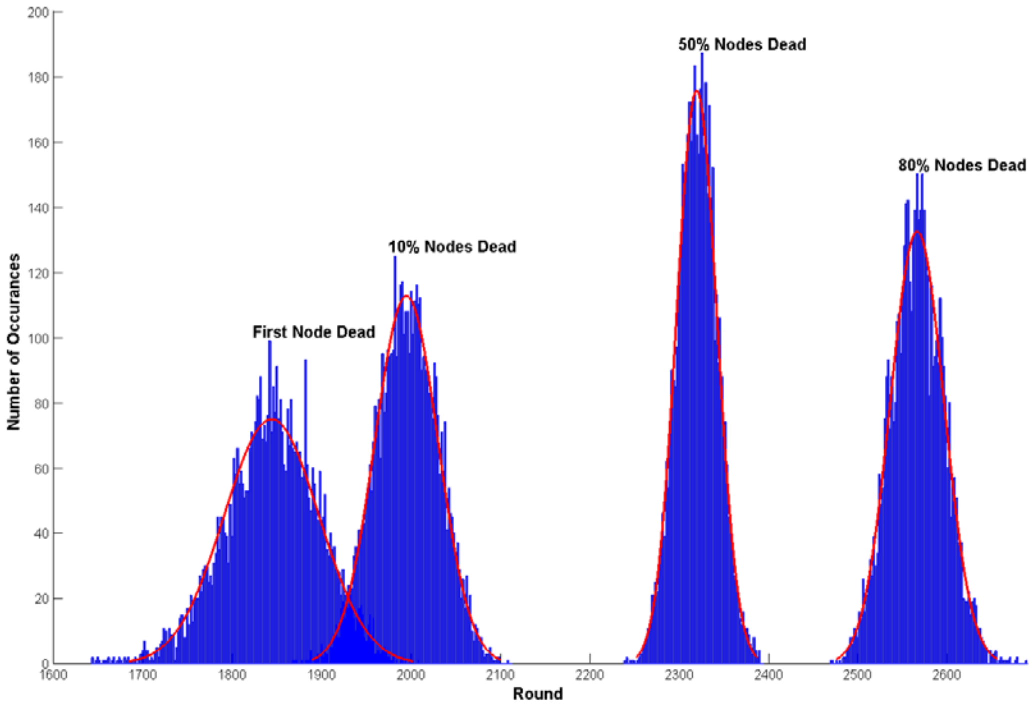


Figure 120. LEACH routing in a multi-gateway WSN. The die out statistics with 5,000 trials are illustrated.

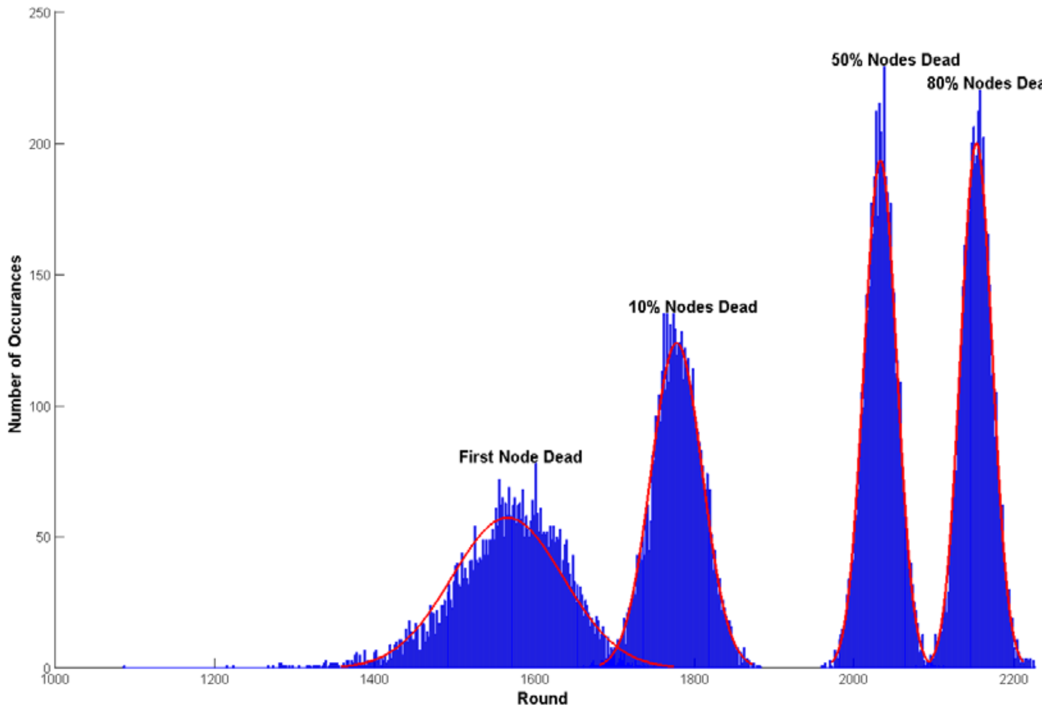


Figure 121. Zone routing with random CH election in a single gateway WSN. The die out statistics with 5,000 trials are illustrated.

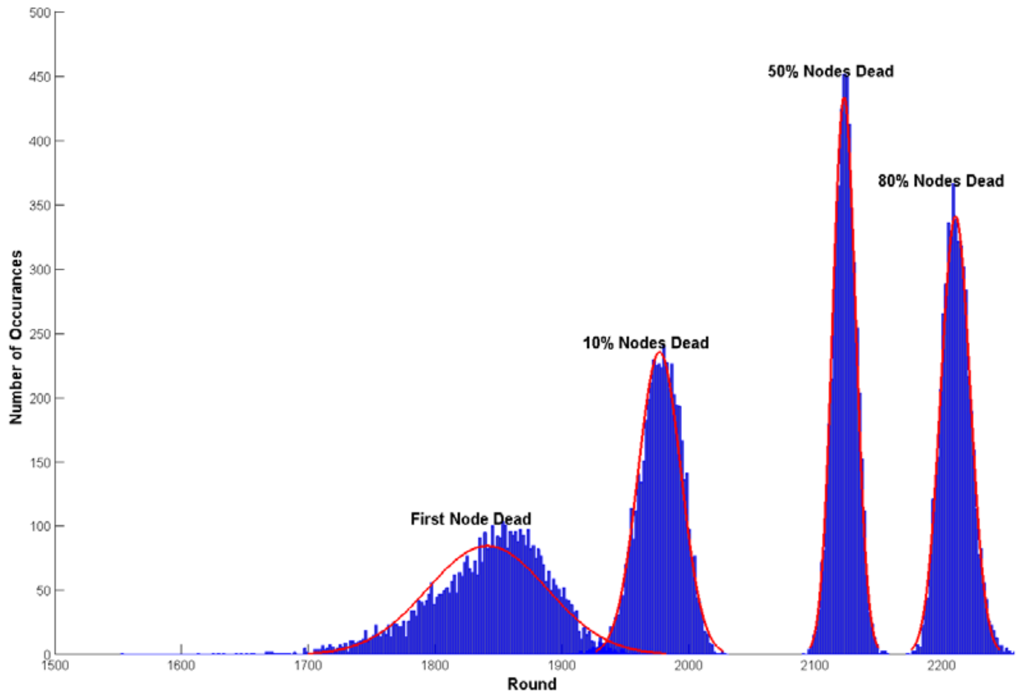


Figure 122. Zone routing with random CH election in a multi-gateway WSN. The die out statistics with 5,000 trials are illustrated.

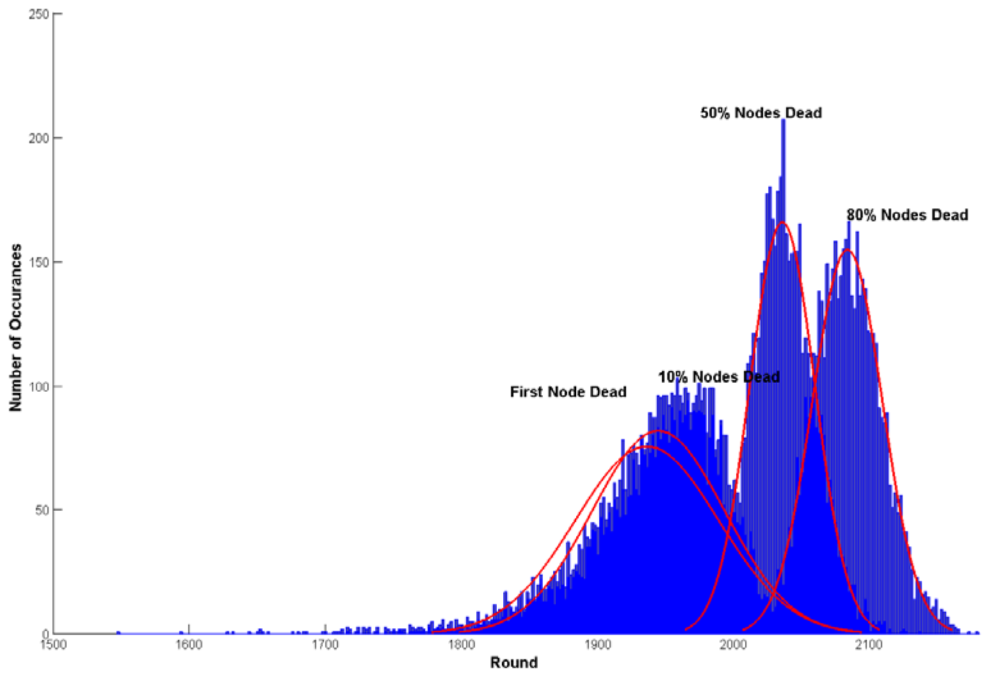


Figure 123. EZone routing in a single gateway WSN. The die out statistics with 5,000 trials are illustrated.

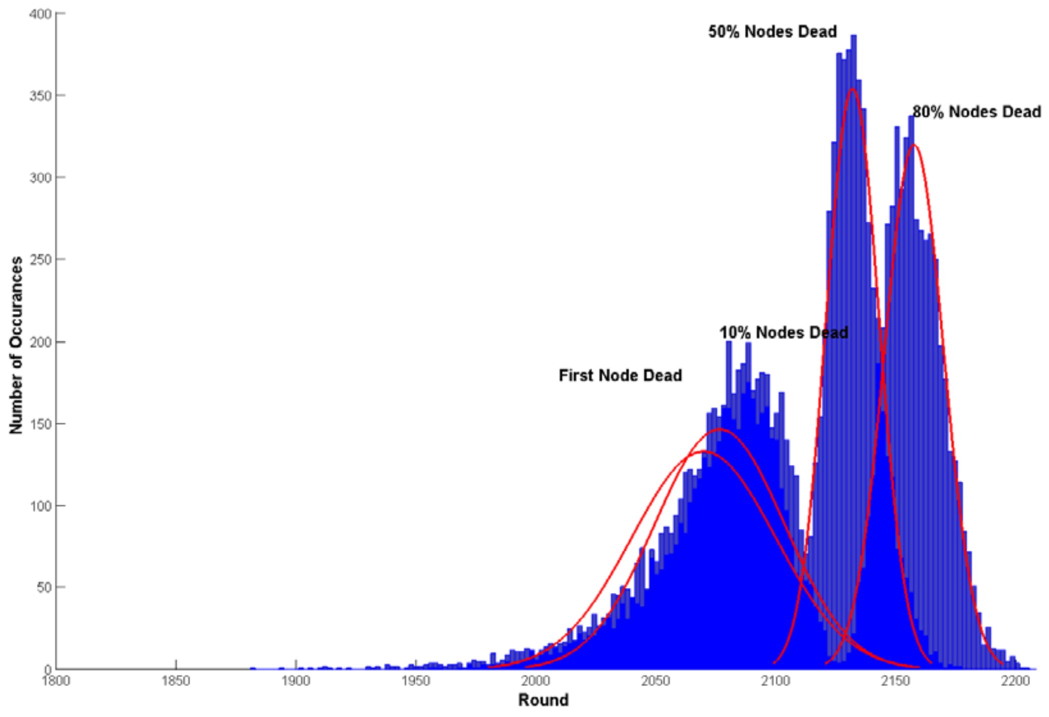


Figure 124. EZone routing in a multi-gateway WSN. The die out statistics with 5,000 trials are illustrated.

B. CHAPTER VI SUMMARY

In this chapter, we described our random variable testing to consider many other network arrangements than what we analyzed in Chapter VI. Our results in Chapter VI were in line with what we presented in Chapter V except we were able to show the impact of an additional gateway on the distribution of our RVs. Our energy efficient zone routing algorithms provides the longest service life with 100 percent coverage, while the LEACH algorithm provides the longest life with at least one node alive. Extending the timeframe of 100 percent service coverage is desired to provide the greatest lasting performance of the entire WSN.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSTIONS AND FUTURE WORK

In this chapter, we summarize the results and contributions of this thesis as well as offer future work opportunities based on our analysis to further improve the networking layer of WSNs.

A. SUMMARY AND CONCLUSIONS

1. Impact of Network Layer Load Balancing

We can achieve significant gains in WSN service lifetime through the use of load balancing in multi-gateway networks. Specifically, implementing an energy efficient cross-layer network routing algorithm can be used to dynamically balance the energy depletion rates of nodes and, subsequently, minimize the overall energy depletion rate of the network resulting in full WSN service coverage.

Network layer load balancing can be used to cause the network to die out in a tactically oriented fashion. From our first algorithm, the direct to gateway routing protocol, and the last algorithm (our energy efficient zone routing protocol-EZone), we covered the variety of ways a network can die out. This information allows a designer to implement or further research the specific type of network algorithm required for a WSN application and, thereby, obtain a desired WSN depletion rate and die out topology.

2. Opportunities Offered by Clustering Algorithms

Clustering algorithms offer additional control of the physical layer through the performance of the networking layer. Allowing the network layer to determine the optimal CHs at any given time achieves a balance of which nodes should realize a high probability of being the CH (nodes in a high power role) and which nodes have a lower probability of being a CH (nodes desired to be in a low power role).

Clustering algorithms allow the physical independence of nodes to be optimally realized; nodes that are not connected wirelessly can make decisions on which of their peers they should communicate with at any given time. Clustering essentially makes this decision for the nodes based on what is best for the network since every node is forced to send their payload to their assigned CH. This wireless independence allows the algorithm to force nodes into certain aggregations which optimize routes chosen in the best interest of the network instead of nodes being personally greedy as the Direct and MTE algorithms allowed.

The largest takeaway of clustering algorithms is how the CHs are determined. The mechanism for determining the CH affects the distribution of how and when nodes die out. If a particular node is chosen to be a CH more than other nodes in a similar area, it will likely die out earlier. This can be an advantage or a disadvantage in that we have some level of control over which nodes die out earlier or later. The advantage is that we can take control of this aspect and prevent high value (critical collection) nodes from being CHs and extend their life to the end of the network. For example, we can employ a clustering algorithm and hierarchically assign nodes a probability of becoming a CH commensurate with the priority of their collected information. If their information is of high priority, their probability to become a CH should be low or zero. However this could be a disadvantage in that selecting a node to be excluded from the CH role requires other nodes to perform the CH role more frequently, which causes them to die out sooner. Conversely, if we want to extend the full service life of the network, CHs should be chosen with energy efficiency in mind so that the energy of all nodes is depleted uniformly, causing them to die out at a similar time. The LEACH algorithm did this in a random fashion, preserving some nodes longer than others.

Clustering algorithms offer an efficient mechanism to perform data aggregation. By doing so, the energy required to transmit a compressed message is less than if packets are not aggregated. This allows a great deal of information to be transmitted a short distance while requiring lower transmission energy to be expended.

3. Performance Gain of Additional Gateway

The inclusion of an additional gateway extended WSN service life in every network layer algorithm except LEACH and offered improved coverage during die out as compared to the single gateway scenario. The use of an additional gateway caused die out to occur in a way that preserved area coverage even while nearby nodes died. As energy efficiency within the algorithm improved, the impact of the additional gateway was lowered. For example, the direct-to-gateway algorithm realized an 83 percent increase in the time when all nodes were alive compared to just a 13 percent and 6 percent gain for zone routing algorithms, respectively (zone and EZone algorithms).

While the additional gateway extended full service node availability (except for LEACH), as nodes began to die out, the additional gateway mitigated gaps in service area coverage as a result of longer wireless propagation distances. This aspect resulted in a more preferred network topology during die out compared to the single gateway cases.

B. TACTICAL NETWORK PROTOCOL RECOMMENDATION

Our zone clustering with energy efficient CH election algorithm (EZone) offers the best opportunity to extend WSN service life while maintaining tactical control of the network layer in both single and multi-gateway configurations. It produced the least variance in energy distribution at any round and smartly balanced cluster and node loading since our zones were implemented based on knowledge of physical layer topology and anticipated application layer loading. This algorithm also demonstrated the advantages of a cross-layer approach by allowing the network layer to make routing decisions based on node battery levels contained at the physical layer.

C. CONTRIBUTIONS OF THIS THESIS

The contributions of this thesis are as follows:

- We surveyed and identified load balancing techniques for WSNs.
- We simulated traditional networking algorithms and identified performance improvements from adding an additional gateway.
- We developed an energy efficient WSN networking algorithm and identified the performance improvements compared to algorithms that do not consider energy efficiency.
- As sensor-node battery levels are depleted and nodes subsequently die out, we showed how the networking algorithm in operation affects the spatial distribution of live nodes and dead nodes in the sensor field and how this affects the continuous service coverage throughout the sensor field.
- We illustrated detailed energy statistics for specific node-gateway(s) arrangement(s) and modeled network die out statistics as random variables

to better characterize the distribution of the algorithms' results over thousands of trials. This technique allowed us to better substantiate the performance of classic network algorithms and our novel energy efficient algorithm.

D. FUTURE WORK

1. Further Optimize the Cluster Approach

Our research of clustering techniques only considered non overlapping clusters; each node could only send packets to the gateway through one CH. Clusters could be allowed to overlap, thereby allowing nodes the decision authority to select which CH they transmit their payload to and when.

This requires the development of a clustering algorithm that takes information from the application layer as well as the physical layer to create energy efficient and application layer efficient clusters in the network. Application layer loading should be altered in a predictable fashion at which point application layer and physical layer load balancing could be incorporated into the network layer to produce clusters that are dynamically efficient at every round or during some cycle of rounds. This technique would better simulate the reality of collection (i.e., in which information is collected non-uniformly throughout the sensor field).

2. Devise and Employ MTE Data Aggregation Strategies to Minimize Hot Node Energy Consumption

Our simulations of the MTE algorithm allowed a hot-node scenario to ensue. Various techniques relating to energy efficiency and our MTE algorithm could be employed such as: 1) allow the hot-node to aggregate data to minimize the final long-haul transmission to the gateway, and 2) extend the link cost metric that was used in this thesis to include other metrics. Other metrics could include transmission distance and other parameters such as hot-node energy level or the number of packets that have gone through it in attempt to balance hot-node energy depletion.

3. Dynamic Zoning Based on Anticipated Sensor Loading

We mentioned that our identification of zones for our zone clustering algorithms was a result of sensor arrangement in the field and anticipated CBR loading at each round to effectively balance total zone throughput at each round. This technique could be employed dynamically such that the network topology is re-zoned periodically as sensor traffic load changes and node die out begins. Periodically re-zoning based on anticipated WSN load changes may offer a technique to balance energy loading throughout the network service life and avoid unnecessary transient network energy reductions.

4. Extend Sensor field Dimensions Beyond Individual Node Communication Range

We made extensive use of the assumption that any node in the WSN was within communication range of the gateway, which allowed for simulation of our direct-to-gateway algorithm, by-passing the LEACH clustering in later rounds if no CHs were chosen, and any node to be employed as a CH any time during clustering simulations. While modern day wireless techniques can communicate efficiently up to several miles, a WSN may be required to communicate at distances greater than any node's communication range. There are several possible options that could be tested to extend our concepts simulated in this thesis in order to allow a larger sensor field than a node's communication range. Specifically, further zoning could be utilized to hierarchically divide a grid space to guarantee communication ranges between zones. This technique would require CHs to communicate with adjacent CHs in a hop-by-hop basis in a path to the gateway. This scenario should also be tested in a multi-gateway configuration to keep the number of CH hops to a minimum.

5. Implement LEACH and EZone in Robust Advanced Simulation Software

Our research primarily utilized MATLAB for simulation. There exist several other WSN simulation software platforms that could further investigate the algorithms in this thesis. We briefly describe a few of these advanced platforms in Appendix B. We propose that our Zone routing algorithms be programmed and explored in these utilities as they may offer more specific and accurate WSN solution results.

6. Implement an Energy Efficient Message Structure

We described a message structure for LEACH that was adopted from [1] that is applicable to any of our clustering algorithms. We did not simulate the impact of the message algorithm, which is another WSN load balancing opportunity.

7. Impact of Varying the Link Cost Parameter for WSNs as Future Work

The distance squared between nodes was utilized as the link cost parameter since the distance is proportional to the energy required at the physical layer for transmission between nodes in our sensor field. Devising a different link cost strategy may offer an opportunity to tailor network traffic to a specific condition, which could offer a benefit to certain designs.

E. FINAL THOUGHTS

As WSNs become more prevalent in society, an understanding of how each layer affects performance is required so that the WSN can be most efficiently tailored to its application. An energy efficient routing strategy offers quantifiable gains to the service life of tactical WSNs. It balances the use of individual battery levels at the node level to maximize the time when all nodes are fully capable. This routing strategy also magnifies the use of a clustering algorithm to balance wireless transmission range coupled with data aggregation to reduce energy demand during transmit operations. Our techniques in this thesis show the importance of load balancing in WSNs and that design creativity at the network layer can have significant impacts on achieving lasting capability of WSN performance. Specifically, implementing energy efficient load balancing techniques at the network layer offers a tactical advantage allowing the DoD to extend network performance and autonomously control the die out topology as the WSN degrades from 100 percent service coverage, improving their effectiveness and suitability in the battlespace.

APPENDIX A. AUTHOR BIOGRAPHY

LT Kevin A. White

United States Navy

Engineering Duty Officer

Lieutenant (LT) Kevin A. White was born in Los Angeles, California (CA) and graduated from Arcadia High School in 1999. He immediately began college at California State Polytechnic University Pomona and earned a Bachelor's Degree in Mechanical Engineering in June 2004, graduating Suma Cum Laude and Mechanical Engineering Valedictorian. Kevin joined the Navy (submarine force) in January 2002 through the Navy's Nuclear Power Officer Candidate Program.

Following college graduation in June 2004, LT White commenced his naval training and officer indoctrination at Officer Candidate School in Pensacola Florida. LT White was commissioned in the Navy in October 2004 and completed Basic Submarine School at Groton Connecticut (January 2005), Nuclear Power School at Charleston, South Carolina (August 2005), and Nuclear Prototype School at Ballston Spa, New York (March 2006).

Following nuclear training, LT White reported to the fast attack submarine *USS Columbia* (SSN 771) at Pearl Harbor, Hawaii in April 2006. LT White participated in various military exercises including Rim of the Pacific 2006, a surge deployment to the Western Pacific in late 2006 and then a 16-month shipyard modernization period. LT White qualified nuclear engineer in June 2008 and participated in Columbia's post availability sea-trials and an Eastern Pacific deployment.

LT White left *USS Columbia* in January 2009 and reported to SPAWAR PEO C4I in San Diego, CA as the submarine integration and installation manager (within PMW-770). At PEO C4I, LT White coordinated and managed submarine communication and network modernization on all submarines in the U.S. Submarine Force. He holds acquisition certifications in program management (Level 2), production quality and manufacturing (Level 1) and systems engineering (SPRDE-SE Level 1) and Lean Six

Sigma LT White lateral transferred to the engineering duty officer community via the June 2010 lateral transfer. LT White completed EDO Basic School in May 2011 earning the Founders Award and conditionally qualified engineering duty officer in August 2011.

LT White left SPAWAR in December 2011 and reported to the Naval Postgraduate School (NPS) in Monterey CA to complete a Master's Degree in Electrical Engineering with emphasis in networking and cyber. While at NPS, LT White also completed the Navy's Joint Professional Military Education curriculum. LT White was selected for promotion to lieutenant commander in July 2013 and will graduate NPS in December 2013.

Following NPS graduation, LT White will report to SPAWAR Bahrain as the officer-in-charge in April 2014.

Kevin married the former Lisa Costanza of Glendora, CA in March 2006 and has a son Tyler (March 2008) and a daughter Rylie (May 2010).

APPENDIX B. MATLAB CODE

The full version of MATLAB code to generate the results reported in Chapters V and VI, which was described by pseudocode in Chapter III and IV is provided in Appendix B. While the code was useful to compare all network algorithms simulated in this research, the structure of each of the algorithms could be easily manipulated to simulate other networking algorithms. The background research performed in choosing a WSN simulation platform and a few manipulations that anyone in the future could consider to reuse this code for their own research are briefly summarized.

1. Simulation Platforms for WSNs

There are many options for one to perform simulations of a WSN. A WSN and associated layers can be modeled using a bottom up approach with mainstream programming languages directly, such as C, C++, Java, etc., which requires extensive expertise in the programming language as well as the protocol, which is desired to be modeled, or it can be modeled from a top-down approach using network specific simulation software such as Qualnet, NS2, or many others. The point is that there are a variety of options to choose from and each has its specific advantages and disadvantages. There are several evaluations in the literature surveying the various pros and cons of each WSN simulation package such as [42]–[44].

Network specific simulation software contains an industry standard buffet of protocol options available for simulation at each layer. Layer specific protocols are based on request for comment (RFC) documentation that has been transformed into a programming language codes for direct use in simulation. The user can select the protocols they desire to simulate at each layer, at which point the software creates a simulation scenario by merging the existing protocols into a runtime program. For example, protocols in NS2 and Qualnet are created using the C++ language. They are lengthy and are the product of many years of development; however, they provide a comprehensive representation of industry standard protocols in use today. The key word here is “in use” as research protocols are not provided with the software and the knowledge base for these paid utilities are small.

We started our early research experimenting with Qualnet. While its graphical user interface was efficient at creating WSN simulations, we quickly realized an intermediate C++ programming ability was insufficient to generate and integrate a custom protocol into Qualnet in the time available and with available manpower. Qualnet does offer tailored contract programming help; however, that was neither in our budget nor our timeframe. As a result, we elected to utilize a MATLAB environment.

A key advantage with MATLAB are the simple built in functions and the ease with which any function in MATLAB can be quickly researched and incorporated into an algorithm. Also, since the user base for MATLAB is so big, there are a huge number of support forums that address literally any problem that has been encountered before. MATLAB is readily available at the Naval Postgraduate School, and its use in the majority of courses during a graduate study guarantees a student’s ability to jump into MATLAB for their thesis work.

2. MATLAB Programming Strategies

Our strategy for WSN MATLAB programming was to think of the WSN layering concept and incorporate each layer necessary with associated assumptions. Thinking of the MAC layer as a TDMA strategy where each node is allocated time in each time division offers a simple loop using rounds to simulate this layer. We can then simulate the application layer of each node generating an *L*-bit message and pushing it to the network layer. The network layer then identifies the route or creates the desired network topology for the round to pass information to the gateway and then use the physical layer to transmit and account for required information. After handling the nuances of decrementing energy to transmit in direct or multi-path propagation, energy to receive, and energy for a CH to aggregate data, the simulations take proper form quickly.

Data presentation obviously is a key aspect of any research. Since our research focused on energy efficiency and extending network lifetime using load balancing techniques in single and multi-gateway scenarios, it made sense to track energy levels, the distribution of energy, and alive and dead nodes each round. Thus, our subplot graphics capture these priorities, and we obtain the energy variance each round as a statistical measure of energy spread, the total WSN energy to obtain energy depletion of the network when all nodes are alive as an indication of network efficiency. Our interest also was the topology of the network as nodes die out, which caused us to display the network topology when the first node died, and at 10 percent, 50 percent, and 80 percent total nodes dead to obtain observations of service area coverage as the network depleted.

3. Comments on our MATLAB Code

While our research considered both single and multi-gateway, we only provide in total the multi-gateway code in the following sections. Single gateway scenarios can be easily generated from the code by removing the second gateway and changing logic that incorporated transmitting information to the closest gateway. All our simulations in Chapter VI used a common sensor field. To achieve this effect, we first generated the uniform sensor filed and our required parameters and passed them into each algorithm. This required the first several lines of code to be suppressed in the following algorithms, indicated by our statement to "SUPPRESS ABOVE IF STARTING WITH COMON WSN PARAMETERS." Maintaining commonality of variables across all algorithms was important to ensure the inputs were accepted by each algorithm.

All data that is generated in our algorithms is meticulously saved in multiple formats. This was to allow access to complete simulation data for post processing as needed. Every figure was sized and set in a specific, yet similar way to achieve scale commonality to alleviate rescaling images as they were copied into this paper. Also each algorithm creates a simulation movie file that captures every fifth frame of our main graphical figure to allow simulations to be viewed after they completed. This was a nice technique to go back and review the simulations to form observations. Since the simulations could not be imbedded in this paper copy, our observations are also supported using the plots at round first dead, 10 percent, 50 percent, and 80 percent total nodes dead.

Our simulations in Chapter VI executed each algorithm many times. Here we allowed the network to be regenerated each time. This required each of the algorithms to be contained in another for loop to execute the code the desired number of iterations. To speed up the simulations, we suppressed majority of our graphics so that the output of each iteration appended the metrics of interest. Our metrics of interest were network die out statistics of round first dead, and the round where 10 percent, 50 percent, and 80 percent of nodes are dead. Once we obtained these arrays of 5,000 values for each parameter, the data was post processed to generate the figures and plots in Chapter VII.

Several of our algorithms involved “structure” programming techniques. Variables that have a period (i.e., S(i).E) were used to constrain multiple variables under a common notation. This technique allowed us to incorporate “meta-data” into our algorithms that were tracked at specific times and used in the network and physical layer. While there are other ways to do this, the use of structures help keep accounting of data under better control.

4. Direct to Multi-gateway

```
%Direct to Multi-gateway simulation

clc;
clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Field Dimensions - x and y maximum (in meters)
xm=50;
ym=50;

%x and y Coordinates of Sink1
sink.x1 = 25;
sink.y1 = -100;
%x and y Coordinates of Sink 2
sink.x2 = 25;
sink.y2 = 150;

%Packet size in bits
L = 2000;

%Number of Nodes in the field
```

```
n=100;

%Energy Model (all values in Joules)
%Initial Energy
Eo=0.5;
%Eelec=Etx=Erx
ETX=50*0.000000001;
ERX=50*0.000000001;
%Transmit Amplifier types
Efs=10*0.000000000001;
Emp=0.0013*0.000000000001;
%Data Aggregation Energy
EDA=5*0.000000001;

%maximum number of rounds
rmax= 2000;

%%%%%%%%%%%%%%%%%%%%%%         END OF PARAMETERS      %%%%%%%%%%%%%%%%%%%%%
%%%%%%   **SUPPRESS ABOVE IF STARTING WITH COMMON WSN PARAMETERS**   %%%%%%

%Computation of do
do=sqrt(Efs/Emp);

%Get the screensize so each figure can be normalized in a similar manner
%for thesis writeup
scrsz = get(0,'ScreenSize');

%Creation of the random Sensor Network
fig = figure(1);
set(fig, 'Position',[1 scrsz(4)*.25 scrsz(3)*.7 scrsz(4)*.75]);

subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
hold on;

for i=1:1:n;
        S(i).xd = SensorX(i);
%     S(i).xd = rand(1,1)*xm;
    XR(i)=S(i).xd; %copy of the x-coordinate outside of the S struct
        S(i).yd = SensorY(i);
%     S(i).yd = rand(1,1)*ym;
    YR(i)=S(i).yd; %copy of the y-coordinate outside of the S struct
    S(i).E=Eo;
    plot(S(i).xd,S(i).yd,'o');
end

xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');

%plot the sink
S(n+1).xd1=sink.x1;
S(n+1).yd1=sink.y1;
S(n+2).xd2=sink.x2;
```

```
S(n+2).yd2=sink.y2;
plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
    'g','MarkerSize',10)
plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
    'g','MarkerSize',10)
axis([-5 xm+5 sink.y1-5 sink.y2+5]) % Set axis of the plot

%plot horizontal boundaries of the sensor field
bottomY=[0,0];
bottomX=[0,xm];
topY = [ym,ym];
topX = [0,xm];

%plot vertical extremes of the sensor field
vertLeftX=[0,0];
vertLeftY=[0,ym];
vertRightX=[xm,xm];
vertRightY=[0,ym];
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
axis([-5 xm+5 sink.y1-5 sink.y2+5]);
set(perimeter,'Color','r','LineWidth',1);
hold off;

%start the plot for the node energy bar graph
energyBar = zeros(1, n);
for iii = 1:n;
    energyBar(iii) = S(iii).E;
end

subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
bar(energyBar);
ylim([0 0.51]);
xlim([-10 110]);
xlabel('Node Number','FontSize',12,'FontWeight','bold');
ylabel('Energy (J)','FontSize',12,'FontWeight','bold');

%Start the plot for 3D Energy Stem
subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
stem3(XR, YR, energyBar, 'Fill', 'g');
hold on
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
set(perimeter,'Color','r','LineWidth',1);
axis([-1 xm+1 -1 ym+1 0 Eo]);
xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
zlabel('Energy (J)','FontSize',12,'FontWeight','bold');
set(get(gca,'xlabel'),'rotation',14);
set(get(gca,'ylabel'),'rotation',338);
grid off;
hold off
```

```
drawnow;

deadNode = zeros(1,n);
energyBar =zeros(1,n);
DEAD_DIRECT_M = zeros(1, rmax);
ENERGY_DIRECT_M = zeros(1, rmax);
ENERGY_VARIANCE_DIRECT_M = zeros(1, rmax);
flag_first_dead = 0;
flag_10P_dead = 0;
flag_50P_dead = 0;
flag_80P_dead = 0;
roundString = '1';

myObj=VideoWriter('MOVIE_DIEOUT_DIRECT_M.avi');
open(myObj);

for round = 1:rmax;
    round

    for aa=1:n;
        %checking if there is a dead node
        if (S(aa).E <= 0);
            deadNode(aa) = 1;
        end
    end

    [notUsed, AliveNodeNo] = find(deadNode == 0);

    for bb = AliveNodeNo;
        %Calculate the distance from each node to the base station
        distance = min( sqrt( (S(bb).xd-(S(n+1).xd1) )^2 + ...
            (S(bb).yd-(S(n+1).yd1) )^2 ), sqrt((S(bb).xd-(S(n+2).xd2) )^2 + ...
            (S(bb).yd-(S(n+2).yd2) )^2 ) ); %(S(n+1) is to the basestation)

        %energy cost for the clusterhead in the zone to aggregate and
        %transmit the message to the basestation
        if (distance > do);
            S(bb).E = S(bb).E - ( (ETX)*(L) + Emp*L*( distance^4 ));
        end
        if (distance <= do);
            S(bb).E = S(bb).E - ( (ETX)*(L)  + Efs*L*( distance^2 ));
        end
    end

    %obtain total system energy at the conclusion of each round and plot
    for cc=1:n;
        energyBar(cc) = S(cc).E;
    end

    %Every round, obtain the energy variance
    ENERGY_VARIANCE_DIRECT_M(round) = var(energyBar);
    ENERGY_DIRECT_M(round)=sum(energyBar); %for Statistics
    DEAD_DIRECT_M(round)=sum(deadNode); %for Statistics

    % Below is used for figure plotting later
    AliveNodeNo;
    [notUsed2, DeadNodeNo] = find(deadNode==1);
    DeadNodeNo;

    AliveX = [];
    AliveY = [];
    DeaDX = [];
    DeaDY = [];
    EnergyAlive = [];
    trackMeAlive = 1;
    trackMeDead = 1;
    for yyy = AliveNodeNo
        AliveX(trackMeAlive) = XR(yyy);
        AliveY(trackMeAlive) =YR(yyy);
        EnergyAlive(trackMeAlive) = S(yyy).E;
        trackMeAlive = trackMeAlive +1;
    end
    for zzz = DeadNodeNo;
        DeaDX(trackMeDead) = XR(zzz);
        DeaDY(trackMeDead) = YR(zzz);
        EnergyDead(trackMeDead) = 0;
        trackMeDead = trackMeDead+1;
    end

    %plot a running bar chart of energy for animation
    figure(1);
    subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
    bar(energyBar);
    ylim([0 0.51]);
    xlim([-10 110]);
    xlabel('Node Number','FontSize',12,'FontWeight','bold');
    ylabel('Energy (J)','FontSize',12,'FontWeight','bold');

    %3D Stem Energy Plot
    subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
    stem3(AliveX, AliveY, EnergyAlive, 'Fill', 'g', 'LineStyle','--');
    hold on;
    if trackMeDead > 1;
        stem3(DeaDX, DeaDY, EnergyDead, 'Fill', 'r', 'LineStyle','--');
    end
    perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
        vertRightX, vertRightY);
    set(perimeter,'Color','r','LineWidth',1);
    xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    zlabel('Energy (J)','FontSize',12,'FontWeight','bold');
    set(get(gca,'xlabel'),'rotation',14);
    set(get(gca,'ylabel'),'rotation',338);
    axis([-1 xm+1 -1 ym+1 0 Eo]);
    grid off;
    hold off;

    deadholder = sum(deadNode);

    % Save the round to a string for display on plots
    roundString = num2str(round);

    subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
    plot(AliveX, AliveY, 'o');
    hold on;
    text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
    text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
    plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
    plot(S(n+1).xd1,S(n+1).yd1,'o', 'MarkerFaceColor','g', ...
        'MarkerSize', 10);
    plot(S(n+2).xd2,S(n+2).yd2,'o', 'MarkerFaceColor','g',...
        'MarkerSize', 10);
    perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
        vertRightX, vertRightY);
    axis([-5 xm+5 sink.y1-5 sink.y2+5]);
    set(perimeter,'Color','r','LineWidth',1);
    xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
    ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
    hold off;
    drawnow

    %turn every 10th frame into a movie
    if mod(round,5)==0;
        MOVIE_DIEOUT_DIRECT_M = getframe(figure(1));
        writeVideo(myObj,MOVIE_DIEOUT_DIRECT_M);
    end

    %find round first node dead plot and save network figures
    if(flag_first_dead == 0);
        if (deadholder >= 1);
            flag_first_dead = 1;
            ROUND_FIRST_DEAD_DIRECT_M = round;
            fig1dead = figure(8);
            set(fig1dead, 'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            plot(AliveX, AliveY, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(S(n+1).xd1,S(n+1).yd1,'o', 'MarkerFaceColor','g',...
                'MarkerSize', 10);
            plot(S(n+2).xd2,S(n+2).yd2,'o', 'MarkerFaceColor','g',...
                'MarkerSize', 10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            drawnow;
            hold off;
            saveas(fig1dead,'1_Node_Dead_grid_DIRECT_M.bmp');
            saveas(fig1dead,'1_Node_Dead_grid_DIRECT_M');
            saveas(fig1dead,'1_Node_Dead_Grid_Energy_DIRECT_M');
            saveas(fig1dead,'1_Node_Dead_Grid_Energy_DIRECT_M.bmp');
        end
    end

    %find the round when 10% of nodes are dead and save network figures
    if(flag_10P_dead == 0);
        if (deadholder/n >= 0.1);
            flag_10P_dead = 1;
            ROUND_10P_DEAD_DIRECT_M = round;
            fig10Pdead = figure(9);
            set(fig10Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            plot(AliveX, AliveY, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
            set(gcf,'Units','normal')
            set(gca,'Position',[.06 .06 .9 .9])
            plot(S(n+1).xd1,S(n+1).yd1,'o', 'MarkerFaceColor','g',...
                'MarkerSize', 10);
            plot(S(n+2).xd2,S(n+2).yd2,'o', 'MarkerFaceColor','g',...
                'MarkerSize', 10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            drawnow;
            hold off;
            saveas(fig10Pdead,'10P_Node_Dead_grid_DIRECT_M.bmp');
            saveas(fig10Pdead,'10P_Node_Dead_grid_DIRECT_M');
            saveas(fig10Pdead,'10P_Node_Dead_Grid_Energy_DIRECT_M');
            saveas(fig10Pdead,'10P_Node_Dead_Grid_Energy_DIRECT_M.bmp');
        end
    end

    %find round when 50% of nodes are dead
    if(flag_50P_dead == 0);
        if (deadholder/n >= 0.5);
            flag_50P_dead = 1;
            ROUND_50P_DEAD_DIRECT_M = round;
            fig50Pdead = figure(10);
            set(fig50Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            plot(AliveX, AliveY, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
            set(gcf,'Units','normal')
            set(gca,'Position',[.06 .06 .9 .9])
            plot(S(n+1).xd1,S(n+1).yd1,'o', 'MarkerFaceColor','g',...
                'MarkerSize', 10);
            plot(S(n+2).xd2,S(n+2).yd2,'o', 'MarkerFaceColor','g',...
                'MarkerSize', 10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5])
            set(perimeter,'Color','r','LineWidth',1);
            xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            drawnow;
            hold off;
            saveas(fig50Pdead,'50P_Node_Dead_grid_DIRECT_M.bmp');
            saveas(fig50Pdead,'50P_Node_Dead_grid_DIRECT_M');
            saveas(fig50Pdead,'50P_Node_Dead_Grid_Energy_DIRECT_M');
            saveas(fig50Pdead,'50P_Node_Dead_Grid_Energy_DIRECT_M.bmp');
        end
    end

    %find round when 50% of nodes are dead and save network figure
    if(flag_80P_dead == 0);
        if (deadholder/n >= 0.8);
            flag_80P_dead = 1;
            ROUND_80P_DEAD_DIRECT_M = round;
            fig80Pdead = figure(11);
            set(fig80Pdead, 'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            plot(AliveX, AliveY, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(S(n+1).xd1,S(n+1).yd1,'o', 'MarkerFaceColor','g',...
                'MarkerSize', 10);
            plot(S(n+2).xd2,S(n+2).yd2,'o', 'MarkerFaceColor','g',...
                'MarkerSize', 10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            drawnow;
            hold off;
            saveas(fig80Pdead,'80P_Node_Dead_grid_DIRECT_M.bmp');
            saveas(fig80Pdead,'80P_Node_Dead_grid_DIRECT_M');
            saveas(fig80Pdead,'80P_Node_Dead_Grid_Energy_DIRECT_M');
            saveas(fig80Pdead,'80P_Node_Dead_Grid_Energy_DIRECT_M.bmp');
        end
    end

    if deadholder == n;
        break;
    end

end

close(myObj);

ALIVE_DIRECT_M = zeros(1,round);
for i = 1:round;
    ALIVE_DIRECT_M(i) = n - DEAD_DIRECT_M(i);
end

RoundDeadStats= [ROUND_FIRST_DEAD_DIRECT_M ROUND_10P_DEAD_DIRECT_M ...
    ROUND_50P_DEAD_DIRECT_M ROUND_80P_DEAD_DIRECT_M];

fig2 = figure(2);
set(fig2,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ALIVE_DIRECT_M(1:round), 'LineWidth', 2);
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Number of Nodes Alive','FontSize',12,'FontWeight','bold');
ylim([0 n+1]);
saveas(figure(2), 'NodesAliveVsRound_DIRECT_M');
saveas(figure(2), 'NodesAliveVsRound_DIRECT_M.bmp');

fig3 = figure(3);
set(fig3,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_DIRECT_M(1:round) , 'LineWidth', 2);
hold on;
plot(RoundDeadStats(1), ENERGY_DIRECT_M(RoundDeadStats(1)),...
    'p', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), ENERGY_DIRECT_M(RoundDeadStats(2)),...
    'd', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_DIRECT_M(RoundDeadStats(3)),...
    's', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_DIRECT_M(RoundDeadStats(4)),...
    '^', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
```

```
ylabel('Total System Energy (J)','FontSize',12,'FontWeight','bold');
leg = legend('Total System Energy','1st Node Dead', '10% Nodes Dead',...
    '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
ylim([-0.1 Eo*n+1]);
saveas(figure(3), 'ENERGY_DIRECT_M');
saveas(figure(3), 'ENERGY_DIRECT_M.bmp');
hold off;

fig4 = figure(4);
set(fig4,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_VARIANCE_DIRECT_M(1:round) , 'LineWidth', 2);
hold on;
plot(RoundDeadStats(1), ENERGY_VARIANCE_DIRECT_M(RoundDeadStats(1)),...
    'p', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), ENERGY_VARIANCE_DIRECT_M(RoundDeadStats(2)),...
    'd', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_VARIANCE_DIRECT_M(RoundDeadStats(3)),...
    's', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_VARIANCE_DIRECT_M(RoundDeadStats(4)),...
    '^', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Energy Variance','FontSize',12,'FontWeight','bold');
leg = legend('Variance of Energy Disribution','1st Node Dead',...
    '10% Nodes Dead', '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
hold off;
saveas(figure(4), 'ENERGY_VARIANCE_DIRECT_M');
saveas(figure(4), 'ENERGY_VARIANCE_DIRECT_M.bmp');

save('DIRECT_M_DATA');
```

Published with MATLAB® R2013a

5. Minimum Transmission Energy with Dijkstra—Multi-gateway

```
% MTE Multi Gateway with DIJKSTRA shortest path
clc;
clear all;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Field Dimensions - x and y maximum (in meters)
xm=50;
ym=50;

%x and y Coordinates of the Sink1
sink.x1 = 25; %0.5*xm;
sink.y1 = -100; %0.5*ym;

%x and y Coordinates of the Sink1
sink.x2 = 25; %0.5*xm;
sink.y2 = 150; %0.5*ym;

%Packet size in bits
L = 2000;

%Number of Nodes in the field
n=100;

%Energy Model (all values in Joules)
%Initial Energy
Eo=0.5;
%Eelec=Etx=Erx
ETX=50*0.000000001;
ERX=50*0.000000001;
%Transmit Amplifier types
Efs=10*0.000000000001;
Emp=0.0013*0.000000000001;
%Data Aggregation Energy
EDA=5*0.000000001;

%maximum number of rounds
rmax= 1500; %9999

%%%%%%%%%%%%%%%%%%%%%%         END OF PARAMETERS      %%%%%%%%%%%%%%%%%%%%%
%%%%%%   **SUPPRESS ABOVE IF STARTING WITH COMMON WSN PARAMETERS**   %%%%%%

%Computation of do
do=sqrt(Efs/Emp);

%Get the screensize so each figure can be normalized in a similar manner
%for thesis writeup
scrsz = get(0,'ScreenSize');
```

```
%Creation of the random Sensor Network
fig = figure(1);
set(fig, 'Position',[1 scrsz(4)*.25 scrsz(3)*.7 scrsz(4)*.75]);
subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
hold on;
axis([-10 xm+10 sink.y1-10 sink.y2+10]);

for i=1:n
    S(i).xd = SensorX(i);
    %     S(i).xd = rand(1,1)*xm;
    XR(i)=S(i).xd; %copy of the x-coordinate outside of the S struct
    S(i).yd = SensorY(i);
    %     S(i).yd = rand(1,1)*ym;
    YR(i)=S(i).yd; %copy of the y-coordinate outside of the S struct
    plot(S(i).xd,S(i).yd,'o');
    S(i).E = Eo;
end

xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');

%plot the sink
S(n+1).xd=sink.x1;
S(n+1).yd=sink.y1;
S(n+2).xd=sink.x2;
S(n+2).yd=sink.y2;
sinkX = [sink.x1 sink.x2];
sinkY = [sink.y1 sink.y2];

plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
    'g','MarkerSize',10); %plots the location of the sink1
plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
    'g','MarkerSize',10);
axis([-5 xm+5 sink.y1-5 sink.y2+5]);

%plot horizontal boundaries of the sensor field
bottomY=[0,0];
bottomX=[0,xm];
topY = [ym,ym];
topX = [0,xm];

%plot vertical extremes of the sensor field
vertLeftX=[0,0];
vertLeftY=[0,ym];
vertRightX=[xm,xm];
vertRightY=[0,ym];
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
set(perimeter,'Color','red','LineWidth',1);
hold off;

%start the plot for the node energy bar graph
```

```
energyBar = zeros(1, n);
for iii = 1:n;
    energyBar(iii) = S(iii).E;
end

subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
bar(energyBar);
ylim([0 0.51]);
xlim([-10 110]);
xlabel('Node Number','FontSize',12,'FontWeight','bold');
ylabel('Energy (J)','FontSize',12,'FontWeight','bold');

%Start the plot for 3D Energy Stem
subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
stem3(XR, YR, energyBar, 'Fill', 'g');
hold on;
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
set(perimeter,'Color','r','LineWidth',1);
axis([-1 xm+1 -1 ym+1 0 Eo]);
xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
zlabel('Energy (J)','FontSize',12,'FontWeight','bold');
set(get(gca,'xlabel'),'rotation',14);
set(get(gca,'ylabel'),'rotation',338);
grid off;
hold off;

ENERGY_MTE_M_DIJKSTRA = zeros(1,rmax); %Initialize ENERGY vector
ALIVE_MTE_M_DIJKSTRA = zeros(1,rmax); %Initialize ALIVE vector
ENERGY_VARIANCE_MTE_M_DIJKSTRA = zeros(1,rmax);
flag_first_dead = 0;
flag_10P_dead = 0;
flag_50P_dead = 0;
flag_80P_dead = 0;
roundString = '1';

myObj=VideoWriter('MTE_M_DIJKSTRA.avi');
open(myObj);

%Setup for DG matrix
index = 1;
PointNode1 = zeros(1,(n+2)^2);
PointNode2 = zeros(1,(n+2)^2);

for i = 1:(n+2)
    for j = 1:(n+2)
        PointNode1(index)=i;
        PointNode2(index)=j;
        index = index + 1;
    end
end
```

```
for round = 1:rmax
    tic
    round

    %for every node, identify the set of nodes that still have energy,
    %ie the set of nodes that are not dead
    for node = 1:n
        node;
        NodesAvailable = []; %Initialize NodesAvailable to an empty matrix
        alive = zeros(1,n);
        for aa = 1:n
            %build a vector of alive nodes
            %this needs to be done each round for each node becasue a
            %nodes battery life could go dead in the middle of a round
            %thus it should not be allowed to be used again in the
            %computation of routing paths
            if ( S(aa).E > 0 )
                alive(aa) = 1;
            end %if

        end %k loop
        NumberAlive = sum(alive);
        %Initialize NodesAvailable to a matrix of zeros each round to
        %eliminate previous results being on the end of this array as it
        %gets smaller and smaller
        NodesAvailable = zeros(1,NumberAlive);
        NodesAvailable = (find(alive == 1)); % this provides the nodes
        %available for routing between, NodesAvailable must be
        %initialized to an empty array at the beginning of the k for
        %loop so as the array becomes smaller, higher order elements
        %do not accidentally remain.
        NodesAvailableAndGateway = [NodesAvailable, (n+1), (n+2)];%include
        %the gateway , if multi gateway, need to have 102 as well

        DSQ = zeros(n+2); %initialize distance squared matrix
        for bb = NodesAvailableAndGateway
            for cc = NodesAvailableAndGateway
                %populate the wireless weighting matrix DSQ
                DSQ(bb,cc)=(S(bb).xd-S(cc).xd)^2+(S(bb).yd-S(cc).yd)^2;
            end %cols
        end %rows

        weightVector = zeros(1,(n+2)^2);
        format long
        index = 1;
        for dd = 1:(n+2) %note for multigateway, it should be 102
            for ee = 1:(n+2) %note for multigateway, it should be 102
                weightVector(index) = DSQ(dd,ee);
                index = index + 1;
            end
        end
        DG = sparse(PointNode1,PointNode2, weightVector);

        hasenergy = S(node).E;

        if (hasenergy > 0) %node will only send its data if it has energy!

            %Now lets obtain the minimum path to the sink.
            %Calculate a path to both sinks and minimize the dist
            [dist1,pathToGateway1,pred1] = graphshortestpath(DG,node,101);
            [dist2,pathToGateway2,pred2] = graphshortestpath(DG,node,102);

            %Obtain the minimum cost path to either gateway 1 or gateway 2
            dist = min(dist1, dist2);
            if (dist == dist1)
                pathToGateway = pathToGateway1;
            else
                pathToGateway = pathToGateway2;
            end

            % now we have a path, and we need to deduct transmit and
            % recieve energy along the path for each node

            %every node except the first node in path and the sink (the
            %last node) should have their energy decremented
            %corresponding to the cost to recieve the message

            trackIndexInPath = 1;
            for ff = pathToGateway

                %decrement energy for ERX along path
                % eliminates energy from being decremented from the first
                %node in the path (i.e the source), and the last node in
                %the path which is the gateway
                if (ff ~= pathToGateway(1)) && ...
                        (ff~=pathToGateway(length(pathToGateway)))
                    S(ff).E = S(ff).E - ERX*L; %cost of the node to RX
                end

                %decrement energy for ETX along path
                if (ff~=pathToGateway(length(pathToGateway))) %the last node
                    %is the gateway, which does not transmit

                    %calculate the distance to the next node in the path
                    distance= sqrt((S(ff).xd - ...
                        S(pathToGateway(trackIndexInPath+1)).xd)^2+...
                        (S(ff).yd -S(pathToGateway(trackIndexInPath + 1)).yd)^2);
                        %the euclidean distance to the next node in path

                    if (distance > do) %mulipath propagation, decrement
                        %energy accordingly
                        S(ff).E = S(ff).E - ( (ETX)*(L) + Emp*L*( distance^4 ));
                        %No data aggregation (EDA) in MTE routing
                    end %if
                    if (distance <= do) %direct path propagation

                        S(ff).E = S(ff).E - ( (ETX)*(L)  + Efs*L*( distance^2 ));
                        %No data aggregation (EDA) in MTE routing
                    end %if
                end %if
                trackIndexInPath = trackIndexInPath + 1;
            end %ff

        end %if (S(j).E > 0) lop
    end %j loop

    %Every round, obtain the number of nodes that are alive
    ALIVE_MTE_M_DIJKSTRA(round) = NumberAlive;
    %note that we get NumberAlive after iterating thru all nodes
    NumberDead = n - NumberAlive;

    %obtain the node energy at the conclusion of each round and plot
    for cc=1:n
        energyBar(cc) = S(cc).E;
    end
    ENERGY_MTE_M_DIJKSTRA(round)=sum(energyBar);


    %Every round, obtain the energy variance
    ENERGY_VARIANCE_MTE_M_DIJKSTRA(round) = var(energyBar);

    DeadNodeNo = find(alive==0);

    AliveX = zeros(1,NumberAlive);
    AliveY = zeros(1,NumberAlive);
    DeaDX = zeros(1,NumberDead);
    DeaDY = zeros(1,NumberDead);
    EnergyAlive = zeros(1,NumberAlive);
    EnergyDead = zeros(1,NumberDead);
    trackMeAlive = 1;
    trackMeDead = 1;
    for yyy = NodesAvailable;
        AliveX(trackMeAlive) = XR(yyy);
        AliveY(trackMeAlive) =YR(yyy);
        EnergyAlive(trackMeAlive) = S(yyy).E;
        trackMeAlive = trackMeAlive +1;
    end
    for zzz = DeadNodeNo;
        DeaDX(trackMeDead) = XR(zzz);
        DeaDY(trackMeDead) = YR(zzz);
        EnergyDead(trackMeDead) = 0;
        trackMeDead = trackMeDead+1;
    end
```

```
%plot a running bar chart of energy for animation
figure(1)
subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
bar(energyBar)
ylim([0 0.51])
xlim([-10 110])
xlabel('Node Number','FontSize',12,'FontWeight','bold')
ylabel('Energy (J)','FontSize',12,'FontWeight','bold')

%3D Stem Energy Plot
subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
stem3(AliveX, AliveY, EnergyAlive, 'Fill', 'g', 'LineStyle','--');
hold on
if trackMeDead > 1;
    stem3(DeaDX, DeaDY, EnergyDead, 'Fill', 'r', 'LineStyle','--');
end
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
set(perimeter,'Color','r','LineWidth',1);
xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold')
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold')
zlabel('Energy (J)','FontSize',12,'FontWeight','bold')
set(get(gca,'xlabel'),'rotation',14)
set(get(gca,'ylabel'),'rotation',338)
axis([-1 xm+1 -1 ym+1 0 Eo]);
grid off
hold off
drawnow

% Save the round to a string for display on plots
roundString = num2str(round);

subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
plot(AliveX, AliveY, 'o');
hold on;
text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
plot(S(n+1).xd,S(n+1).yd,'o', 'MarkerFaceColor','g', 'MarkerSize', 10);
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
    'g','MarkerSize',10)
axis([-5 xm+5 sink.y1-5 sink.y2+5])
set(perimeter,'Color','r','LineWidth',1);
xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
hold off;
drawnow

% turn every 5th frame into a movie
if mod(round,2)==0;
```

```
        MOVIE_MTE_DIJKSTRA = getframe(figure(1));
        writeVideo(myObj,MOVIE_MTE_DIJKSTRA);
    end

    %Flag the round the first node dies
    if (flag_first_dead == 0)
        if (NumberDead > 0);
            flag_first_dead = 1;
            ROUND_FIRST_DEAD_MTE_M_DIJKSTRA = round;
            fig1dead = figure(8)
            set(fig1dead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(AliveX, AliveY, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
            plot(S(n+1).xd,S(n+1).yd,'o','MarkerFaceColor','g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k',...
                'MarkerFaceColor','g','MarkerSize',10)
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            drawnow;
            hold off;
            saveas(fig1dead,'1_Node_Dead_grid_MTE_M_DIJKSTRA.bmp');
            saveas(fig1dead,'1_Node_Dead_grid_MTE_M_DIJKSTRA');
            saveas(figure(1),'1_Node_Dead_Grid_Energy_MTE_M_DIJKSTRA');
            saveas(figure(1),'1_Node_Dead_Grid_Energy_MTE_M_DIJKSTRA.bmp');
        end %if
    end %if

    %find round when 10% of nodes are dead
    if(flag_10P_dead == 0);
        if (NumberDead >= n*0.1);
            flag_10P_dead = 1;
            ROUND_10P_DEAD_MTE_M_DIJKSTRA = round;
            fig10Pdead = figure(9);
            set(fig10Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(AliveX, AliveY, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
            plot(S(n+1).xd,S(n+1).yd,'o', 'MarkerFaceColor',...
                'g', 'MarkerSize', 10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k',...
                'MarkerFaceColor','g','MarkerSize',10);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            drawnow;
            hold off;
            saveas(fig10Pdead,'10P_Node_Dead_grid_MTE_M_DIJKSTRA.bmp');
            saveas(fig10Pdead,'10P_Node_Dead_grid_MTE_M_DIJKSTRA');
            saveas(figure(1),'10P_Node_Dead_Grid_Energy_MTE_M_DIJKSTRA');
            saveas(figure(1),'10P_Node_Dead_Grid_Energy_MTE_M_DIJKSTRA.bmp');
        end
    end

    %find round when 50% of nodes are dead
    if(flag_50P_dead == 0);
        if (NumberDead >= n*0.5);
            flag_50P_dead = 1;
            ROUND_50P_DEAD_MTE_M_DIJKSTRA = round;
            fig50Pdead = figure(10);
            set(fig50Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(AliveX, AliveY, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
            plot(S(n+1).xd,S(n+1).yd,'o', 'MarkerFaceColor','g',...
                'MarkerSize', 10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k',...
                'MarkerFaceColor','g','MarkerSize',10)
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            drawnow;
            hold off;
            saveas(fig50Pdead,'50P_Node_Dead_grid_MTE_M_DIJKSTRA.bmp');
            saveas(fig50Pdead,'50P_Node_Dead_grid_MTE_M_DIJKSTRA');
            saveas(figure(1),'50P_Node_Dead_Grid_Energy_MTE_M_DIJKSTRA');
            saveas(figure(1),'50P_Node_Dead_Grid_Energy_MTE_M_DIJKSTRA.bmp');
        end
    end

    %find round when 80% of nodes are dead and save network figure
    if(flag_80P_dead == 0);
```

```
        if (NumberDead >= n*0.8);
            flag_80P_dead = 1;
            ROUND_80P_DEAD_MTE_M_DIJKSTRA = round;
            fig80Pdead  = figure(11);
            set(fig80Pdead, 'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(AliveX, AliveY, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(DeaDX, DeaDY, 'r .', 'MarkerSize', 20);
            plot(S(n+1).xd,S(n+1).yd,'o','MarkerFaceColor''g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k',...
                'MarkerFaceColor','g','MarkerSize',10);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1)
            xlabel ('x-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel ('y-Grid Axis (m)','FontSize',12,'FontWeight','bold');
            drawnow;
            hold off;
            saveas(fig80Pdead,'80P_Node_Dead_grid_MTE_M_DIJKSTRA.bmp');
            saveas(fig80Pdead,'80P_Node_Dead_grid_MTE_M_DIJKSTRA');
            saveas(figure(1),'80P_Node_Dead_Grid_Energy_MTE_M_DIJKSTRA');
            saveas(figure(1),'80P_Node_Dead_Grid_Energy_MTE_M_DIJKSTRA.bmp');
        end
    end

    %break out of the loop when all nodes are dead
    if ALIVE_MTE_M_DIJKSTRA(round) == 0;
        break;
    end

    toc
end %i loop (all rounds are done!)

close(myObj);

RoundDeadStats= [ROUND_FIRST_DEAD_MTE_M_DIJKSTRA ROUND_10P_DEAD_MTE_M_DIJKSTRA ...
    ROUND_50P_DEAD_MTE_M_DIJKSTRA ROUND_80P_DEAD_MTE_M_DIJKSTRA];

fig2 = figure(2);
set(fig2,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ALIVE_MTE_M_DIJKSTRA(1:round) , 'LineWidth', 2);
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Number of Nodes Alive','FontSize',12,'FontWeight','bold');
ylim([0 n+1]);
saveas(figure(2), 'NodesAliveVsRound_MTE_M_DIJKSTRA');
saveas(figure(2), 'NodesAliveVsRound_MTE_M_DIJKSTRA.bmp');

fig3 = figure(3);
set(fig3,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_MTE_M_DIJKSTRA(1:round) , 'LineWidth', 2);
hold on;
plot(RoundDeadStats(1), ENERGY_MTE_M_DIJKSTRA(RoundDeadStats(1)),...
    'p', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), ENERGY_MTE_M_DIJKSTRA(RoundDeadStats(2)),...
    'd', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_MTE_M_DIJKSTRA(RoundDeadStats(3)),...
    's', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_MTE_M_DIJKSTRA(RoundDeadStats(4)),...
    '^', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Total System Energy (J)','FontSize',12,'FontWeight','bold');
leg = legend('Total System Energy','1st Node Dead', '10% Nodes Dead',...
    '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
ylim([-0.1 Eo*n+1]);
hold off;
saveas(figure(3), 'ENERGY_MTE_M_DIJKSTRA');
saveas(figure(3), 'ENERGY_MTE_M_DIJKSTRA.bmp');

fig4 = figure(4);
set(fig4,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_VARIANCE_MTE_M_DIJKSTRA(1:round) , 'LineWidth', 2);
hold on;
plot(RoundDeadStats(1), ENERGY_VARIANCE_MTE_M_DIJKSTRA(RoundDeadStats(1)),...
    'p', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), ENERGY_VARIANCE_MTE_M_DIJKSTRA(RoundDeadStats(2)),...
    'd', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_VARIANCE_MTE_M_DIJKSTRA(RoundDeadStats(3)),...
    's', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_VARIANCE_MTE_M_DIJKSTRA(RoundDeadStats(4)),...
    '^', 'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Energy Variance (J^2)','FontSize',12,'FontWeight','bold');
leg = legend('Variance of Energy Disribution','1st Node Dead',...
    '10% Nodes Dead', '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
hold off;
saveas(figure(4), 'ENERGY_VARIANCE_MTE_M_DIJKSTRA');
saveas(figure(4), 'ENERGY_VARIANCE_MTE_M_DIJKSTRA.bmp');

save('MTE_M_DIJKSTRA_DATA');
```

Published with MATLAB® R2013a

6. LEACH—Multi-gateway

```
%LEACH Multi gateway

clc;
clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Field Dimensions - x and y maximum (in meters)
xm=50;
ym=50;

%x and y Coordinates of Sink 1
sink.x1 = 25;
sink.y1 = -50;
%x and y Coordinates of Sink 2 (FOR MULTIGATEWAY CASE)
sink.x2 = 25;
sink.y2 = 100;

%Packet size in bits
L = 2000;

%Number of Nodes in the field
n=100;

%Optimal Election Probability of a node to become cluster head
p=0.05;

%Energy Model (all values in Joules)
%Initial Energy
Eo=0.5;

%Eelec=Etx=Erx
ETX=50*0.000000001;
ERX=50*0.000000001;
%Transmit Amplifier types
Efs=10*0.000000000001;
Emp=0.0013*0.000000000001;
%Data Aggregation Energy
EDA=5*0.000000001;

%maximum number of rounds
rmax= 4000; %9999

%%%%%%%%%%%%%%%%%%%%%%         END OF PARAMETERS      %%%%%%%%%%%%%%%%%%%%%
%%%%%%   **SUPPRESS ABOVE IF STARTING WITH COMMON WSN PARAMETERS**   %%%%%%

%Computation of do
do=sqrt(Efs/Emp);
```

```
%Get the screensize so each figure can be normalized in a similar manner
%for thesis writeup
scrsz = get(0,'ScreenSize');

%Creation of the random Sensor Network
fig = figure(1);
set(fig, 'Position',[1 scrsz(4)*.25 scrsz(3)*.7 scrsz(4)*.75]);

subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
hold on

for i=1:n;
    S(i).xd = SensorX(i);
    %     S(i).xd = rand(1,1)*xm;
    XR(i)=S(i).xd; %copy of the x-coordinate outside of the S struct
    S(i).yd = SensorY(i);
    %     S(i).yd = rand(1,1)*ym;
    YR(i)=S(i).yd; %copy of the y-coordinate outside of the S struct
    S(i).G=0;
    %initially there are no cluster heads, only nodes
    S(i).type='N';
    S(i).E=Eo;
    plot(S(i).xd,S(i).yd,'o');
end

xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');

%plot the sink
S(n+1).xd1=sink.x1;
S(n+1).yd1=sink.y1;
S(n+2).xd2=sink.x2;
S(n+2).yd2=sink.y2;
plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
    'g','MarkerSize',10)
plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
    'g','MarkerSize',10)
axis([-5 xm+5 sink.y1-5 sink.y2+5])

%plot horizontal boundaries of the sensor field
bottomY=[0,0];
bottomX=[0,xm];
topY = [ym,ym];
topX = [0,xm];

%plot vertical extremes of the sensor field
vertLeftX=[0,0];
vertLeftY=[0,ym];
vertRightX=[xm,xm];
vertRightY=[0,ym];
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
axis([-5 xm+5 sink.y1-5 sink.y2+5])
set(perimeter,'Color','r','LineWidth',1);
figure(1);
hold off

%start the plot for the node energy bar graph
energyBar = zeros(1, n);
for iii = 1:n;
    energyBar(iii) = S(iii).E;
end

subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
bar(energyBar);
ylim([0 0.51]);
xlim([-10 110]);
xlabel('Node Number','FontSize',12,'FontWeight','bold');
ylabel('Energy (J)','FontSize',12,'FontWeight','bold');

%Start the plot for 3D Energy Stem
subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
stem3(XR, YR, energyBar, 'Fill', 'g');
hold on;
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
set(perimeter,'Color','r','LineWidth',1);
axis([-1 xm+1 -1 ym+1 0 Eo]);
xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
zlabel('Energy (J)','FontSize',12,'FontWeight','bold');
set(get(gca,'xlabel'),'rotation',14);
set(get(gca,'ylabel'),'rotation',338);
grid off;
hold off;

%counter for CHs, initializes the countCHs metric to zero
countCHs=0;
%counter for CHs per round
rcountCHs=0;
cluster=1;

countCHs;
rcountCHs=rcountCHs+countCHs;

flag_first_dead = 0;
flag_10P_dead = 0;
flag_50P_dead = 0;
flag_80P_dead = 0;
roundString = '1';
ENERGY_VARIANCE_LEACH_M = zeros(1, rmax);
CLUSTERHS_LEACH_M = zeros(1, rmax);
ENERGY_LEACH_M = zeros(1, rmax);
```

```
DEAD_LEACH_M = zeros(1, rmax);

myObj=VideoWriter('MOVIE_DIEOUT_LEACH_M.avi');
open(myObj);

for r=0:rmax;
    r

    %Initialize arrays
    ClusterHeads = [];
    ClusterHeadsX = [];
    ClusterHeadsY = [];

    %Reset G and cl every 1/p rounds
    if(mod(r, round(1/p) )==0);
        for bbb=1:n;
            S(bbb).G=0;
            S(bbb).cl=0;
        end
    end

    for aaa=1:n;
        if (S(aaa).E<=0);
            S(aaa).type='D';
        end
        if S(aaa).E>0;
            S(aaa).type='N';
        end
    end

    cluster=1;
    in_while = 0;
    while(cluster==1);
        for bb=1:n; %this loop goes through each node and identifies the CHs
            if(S(bb).E>0);
                temp_rand = rand;
                if ( (S(bb).G) <= 0); %Only chooses nodes that belong to
                    %set G (nodes that havent been clusterheads in last
                    %1/p rounds) as clusterheads

                    %Election of Cluster Heads
                    if(temp_rand <= (p/(1-p*mod(r,round(1/p)))));

                        S(bb).type = 'C';
                        S(bb).G = round(1/p)-1; %Annotates the node has been
                        %a CH in the last 1/p rounds
                        C(cluster).xd = S(bb).xd;
                        C(cluster).yd = S(bb).yd;
                        ClusterHeadsX(cluster) = S(bb).xd;
                        ClusterHeadsY(cluster) = S(bb).yd;
                        ClusterHeads(cluster) = bb;
                        %Incoporate multigateway below
                        distance = min([(sqrt( (S(bb).xd-(S(n+1).xd1))^2 +...
                          (S(bb).yd-(S(n+1).yd1))^2))(sqrt((S(bb).xd-(S(n+2).xd2) )^2+...
                           (S(bb).yd-(S(n+2).yd2) )^2 ))]); %Takes the
                            %shortest distance to either Sink1 or Sink 2

                        C(cluster).distance = distance;
                        C(cluster).id = bb; %The C(cluster).id is id of the CH
                        X(cluster)=S(bb).xd;
                        Y(cluster)=S(bb).yd;
                        cluster=cluster+1;

                        %Calculation of Energy dissipated (ONLY FOR CHs)
                        distance;
                        if (distance > do); %MultiPath Propagation
                            S(bb).E=S(bb).E - ( (ETX+EDA)*(L) + Emp*L*( distance^4 ));
                        end
                        if (distance <= do); %DirectPath Propagation
                            S(bb).E=S(bb).E - ( (ETX+EDA)*(L)  + Efs*L*( distance^2 ));
                        end
                    end

                end
            end
        end
        in_while = in_while+1;
        if in_while == 100;
            break
        end
    end

    CLUSTERHS_LEACH_M(r+1)=cluster-1;


    %Each node picks its closest clusterhead
    %Every node it checks distance to each basestation and the distance to
    %each clusterhead, the output being the smallest distance, which is the
    %distance to transmit the nodes' energy
    for cc=1:n; % Check all the nodes
        if ( S(cc).type=='N' && S(cc).E>0 );
            if(cluster-1>=1)% What if no clusterheads are chosen... i.e. cluster-1 = 0??
                min_dis = min([(sqrt( (S(cc).xd-(S(n+1).xd1) )^2 +...
                    (S(cc).yd-(S(n+1).yd1) )^2 )) (sqrt( (S(cc).xd-(S(n+2).xd2) )^2 +...
                    (S(cc).yd-(S(n+2).yd2) )^2 ))]); %Each 'N' node identifies the
                    %minimum distance to the basestation
                min_dis_cluster = 1;
                for dd=1:cluster-1; %cluster - 1 is the number of CHs for that round
                    temp = min(min_dis, sqrt( (S(cc).xd-C(dd).xd)^2 +...
                        (S(cc).yd-C(dd).yd)^2 ) ); %take smaller of distance to
                        %basestation or distance to CH
                    if ( temp < min_dis )
                        min_dis=temp;
                        min_dis_cluster = dd;
                    end
                end

                %Energy dissipated by the node to transmit the message to
                %its clusterhead
                min_dis;
                if (min_dis>do); %MultiPath Propagation
                    S(cc).E = S(cc).E - ( ETX*(L) + Emp*L*( min_dis^4));
                end
                if (min_dis<=do); %DirectPath Propagation
                    S(cc).E = S(cc).E- ( ETX*(L) + Efs*L*( min_dis^2));
                end
                %Energy dissipated for the node's clusterhead to recieve
                %and aggregate the message.
                if(min_dis>0);
                    S(C(min_dis_cluster).id).E=S(C(min_dis_cluster).id).E-((ERX+EDA)*L );
                end
                S(cc).min_dis=min_dis;
                S(cc).min_dis_cluster = min_dis_cluster;
            end
            if (cluster == 1);  % this is the case where no clusterheads are chosen
                %nodes communicate directly with sink
                min_dis = min([(sqrt( (S(cc).xd-(S(n+1).xd1) )^2 +...
                    (S(cc).yd-(S(n+1).yd1) )^2 )) (sqrt( (S(cc).xd-(S(n+2).xd2) )^2 +...
                    (S(cc).yd-(S(n+2).yd2) )^2 ))]);
                %Energy dissipated by the node to transmit the message to
                %closest basestation in case there are no clusterheads
                %chosen
                disp('NO CLUSTERHEADS CHOSEN ... SENDING DIRECT TO SINK')
                if (min_dis>do); %MultiPath Propagation
                    S(cc).E = S(cc).E - ( ETX*(L) + Emp*L*( min_dis^4));
                end
                if (min_dis<=do); %DirectPath Propagation
                    S(cc).E = S(cc).E- ( ETX*(L) + Efs*L*( min_dis^2));
                end
            end
        end
    end

    %obtain the energy variance and determine dead nodes
    energyBar = zeros(1, n);
    deadholder = 0;
    for www = 1:n;
        energyBar(www) = S(www).E;
        if S(www).E <=0;
            deadholder = deadholder+1;
        end
    end
    DEAD_LEACH_M(r+1)=deadholder;

    ENERGY_VARIANCE_LEACH_M(r+1) = var(energyBar);
    ENERGY_LEACH_M(r+1) = sum(energyBar);

    %plot a running bar chart of energy for animation
    figure(1);
    subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
    bar(energyBar);
    ylim([0 0.51]);
    xlim([-10 110]);
    xlabel('Node Number','FontSize',12,'FontWeight','bold');
    ylabel('Energy (J)','FontSize',12,'FontWeight','bold');
    drawnow


    % Save the round to a string for display on plots
    roundString = num2str(r);

    trackitAlive = 1;
    trackitDead = 1;
    %preallocate for speed
    RoundDead = zeros(1, deadholder);
    RoundDeadx = zeros(1, deadholder);
    RoundDeady = zeros(1, deadholder);
    EnergyDead = zeros(1, deadholder);
    RoundAlive = zeros(1, n - deadholder);
    RoundAlivex = zeros(1, n - deadholder);
    RoundAlivey = zeros(1, n - deadholder);
    EnergyAlive = zeros(1, n - deadholder);
    RoundCHx = [];
    RoundCHy = [];
    RoundCHs = [];
    NormalNode = [];
    NormalNodex = [];
    NormalNodey = [];

    for hh = 1:n;
        if S(hh).E > 0;
            RoundAlive(trackitAlive) = hh;
            RoundAlivex(trackitAlive) = XR(hh);
            RoundAlivey(trackitAlive) = YR(hh);
            EnergyAlive(trackitAlive) = S(hh).E;
            trackitAlive = trackitAlive +1;
        end
        if S(hh).E <=0;
            RoundDead(trackitDead)= hh;
            RoundDeadx(trackitDead)= XR(hh);
            RoundDeady(trackitDead)= YR(hh);
            EnergyDead(trackitDead) = 0;
            trackitDead = trackitDead + 1;
        end
    end

    %3D Stem Energy Plot
```

```matlab
    subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
    stem3(RoundAlivex, RoundAlivey, EnergyAlive, 'Fill', 'g', 'LineStyle','--');
    hold on;
    if trackitDead > 1;
        stem3(RoundDeadx, RoundDeady, EnergyDead, 'Fill', 'r', 'LineStyle','--');
    end
    perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
        vertRightX, vertRightY);
    set(perimeter,'Color','r','LineWidth',1);
    xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    zlabel('Energy (J)','FontSize',12,'FontWeight','bold');
    set(get(gca,'xlabel'),'rotation',14);
    set(get(gca,'ylabel'),'rotation',338);
    axis([-1 xm+1 -1 ym+1 0 Eo]);
    grid off;
    hold off;
    drawnow;

    subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
    plot(XR,YR, 'o');
    hold on;
    text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
    text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
    plot(ClusterHeadsX,ClusterHeadsY, 'k *');
    if deadholder > 0;
        plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
    end
    plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
        'g','MarkerSize',10);
    plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
        'g','MarkerSize',10);
    perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
        vertRightX, vertRightY);
    axis([-5 xm+5 sink.y1-5 sink.y2+5]);
    set(perimeter,'Color','r','LineWidth',1);
    xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    hold off;
    drawnow;

    %turn every 5th frame into a movie
    if mod(r,5)==0;
        MOVIE_DIEOUT_LEACH_M = getframe(figure(1));
        writeVideo(myObj,MOVIE_DIEOUT_LEACH_M);
    end

    %find round first node dead
    if(flag_first_dead == 0);
        if (deadholder >= 1);
            flag_first_dead = 1;
            ROUND_FIRST_DEAD_LEACH_M = r;
            fig1dead = figure(8);
            set(fig1dead, 'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig1dead,'1_Node_Dead_grid_LEACH_M.bmp');
            saveas(fig1dead,'1_Node_Dead_grid_LEACH_M');
            saveas(figure(1),'1_Node_Dead_Grid_Energy_LEACH_M');
            saveas(figure(1),'1_Node_Dead_Grid_Energy_LEACH_M.bmp');
        end
    end

    %find the round when 10% of nodes are dead and save network figures
    if(flag_10P_dead == 0);
        if (deadholder >= 0.1*n);
            flag_10P_dead = 1;
            ROUND_10P_DEAD_LEACH_M = r;
            fig10Pdead = figure(9);
            set(fig10Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor','g',...
                'MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor','g',...
                'MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig10Pdead,'10P_Node_Dead_grid_LEACH_M.bmp');
            saveas(fig10Pdead,'10P_Node_Dead_grid_LEACH_M');
            saveas(figure(1),'10P_Node_Dead_Grid_Energy_LEACH_M');
            saveas(figure(1),'10P_Node_Dead_Grid_Energy_LEACH_M.bmp');
        end
    end

    %find round when 50% of nodes are dead
    if(flag_50P_dead == 0);
        if (deadholder >= 0.5*n);
            flag_50P_dead = 1;
            ROUND_50P_DEAD_LEACH_M = r;
            fig50Pdead = figure(10);
            set(fig50Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor','g',...
                'MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor','g',...
                'MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig50Pdead,'50P_Node_Dead_grid_LEACH_M.bmp');
            saveas(fig50Pdead,'50P_Node_Dead_grid_LEACH_M');
            saveas(figure(1),'50P_Node_Dead_Grid_Energy_LEACH_M');
            saveas(figure(1),'50P_Node_Dead_Grid_Energy_LEACH_M.bmp');
        end
    end

    %find round when 80% of nodes are dead and save network figure
```

```
    if(flag_80P_dead == 0);
        if (deadholder >= 0.8*n);
            flag_80P_dead = 1;
            ROUND_80P_DEAD_LEACH_M = r;
            fig80Pdead = figure(11);
            set(fig80Pdead, 'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor','g',...
                'MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor','g',...
                'MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig80Pdead,'80P_Node_Dead_grid_LEACH_M.bmp');
            saveas(fig80Pdead,'80P_Node_Dead_grid_LEACH_M');
            saveas(figure(1),'80P_Node_Dead_Grid_Energy_LEACH_M');
            saveas(figure(1),'80P_Node_Dead_Grid_Energy_LEACH_M.bmp');
        end
    end

    if deadholder == n;
        break;
    end

end

close(myObj);

ALIVE_LEACH_M = zeros(1,r);
for ii = 0:r
    ALIVE_LEACH_M(ii+1) = n - DEAD_LEACH_M(ii+1);
end

RoundDeadStats= [ROUND_FIRST_DEAD_LEACH_M ROUND_10P_DEAD_LEACH_M ...
    ROUND_50P_DEAD_LEACH_M ROUND_80P_DEAD_LEACH_M];
```

```
fig2 = figure(2);
set(fig2,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ALIVE_LEACH_M, 'LineWidth', 2);
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Number of Nodes Alive','FontSize',12,'FontWeight','bold');
ylim([0 n+1]);
saveas(figure(2), 'NodesAliveVsRound_LEACH_M');
saveas(figure(2), 'NodesAliveVsRound_LEACH_M.bmp');


fig3 = figure(3);
set(fig3,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_LEACH_M(1:r), 'LineWidth', 2);
hold on;
plot(RoundDeadStats(1), ENERGY_LEACH_M(RoundDeadStats(1)), 'p',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), ENERGY_LEACH_M(RoundDeadStats(2)), 'd',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_LEACH_M(RoundDeadStats(3)), 's',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_LEACH_M(RoundDeadStats(4)), '^',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Total System Energy (J)','FontSize',12,'FontWeight','bold');
ylim([-0.1 Eo*n+1]);
saveas(figure(3), 'TotalSystemEnergy_LEACH_M');
saveas(figure(3), 'TotalSystemEnergy_LEACH_M.bmp');

fig4 = figure(4);
set(fig4,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_VARIANCE_LEACH_M(1:r), 'LineWidth', 2);
hold on;
plot(RoundDeadStats(1), ENERGY_VARIANCE_LEACH_M(RoundDeadStats(1)), 'p',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), ENERGY_VARIANCE_LEACH_M(RoundDeadStats(2)), 'd',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_VARIANCE_LEACH_M(RoundDeadStats(3)), 's',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_VARIANCE_LEACH_M(RoundDeadStats(4)), '^',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Energy Variance (J^2)','FontSize',12,'FontWeight','bold');
leg = legend('Variance of Energy Disribution','1st Node Dead',...
    '10% Nodes Dead', '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
hold off;
```

```
saveas(figure(4), 'ENERGY_VARIANCE_LEACH_M');
saveas(figure(4), 'ENERGY_VARIANCE_LEACH_M.bmp');

%Filter the Clusterheads per round for plotting
filterRounds = 50
c = 1/filterRounds;
for i = 1:filterRounds
    b(i) = c; %b is vector for matlab filter()
end
a = 1; %a is vector for matlab filter()
filtered_CLUSTERHS_LEACH_M = filter(b,a, CLUSTERHS_LEACH_M(1:r));

fig5 = figure(5)
set(fig5,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(CLUSTERHS_LEACH_M(1:r), 'LineWidth', 1);
hold on;
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
plot(filtered_CLUSTERHS_LEACH_M, 'r -', 'LineWidth', 2);
plot(RoundDeadStats(1), filtered_CLUSTERHS_LEACH_M(RoundDeadStats(1)), 'p',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), filtered_CLUSTERHS_LEACH_M(RoundDeadStats(2)), 'd',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), filtered_CLUSTERHS_LEACH_M(RoundDeadStats(3)), 's',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), filtered_CLUSTERHS_LEACH_M(RoundDeadStats(4)), '^',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
xlabel('Round','FontSize',12,'FontWeight','bold')
ylabel('Number Of Cluster Heads','FontSize',12,'FontWeight','bold')
leg = legend('Clusterheads per round','50 Round Moving Average Filter',...
    '1st Node Dead', '10% Nodes Dead', '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
hold off
saveas(figure(5), 'ClusterheadsPerRound_LEACH_M');
saveas(figure(5), 'ClusterheadsPerRound_LEACH_M.bmp');

save('LEACH_M_DATA');
```

Published with MATLAB® R2013a

7. Zone Clustering with Random CH Electio—Multi-gateway

```
% User identified Zone Based Protocol with random CH Election Multi Gateway

clc;
clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Field Dimensions - x and y maximum (in meters)
xm=50;
ym=50;

%x and y Coordinates of Sink1
sink.x1 = 25;
sink.y1 = -100;

%x and y Coordinates of Sink 2
sink.x2 = 25;
sink.y2 = 150;

%Packet size in bits
L = 2000;

%Number of Nodes in the field
n = 100;

% Number to zones to subdivide the field
z = 5;

%Energy Model (all values in Joules)
%Initial Energy
Eo=0.5;
%Eelec=Etx=Erx
ETX=50*0.000000001;
ERX=50*0.000000001;
%Transmit Amplifier types
Efs=10*0.000000000001;
Emp=0.0013*0.000000000001;
%Data Aggregation Energy
EDA=5*0.000000001;

%maximum number of rounds
rmax= 9999;

%%%%%%%%%%%%%%%%%%%%%%         END OF PARAMETERS      %%%%%%%%%%%%%%%%%%%%%
%%%%%%   **SUPPRESS ABOVE IF STARTING WITH COMMON WSN PARAMETERS**   %%%%%%

%Computation of do
do=sqrt(Efs/Emp);

%Get the screensize so each figure can be normalized in a similar manner
%for thesis writeup
scrsz = get(0,'ScreenSize');

%Creation of the random Sensor Network
fig = figure(1);
set(fig, 'Position',[1 scrsz(4)*.25 scrsz(3)*.7 scrsz(4)*.75])

subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
hold on;
```

```
for i=1:n
    S(i).xd = SensorX(i);
    %     S(i).xd = rand(1,1)*xm;
    XR(i)=S(i).xd; %copy of the x-coordinate outside of the S struct
    S(i).yd = SensorY(i);
    %     S(i).yd = rand(1,1)*ym;
    YR(i)=S(i).yd; %copy of the y-coordinate outside of the S struct
    S(i).E=Eo;
    S(i).totalAsCH = 0;
    plot(S(i).xd,S(i).yd,'o');
end

xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold')
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold')

%plot the sink
S(n+1).xd1=sink.x1;
S(n+1).yd1=sink.y1;
S(n+2).xd2=sink.x2;
S(n+2).yd2=sink.y2;
plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor','g',...
    'MarkerSize',10);
plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor','g',...
    'MarkerSize',10)
axis([-5 xm+5 sink.y1-5 sink.y2+5])

%plot horizontal boundaries of the sensor field
bottomY=[0,0];
bottomX=[0,xm];
topY = [ym,ym];
topX = [0,xm];

%plot vertical extremes of the sensor field
vertLeftX=[0,0];
vertLeftY=[0,ym];
vertRightX=[xm,xm];
vertRightY=[0,ym];
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
axis([-5 xm+5 sink.y1-5 sink.y2+5]);
set(perimeter,'Color','r','LineWidth',1);

%plot the vertical zone partitions
for bb = 1:z-1;
    vertX = [xm/z*bb, xm/z*bb];
    vertY = [0 , ym];
    line(vertX,vertY, 'Color', 'r');
end
figure(1);
hold off;
```

```
%start the plot for the node energy bar graph
energyBar = zeros(1, n);
for iii = 1:n;
    energyBar(iii) = S(iii).E;
end

subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
bar(energyBar);
ylim([0 0.51]);
xlim([-10 110]);
xlabel('Node Number','FontSize',12,'FontWeight','bold');
ylabel('Energy (J)','FontSize',12,'FontWeight','bold');

%Start the plot for 3D Energy Stem
subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
stem3(XR, YR, energyBar, 'Fill', 'g');
hold on;
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
for bb = 1:z-1;
    vertX = [xm/z*bb, xm/z*bb];
    vertY = [0 , ym];
    line(vertX,vertY, 'Color', 'r');
end
set(perimeter,'Color','r','LineWidth',1);
axis([-1 xm+1 -1 ym+1 0 Eo]);
xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
zlabel('Energy (J)','FontSize',12,'FontWeight','bold');
set(get(gca,'xlabel'),'rotation',14);
set(get(gca,'ylabel'),'rotation',338);
grid off;
hold off;

%Partition the field into zones, assign each point a zone starting from x =
%0 and increasing to x max.  Zones are verticle zones in the field.
for dd = 1:n;
    S(dd).zone = ceil(S(dd).xd/(xm/z)); %each zone only corresponds
    %to its x coordinate
end

%Set flags, preallocate arrays for speed
ENERGY_ZONE_M = zeros(1, rmax);
ENERGY_VARIANCE_ZONE_M = zeros(1, rmax);
DEAD_ZONE_M = zeros(1, rmax);
CLUSTERHS_ZONE_M = zeros(1, rmax);
flag_first_dead = 0;
flag_10P_dead = 0;
flag_50P_dead = 0;
flag_80P_dead = 0;
roundString = '1';
```

```
myObj=VideoWriter('MOVIE_DIEOUT_ZONE_M.avi');
open(myObj);

for round = 1:rmax; % Perform zone based CH rotation every round
    round

    for ee=1:n;
        if (S(ee).E <= 0);
            S(ee).type = 'D';
        end
        if (S(ee).E > 0);
            S(ee).type='N';
        end
    end

    %Initialize arrays
    ClusterHeads = [];
    ClusterHeadsX = [];
    ClusterHeadsY = [];

    clustersHeadsPerRound = 0;

    for jj = 1:z; %do it for every zone

        %initialize arrays & variables that are used each round:
        zoneCHinZone = [];
        zone = [];
        index = 1;
        zoneCHx = 0;
        zoneCHy = 0;
        distance = 0;

        for kk = 1:n; %first lets separate the nodes into their zones
            %if they have energy
            if(S(kk).zone == jj) && (S(kk).E > 0); %only nodes that have
                %energy can be included in all calculations
                zone(index) = kk;
                index = index + 1;
            end
        end

        %now there is a vector “zone” with the id's of members S in the zone
        %randomly select a node from from the “zone” array to be a CH

        %randomly select an array position in zone then identify what node
        %was selected
        if (length(zone)>1); %if (length(zone)>1) is required for when
            %there is only 1 node left in a zone
            zoneCHinZone = randi(length(zone),1,1); % randomly chooses 1
            %integer from 1 to length(zone)
            zoneCH = zone(zoneCHinZone); %gives the original index of the CH
        else
            zoneCH = zone; %there is only one node left in the zone so it
            %must be a CH
        end

        if (~isempty(zoneCH));

            clustersHeadsPerRound = clustersHeadsPerRound + 1;

            %if inside this if statement, that means there is at least one
            %node in the current zone.
            %This must be in an if statement becasue otherwise there would
            %be errors as one zone completely dies out but other zones have
            %not. Thus if a zone has no remaining nodes with energy, it
            %simply bypasses to the next zone

            S(zoneCH).totalAsCH = S(zoneCH).totalAsCH + 1;
            S(zoneCH).type = 'C';
            zoneCHx = S(zoneCH).xd;
            zoneCHy = S(zoneCH).yd;
            ClusterHeads(jj) = zoneCH;
            ClusterHeadsX(jj)= zoneCHx;
            ClusterHeadsY(jj)= zoneCHy;

            %Identify the smaller distance to the Sink 1 or 2;
            distance = min(sqrt((zoneCHx-S(n+1).xd1)^2 + (zoneCHy-S(n+1).yd1)^2),...
                sqrt((zoneCHx-S(n+2).xd2)^2 + (zoneCHy-S(n+2).yd2)^2));



            %energy cost for the CH in the zone to aggregate its
            %own sensor data and and transmit the message to the gateway
            if (distance > do);
                S(zoneCH).E=S(zoneCH).E - ( (ETX+EDA)*(L) + Emp*L*( distance^4 ));
            end
            if (distance <= do);
                S(zoneCH).E=S(zoneCH).E - ( (ETX+EDA)*(L)  + Efs*L*( distance^2 ));
            end

            %the total number of nodes under the CH is length(zone)-1
            %then the CH will recieve length(zone)-1 messages

            %energy cost for the CH to recieve and aggregate
            %messages from its nodes.
            S(zoneCH).E = S(zoneCH).E - (ERX+EDA)*L*(length(zone)-1);

            %Now, for each node in the zone, except for the zone's CH,
            %Let's deduct energy cost for each node to send the message to
            %the CH.

            %Iterate through each node in the zone except for the
            %cluster head node.
            %Use the “zone” array since it is a vector for each node in
            %the zone (but dont include the CH)

            for gg = zone;
                %for each iteration, gg represents the ID of the node we are
                %ranging to the CH

                if (gg ~= zoneCH);
                    %calculate the distance to the CH
                    distance = sqrt( (S(gg).xd - zoneCHx)^2 + (S(gg).yd - zoneCHy)^2 );

                    %Energy cost to transmit L bits to CH
                    if (distance > do); %Multipath Propagation
                        S(gg).E = S(gg).E - (ETX*L + Emp*L*(distance^4));
                    end
                    if (distance <= do);%Direct path propagation
                        S(gg).E = S(gg).E - (ETX*L + Efs*L*(distance^2));
                    end
                end %gg~= zoneCH
            end%gg=zone
        end %(~isempty(zoneCH))
    end

    CLUSTERHS_ZONE_M(round) = clustersHeadsPerRound;

    %Plot and obtain desired stats

    %obtain the energy variance and determine dead nodes
    energyBar = zeros(1, n);
    deadholder = 0;
    for www = 1:n;
        energyBar(www) = S(www).E;
        if S(www).E <=0;
            deadholder = deadholder+1;
        end
    end
    DEAD_ZONE_M(round)=deadholder;

    ENERGY_VARIANCE_ZONE_M(round) = var(energyBar);
    ENERGY_ZONE_M(round)=sum(energyBar);

    %plot a running bar chart of energy for animation
    figure(1);
    subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
    bar(energyBar);
    ylim([0 0.51]);
    xlim([-10 110]);
    xlabel('Node Number','FontSize',12,'FontWeight','bold');
    ylabel('Energy (J)','FontSize',12,'FontWeight','bold');
    drawnow;

    % Save the round to a string for display on plots
```

```
    roundString = num2str(round);

    trackitAlive = 1;
    trackitDead = 1;
    %preallocate for speed
    RoundDead = zeros(1, deadholder);
    RoundDeadx = zeros(1, deadholder);
    RoundDeady = zeros(1, deadholder);
    EnergyDead = zeros(1, deadholder);
    RoundAlive = zeros(1, n - deadholder);
    RoundAlivex = zeros(1, n - deadholder);
    RoundAlivey = zeros(1, n - deadholder);
    EnergyAlive = zeros(1, n - deadholder);
    RoundCHx = [];
    RoundCHy = [];
    RoundCHs = [];
    NormalNode = [];
    NormalNodex = [];
    NormalNodey = [];

    for hh = 1:n
        if S(hh).E > 0;
            RoundAlive(trackitAlive) = hh;
            RoundAlivex(trackitAlive) = XR(hh);
            RoundAlivey(trackitAlive) = YR(hh);
            EnergyAlive(trackitAlive) = S(hh).E;
            trackitAlive = trackitAlive +1;
        end

        if S(hh).E <=0;
            RoundDead(trackitDead)= hh;
            RoundDeadx(trackitDead)= XR(hh);
            RoundDeady(trackitDead)= YR(hh);
            EnergyDead(trackitDead) = 0;
            trackitDead = trackitDead + 1;
        end
    end

    %3D Stem Energy Plot
    subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
    stem3(RoundAlivex, RoundAlivey, EnergyAlive, 'Fill', 'g', 'LineStyle','--');
    hold on;
    if trackitDead > 1;
        stem3(RoundDeadx, RoundDeady, EnergyDead, 'Fill', 'r', 'LineStyle','--');
    end
    perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
        vertRightX, vertRightY);
    set(perimeter,'Color','r','LineWidth',1);
    for bb = 1:z-1;
        vertX = [xm/z*bb, xm/z*bb];
        vertY = [0 , ym];
        line(vertX,vertY, 'Color', 'r');
    end
    xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    zlabel('Energy (J)','FontSize',12,'FontWeight','bold');
    set(get(gca,'xlabel'),'rotation',14);
    set(get(gca,'ylabel'),'rotation',338);
    axis([-1 xm+1 -1 ym+1 0 Eo]);
    grid off;
    hold off;
    drawnow

    subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
    plot(XR,YR, 'o');
    hold on;
    text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
    text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
    plot(ClusterHeadsX,ClusterHeadsY, 'k *');
    if deadholder > 0;
        plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
    end
    plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor','g',...
    'MarkerSize',10);
    plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor','g',...
        'MarkerSize',10);
    perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
        vertRightX, vertRightY);
    axis([-5 xm+5 sink.y1-5 sink.y2+5]);
    set(perimeter,'Color','r','LineWidth',1);
    for bb = 1:z-1;
        vertX = [xm/z*bb, xm/z*bb];
        vertY = [0 , ym];
        line(vertX,vertY, 'Color', 'r');
    end
    xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    hold off
    drawnow


    %turn every 5th frame into a movie
    if mod(round,5)==0;
        MOVIE_DIEOUT_ZONE_M = getframe(figure(1));
        writeVideo(myObj,MOVIE_DIEOUT_ZONE_M);
    end


    %find round first node dead
    if(flag_first_dead == 0);
        if (deadholder >= 1);
            flag_first_dead = 1;
            ROUND_FIRST_DEAD_ZONE_M = round;
            fig1dead = figure(8);
            set(fig1dead, 'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, ...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1)
            for bb = 1:z-1;
                vertX = [xm/z*bb, xm/z*bb];
                vertY = [0 , ym];
                line(vertX,vertY, 'Color', 'r');
            end
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig1dead,'1_Node_Dead_grid_ZONE_M.bmp');
            saveas(fig1dead,'1_Node_Dead_grid_ZONE_M');
            saveas(fig1dead,'1_Node_Dead_Grid_Energy_ZONE_M');
            saveas(fig1dead,'1_Node_Dead_Grid_Energy_ZONE_M.bmp');
        end
    end

    %find the round when 10% of nodes are dead and save network figures
    if(flag_10P_dead == 0);
        if (deadholder >= 0.1*n);
            flag_10P_dead = 1;
            ROUND_10P_DEAD_ZONE_M = round;
            fig10Pdead = figure(9);
            set(fig10Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            for bb = 1:z-1;
                vertX = [xm/z*bb, xm/z*bb];
                vertY = [0 , ym];
                line(vertX,vertY, 'Color', 'r');
            end
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig10Pdead,'10P_Node_Dead_grid_ZONE_M.bmp');
            saveas(fig10Pdead,'10P_Node_Dead_grid_ZONE_M');
            saveas(fig10Pdead,'10P_Node_Dead_Grid_Energy_ZONE_M');
            saveas(fig10Pdead,'10P_Node_Dead_Grid_Energy_ZONE_M.bmp');
        end
    end

    %find round when 50% of nodes are dead
    if(flag_50P_dead == 0);
        if (deadholder >= 0.5*n);
            flag_50P_dead = 1;
            ROUND_50P_DEAD_ZONE_M = round;
            fig50Pdead = figure(10);
            set(fig50Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, ...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            for bb = 1:z-1;
                vertX = [xm/z*bb, xm/z*bb];
                vertY = [0 , ym];
                line(vertX,vertY, 'Color', 'r');
            end
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig50Pdead,'50P_Node_Dead_grid_ZONE_M.bmp');
            saveas(fig50Pdead,'50P_Node_Dead_grid_ZONE_M');
            saveas(fig50Pdead,'50P_Node_Dead_Grid_Energy_ZONE_M');
            saveas(fig50Pdead,'50P_Node_Dead_Grid_Energy_ZONE_M.bmp');
        end
    end

    %find round when 80% of nodes are dead and save network figure
    if(flag_80P_dead == 0);
        if (deadholder >= 0.8*n);
            flag_80P_dead = 1;
            ROUND_80P_DEAD_ZONE_M = round;
            fig80Pdead = figure(11);
            set(fig80Pdead, 'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold');
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            for bb = 1:z-1;
                vertX = [xm/z*bb, xm/z*bb];
                vertY = [0 , ym];
                line(vertX,vertY, 'Color', 'r');
            end
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig80Pdead,'80P_Node_Dead_grid_ZONE_M.bmp');
            saveas(fig80Pdead,'80P_Node_Dead_grid_ZONE_M');
            saveas(fig80Pdead,'80P_Node_Dead_Grid_Energy_ZONE_M');
            saveas(fig80Pdead,'80P_Node_Dead_Grid_Energy_ZONE_M.bmp');
        end
    end

    if deadholder == n;
        break;
    end
end

close(myObj);

ALIVE_ZONE_M = zeros(1,round);
for ii = 1:round
    ALIVE_ZONE_M(ii) = n - DEAD_ZONE_M(ii);
end

RoundDeadStats= [ROUND_FIRST_DEAD_ZONE_M ROUND_10P_DEAD_ZONE_M ...
    ROUND_50P_DEAD_ZONE_M ROUND_80P_DEAD_ZONE_M];


fig2 = figure(2);
set(fig2,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ALIVE_ZONE_M, 'LineWidth', 2);
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Number of Nodes Alive','FontSize',12,'FontWeight','bold');
ylim([0 n+1]);
saveas(figure(2), 'NodesAliveVsRound_ZONE_M');
saveas(figure(2), 'NodesAliveVsRound_ZONE_M.bmp');

fig3 = figure(3);
set(fig3,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_ZONE_M(1:round), 'LineWidth', 2);
hold on;
plot(RoundDeadStats(1), ENERGY_ZONE_M(RoundDeadStats(1)), 'p',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), ENERGY_ZONE_M(RoundDeadStats(2)), 'd',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_ZONE_M(RoundDeadStats(3)), 's',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_ZONE_M(RoundDeadStats(4)), '^',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Total System Energy (J)','FontSize',12,'FontWeight','bold');
leg = legend('Total System Energy','1st Node Dead', '10% Nodes Dead',...
    '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
hold off;
ylim([-0.1 Eo*n+1]);
saveas(figure(3), 'ENERGY_Zone_M');
saveas(figure(3), 'ENERGY_Zone_M.bmp');
```

```
fig4 = figure(4);
set(fig4,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_VARIANCE_ZONE_M(1:round), 'LineWidth', 2);
hold on;
plot(RoundDeadStats(1), ENERGY_VARIANCE_ZONE_M(RoundDeadStats(1)), 'p',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), ENERGY_VARIANCE_ZONE_M(RoundDeadStats(2)), 'd',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_VARIANCE_ZONE_M(RoundDeadStats(3)), 's',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_VARIANCE_ZONE_M(RoundDeadStats(4)), '^',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Energy Variance','FontSize',12,'FontWeight','bold');
leg = legend('Variance of Energy Disribution','1st Node Dead',...
    '10% Nodes Dead', '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
hold off;
saveas(figure(4), 'ENERGY_VARIANCE_ZONE_M');
saveas(figure(4), 'ENERGY_VARIANCE_ZONE_M.bmp');

fig5 = figure(5) ;
set(fig5,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(CLUSTERHS_ZONE_M(1:round), 'LineWidth', 2);
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Number Of Cluster Heads','FontSize',12,'FontWeight','bold');
hold off;
saveas(figure(5), 'ClusterheadsPerRound_ZONE_M');
saveas(figure(5), 'ClusterheadsPerRound_ZONE_M.bmp');

save('ZONE_M_DATA')
```

Published with MATLAB® R2013a

8. Zone Clustering with Energy Efficient CH Election—Multi-gateway

```
% User identified Multi Gateway Zone Protocol with Energy Efficient CH Election

clc;
clear all;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Field Dimensions - x and y maximum (in meters)
```

```
xm=50;
ym=50;

x and y Coordinates of Sink1
sink.x1 = 25;
sink.y1 = -100;

x and y Coordinates of Sink 2
sink.x2 = 25;
sink.y2 = 150;

Packet size in bits
L = 2000;

Number of Nodes in the field
n = 100;

Number to zones to subdivide the field
z = 5;

Energy Model (all values in Joules)
Initial Energy
Eo=0.5;
Eelec=Etx=Erx
ETX=50*0.000000001;
ERX=50*0.000000001;
Transmit Amplifier types
Efs=10*0.000000000001;
Emp=0.0013*0.000000000001;
Data Aggregation Energy
EDA=5*0.000000001;

maximum number of rounds
rmax= 9999;

%%%%%%%%%%%%%%%%%%%%%%         END OF PARAMETERS      %%%%%%%%%%%%%%%%%%%%%
%%%%%%   **SUPPRESS ABOVE IF STARTING WITH COMMON WSN PARAMETERS**   %%%%%%

%Computation of do
do=sqrt(Efs/Emp);

%Get the screensize so each figure can be normalized in a similar manner
%for thesis writeup
scrsz = get(0,'ScreenSize');

%Creation of the random Sensor Network
fig = figure(1);
set(fig, 'Position',[1 scrsz(4)*.25 scrsz(3)*.7 scrsz(4)*.75]);

subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
hold on;
```

```
for i=1:n
    S(i).xd = SensorX(i);
    %     S(i).xd = rand(1,1)*xm;
    XR(i)=S(i).xd; %copy of the x-coordinate outside of the S struct
    S(i).yd = SensorY(i);
    %     S(i).yd = rand(1,1)*ym;
    YR(i)=S(i).yd; %copy of the y-coordinate outside of the S struct
    S(i).E=Eo;
    S(i).totalAsCH = 0;
    plot(S(i).xd,S(i).yd,'o');
end

xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');

%plot the sink
S(n+1).xd1=sink.x1;
S(n+1).yd1=sink.y1;
S(n+2).xd2=sink.x2;
S(n+2).yd2=sink.y2;
plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
    'g','MarkerSize',10);
plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
    'g','MarkerSize',10)
axis([-5 xm+5 sink.y1-5 sink.y2+5])

%plot horizontal boundaries of the sensor field
bottomY=[0,0];
bottomX=[0,xm];
topY = [ym,ym];
topX = [0,xm];

%plot vertical extremes of the sensor field
vertLeftX=[0,0];
vertLeftY=[0,ym];
vertRightX=[xm,xm];
vertRightY=[0,ym];
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
axis([-5 xm+5 sink.y1-5 sink.y2+5]);
set(perimeter,'Color','r','LineWidth',1);

%plot the vertical zone partitions
for bb = 1:z-1;
    vertX = [xm/z*bb, xm/z*bb];
    vertY = [0 , ym];
    line(vertX,vertY, 'Color', 'r');
end
hold off;

%start the plot for the node energy bar graph
energyBar = zeros(1, n);
```

```
for iii = 1:n
    energyBar(iii) = S(iii).E;
end

subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
bar(energyBar);
ylim([0 0.51]);
xlim([-10 110]);
xlabel('Node Number','FontSize',12,'FontWeight','bold');
ylabel('Energy (J)','FontSize',12,'FontWeight','bold');

%Start the plot for 3D Energy Stem
subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
stem3(XR, YR, energyBar, 'Fill', 'g');
hold on;
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
set(perimeter,'Color','r','LineWidth',1);
for bb = 1:z-1;
    vertX = [xm/z*bb, xm/z*bb];
    vertY = [0 , ym];
    line(vertX,vertY, 'Color', 'r');
end
axis([-1 xm+1 -1 ym+1 0 Eo]);
xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
zlabel('Energy (J)','FontSize',12,'FontWeight','bold');
set(get(gca,'xlabel'),'rotation',14);
set(get(gca,'ylabel'),'rotation',338);
grid off;
hold off;


%Partition the field into zones, assign each point a zone starting from x =
%0 and increasing to x max.  Zones are verticle zones in the field.
for dd = 1:n;
    S(dd).zone = ceil(S(dd).xd/(xm/z)); %each zone only corresponds to its x coordinate
end

%Set flags, preallocate arrays for speed

ENERGY_E_ZONE_M = zeros(1, rmax);
ENERGY_VARIANCE_E_ZONE_M = zeros(1, rmax);
DEAD_E_ZONE_M = zeros(1, rmax);
CLUSTERHS_E_ZONE_M = zeros(1, rmax);
flag_first_dead = 0;
flag_10P_dead = 0;
flag_50P_dead = 0;
flag_80P_dead = 0;
roundString = '1';

myObj=VideoWriter('MOVIE_DIEOUT_E_ZONE_M.avi');
```

```
open(myObj);

for round = 1:rmax; % Perform zone based CH rotation every round
    round

    for ee=1:n;
        if (S(ee).E <= 0);
            S(ee).type = 'D';
        end
        if (S(ee).E > 0);
            S(ee).type='N';
        end
    end

    %Initialize arrays
    ClusterHeads = [];
    ClusterHeadsX = [];
    ClusterHeadsY = [];

    clustersHeadsPerRound = 0;

    for jj = 1:z; %do it for every zone

        %initialize arrays & variables that are used each round:
        EnergyOfNodesInZone = [];
        zone = [];
        index = 1;
        zoneCHx = 0;
        zoneCHy = 0;
        distance = 0;

        for kk = 1:n; %first lets separate the nodes into their zones if
            %they have energy
            if(S(kk).zone == jj) && (S(kk).E > 0); %only nodes that have
                %energy can be included in all calculations
                zone(index) = kk;
                EnergyOfNodesInZone(index)=S(kk).E;
                index = index + 1;
            end
        end

        %now there is a vector “zone” with the id's of members S in the zone
        %randomly select a node from from the “zone” array to be a CH

        %randomly select an array position in zone then identify what node
        %was selected
        if (length(zone)>1); %if (length(zone)>1) is required for when there
            %is only 1 node left in a zone

            [maxEinZone,IndexOfMax] = max(EnergyOfNodesInZone);

            zoneCH = zone(IndexOfMax); %gives the original index of the CH
        else
            zoneCH = zone; %there is only one node left in the zone so it must be a CH
        end

        if (~isempty(zoneCH));

            clustersHeadsPerRound = clustersHeadsPerRound + 1;

            %if inside this if statement, that means there is at least one
            %node in the current zone
            %this must be in an if statement becasue otherwise there would
            %be errors as one zone completely dies out but other zones have
            %not. Thus if a zone has no remaining nodes with energy, it
            %simply bypasses to the next zone

            S(zoneCH).totalAsCH = S(zoneCH).totalAsCH + 1;
            S(zoneCH).type = 'C';
            zoneCHx = S(zoneCH).xd;
            zoneCHy = S(zoneCH).yd;
            ClusterHeads(jj) = zoneCH;
            ClusterHeadsX(jj)= zoneCHx;
            ClusterHeadsY(jj)= zoneCHy;

            %Identify the smaller distance to the Sink 1 or 2;
            distance = min(sqrt((zoneCHx-S(n+1).xd1)^2 + ...
                (zoneCHy-S(n+1).yd1)^2), sqrt((zoneCHx-S(n+2).xd2)^2 +...
                (zoneCHy-S(n+2).yd2)^2));

            %energy cost for the CH in the zone to aggregate its own sensor data and and
            %transmit the message to the basestation
            if (distance > do);
                S(zoneCH).E=S(zoneCH).E - ( (ETX+EDA)*(L) + Emp*L*( distance^4 ));
            end
            if (distance <= do);
                S(zoneCH).E=S(zoneCH).E - ( (ETX+EDA)*(L)  + Efs*L*( distance^2 ));
            end

            %the total number of nodes under the CH is length(zone)-1
            %then the CH will recieve length(zone)-1 messages

            %energy cost for the CH to recieve and aggregate messages from its
            %nodes.
            S(zoneCH).E = S(zoneCH).E - (ERX+EDA)*L*(length(zone)-1);

            %Now, for each node in the zone, except for the zone's CH,
            %Let's deduct energy cost for each node to send the message to
            %the CH

            %Iterate through each node in the zone except for the
            %cluster head node.
            %Use the “zone” array since it is a vector for each node in
            %the zone (but dont include the CH)
            for gg = zone;
                %for each iteration, gg represents the ID of the node we are
                %ranging to the CH

                if (gg ~= zoneCH);
                    %calculate the distance to the CH
                    distance = sqrt( (S(gg).xd - zoneCHx)^2 + (S(gg).yd - zoneCHy)^2 );

                    %Energy cost to transmit L bits to CH
                    if (distance > do); %Multipath Propagation
                        S(gg).E = S(gg).E - (ETX*L + Emp*L*(distance^4));
                    end
                    if (distance <= do);%Direct path propagation
                        S(gg).E = S(gg).E - (ETX*L + Efs*L*(distance^2));
                    end
                end %gg~= zoneCH
            end%gg=zone
        end %(~isempty(zoneCH))
    end

    CLUSTERHS_E_ZONE_M(round) = clustersHeadsPerRound;

    %obtain the energy variance and determine dead nodes
    energyBar = zeros(1, n);
    deadholder = 0;
    for www = 1:n;
        energyBar(www) = S(www).E;
        if S(www).E <=0;
            deadholder = deadholder+1;
        end
    end
    DEAD_E_ZONE_M(round)=deadholder;

    ENERGY_VARIANCE_E_ZONE_M(round) = var(energyBar);
    ENERGY_E_ZONE_M(round)=sum(energyBar);

    %plot a running bar chart of energy for animation
    figure(1);
    subplot 221; subplot('Position',[0.06 0.55 0.42 0.41]);
    bar(energyBar);
    ylim([0 0.51]);
    xlim([-10 110]);
    xlabel('Node Number','FontSize',12,'FontWeight','bold');
    ylabel('Energy (J)','FontSize',12,'FontWeight','bold');
    drawnow;

    % Save the round to a string for display on plots
    roundString = num2str(round);

    trackitAlive = 1;
    trackitDead = 1;
```

```
%preallocate for speed
RoundDead = zeros(1, deadholder);
RoundDeadx = zeros(1, deadholder);
RoundDeady = zeros(1, deadholder);
EnergyDead = zeros(1, deadholder);
RoundAlive = zeros(1, n - deadholder);
RoundAlivex = zeros(1, n - deadholder);
RoundAlivey = zeros(1, n - deadholder);
EnergyAlive = zeros(1, n - deadholder);
RoundCHx = [];
RoundCHy = [];
RoundCHs = [];
NormalNode = [];
NormalNodex = [];
NormalNodey = [];

for hh = 1:n
    if S(hh).E > 0;
        RoundAlive(trackitAlive) = hh;
        RoundAlivex(trackitAlive) = XR(hh);
        RoundAlivey(trackitAlive) = YR(hh);
        EnergyAlive(trackitAlive) = S(hh).E;
        trackitAlive = trackitAlive +1;
    end

    if S(hh).E <=0;
        RoundDead(trackitDead)= hh;
        RoundDeadx(trackitDead)= XR(hh);
        RoundDeady(trackitDead)= YR(hh);
        EnergyDead(trackitDead) = 0;
        trackitDead = trackitDead + 1;
    end
end

%3D Stem Energy Plot
subplot 223; subplot('Position',[0.06 0.06 0.42 0.41]);
stem3(RoundAlivex, RoundAlivey, EnergyAlive, 'Fill', 'g', 'LineStyle','--');
hold on;
if trackitDead > 1;
    stem3(RoundDeadx, RoundDeady, EnergyDead, 'Fill', 'r', 'LineStyle','--');
end
perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
    vertRightX, vertRightY);
set(perimeter,'Color','r','LineWidth',1);
for bb = 1:z-1;
    vertX = [xm/z*bb, xm/z*bb];
    vertY = [0 , ym];
    line(vertX,vertY, 'Color', 'r');
end
xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
zlabel('Energy (J)','FontSize',12,'FontWeight','bold');
```

```
    set(get(gca,'xlabel'),'rotation',14);
    set(get(gca,'ylabel'),'rotation',338);
    axis([-1 xm+1 -1 ym+1 0 Eo]);
    grid off;
    hold off;
    drawnow

    subplot(2,2,[2 4]); subplot('Position',[0.55 0.06 0.42 0.9]);
    plot(XR,YR, 'o');
    hold on;
    text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
    text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
    plot(ClusterHeadsX,ClusterHeadsY, 'k *');
    if deadholder > 0;
        plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
    end
    plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
        'g','MarkerSize',10);
    plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
        'g','MarkerSize',10);
    perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, vertLeftY,...
        vertRightX, vertRightY);
    axis([-5 xm+5 sink.y1-5 sink.y2+5]);
    set(perimeter,'Color','r','LineWidth',1);
    for bb = 1:z-1;
        vertX = [xm/z*bb, xm/z*bb];
        vertY = [0 , ym];
        line(vertX,vertY, 'Color', 'r');
    end
    xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
    hold off
    drawnow

    %turn every 5th frame into a movie
    if mod(round,5)==0;
        MOVIE_DIEOUT_E_ZONE_M = getframe(figure(1));
        writeVideo(myObj,MOVIE_DIEOUT_E_ZONE_M);
    end

    %find round first node dead
    if(flag_first_dead == 0);
        if (deadholder >= 1);
            flag_first_dead = 1;
            ROUND_FIRST_DEAD_E_ZONE_M = round;
            fig1dead = figure(8);
            set(fig1dead, 'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1)
            for bb = 1:z-1;
                vertX = [xm/z*bb, xm/z*bb];
                vertY = [0 , ym];
                line(vertX,vertY, 'Color', 'r');
            end
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig1dead,'1_Node_Dead_grid_E_ZONE_M.bmp');
            saveas(fig1dead,'1_Node_Dead_grid_E_ZONE_M');
            saveas(figure(1),'1_Node_Dead_Grid_Energy_E_ZONE_M');
            saveas(figure(1),'1_Node_Dead_Grid_Energy_E_ZONE_M.bmp');
        end
    end

    %find the round when 10% of nodes are dead and save network figures
    if(flag_10P_dead == 0);
        if (deadholder >= 0.1*n);
            flag_10P_dead = 1;
            ROUND_10P_DEAD_E_ZONE_M = round;
            fig10Pdead = figure(9);
            set(fig10Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX, ...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            for bb = 1:z-1;
                vertX = [xm/z*bb, xm/z*bb];
                vertY = [0 , ym];
                line(vertX,vertY, 'Color', 'r');
            end
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig10Pdead,'10P_Node_Dead_grid_E_ZONE_M.bmp');
            saveas(fig10Pdead,'10P_Node_Dead_grid_E_ZONE_M');
            saveas(figure(1),'10P_Node_Dead_Grid_Energy_E_ZONE_M');
            saveas(figure(1),'10P_Node_Dead_Grid_Energy_E_ZONE_M.bmp');
        end
    end

    %find round when 50% of nodes are dead
    if(flag_50P_dead == 0);
        if (deadholder >= 0.5*n);
            flag_50P_dead = 1;
            ROUND_50P_DEAD_E_ZONE_M = round;
            fig50Pdead = figure(10);
            set(fig50Pdead,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            for bb = 1:z-1;
                vertX = [xm/z*bb, xm/z*bb];
                vertY = [0 , ym];
                line(vertX,vertY, 'Color', 'r');
            end
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig50Pdead,'50P_Node_Dead_grid_E_ZONE_M.bmp');
            saveas(fig50Pdead,'50P_Node_Dead_grid_E_ZONE_M');
            saveas(figure(1),'50P_Node_Dead_Grid_Energy_E_ZONE_M');
            saveas(figure(1),'50P_Node_Dead_Grid_Energy_E_ZONE_M.bmp');
        end
    end

    %find round when 80% of nodes are dead and save network figure
    if(flag_80P_dead == 0);
        if (deadholder >= 0.8*n);
            flag_80P_dead = 1;
            ROUND_80P_DEAD_E_ZONE_M = round;
            fig80Pdead = figure(11);
            set(fig80Pdead, 'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
            set(gcf,'Units','normal');
            set(gca,'Position',[.06 .06 .9 .9]);
            plot(XR,YR, 'o');
            hold on;
            text(xm-10,-10,'Round','FontSize',12,'FontWeight','bold')
            text(xm-10,-17,roundString,'FontSize',12,'FontWeight','bold');
            plot(ClusterHeadsX,ClusterHeadsY, 'k *');
            if deadholder > 0;
                plot(RoundDeadx, RoundDeady, 'r .', 'MarkerSize', 20);
            end
            plot(sink.x1,sink.y1,'o', 'MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10)
            plot(sink.x2,sink.y2,'o','MarkerEdgeColor','k','MarkerFaceColor',...
                'g','MarkerSize',10);
            perimeter = plot(bottomX,bottomY,topX, topY, vertLeftX,...
                vertLeftY, vertRightX, vertRightY);
            axis([-5 xm+5 sink.y1-5 sink.y2+5]);
            set(perimeter,'Color','r','LineWidth',1);
            for bb = 1:z-1;
                vertX = [xm/z*bb, xm/z*bb];
                vertY = [0 , ym];
                line(vertX,vertY, 'Color', 'r');
            end
            xlabel('x-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            ylabel('y-Grid-Axis (m)','FontSize',12,'FontWeight','bold');
            hold off;
            drawnow;
            saveas(fig80Pdead,'80P_Node_Dead_grid_E_ZONE_M.bmp');
            saveas(fig80Pdead,'80P_Node_Dead_grid_E_ZONE_M');
            saveas(figure(1),'80P_Node_Dead_Grid_Energy_E_ZONE_M');
            saveas(figure(1),'80P_Node_Dead_Grid_Energy_E_ZONE_M.bmp');
        end
    end

    if deadholder == n;
        break;
    end
end
```

```
close(myObj);

ALIVE_E_ZONE_M = zeros(1,round);
for ii = 1:round
    ALIVE_E_ZONE_M(ii) = n - DEAD_E_ZONE_M(ii);
end

RoundDeadStats= [ROUND_FIRST_DEAD_E_ZONE_M ROUND_10P_DEAD_E_ZONE_M ...
    ROUND_50P_DEAD_E_ZONE_M ROUND_80P_DEAD_E_ZONE_M]

fig2 = figure(2);
set(fig2,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ALIVE_E_ZONE_M, 'LineWidth', 2);
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Number of Nodes Alive','FontSize',12,'FontWeight','bold');
ylim([0 n+1]);
saveas(figure(2), 'NodesAliveVsRound_E_ZONE_M');
saveas(figure(2), 'NodesAliveVsRound_E_ZONE_M.bmp');

fig3 = figure(3);
set(fig3,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_E_ZONE_M(1:round), 'LineWidth', 2);
hold on
plot(RoundDeadStats(1), ENERGY_E_ZONE_M(RoundDeadStats(1)), 'p',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(2), ENERGY_E_ZONE_M(RoundDeadStats(2)), 'd',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_E_ZONE_M(RoundDeadStats(3)), 's',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_E_ZONE_M(RoundDeadStats(4)), '^',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Total System Energy (J)','FontSize',12,'FontWeight','bold');
leg = legend('Total System Energy','1st Node Dead', '10% Nodes Dead',...
    '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
ylim([-0.1 Eo*n+1]);
hold off;
saveas(figure(3), 'ENERGY_E_ZONE_M');
saveas(figure(3), 'ENERGY_E_ZONE_M.bmp');

fig4 = figure(4);
set(fig4,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(ENERGY_VARIANCE_E_ZONE_M(1:round), 'LineWidth', 2);
hold on;
plot(RoundDeadStats(1), ENERGY_VARIANCE_E_ZONE_M(RoundDeadStats(1)), 'p',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
```

```
plot(RoundDeadStats(2), ENERGY_VARIANCE_E_ZONE_M(RoundDeadStats(2)), 'd',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(3), ENERGY_VARIANCE_E_ZONE_M(RoundDeadStats(3)), 's',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
plot(RoundDeadStats(4), ENERGY_VARIANCE_E_ZONE_M(RoundDeadStats(4)), '^',...
    'MarkerSize', 15, 'MarkerEdgeColor','k','MarkerFaceColor','c');
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Energy Variance (J^2)','FontSize',12,'FontWeight','bold');
leg = legend('Variance of Energy Disribution','1st Node Dead',...
    '10% Nodes Dead', '50% Nodes Dead', '80% Nodes Dead');
set(leg,'FontWeight','bold');
hold off;
saveas(figure(4), 'ENERGY_VARIANCE_E_ZONE_M');
saveas(figure(4), 'ENERGY_VARIANCE_E_ZONE_M.bmp');

fig5 = figure(5) ;
set(fig5,'Position',[1 scrsz(4)*.26 scrsz(3)*.7 scrsz(4)*.75]);
plot(CLUSTERHS_E_ZONE_M(1:round), 'LineWidth', 2);
set(gcf,'Units','normal');
set(gca,'Position',[.06 .06 .9 .9]);
xlabel('Round','FontSize',12,'FontWeight','bold');
ylabel('Number Of Cluster Heads','FontSize',12,'FontWeight','bold');
set(leg,'FontWeight','bold');
hold off;
saveas(figure(5), 'ClusterheadsPerRound_E_ZONE_M');
saveas(figure(5), 'ClusterheadsPerRound_E_ZONE_M.bmp');

save('E_ZONE_M_DATA')
```

Published with MATLAB® R2013a

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” *System Sciences, Proceedings of the 33rd Annual Hawaii International Conference*, Maui, HI, 2000, pp. 1–10.

[2] W. B. Heinzelman, A. P. Chandrakasan and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, “Wireless Sensor Networks: A Survey,” *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[4] “TinyOS.” [Online]. Available: http://www.tinyos.net/. [Accessed 4 Sept. 2013].

[5] J. Hao and H. Mouftah, “Routing protocols for duty cycled wireless snesor networks: A survey,” *IEEE Communications Magazine*, vol. 50, no. 12, pp. 116–123, 2012.

[6] C. Cirstea, “Energy efficient routing protocols for wireless sensor networks: A survey,” *Proceedings of IEEE 17th International Symposium on Design and Technology in Electronic Packaging (SIITME)*, Timişoara, Romania, 2011, pp. 277–282.

[7] A. A. Abbasi and M. Younis, “A survey on clustering algorithms for wireless sensor networks,” *Computer Communications*, vol. 30, no. 14–15, pp. 2826–2841, 2007.

[8] K. Akkaya and M. Younis, “A survey on routing protocols for wireless sensor networks,” *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.

[9] D. Goyal and M. R. Tripathy, “Routing protocols in wireless sensor networks: A survey,” *Proceedings of IEEE Second International Conference on Advanced Computing & Communication Technologies (ACCT)*, Rohtak, India, 2012, pp. 474–480.

[10] “Libelium,” Libelium Communicaciones. [Online]. Available: http://www.libelium.com/products/waspmote/. [Accessed 23 Sep. 2013].

[11] National Instruments, “NI wireless sensor network starter kit,” National Instrumetns Corporation, 2012. [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/206916. [Accessed 9 Sep. 2013].

[12] A. Bielsa, “Detecting radiation levels in Fukushima: an example of crowdsourcing,” 22 Feb. 2013. [Online]. Available:

http://www.libelium.com/fukushima_crowdsourcing_radiation_social_project/. [Accessed 23 Sep. 2013].

[13] "Smart Santander: Future internet research & experimentation." [Online]. Available: http://www.smartsantander.eu/. [Accessed 2013 Sep. 23].

[14] A. Bielsa, "Smart City Project in Santander to monitor environmental parameters," 22 February 2013. [Online]. Available: http://www.libelium.com/smart_santander_environment_smart_ctiy/. [Accessed 23 Sep.2013].

[15] A. Bielsa, "Smart Agriculture project in Galicia to monitor vineyards with Waspmote," 8 Jun. 2012. [Online]. Available: http://www.libelium.com/smart_agriculture_vineyard_sensors_waspmote/. [Accessed 23 Sep. 2013].

[16] I. Bosch, J. Lloret, S. Sendra and A. Serrano, "A wireless sensor network for vineyard monitoring that uses image processing," *Sensors*, vol. 2011, no. 11, pp. 6165–6196, 2011.

[17] T. Arampatzis and J. Lygeros, "A survey of applications of wireless sensors and wireless sensor networks," *Proceedings of the 13th Mediterranean Conference on Control and Automation*, Limassol, Cyprus, 2005, pp. 719–724.

[18] R. Merritt, "Slideshow: Imagining a trillion sensor world," *EE Times*, 28 Oct. 2013. [Online]. Available: http://www.eetimes.com/document.asp?doc_id=1319913. [Accessed 19 Nov. 2013]

[19] J. Eberspacher, C. Bettstetter, H.-J. Vogel and C. Hartman, *GSM Architecture, Protocols and Services*, 3rd ed. West Sussex: Wiley, 2009.

[20] A. A. Abidi, "Direct-conversion radio transceivers for digital communications," *IEEE Journal Of Solid State Circuits*, vol. 30, no. 12, pp. 1399–1409, 1995.

[21] W. Zou, Chen, C. Pengpeng and H. Yang, "A High-linearity Energy-Efficient CMOS PA for Wireless Environment Monitoring," *Proceedings of IEEE Fourth International Conference on Digital Manufacturing & Automation*, Qindao, Shandong, China, 2013, pp. 32–35.

[22] J. Lavaei, A. Babakhani, A. Hajimiri and J. C. Doyle, "Passively controllable smart antennas," *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2010)*, Miami, FL, 2010, pp. 1–6.

[23] I. Akyildiz, T. Melodia and K. Chowdury, "Wireless multimedia sensor networks: A survey," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 32–39, 2007.

[24] W. Ye, J. Heidemann and D. Estrin, “An energy-efficient MAC protocol for wireless sensor networks,” *Proceedsing of IEEE International Conference on Computer Communications (INFOCOM)*, New York, 2012, pp. 1567–1576.

[25] K. Jamieson, H. Balakrishnan and Y. C. Tay, “Sift: A MAC protocol for event-driven,” *Wireless Sensor Networks*, Springer Berlin Heidelberg, pp. 260–275, 2006.

[26] N. A. Pantazis, S. A. Nikolidakis and D. D. Vergados, “Energy-efficient routing protocols in wireless sensor networks: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 551–591, 2013.

[27] M. Patil and R. C. Biradar, “A survey on routing protocols in wireless sensor networks,” *Proceedings of IEEE International Conference on Networks* (ICON), Singapore, 2012, pp. 86–91.

[28] C. Wei, J. Yang and Y. Gao, “Cluster-based routing protocols in wireless sensor networks: A survey,” *Proceedings of IEEE International Conference on Computer Science and Network Technology*, Harbin, China, 2011, pp. 1659–1663.

[29] B. Puccinelli and M. Haenggi, “Arbutus: Network-layer load balancing for wireless sensor networks,” *Proceedings of Wireless Communications and Networking Conference (WCNC)*, Las Vegas, NV, 2008, pp 2063–2068.

[30] W. Stallings, *Data and Computer Communications*, 9th ed., Upper Saddle River: Prentice Hall, 2011.

[31] S. K. Singh, M. P. Singh and D. K. Singh, “Energy efficient homogenous clustering algorithm for wireles sensor networks,” *International Journal of Wireless Mobile Networks*, vol. 2, no. 3, pp. 49–61, 2010.

[32] J. Postal, “RFC 792: Internet control message protocol,” San Diego, CA: InterNet Network Working Group, 1981.

[33] C.-W. Hsu, C.-S. Shieh and W. K. Lai, “A multi-path routing protocol with reduced control messages for wireless sensor networks,” *Intelligent Information Hiding and Multimedia Signal Processing*, pp. 671–675, 2007.

[34] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan and D. Estrin, “A wireless sensor network for structural monitoring,” *Proceedings of ACM International conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD,2004, pp. 13–24.

[35] W. Chao, Z. Xingming, C. Wenping and N. Xiaona, “Load balancing algorithm using flow chopping to avoid packet reordering,” *International Forum On Information Technology and Applications*, Chengdu, China, 2009, pp. 193–197.

[36] M. Li and Y. Jing, "Feedback congestion control protocol for wireless sensor networks," *Proceedings of IEEE Chinese Control and Decision Conference*, Taiyuan, China, 2012, pp. 4217–4220.

[37] M. Hefeida, T. Canli, A. Kshemkalyani and A. Khokhar, "Context modeling in collaborative sensor network applications," *Proceedings of the International Conference on Collaboration Technologies and Systems*, Philadelphia, PA, 2011 pp. 274-279.

[38] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Proceedings of ACM International Conference on Mobility Computing and Networking*, Boston, MA, 2000, pp. 56 –67.

[39] F. Ye, H. Luo, J. Cheng, S. Lu and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," *Proceedings of ACM International Conference on Mobility Computing and Networking (MobiCom)*, Atlanta, GA, 2012, pp. 148–159.

[40] G. Smaragdakis, I. Matta and A. Bestavros, "SEP: A stable election protocol for clustered heterogeneous wireless sensor networks," *Boston University Computer Science Department*, Boston, MA, 2004.

[41] A. Hayter, *Probability and Statistics for Engineers and Scientists*, 4th ed., Boston: Brooks/Cole, 2012.

[42] J. Lessman, P. Janacik, L. Lachev and D. Orfanus, "Comparative study of wireless network simulators," *Proceedings of IEEE International Conference on Networking (ICN)*, Cancun, Mexico, 2008, pp. 517–523.

[43] D. Orfanus, J. Lessmann, P. Janacik and L. Lachev, "Performance of wireless network simulators: a case study," *Proceedings of ACM 3rd workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, Vancouver, Canada, 2008, pp. 59–66.

[44] L. Hogie, P. Bouvry and F. Guinand, "An overview of MANETs simulation," *Electronic notes in theoretical computer science 150*, no. 1, vol. 150, no. 1, pp. 81–101, 2006.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California