Distributed Algorithms for Tasking Large Sensor Networks

Shashank Mehrotra

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

July 10, 2001
Blacksburg, Virginia

Keywords: Distributed Algorithms, Distributed Sensor Networks, Energy Efficient Protocols, Wireless Micro-Sensors

Distributed Algorithms for Tasking Large Sensor Networks

Shashank Mehrotra

Dr. Mark T. Jones, Chair

The Bradley Department of Electrical and Computer Engineering

(Abstract)

Recent advances in wireless communications along with developments in low-power circuit design and micro-electro mechanical systems (MEMS) have heralded the advent of compact and inexpensive wireless micro-sensor devices. A large network of such sensor nodes capable of communicating with each other provides significant new capabilities for automatically collecting and analyzing data from physical environments.

A notable feature of these networks is that more nodes than are strictly necessary may be deployed to cover a given region. This permits the system to provide reliable information, tolerate many types of faults, and prolong the effective service time. Like most wireless systems, achieving low power consumption is a key consideration in the design of these networks. This thesis presents algorithms for managing power at the distributed system level, rather than just at the individual node level. These distributed algorithms allocate work based on user requests to the individual sensor nodes that comprise the network. The primary goal of the algorithms is to provide a robust and scalable approach for tasking nodes that prolongs the effective life of the network.

Theoretical analysis and simulation results are presented to characterize the behavior of these algorithms. Results obtained from simulation experiments indicate that the algorithms can achieve a significant increase in the life of the network. In some cases this may be by an order of magnitude. The algorithms are also shown to ensure a good quality of sensor coverage while improving the network life. Finally, they are shown to be robust to faults and scale to large numbers of nodes.

Dédié mes parents, pour leur confiance et leur patience

Acknowledgments

I have often heard that writing the acknowledgements is one the harder yet more fulfilling parts of writing a thesis. I undertake this task with the hope I can adequately express my appreciation for all the assistance and good will I have received while conducting this research.

I first met Dr. Mark Jones during my early days at Virginia Tech when I was struggling to define research goals and a direction for my graduate work. Two years later, I am still wondrous of his patience, wit, and ability to make complex concepts look easy (when they really are not!). His insightful comments and encouraging words have helped me navigate my way through the endless maze that is graduate research. I thank him for all the motivation and also the helpful hints on comma usage! I will cherish my association with him.

I want to thank Dr Peter Athanas for all his encouragement and support. I will remember most his ability to listen keenly and his support of me in times of personal turmoil.

I want to express my appreciation to Dr. Scott Midkiff for serving on my committee. In the relatively little I learnt of him as an instructor I found him to be a helpful and meticulous person. His high academic standards have motivated me through many nights spent working on class projects!

I am grateful for all the help and ideas provided by Dr. Jae Hong Park on the DSN team. He has assisted me immensely while the foundation for this work was being created; many ideas and concepts in this thesis were developed and implemented after brainstorming sessions with him. Interacting with him has been an enjoyable and learning experience.

Jonathan Scalera a.k.a. 'F/S' has been a good friend and a great help while writing this thesis. I will remember fondly the night-outs we both spent in the lab hunched over keyboards typing out our respective theses, and listening to WolfFM (or was it the other way around?). I wish him much success for the future.

The Configurable Computing Lab has been my second home for much of my graduate life at Virginia Tech. The people in the lab, past and present, have always been ready to offer ideas, assistance, and help with debugging problems. I have had a wonderful affiliation with them. To each one of them I express my thankfulness and best wishes for their future endeavors.

To my roommates and friends in Blacksburg I convey my gratitude. They have borne the brunt of my thesis-writing induced erratic behavior but have always remained supportive. My stay here would not have been the same without them.

Lastly, I want to thank my family near and far for inspiring, encouraging, and bribing me to complete this thesis. That I have done so is a testament to their belief.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The past few years have seen the rapid proliferation of small wireless devices for personal communication. Such devices are achieving increasing levels of connectivity with large networks such as the Internet, and among themselves as peer-to-peer networks. As an example, consider that the number of mobile phone subscribers world-wide was expected to be approximately 560 million at the end of year 2000 [1]. Alongside these recent advancements in wireless networks [2], there have been significant developments in low-power digital circuit design, sensing technology, and Micro Electro-Mechanical Systems (MEMS) [3, 4]. The amalgamation of all these technologies has sparked great interest in creating miniature units that combine physical sensing and wireless communication - effectively, a wireless micro-sensor device.

Large numbers of such *sensor nodes* can be disseminated in a region, and can automatically collect and analyze data from the physical environment. A large network of such nodes collaborating their sensing efforts offers significant new opportunities in the study, monitoring, and maintenance of physical environments.

Like most wireless systems, sensor networks must effectively manage the power consumption of individual nodes to achieve satisfactory system lifetimes. It is imperative to design low-power hardware, and power-aware operating systems so that power savings can be achieved at the node level [3]. This thesis introduces the concept of power management at the distributed system level. It contributes a set of distributed algorithms that conserve power by efficiently managing and monitoring the workload of sensor nodes. Also presented is a theoretical background and analysis of these distributed algorithms along with extensive simulation results to demonstrate their applicability for such sensor networks.

1.1 Motivation

1.1.1 What is a Distributed Sensor Network?

A Distributed Sensor Network consists of multiple sensor nodes that are capable of communicating with each other and collaborating on a common sensing goal [5]. To some extent, such a network is no different from the conventional perception of a distributed network, for example, the Internet, or a cellular phone network. Both types of networks consist of nodes which can talk to each other, which may have dissimilar capabilities, and which may be spatially separated. Notwithstanding these similarities, considerable differences exist between these two types of networks. As a result, many of the designs and protocols used in conventional networks do not apply to distributed sensor networks [6]. Although there is no unique picture of what a distributed sensor network is expected to look like, some characteristics and their implications are discussed below.

A distributed sensor network is expected to perform a very different function from traditional networks, which are typically built to transfer data between nodes. In contrast, many nodes in a sensor network combine their sensing resources locally, and then communicate the combined result to a user. The user is not concerned with making requests to individual nodes, she is more likely to want information from a region or a set of nodes [7]. In fact, the user may not even have enough information to individually address nodes.

The size of a node is an important consideration. Nodes need to have small form factors so that they may be located unobtrusively in the environment targeted for monitoring. If nodes are being dispersed in inhospitable terrain such as a forest, a rugged and robust construction is required. The restriction in size is closely related to the amount of energy available to a node. Given a small form factor, a node has a fairly limited energy budget. Because the cost of replacing the battery may be high, nodes are expected to be thrown away once their energy supply is exhausted. Once nodes die, the only way to replenish the sensing capabilities of the network is to add more nodes to cover the depleted region.

It is infeasible, if not impossible to construct a distributed sensor network such that nodes communicate via a wired medium. For most urban or non-urban applications, it is impractical to network a large number of sensor nodes with wires given the potential obstructions. Therefore, it is assumed that most sensor networks will be constructed with a wireless topology.

A distributed sensing system such as the one being described is largely a collection of unattended devices that must be self-organizing and capable of high levels of fault tolerance. Nodes may fail often, or they may die when their energy quota is expended. The system must have multiple levels of redundancy built into it to allow for these failures. This can be achieved by using more sensor nodes than is strictly necessary to cover an area. Redundancy in the location of nodes and their numbers allows for node attrition and increases the reliability of the system [8].

Finally, the configuration and topology of the sensor network may be rapidly changing in the presence of a hostile environment, a large volume of assigned work, and nodes that fail routinely. Conventional protocols may be inadequate to manage such situations; new protocols are required to deliver effective coverage and ensure longevity.

1.1.2 Scenarios of Operation for a Sensor Network

The power of a distributed sensor network lies in its ability to deliver information from far flung areas in response to questions posed by a user. These far flung areas may constitute regions that are not suitable for human exploration. Examples include dense forests, remote planetary surfaces, radioactive zones, highly polluted environments, or mine fields. Alternatively, nodes could be deployed in military scenarios to detect and monitor the movements of personnel or combat vehicles. Uses can be visualized for industry, offices, and urban scenarios. In these settings, miniature nodes may be tacked onto objects to monitor production or inventory [6].

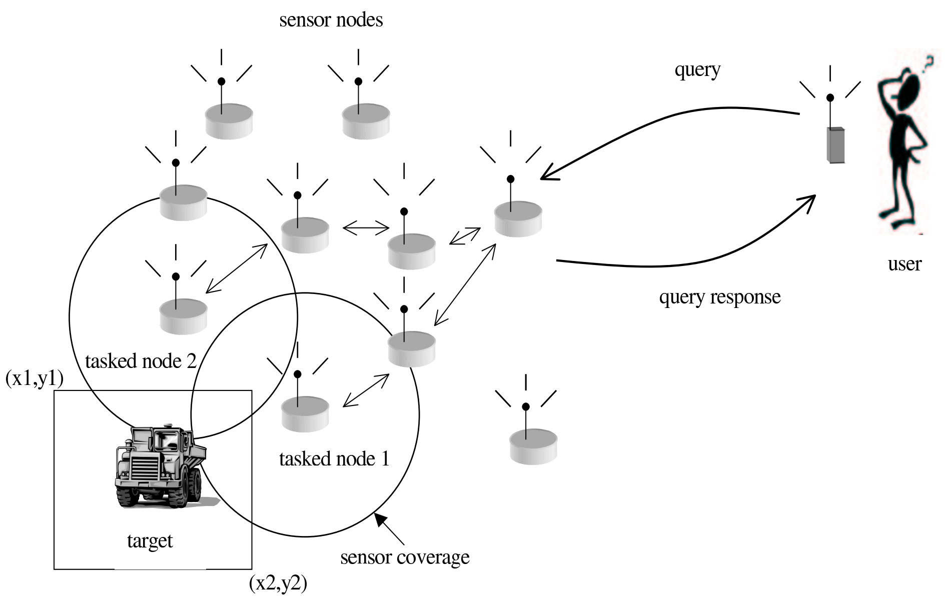


Figure 1.1: A typical scenario for operation of a sensor network.

Figure 1.1 depicts a scenario where a user poses a *query* to the sensor network. In this case the query is, "are there any tracked vehicles in the region bounded by the rectangular box [(x1, y1), (x2, y2)] ?". The query is routed to the appropriate nodes in the network, which are then *tasked* by that user query. If a vehicle is detected, then the nodes indicate a positive response which propagates back to the user. The querying approach is probably the most common mode in which such a system may be employed.

In another proposed mode of operation, nodes may lie dormant for long periods waiting for some environmental occurrence. A forest fire management system is envisaged as an example. For such a system, many thousands of nodes may be preemptively deployed in the area at risk. An increase in temperature or the CO_2 level may trigger an event management module, which sends a signal to the user. Many sensors may combine their sensing efforts to yield early information to help fight these fires. Using these networks effectively can lead to a reduction in false alarms [9].

1.1.3 Distributed Tasking Algorithms

It may be possible to have a centralized control point in the network that allocates node-level tasks, oversees load balancing, and manages the entire network. This solution, however, suffers the disadvantages posed by centralized systems, i.e., low fault tolerance, low scalability, communication bottlenecks, and the need for continuous updates [6]. Because the sensing capabilities of these sensor networks are distributed, it can be intuitively visualized that schemes for power management and control also need to be distributed in nature.

Motivated by this reasoning, a set of distributed algorithms are proposed that manage power at the system level by assigning work (user queries) to sensor nodes in such a way as to manage and balance the power consumption among the nodes. These tasking algorithms determine which subset of nodes will be tasked with a given user query, and which nodes will be switched off to conserve power. Another goal of these algorithms is to rotate the assignments of tasks across redundant nodes to prevent dead-spots in the area covered by the sensor network.

1.2 Thesis Organization

This thesis is organized as follows. Chapter 2 gives an overview of ongoing research in distributed sensor networks. Various aspects of these networks are explored to emphasize the need for energy conservation across the entire system.

In Chapter 3, a purely distributed algorithm for the selection of Application Query Servers is presented. A theoretical background of the algorithm is presented. The algorithm is discussed in the context of execution times, scalability, and faults such as node failure. Variations in the basic algorithm are proposed to make it more efficient.

Chapter 4 introduces approaches to distributed tasking of the sensor network using AQSs. These approaches attempt to (a) prolong the life of the network by efficiently utilizing the redundancy in node locations, (b) load balance the network by selecting appropriate nodes to task for each user query, and (c) ensure good quality of sensor coverage for queries.

Extensive simulations are conducted to explore the viability of these tasking algorithms. Chapter 5 presents an overview of the simulation tools, models, and methodologies. Chapter 5 also includes a discussion of two expected scenarios in which such networks may be deployed and operated. These scenarios are constructed and used in simulations as part of the experimental process.

Chapter 6 introduces metrics to evaluate the data collected in simulations. These metrics are applied to the scenarios described in Chapter 5. Results are presented along with a discussion of the observed trends. An analysis of the results obtained indicates that these distributed algorithms can increase the effective sensor network lifetime while maintaining minimal dead-spots in the area covered by sensors.

In Chapter 7, the assumptions used in constructing simulation models are validated against real system measurements and intuition about the operation of the network. These models are also verified to be an accurate representation of the system under study by performing multiple tests and checks in simulations.

Finally, Chapter 8 gives concluding remarks and outlines some possible future work in this area. This includes a discussion of the areas where the existing models need to be augmented and other ramifications of this work.

Chapter 2

Overview of Related Work

Distributed sensor networks have many operating requirements and characteristics that distinguish them from the traditional view of a wireless network. Some of these differences were outlined in Section 1.1. Many research efforts are currently underway to develop solutions and designs that best address these differences. The most stringent of these requirements is that all the hardware, software, and protocols for such networks must be designed to be *highly* energy efficient. This is the primary goal underlying most of the work outlined in this chapter.

This chapter begins by highlighting efforts to create efficient hardware implementations of a micro-sensor node. It then proceeds to analyze network layer issues, attempts at profiling energy consumption, strategies for collaborative processing of data, and optimization based techniques for the optimal placement of sensor nodes.

An understanding of the efforts being currently undertaken provides insight into designing distributed algorithms for the efficient tasking and power management of these networks.

2.1 Node and Network Architecture

The design of a sensor node is motivated by the need to create an inexpensive device with a small form factor and low power dissipation. This section outlines the efforts of various research groups in developing viable node architectures, and the corresponding perception of how these nodes operate in a network. The distributed tasking algorithms presented in this thesis draw upon the designs summarized in this section to construct simulation models. Moreover, understanding the sensing and communication capabilities of these prototype nodes allows the distributed algorithms to be more efficiently structured and fine-tuned.

The Wireless Integrated Network Sensors (WINS) project [10] at UCLA and Rockwell has led to the development of sensor nodes that integrate sensing, processing, and communication on micro-sensor platforms [11]. These sensors are fabricated using low-power wireless integrated microsensor (LWIM) technology [4], and are capable of forming self-assembling, multihop networks. A new generation of WINS nodes currently being developed by Sensoria Corporation provides a development platform to monitor and process multiple sensor signals [12, 13]. Sensor data and messages are exchanged amongst nodes via a radio frequency (RF) modem. These WINS nodes are capable of fielding seismic, acoustic, and infrared sensing elements [1]. Each of these individual sensors is capable of detecting targets at different ranges, for example, the seismic sensor can detect tracked vehicles at distances greater than it can detect personnel. This creates an important requirement for the tasking algorithms; they must be aware of the effective coverage radii of individual sensors to optimize assignment of tasks to nodes.

The Smart-Dust project [14] explores an alternative to the traditional view of radio linked sensor networks. The project uses MEMS-based sensor nodes [15] approximately a cubic millimeter in size which communicate with a base station via a free space optical link. These nodes store no more than 1 Joule of energy and exhibit power consumption at microwatt levels. Individual nodes may be scattered in a region or attached to the objects that require monitoring. A user may then communicate with these nodes using a mobile base station unit equipped with a transceiver unit.

[1]The term *sensor node* pertains to a *node* in the sensor network. This term is used interchangeably with *node*. The terms *sensing elements* or *sensors* are used for individual sensors packaged within a node

The downlink communication pattern between the base station and a node employs an external illuminating laser to download command signals to the node. The node uses the energy of the laser via a CCR (Corner Cube Retro-reflector) [16] to reflect the signal back to the base station. Analysis by the researchers shows that a range of a few hundred meters with a bit rate of a several kilobits per second is achievable.

The micro-Adaptive Multi-domain Power-aware Sensors (μAMPS) project [17] is designing power-aware nodes, software for those nodes, and protocols for communication between the nodes [7]. The hardware design of these nodes incorporates techniques such as dynamic voltage scaling to make energy-quality tradeoffs for a low processor duty cycle. Another focus of this project is the development of power management techniques at the software level by restructuring of the data processing algorithms [18]. Experimental results are presented for two common signal processing applications, finite impulse response filtering and image decoding. An implementation of the former is shown to suffer a degradation of only 10% if the available energy per sample is reduced by 50%. The modified image decoding application reconstructs an image at nearly 90% of its original quality while reducing energy consumption by 75%.

The PicoRadio project [19] aims to create a single-chip implementation of a low-power ($< 500\mu$W power dissipation), configurable radio device, and to network large numbers of such devices. This effort targets the energy efficiencies that can be realized through careful optimization of the radio network, particularly the physical, medium access control (MAC), and network layers [20].

Rockwell Science Center [21] has developed a prototype micro-sensor unit [22] with a modular design that allows the incorporation of multiple sensing modules within a node. The sensor, processor, and communication modules are mounted on circuit boards that are stacked together and connected via a system bus. The software platform consists of a real-time, multi-tasking kernel based on the MicroC/OS.

2.2 Energy Usage Characterization

Energy consumption patterns of individual nodes and of the entire network must be systematically characterized and profiled. This process yields a better understanding of where to apply tradeoffs in the design of algorithms and hardware for nodes.

Studies conducted by Katz and Stemm [23] provide information on the energy consumption of small wireless devices and network interfaces such as wireless PDAs and wireless LAN adapters. They show that the time spent in the idle state (where data is not actually being received or sent but the network interface is operational) dominates power consumption in these devices. Transport and application layer optimizations, primarily based on switching off the network interface, are proposed to minimize the power consumption. Therefore, a key observation is that nodes which are not being tasked or relaying messages must conserve resources by switching off their network interface.

Srivastava, *et al.*, classify energy consumption by introducing the concept of energy consumers and producers [24]. The processor, radio, and sensing elements of a node are the consumers, while the static battery source is the producer. The energy consumers are profiled for a set of prototype nodes, which include the node developed by Rockwell [22]. Another contribution they make is to introduce three battery models, namely, the linear model, the discharge rate dependent model, and the relaxation model. In the linear model, the battery is considered to be a bucket of energy. Energy is linearly drawn from this bucket by the energy consumers. The discharge rate dependent model considers the rate at which energy is drawn from the battery to compute the remaining battery life. At high discharge rates, the capacity of the battery is reduced. Finally, the relaxation model takes into account a phenomenon seen in real-life batteries where the battery's voltage recovers if the discharge rate is decreased.

Srivastava, *et al.*, use the information derived from their study of prototype nodes to create SensorSim [25], a simulation infrastructure built on top of ns-2, a popular discrete event simulator. Their tool specifically targets the evaluation and performance analysis of sensor networks. In addition to the capabilities offered by traditional simulation tools, SensorSim incorporates features to model power usage and study new communication protocols for sensor nodes. An interesting feature of this simulation framework is that physical nodes can be combined with simulated nodes to conduct hybrid simulations.

A network-level perspective on energy usage is presented in [26]. By simulating data gathering, simulating energy consumption, and using analytical methods, the authors establish upper bounds on the lifetime of a sensor network. The relationship of these bounds with the available initial energy, number of nodes, base station location, and other factors is explored. These bounds may be near-optimal if the deployment of nodes can be controlled by the user.

2.3 Inter-Node Communication and Routing

Network layer design for large sensor networks [5] also has to work within the same energy conservation goals that motivate node architecture. Optimization of the network protocol stack is a very important consideration if substantial energy savings are being targeted. Any gains achieved by efficient node design may be easily offset if non power-aware routing or communication schemes are used. Much of the work outlined in this section is compared to traditional network protocols to illustrate this key point.

Estrin, *et al.*, examine large sensor networks and their applications in [6]. They propose a set of distributed algorithms running on each node that are responsible for all the sensing and communication tasks. A clustering algorithm for selecting local cluster heads is described, and its advantages characterized in terms of robustness and energy savings. On-demand and *a priori* routing algorithms are discussed in [6] as they are applied to large networks. Because sensor nodes may be dormant for long time periods, on-demand algorithms may achieve greater energy savings because no traffic is generated periodically as in *a priori* protocols.

Estrin, *et al.*, discuss routing in sensor networks [27] and introduce two algorithms, the Basic Energy Conservation Algorithm (BECA) and the Adaptive Fidelity Energy Conservation Algorithm (AFECA). The former switches off the radio periodically and trades off higher route latency for power savings. The latter uses information about node density to adaptively let neighboring nodes handle network traffic. Using comparisons with Ad-hoc On Demand Distance Vector (AODV) routing, they report energy savings of up to 40% with BECA with a 50% duty cycle. AFECA is shown to achieve a two-fold increase in lifetime with a four-fold increase in node density.

Estrin, *et al.*, also describe a method to express communication patterns between the nodes via directed diffusion [28]. In directed diffusion, nodes express a willingness to *publish* or disseminate data. Other nodes may express *subscribtion* requests for obtaining that data. These *subscribtion* requests are used to establish gradients between these two sets of nodes. Gradients can also be used to model physical or logical attributes of the path between two nodes, for example, the link capacity. A message propagates between nodes along the paths with the maximum gradient until it reaches the node(s) which desires that data. Data may be locally modified, aggregated, or cached at intermediate nodes along the route.

Pottie, *et al.*, [29] describe a protocol suite for sensor networks beginning with the allocation and organization of TDMA-like time slots, to establishing routes among the nodes. The algorithm for the discovery of time slots proceeds without global knowledge of other nodes, or frame synchronization. A MAC protocol for interaction between a mobile node and the rest of the network is also discussed. The underlying idea behind these protocols is to minimize messages among nodes and limit the amount of information stored locally in registries to conserve energy. This is in contrast to protocols for traditional applications where energy constraints are not as strict.

The PicoRadio research effort [20] addresses the network layer by introducing designs for the MAC layer using dynamic channel assignment techniques. They propose using multiple communication channels to alleviate the problems of contention and collision. They characterize multihop routing schemes [30, 31] in terms of the energy required to transfer useful bits of information across the network. They find that sending a packet along multiple hops of shorter lengths is more efficient than using one longer hop.

2.4 Data Fusion

For a computational system such as a distributed sensor network, it is desirable for nodes to share raw sensor data with each other so that the quality of the final output may be increased. A single node may report erroneous values that may need to be averaged with the values reported by other nodes. Alternatively, the sensor coverage of a single node may not be adequate to provide the final output, and may require combination with the coverage of neighboring nodes. Both these cases require some form of collaborative processing between nodes to achieve the levels of service and reliability desired by the end user. A listing of the advantages and implications of redundant multi-sensor systems can be found in [8].

Another motivation for data aggregation is that raw data is much more expensive to transmit over a radio link than locally processing that data at the node level (for a comparison of the typical costs associated with these operations see [11]). Finally, it may be noted that for sensor networks, an end user is typically concerned with obtaining estimates or aggregate values from a group of sensors, and not individual node data.

The distributed tasking algorithms introduced in this thesis achieve their power conservation goals primarily by relying on the redundancy found in these sensor networks. Additionally, understanding how data is fused between nodes also gives these tasking algorithms a perspective on which nodes are essential to the tasking process, and which nodes can be switched off to conserve power.

A large body of literature addresses collaborative signal processing for distributed systems such as sensor networks. While it is impossible to list all sources, a few efforts in this direction are outlined. Beamforming is the process of aggregating co-related sensor signals using filters, to enhance or optimize some property of the original signals [32]. Yao, *et al.*, consider blind beamforming for a randomly located sensor array [33]. A beamforming problem in the absence of sensor locations is referred to as blind beamforming. The approach employed by the authors uses only the measured sensor data to form a correlation matrix, which is then processed using the maximum power criterion. This method is well suited to sensor networks that typically contain sensor node arrays whose geometry is irregular. A potential drawback of this approach is its dependence on time synchronization between the sensors.

Research conducted by the μAMPS group [32] explores the energy savings afforded by using a cluster head to collect beamformed data vs. a direct transmission scheme, and vs. a multi-hop communication scheme. They report that the clustering algorithm shows a reduction by a factor of six in the energy consumed, and a two-fold increase in the total lifetime of the network over the other two schemes. They also perform hardware comparisons of two beamforming algorithms, specifically, Maximum Power Beamforming and Least Mean Square Squares (LMS). The LMS algorithm is shown to be superior to the Maximum Power Beamforming algorithm; it consumes $1/5th$ the energy with a 3 dB loss in performance.

Yoo and Chien use the Rockwell sensor node [22] to classify seismic and acoustic data signatures for target recognition [34]. They extend the target recognition capabilities of the node to target tracking by simulating two co-operative processing algorithms. The first uses Wave Intensity Comparison (WIC) of the seismic readings. The second operates on acoustic data using a technique called Interaural Time Differencing (ITD). These approaches are less computationally complex than beamforming. Additionally, ITD is shown to be less dependent on clock synchronization than beamforming.

Iyengar, *et al.*, [35] propose to architect a sensor network as a multi-level deBruijn network. Such a configuration is shown to achieve fault tolerance along with a simple and decentralized routing scheme. This work also includes an overview of the sensor integration problem, i.e., combining the outputs of n sensors to produce a meaningful output, given that f $(f < n)$ sensors produce faulty information. The construction of the network as described allows the search for a fault in the network to be narrowed to two faulty nodes or communication links.

2.5 Optimal Placement of Sensor Nodes

As noted in the introduction, a distributed sensor network is characterized by many redundant nodes placed in the region of interest. The eventual deployment of these nodes, i.e., determining where to locate these nodes given a set of fixed constraints, is a problem that has not been as widely addressed as other aspects of these networks. Node deployment is affected by many factors, some of them being the sensor capabilities of individual nodes, radio propagation characteristics, and the topology of the region. Other constraints may include introducing a degree of overlap in the sensor coverage of two nodes so that they may collaborate their data processing efforts. For the system to operate effectively, redundancy should be utilized in a way that single points of failure in the communication network are avoided. This constraint may require knowledge of the underlying network protocols. Effective deployment of nodes can greatly impact the efficiency achieved by distributed tasking algorithms proposed in this thesis. This is chiefly because of the inherent assumption on the part of the algorithms that many nodes are located close enough to each other to permit some measure of load balancing.

Some of the related work in determining optimal placement of sensors focuses on placing sensors for structural analysis, optimizing placement for range finders, and placement of acoustic sensor arrays [36, 37, 38]. Spall and Sadegh [36] deal with determining an optimal configuration of sensors for systems where the prior knowledge is either sparse or too complex to construct reliable models. The authors base their criterion of optimality on maximizing the cumulative sensor response, and minimizing the correlation between the individual sensor responses. This approach has benefits in situations where exact measurements and models are not available initially, and some arbitrary or near arbitrary initial configuration must be chosen. Various optimization techniques are compared as part of this study, specifically, Finite Difference Stochastic Approximation (FDSA), Simultaneous Perturbation Stochastic Approximation (SPSA), Deterministic Gradient Based Techniques, Simulated Annealing, and Genetic Algorithms. SPSA is found to be preferable because it offers savings in the number of observations per gradient approximation and does not need detailed model information.

Another class of problems that bears close resemblance to the problem at hand is the optimal location of transmitters, or base stations for wireless and cellular networks. The primary difference between these two problems is that the sensor network is assumed to be collaborative and redundant unlike a cellular network. Any placement approach for sensor nodes must also take into account the expense and difficulty in re-deploying nodes. This is chiefly due to the limited lifespan of nodes, and the fact that their battery sources are non-replaceable. Cellular networks, in contrast, do not suffer from this limitation.

Sherali, *et al.*, [39] outline the differences between the traditional facility location problem and that of placing a set of radio transmitters optimally. They approach the problem by constructing a discrete grid to specify the radio coverage region associated with one or more transmitters. Their model description decomposes into a non-linear programming problem, which they solve using non-linear optimization algorithms such as Hooke and Jeeves' method, quasi-Newton techniques, and conjugate gradient methods. They describe their search methods as viable algorithmic procedures in addressing problems of this nature.

Wright, *et al.*, have developed a software visualization tool called Wireless System Engineering (WISE) [40, 41] to plan where to place base stations. Their approach uses a combination of computational geometry, Computer Aided Design (CAD), and numerical optimization methods. The authors show that the objective functions derived from radio propagation models are typically discontinuous and, therefore, choose an optimization scheme based on the *simplex* direct search method. The application of WISE software is primarily for locating base stations within buildings. Local minima in the objective functions are considered acceptable solutions because many competing locations may exist within a single building. Sensor networks may share this characteristic of multiple locations.

Howitt and Ham [42] propose a technique for base station location by employing a global optimization strategy. They contend that due to the inherent non-smooth and non-convex nature of local objective functions, modeling the objective function with a stochastic process yields substantial improvements.

2.6 Summary

This chapter outlines literature that addresses node architecture, collaborative processing and detection algorithms, and communication protocols for distributed sensor networks. Most of the techniques proposed are motivated by the need to minimize the energy consumption of nodes. A significant observation in this context is that design aspects across all layers must be considered relative to each other to achieve major energy savings. An inefficient scheme at any layer may adversely affect overall performance. Power management objectives must be aggressively pursued, if these networks are to take shape as large, unattended, sensing systems capable of operating for substantial time periods.

Chapter 3

Distributed Algorithms for Electing Application Query Servers

To achieve distributed management of tasks in the network, the concept of an *Application Query Server* (AQS) is introduced. An AQS is essentially a node in the network that oversees the assignment, arbitration, and control of user queries for all the nodes in its geographic region of the network. This task assignment should take into account the energy level of the nodes, the current workload of the nodes, and the capabilities of the nodes. As indicated by the name, the AQS operates at a layer of abstraction higher than a cluster-head for communication or routing purposes [6, 32]. Instead, the AQS advertises itself to the routing layer to allow queries destined for its region to be routed to it.

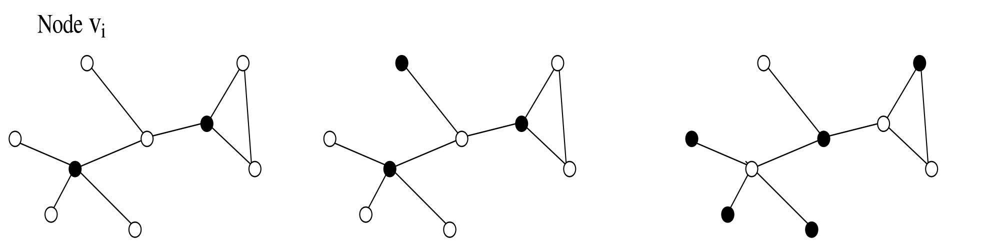
An AQS is a node that manages a set of nodes in a region. The goal of the AQS is to extend the service life of the network by assigning tasks to individual nodes in a power-aware fashion. As noted in Chapter 2, a sensor node may have different coverage areas for each of its individual sensing elements. This requires selecting an AQS for each *application* or sensing element that a node may contain. An AQS does not necessarily have to cover its entire region with its own sensing elements. However, it is in contact with nodes that cover that area.

Achieving energy savings using AQSs requires that, as indicated in [23], some nodes in the region and their radios are shutdown when their services are not required for tasking or communication purposes. AQSs may also sleep for long periods of inactivity and may be woken on request from the network. The underlying assumption in using this approach is that the sensor field is quite dense and allows significant levels of redundancy. If the sensor field is not redundant, then to some extent, one does not need to consider efficiency because every sensor in the a region will be needed to service a query.

In addition to efficiently assigning tasks, an AQS must balance energy consumption across nodes to avoid the formation of dead-spots in the areas covered by the sensor network. This goal when combined with energy scavenging techniques, such as recovering energy from the environment by using solar cells [16], gives nodes an opportunity to rejuvenate their energy source.

This distributed tasking algorithm operates in two stages. The first stage is the localized election of application query servers from among the nodes in a network. The algorithm to conduct these localized elections is based on existing distributed algorithms for the selection of a maximal independent set (MIS) for simple graphs. These algorithms are particularly appealing for this application because of their asynchronous and distributed nature that makes them suitable for sensor networks. These algorithms are also seen to exhibit scalable and fault-tolerant properties. This chapter introduces the stage of the algorithm used to elect AQSs. The discussion is prefaced with a theoretical analysis of the algorithm. Variations in the basic algorithm and its fault-tolerant abilities are also characterized.

In the second stage of the algorithm, elected AQSs use distributed methods to task individual nodes with user queries. The goal of these schemes is to select an appropriate subset of nodes in a region that are assigned tasks. A discussion of these methods is given in Chapter 4



(a) An independent set of nodes. **(b)** A maximal independent set of nodes **(c)** A maximum independent set of nodes.

● A node in the independent set

○ A node **not** in independent set

Figure 3.1: Independent sets for a simple graph.

3.1 Selecting a Maximal Independent Set

3.1.1 Definitions

Before discussing the problem of electing application query servers from among the set of sensor nodes, the following definitions are presented.

The sensor network is represented by a simple graph $G = (V, E)$. Each vertex v_i represents a sensor node in the network. Each edge (v_i, v_j) represents two adjacent nodes v_i and v_j. The adjacency criteria could be, for example, two nodes that have an overlap in the area covered by their radios. Two nodes that are adjacent are said to be *neighbors* of each other.

Further, consider the definitions of the following properties of simple graphs [43]. Given a simple graph G with a vertex set V and edge set E; an *independent set* of G is a subset S of V, such that no two vertices of S are adjacent in G. A independent set S of G is called a *maximal independent set* if all the vertices of G are either in S, or adjacent to a vertex in S. Finally, an independent set S of G is a *maximum independent set* if there is no independent set S' with $|S'| > |S|$. A pictorial representation of these sets is shown in Figure 3.1.

A maximal independent set S in G can be considered to be a set of AQSs. This is so because by definition each and every node in V is either an element of S or is adjacent to an element of S. Therefore, the elements of S are collectively responsible for all the nodes in the sensor network, i.e., V. Consider Figure 3.1(a); node v_i is neither in the independent set nor adjacent to a node in the independent set. Therefore, it is not accounted for by any AQS. In contrast, Figure 3.1(b) represents a MIS where all nodes are either AQSs or are adjacent to an AQS. Figure 3.1(c) shows a maximum independent set. To summarize, the problem of electing AQSs for the management of nodes can be viewed as one of finding a maximal independent set (MIS) from among the nodes in the network.

3.1.2 Algorithm to Generate a Maximal Independent Set

The maximal independent set of nodes chosen to be the AQSs should ideally be the *minimum* maximal independent set, i.e., the maximal independent set of minimum size. This is because the smallest possible number of AQSs should be chosen to manage other nodes. The problem of determining a maximal independent set of size K or less where $K \leq |V|$ is known to be NP-complete [44]. It is also known that the problem of finding a maximum independent set of a given size is an NP-complete [45].

In contrast, a simple linear time algorithm exists for determining a maximal independent set of a simple graph. While this may not lead to the best selection of AQSs, i.e., the size of the maximal independent set chosen may be larger than the minimum maximal independent set, this problem can be effectively solved due to its polynomial bound. Such an algorithm is shown in Figure 3.2. The operation of this algorithm is simple. At each iteration a vertex is chosen from the set V and a check is performed to determine if it belongs to the set $N_G(I)$ where the set $N_G(I)$ is defined to be the *neighbor set* of I in G [43]. If not, it is added to the independent set I. The algorithm iterates over all the vertices in V. It can be seen from Figure 3.2 that the algorithm executes with a polynomial complexity.

A parallel solution to the maximal independent problem that is NC-complete is proposed by Karp and Widgerson [46]. Another approach that has been successfully employed in this regard is to

```
1:    $I = \emptyset$
2:    for all $v \in V$ do
3:        if $v \notin N_G(I)$ then
4:            $I = I \cup v$
5:        end if
6:    end do
```

Figure 3.2: Sequential algorithm to determine a maximal independent set.

generate a maximal independent set based on the Monte Carlo method. This method is attributed to Luby [47]. The Monte Carlo algorithm proposed by Luby selects an initial independent set I', and then increments it to obtain a maximal independent set I. The following rule determines if a node v_i is a member of the independent set I'.

Rule 3.1 *For each vertex $v_i \in V$, choose a distinct random number $\rho(v_i)$. Let $v_i \in I' \Leftrightarrow \rho(v_i) > \rho(v_j)$, $\forall$ v_j adjacent to v_i.*

The algorithm is presented in Figure 3.3. After determining the initial independent set I' in step 5, the vertices adjacent to the vertices in I' are determined in step 7. These vertices constitute set $N_G(I')$. In step 8, the union of sets I' and $N_G(I')$ is removed from V'. The subgraph of G induced by the vertices remaining in V', i.e., $G(V')$, is constructed in step 9. This loop iterates until this vertex induced subgraph is empty; at this point the maximal independent set I is obtained. Analysis by Luby shows that the expected number of iterations required to generate the set I is $EO(log(|V|))$.

Given the distributed nature of the sensor network, one needs a method for determining a maximal independent set that can be executed by all the nodes in parallel. The basic algorithm introduced by Luby is used by Jones and Plassmann [48] to create a method for vertex coloring a graph in parallel. The process by which maximal independent sets are created for this parallel coloring algorithm is an improvement over Luby's algorithm in two important respects. Firstly, the expected execution time is shown to be smaller than the bound derived by Luby. Secondly, this algorithm is asynchronous in nature. The algorithm to elect AQSs that is introduced in this thesis is based on

```
1:     $I = \emptyset$
2:     $V' = V$
3:     $G' = G$
4:     while $G' \neq \emptyset$ do
5:         Select an independent set $I'$ in $G'$
6:         $I = I \cup I'$
7:         $H = I' \cup N_G(I')$
8:         $V' = V' \setminus H$
9:         $G' = G(V')$
10:     end do
```

Figure 3.3: Luby's Monte Carlo algorithm to determine a maximal independent set.

this coloring algorithm.

3.2 Basic AQS Election Algorithm

This section describes the basic AQS election algorithm. An analysis later Section 3.2 proves that the algorithm is fast, scales gracefully to a large number of sensor nodes, and is robust to failure.

3.2.1 Description of the Algorithm

Some definitions are required prior to presenting the algorithm.

S = set of sensor nodes

$|S|$= cardinality of S

s_i = sensor node i

$A(s_i)_k$ = effective coverage area of sensor node i for the sensor type k

$R(s_i)$ = effective radio coverage area of sensor node i

$R_s((s_i)_k)$ = effective coverage radius of sensor node i for the sensor type k

$R_r(s_i)$ = effective radio coverage radius of sensor node i

node status = {battery status, sensor capabilities, $A(s_i)$, $R(s_i)$, workload}
G_r = the intersection graph where s_i is a vertex in the graph, and an edge e_{ij} exists if and only if $R(s_i)$ intersects with $R(s_j)$

A set of application query servers is chosen such that each s_i is either an application query server or $R(s_i)$ intersects with the $R(s_i)$ of an application query server. As explained in the previous section, these application query servers form a maximal independent set in the intersection graph G_r. Each application query server, s_k, maintains the node status of each s_j where $R(s_k)$ intersects with $R(s_j)$. The algorithm is presented in Figure 3.4. Upon completion, I forms the MIS.

```
1:    assign each $s_i$ a random number, $0 < r(i) < 1$
2:    $P = S$
3:    $I = \emptyset$
4:    $H = G_r$
5:    while ($P \neq \emptyset$) do
6:        $V = \{s_i \in H : r(i) > r(j)\ for\ each\ e_{ij}\}$
7:        $P = P \setminus V$
8:        $P = P \setminus$ adjoining ($V$)
9:        $I = I \cup V$
10:       $H$ is now the graph induced by $P$
11:   end while
```

Figure 3.4: The basic algorithm for selection of AQSs.

The operation of this algorithm is similar to the one in Figure 3.3. At each iteration, the set of nodes V, with random neighbors greater than their neighbors' random numbers is chosen from the remaining graph, H. This set, V, is added to the maximal independent set and then subtracted from H, along with all nodes which are neighbors of H. The algorithm is presented in graph notation for the purpose of clarity. In the actual implementation, the algorithm proceeds in a purely distributed manner without the formation of graphs.

3.2.2 Distributed Operation of the Algorithm

The algorithm requires no global communication; all communication is only with the nearest neighbor. The operation of the algorithm at the sensor node level can be expressed as follows. At the beginning of the algorithm, each node broadcasts its random number to its neighbors. Following this initial message, the only information sent out by a node is (a) when it is included in the MIS, or (b) when it knows that it does not need to be in the MIS. Due to the condition under which nodes are included in the initial independent set (Rule 3.1), a node needs to send this information only to its neighbors whose random number is *lower* than its random number. However, because a broadcast medium (wireless) is being assumed for this discussion, nodes that receive messages from neighbors with lower random numbers ignore those messages.

The asynchronous nature of the algorithm is illustrated in Figure 3.5, where node 7 must wait on information from nodes 1 and 5 before deciding whether to include itself in the independent set. Once node 7 has made a decision, it communicates this to its other neighbors which are then free to take their own decisions.

3.2.3 Scalability and Fault Tolerance

If it assumed that the number of neighbors of any node in G_r remains bounded as $|S|$ grows, the loop executes $EO(\log(|S|)/\log\log(|S|))$ iterations. This is shown by the analysis of the parallel graph coloring heuristic in [48]. This assumption is not excessively restrictive; for most sensor network applications the density of nodes in a particular region is expected to be nearly constant even as the total number of nodes across the network grows. Therefore, the execution time of this algorithm grows *very* slowly with an increase in $|S|$. This leads to the belief that the algorithm is scalable to a large number of nodes. Experimental results presented in Chapter 6 confirm this belief.

The algorithm, as presented, assumes symmetric and reliable communication between each pair of nodes. This may not always be the case, for example, if a reliable underlying transport protocol is not being used. The effect of lost messages on the algorithm is now considered. Consider a failure

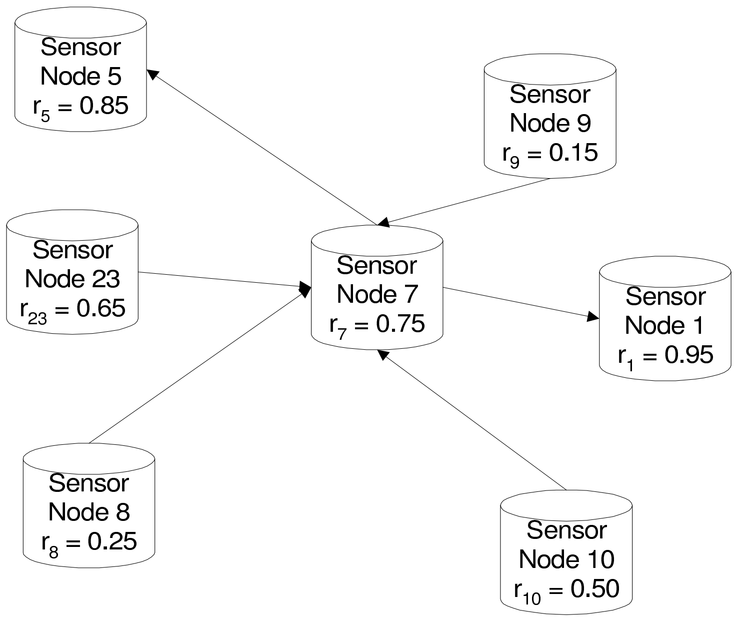

Sensor Node 5 $r_5 = 0.85$
Sensor Node 9 $r_9 = 0.15$
Sensor Node 23 $r_{23} = 0.65$
Sensor Node 7 $r_7 = 0.75$
Sensor Node 1 $r_1 = 0.95$
Sensor Node 8 $r_8 = 0.25$
Sensor Node 10 $r_{10} = 0.50$

Figure 3.5: An example scenario for an AQS election. Flow of information from various nodes with respect to Node 7 is shown.

in the exchange of r values between s_1 and s_2 such that s_1 receives $r(2)$, but s_2 does not receive $r(1)$ (asymmetric communication). This failure can be considered for the two separate cases shown in Figure 3.6.

Case 1. If $r(1) > r(2)$, then both nodes will proceed normally through the algorithm, with the potential that both of them may become an AQS. An extra elected AQS can be considered inefficient, but if the communication between the two nodes is unreliable then this may be a desirable result.

Case 2. If $r(1) < r(2)$, then s_2 will proceed normally and correctly through the algorithm and s_1 will wait for a message from s_2 indicating whether or not s_2 is an AQS. If the message is received (assuming that s_2 is broadcasting only to its neighbors), then s_1 proceeds normally and correctly through the algorithm. If this message is not received (assuming that s_2 is communicating only to its neighbors and does not know about s_1), then s_1 will wait indefinitely for a message. To prevent a deadlock, a timeout is introduced in the election process. After the timeout period, if for any

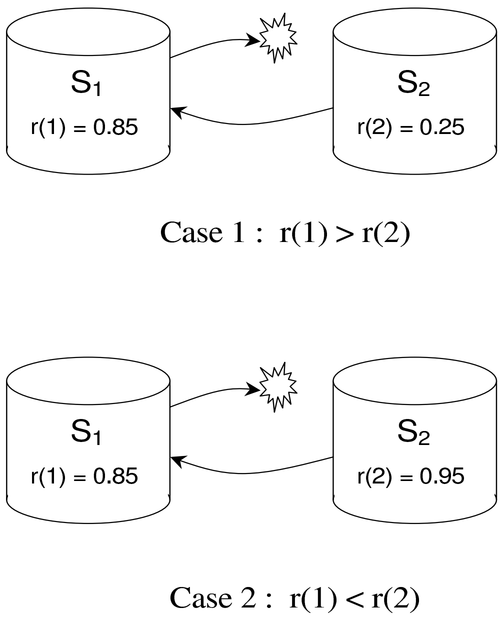


Figure 3.6: Asymmetric communication between two nodes due to an unreliable link.

reason a node does not receive a message it expects from a neighbor regarding inclusion in the independent set *and* none of its neighbors have been elected as an AQS, then it elects itself as an AQS and lets its neighbors know. Again, this may result in an extra AQS, but if communication is unreliable then this may be desirable.

Simulation experiments performed to study the algorithms consider the effect of link failure and packet errors on the execution of the election algorithm. Results from these simulations presented in Chapter 6 indicate that the performance of the algorithm scales gracefully in the presence of these failures.

3.3 Extensions to the Algorithm

In the basic algorithm, the decision on whether or not a node s_i is included in the MIS depends on the value of its random number $r(i)$, and the random numbers of its neighbors. Choosing an AQS based solely on this random number does not take into account the node status. For example, it might be desirable to select application query servers from those nodes with the longest remaining battery life. In this case, the remaining battery life can be defined as an integer x, such that $0 \leq x \leq k$. The algorithm can then be modified to redefine $r(i) = x + a$, where a is a random number between 0 and 1. With this modification, the nodes with the longest remaining battery life are always elected over other nodes, however, including the term a in $r(i)$ helps to break ties randomly.

The communication pattern is unchanged. As before, only the $r(i)$ value and messages indicating inclusion or non-inclusion in the MIS are exchanged between nodes. This technique can be used to define $r(i)$ as any function of desired node characteristics, thereby leading to the election of the most *appropriate* node as the AQS for a given region. This scheme is motivated by the balanced graph coloring algorithms introduced by Gjertsen, *et al.* [49]. These algorithms rely on adding weights to the initial random numbers to improve the graph colorings obtained. It is proved in [49] that the expected execution time of this algorithm remains $EO(\log(|S|)/\log\log(|S|))$. Experimental results presented in Chapter 6 confirm that the running time of the algorithm remains bounded.

This scheme can be modified as follows so that it is not dependent on the relative magnitudes of a and x. Define $r(i)$ to be the tuple $\{x, a\}$. For nodes s_i and s_j, define $r(i) > r(j)$ if $(x_i > x_j)$ or if $(x_i = x_j$ and $a_i > a_j)$.

Another variation in the basic algorithm can be introduced by considering the definition of two adjacent nodes. The basic algorithm defines two nodes s_i and s_j to be adjacent if and only if $R(s_i)$ intersects with $R(s_j)$. The algorithm can also be extended to a different type of graph, where adjacency is defined by a criterion other than radio connectivity. An example of such a criterion is the sensor range. To understand the advantage in doing this, consider the case when the sensor range is much larger than the radio range. For this scenario, the graph based on the overlap in sensor coverage is more dense than the one based on radio coverage. If the radio connectivity graph is used to elect AQSs, then a larger number of AQSs will be chosen than if the sensor connectivity graph is used. A larger number of AQSs can be considered inefficient because each AQS makes its task assignments independently (task assignment is discussed in detail in Chapter 4). This can lead to inefficiency if a node is assigned the same task by two different AQSs. In contrast, if the graph is based on the overlap in sensor coverage fewer AQSs are chosen. This leads to a more efficient allocation of tasks to the nodes with fewer conflicts from other AQSs.

Therefore, in certain cases it may be advantageous to use the graph based on sensor coverage overlap. However, the communication pattern between nodes may change as a result of this condition. To understand the effect on inter-node communication, consider the following scenario. Define two nodes s_i and s_j to be adjacent if and only if $A(s_i)_k$ intersects with $A(s_j)_k$. The operation of the algorithm is not affected by this new definition because the algorithm operates only on the underlying connectivity graph. The communication patterns are considered for the case when the sensor range is smaller than the radio range, and vice-versa. For both these cases, an example of a connectivity graph is shown in Figure 3.7.

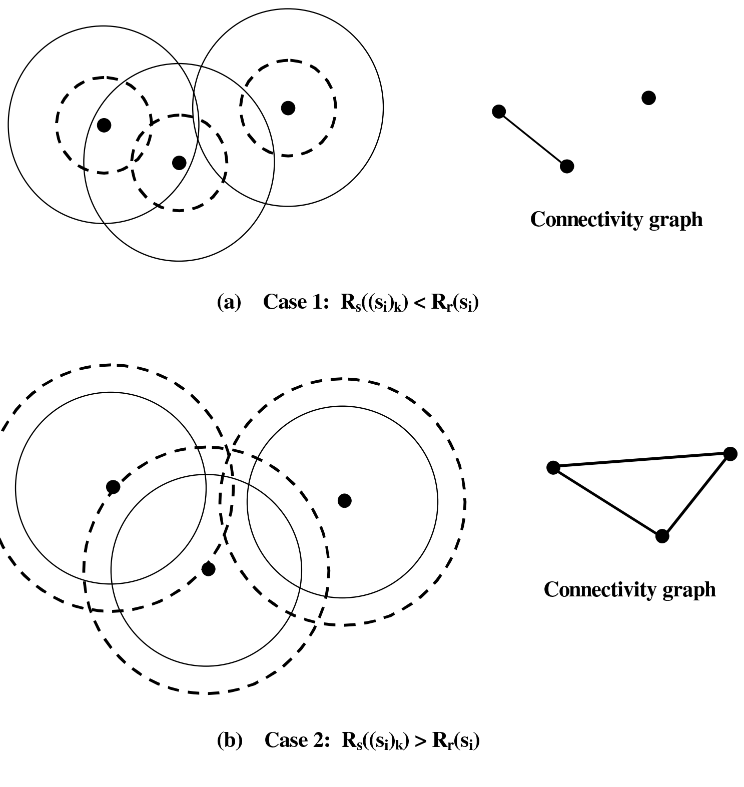

Connectivity graph
(a) Case 1: $R_s((s_i)_k) < R_r(s_i)$
Connectivity graph
(b) Case 2: $R_s((s_i)_k) > R_r(s_i)$

Figure 3.7: Connectivity graphs based on sensor range. The circle with the broken line represents the sensor range whereas the circle with the unbroken line represents the radio range.

Case 1. If $R_s((s_i)_k) \leq R_r(s_i)$ (Figure 3.7 (a)), then the required radio communication is still only a single hop. The algorithm at each node proceeds exactly as described earlier in Figure 3.5.

Case 2. If $R_s((s_i)_k) > R_r(s_i)$ (Figure 3.7 (b)), then the communication between nodes is no longer a single hop. A message between two nodes s_i and s_j may need to propagate over multiple hops. Nevertheless, the basic algorithm remains the same.

The optimizations mentioned in this section have been incorporated in the simulation models used to study the operation of the distributed election algorithm. Results for these are presented in Chapter 6.

3.4 Summary

This chapter introduces the concept of an Application Query Server (AQS), which is a node in the network that manages and assigns user queries to nodes within its region. AQSs are chosen for the network such that each node in the network is either an AQS or is adjacent to another AQS. This problem is reduced to one of finding a maximal independent set (MIS) of nodes for the network. A theoretical background of the MIS problem for simple graphs is presented. A distributed algorithm by Luby [47] that chooses a maximal independent set is analyzed, and extended to an algorithm for the election of application query servers. The election algorithm is analyzed from the perspective of its fault-tolerance and scalability to a large number of nodes. Finally, variations in the basic algorithm are proposed to fine-tune its performance. The first variation attempts to select the nodes with the longest remaining battery lives as AQSs. The second variation explores using intersection graphs other than the basic radio connectivity graph to elect AQSs.

Chapter 4

Assignment of Tasks Using Application Query Servers

This chapter considers the operation of the distributed tasking algorithm following the election of a set of AQSs. As a result of the election, the geographical confines of the network are sub-divided into regions with an AQS being responsible for each region. Each node is a part of at least one such region; some nodes may belong to more than one region. At this point, the sensor network is ready to be tasked with user queries. Application query servers advertise themselves to the routing layer of the network. This causes a user query designated for a particular region of the network to be routed to the application query server(s) responsible for that region.

It is the responsibility of an AQS to assign tasks related with user queries to the nodes within its region. To accomplish this objective, an AQS must maintain a record of the status of nodes within its area of responsibility. This status information includes the remaining battery life of a node, the current activity status of a node (awake, asleep, or dead), the sensor capabilities of the node, and the current task load of the node. This requires nodes to periodically inform their AQSs of their battery status. The remainder of this chapter describes how an AQS uses this status information to assign tasks to nodes within its area of responsibility.

4.1 Energy Aware Task Assignment

4.1.1 State Transitions at a Node

As noted earlier, a high degree of node redundancy is a characteristic feature of distributed sensor networks. This property allows for only a subset of nodes in a given region to be tasked for actually servicing a user query. Before addressing the assignment of tasks to this subset of nodes consider the following observation. When a node is not actively being tasked or relaying messages, it should be powered down to a sleep state to conserve energy. To achieve any substantial energy gains in this sleep state, the radio interface must be switched off [23]. Nodes may toggle themselves between the awake and asleep state periodically to check for new tasks. The AQS could then determine the sleep period of a node and wait for it to power up before it dispatching it a task. However, this presents a difficulty if the AQS *must* task a node while it is asleep. A low-power wakeup radio [50, 20], similar to a pager, can be attached to a node that allows the node to be woken up on demand. This wakeup radio is only meant to allow the node to transition from the sleep to the awake state. It allows only for a small amount of information relevant to this process to be sent to a node. The use of the wakeup radio is assumed in the simulations; it allows nodes to remain in the sleep state for long time periods because they can be woken up when required by the AQS. Nevertheless, the use of such a system is not required for the proper functioning of the tasking algorithms.

Figure 4.1 shows a state diagram showing the transitions between several states of a node. The *awake/idle* state is where the network interface is operational and the node can send or receive data. The *asleep* state was described in the previous paragraph. A node enters the *transmit* or *receive* states to send or receive data, and returns immediately to the *awake* state. The *dead* state is reached when the battery of the node is completely drained, and it cannot function any longer.

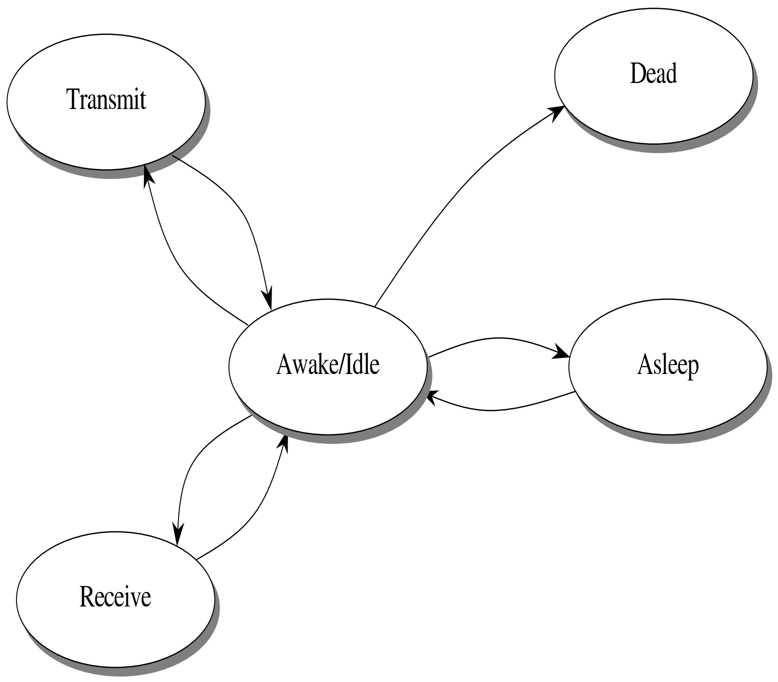


Figure 4.1: Transitions between states at a node.

4.1.2 The Grid Map

An AQS may use two approaches to process a user level query into task assignments for nodes. A simplistic approach, *Approach I*, involves receiving a query at the AQS, and activating all nodes in its range to retrieve the information required by the query. Approach I requires relatively little information to be maintained at the AQS, and in some sense is a brute force approach to tasking. The second approach, *Approach II*, is motivated by the fact that nodes need to be kept asleep as long as possible to achieve meaningful energy savings. Therefore, Approach II tries to select a set of appropriate nodes to task by taking into account the remaining battery life at each node, number of active tasks at a node, and the geographical region of the query. All other nodes in the region of the AQS are allowed to transition to the asleep state.

In Approach II, an AQS forms a *grid map* by subdividing its region into a rectangular grid. The purpose of this map is to represent sensor coverage of various nodes in the AQS's region. Therefore, the size of the grid map must be such that it includes the sensor coverage of all the nodes that

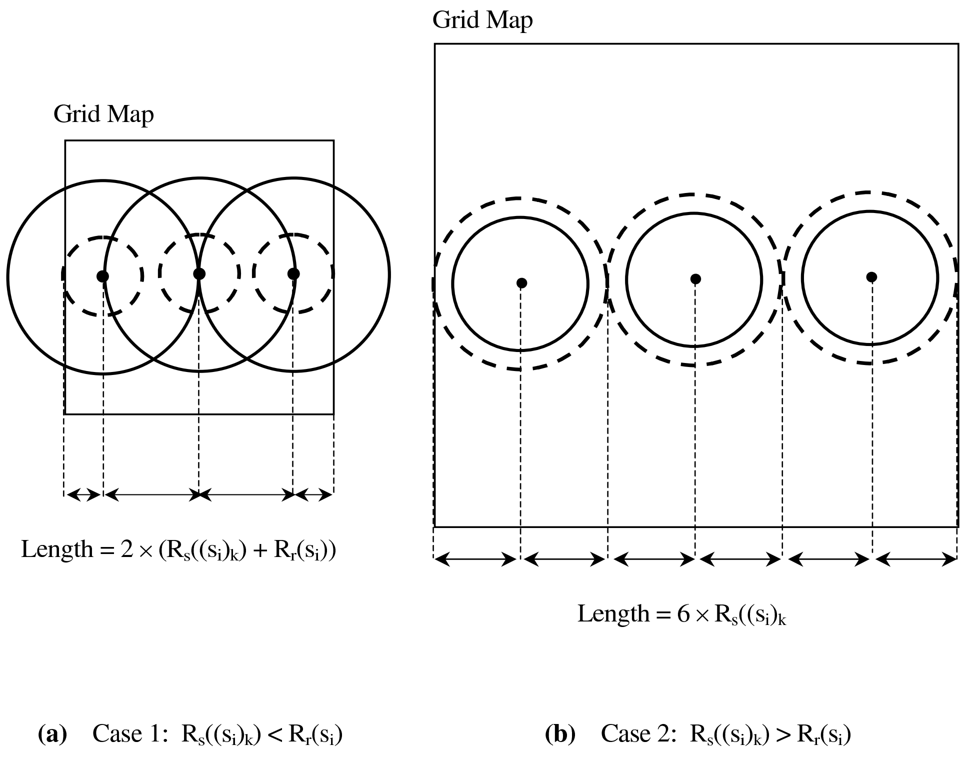

Grid Map
Length = $2 \times (R_s((s_i)_k) + R_r(s_i))$
(a) Case 1: $R_s((s_i)_k) < R_r(s_i)$
Grid Map
Length = $6 \times R_s((s_i)_k$
(b) Case 2: $R_s((s_i)_k) > R_r(s_i)$

Figure 4.2: Size of the grid map/geographical region of an AQS. The circle with the broken line represents the sensor range whereas the circle with the unbroken line represents the radio range.

may lie in its region of responsibility. As seen from Figure 4.2, the size of the grid map is different for the case when the sensor coverage radius is smaller than the radio radius (Figure 4.2 (a)) and vice-versa (Figure 4.2 (b)). For both these cases, the area covered by the grid map is also defined as its geographical region of responsibility.

A cell in the grid map is colored to denote the relative position of a node from the AQS. Cells around that particular cell are colored with the same color to indicate the sensor coverage of that node. Eventually, when such a map has been constructed, a colored cell on the grid map represents the presence of sensor coverage at that location, while an un-colored cell denotes the absence of sensor coverage. Such a grid map is shown in Figure 4.3. The center of this map is drawn relative to the position of the AQS. The black region corresponds to sensor coverage provided by the AQS itself. Other shaded regions correspond to nodes around the AQS within its region. All un-shaded

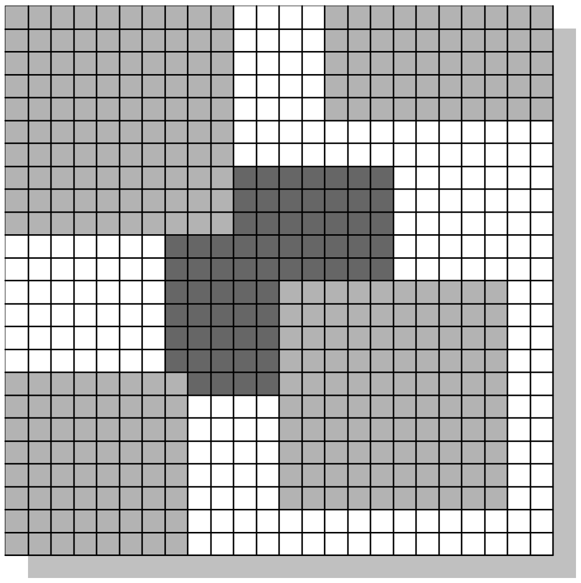


Figure 4.3: Representation of a grid map at an AQS.

blocks correspond to uncovered area. The assignment of tasks to nodes is a two step process. The first step consists of constructing the grid map by selecting nodes that satisfy some fitness criteria. The second step consists of actually assigning tasks to a subset of the nodes that are currently in the grid map.

4.1.3 Construction of the Grid Map

At all times during its operation, the AQS maintains a set B that is a subset of all the nodes in its region. This set essentially contains those nodes that *can* be used to service a query that is sent to its region. The nodes in B are placed on the grid map and the sensor coverage associated with them is used to color the pixels of the grid map.

Before discussing how the set B is chosen, the following nomenclature is defined. Define set $N = \{s_1, s_2 \ldots s_n\}$ to be all the *alive* nodes in the region of the AQS. Further, define $b(s_i)$ to be the current energy level of node $s_i \in N$. Finally, define r_{index} to the redundancy index, i.e. the value of the sensor coverage overlap between two nodes at which they are considered to be redundant with respect to each other.

The selection of B is described using the algorithm given in Figure 4.4. The operation of the algorithm can be explained as follows. In steps 3 and 4, a node that does not have any coverage overlap with any node currently in the grid map is added to set B. If a node s_i does have an overlap with a node $s_j \in B$, two cases may occur. If the overlap of s_i and s_j exceeds the value of the redundancy index r_{index}, then the node with the greater energy level is selected in B and the other node is removed from B (steps 8, 9 and 10). Alternatively, if the two nodes are not redundant both are added to B (step 13). To summarize, a node is added to set B under 3 conditions, (a) if it has no overlap with any node currently on the grid map, (b) if it is redundant with a node in the grid map with a lower energy level than itself, and (c) if its coverage overlaps but is not redundant with any node on the grid map. The only condition under which a node is removed from the grid map is when it is redundant with another node with a greater energy level.

```
1:   $B = \emptyset$
2:   for all $s_i \in N$ do
3:       if $(A(s_i)_k \cap A(s_j)_k) = \emptyset \;\; \forall \; s_j \in B$ then do
4:           $B = B \cup s_i$
5:       else
6:           for all $s_j \in B : A(s_i)_k \cap A(s_j)_k) \neq \emptyset$ do
7:               if $(A(s_i)_k \cap A(s_j)_k) > r_{index}$ then do
8:                   if $b(s_i) > b(s_j)$ then do
9:                       $B = B \cup s_i$
10:                      $B = B \setminus s_j$
11:                  end if
12:              else
13:                  $B = B \cup s_i$
14:              end if
15:          end do
16:      end if
17:  end do
```

Figure 4.4: Algorithm for constructing the grid map.

```
1:   $S = \emptyset$
2:   $A = \emptyset$
3:   for each $q \in Q$
4:       $q = \{q_{id}, \{\emptyset\}\}$
5:       select $\{s_1, s_2 \ldots s_m\}$ in $B$ which will service $q$
6:       $q = \{q_{id}, \{s_1, s_2 \ldots s_m\}\}$
7:   end for
8:   for each node $s_i \in B$
9:       if $\exists\ q \in Q : q$ is serviced by $s_i$, and $s_i \notin A$
10:          $A = A \cup s_i$
11:  end for
12:  $S = N \setminus A$
```

Figure 4.5: Algorithm for assigning tasks to nodes in the grid map.

4.1.4 Using the Grid Map to Assign Tasks

Following the construction of the grid map, an AQS attempts to assign tasks to the nodes in the grid map (set B) by using the algorithm described in Figure 4.5. Prior to a discussion of this tasking algorithm some additional terminology is introduced. Define A to be the set of all the currently *awake* nodes that are being tasked, and S to be the set of all the *asleep* nodes. Therefore, $N = A \cup S$. Q is the set of currently active queries at an AQS. Each query $q \in Q$ is organized as the following tuple: $\{q_{id}, \{s_1, s_2 \ldots s_m\}\}$ where q_{id} is the query identifier and $s_1, s_2 \ldots s_m$ are the nodes that are being tasked by this query.

It is observed that many queries sent to an AQS in the network may require the AQS to task only a portion of the region it covers. Therefore, not all nodes in set B need to be in the *awake* state. Only a subset of the nodes in B, i.e., A need to be *awake* at a given time to actually service those queries. The remainder of the nodes are in the *asleep* state, i.e. S. The algorithm in Figure 4.5 describes the partitioning of nodes in N into sets A and S.

The important step in this algorithm is the method for the selection of the query tuple for a query q in step 5. This is done by calculating the intersection of the geographical area specified by the query with the corresponding area in the grid map. This is shown in Figure 4.6. All nodes whose sensor coverage lies within this region are added to the tuple. In step 9, the AQS calculates if a node in B needs to be *awake* for any of these queries. All such nodes are added to the set A. Once the construction of these sets is complete, nodes in A are sent wakeup messages and task assignments whereas the nodes in S are sent messages that enable them to enter the *asleep* state.

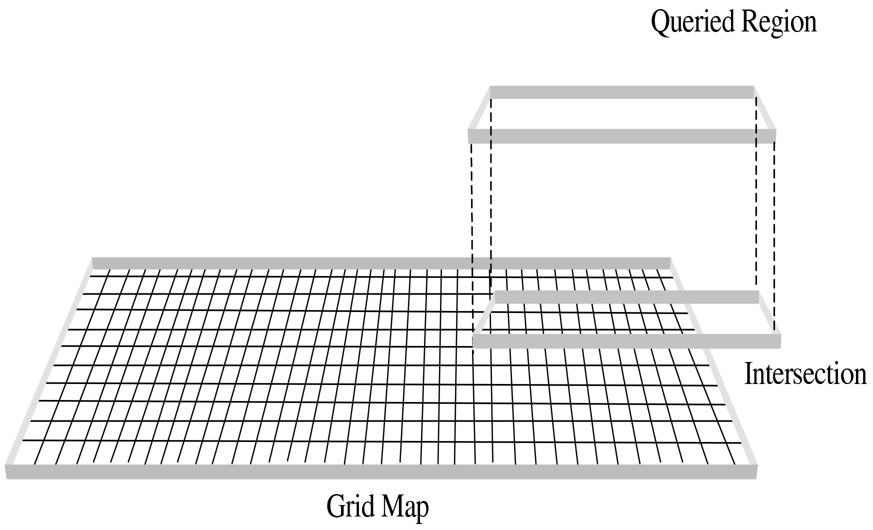


Figure 4.6: Intersection of the region of a query with the corresponding region in the grid map.

Because nodes in A report their battery level to the AQS periodically, the grid map must be redrawn to reflect these updated values. Each time the grid map is redrawn, sets B and A are re-calculated and new nodes take over the existing queries. To understand the effect of reconstructing this grid map, consider the following scenario. Battery updates are received very often from nodes that are awake and the period for redrawing the grid map is small. In this situation, nodes are quickly toggled between the awake and asleep state in the grid map. This leads to a situation where the set B, and therefore the set A are re-calculated too rapidly. On the other hand, if the battery updates are received at widely spaced intervals and the grid map redraw periods are comparatively large, sets B and A remain fairly constant for long periods of time. This may lead to the excessive energy consumption of the nodes in A. Therefore, the frequency of these updates and the frequency of reconstructing the grid map should be infrequent when the activity in the network is low, and somewhat higher for more active periods of the network.

The assignment of tasks using methods described above requires the mapping of sensor nodes to physical locations in the network. Nodes may need to be aware of their absolute or relative geographical positions in the network. Approaches for the geo-location of nodes are discussed in [51, 52].

4.2 Quality of Sensor Coverage for User Queries

While extending the life of the network by increasing $|S|$ is an important objective of these tasking models, it is also necessary to ensure that adequate sensor coverage is achieved for all queries. Because there are at most $|N|$ nodes which can be tasked by the AQS, one cannot achieve coverage greater than would be achieved by activating all those nodes simultaneously. Two factors determine whether the quality of coverage equivalent to turning on all nodes can be achieved.

(a) The first is the criteria used to determine the value of the redundancy index r_{index}. It is obvious that if a node's sensor coverage has no overlap with any other in node in N, then it must be included in B. Also, a node s_i may be replaced in the grid map by node s_j if s_i and s_j are located at the same position, and $b(s_j) > b(s_i)$. To conserve resources, the restriction that the nodes be exactly co-located can be relaxed; for example two nodes may be defined to be replaceable with each other if the coverage of one overlaps that of the other by 80%. This percentage, i.e., the redundancy index (RI), is varied in simulation experiments. The drawback of this approach is that the queried area and the actual tasked area may differ if low values are used for the RI. This heuristic however, provides a means to tradeoff network life with the quality of coverage. It may also be employed to increase the life of the network for periods when queries have relatively non-critical boundary requirements, or when prolonging the network life is more important than maintaining high fidelity for queries.

(b) The second factor that has bearing on sensor coverage is how nodes are added to the query tuples in step 5 of Figure 4.5. As shown in Figure 4.6, this method involves intersecting the area defined by the query with the corresponding region in the grid map and selecting nodes that lie in the intersection. The shape of the query and the approximation of a sensor's coverage in the grid map must accurately represent the region of the query and the actual sensor coverage respectively. For this implementation a simple rectangular representation is chosen for both the queries and the sensor coverage. This simplifies the calculation of the intersection. These representations can be extended to better approximations. Note that for queries that lie across multiple AQSs, the sum of queried areas at those AQSs provides the total coverage.

Also note that some collaborative signal processing algorithms may require the use of overlapping sensor node coverage areas; the methods for choosing B and adding nodes to query tuples can be modified to address this requirement.

4.3 Potential Inefficiencies in Task Assignment

An AQS is responsible for the physical area associated with the sensors of all the nodes in its region. Because the set of AQSs is essentially a MIS for the nodes in the network, it is noted that a node could have more than one neighboring AQS. In addition, a query could be sent to a region that contains multiple AQSs, i.e., the queried area could extend beyond the region associated with a single AQS. Both these situations are illustrated in Figure 4.7. Methods for assigning tasks must take both of these overlapping situations into account to avoid overloading a node. The basic problem is that one node could be assigned tasks by two different AQSs simultaneously; in this situation, neither AQS knows about the assignment of the other AQS. This may overload the node. This problem is considered for intersection graphs based on the radio coverage, and those based on the sensor coverage.

4.3.1 Intersection Graph Based on Radio Coverage

The potential inefficiencies of such an intersection graph are explored by considering the case of a sensor range greater than the radio range separately from the case of a sensor range smaller than the radio range.

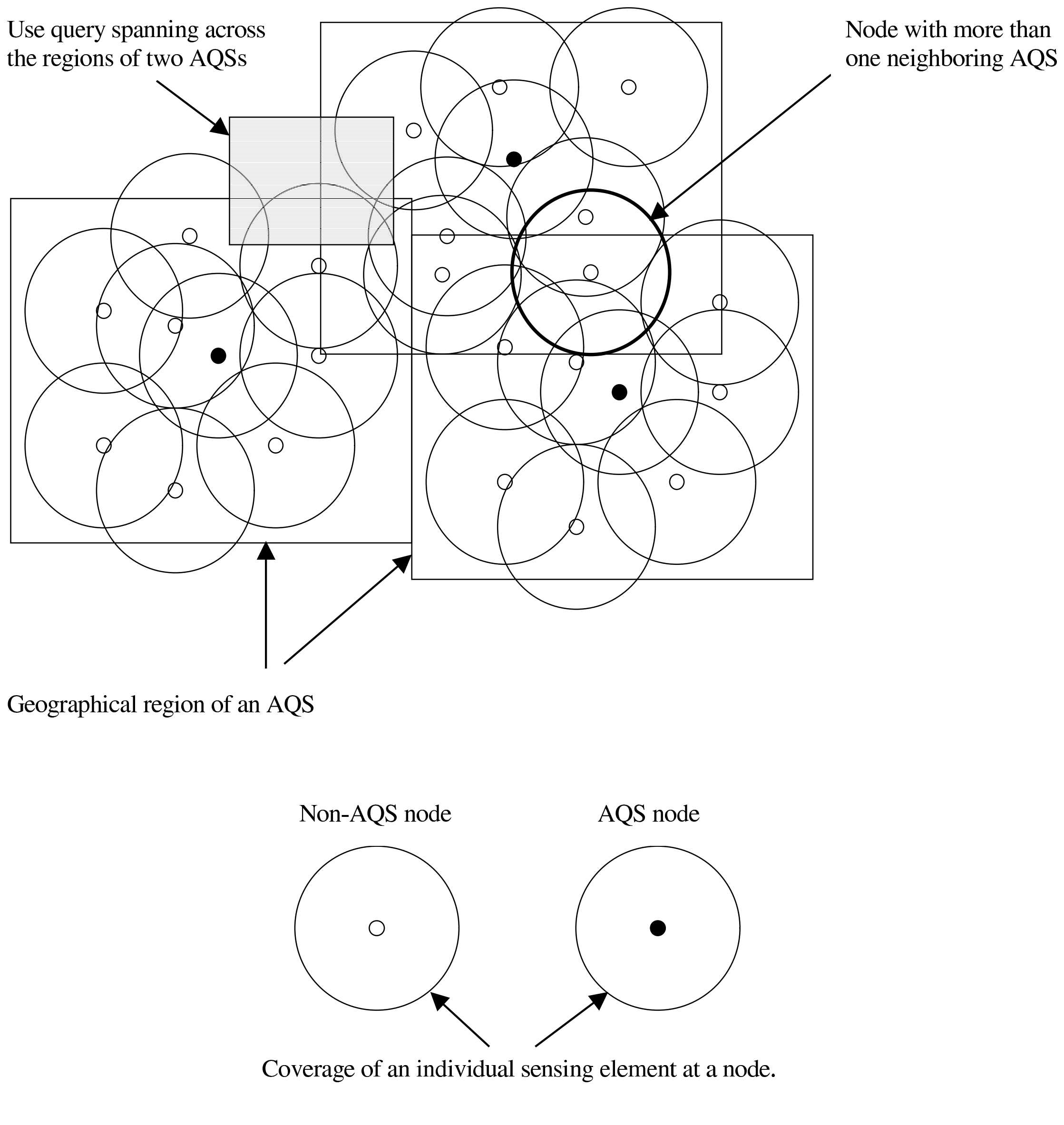

Use query spanning across the regions of two AQSs
Node with more than one neighboring AQS
Geographical region of an AQS
Non-AQS node
AQS node
Coverage of an individual sensing element at a node.

Figure 4.7: Overlapping regions of Application Query Servers.

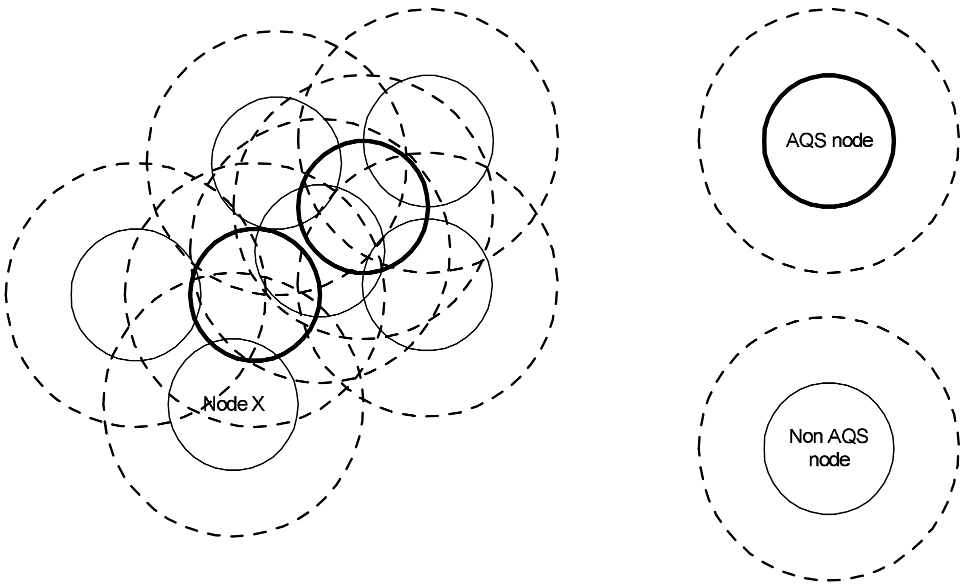


Figure 4.8: Intersection graph based on radio coverage, Case 1(a): $R_s((s_i)_k) > R_r(s_i)$. The circle with the broken line represents the sensor range whereas the circle with the unbroken line represents the radio range.

Case 1(a). *For s_i, the range of the sensors for application k, $R_s((s_i)_k)$, is greater than the radio range, $R_r(s_i)$.* As shown in Figure 4.8, this case has the difficulty that significant overlap in coverage area may occur between nodes that belong to different AQSs. Consider the non-AQS node, *Node X*, as an example. This node has significant overlap in coverage with several nodes that do not belong to the same AQS as it does. This creates a potential lack of efficiency if each AQS makes its assignment separately.

Case 2(a). *For s_i, the range of the sensors for application k, $R_s((s_i)_k)$, is smaller than the radio range, $R_r(s_i)$.* As shown in Figure 4.9, the nodes whose sensor coverage intersects with only one AQS are simple to handle; they are completely managed by that AQS. The overlap in sensor coverage between nodes belonging to two or more AQSs may still lead to inefficient task assignment. However, because the sensor range is smaller than the radio range this happens more infrequently than in Case 1(a).

To resolve this problem without excessive coordination between AQSs, the following scheme could be

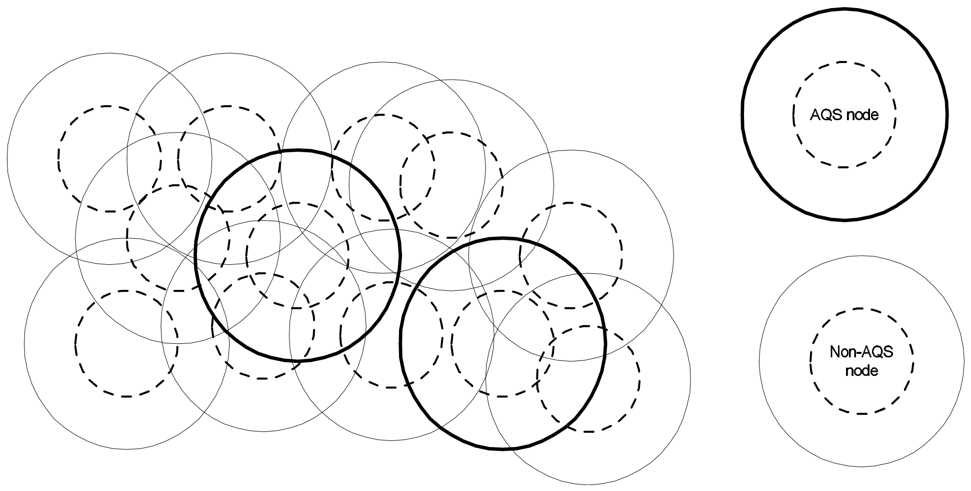


Figure 4.9: Intersection graph based on radio coverage, Case 2(a): $R_s((s_i)_k) < R_r(s_i)$. The circle with the broken line represents the sensor range whereas the circle with the unbroken line represents the radio range.

used. A node notifies each neighboring AQS whenever it is assigned a task (this can be accomplished via broadcast). Each neighboring AQS then ACKs that message. If a node receives a second task assignment from any AQS before it receives an ACK from that AQS, then it assumes the assignment was made without considering its current load and rejects the new task. The AQS must then decide to reaffirm the assignment or task a substitute node. This method of conflict resolution allows boundary conflicts to be avoided while maintaining only nearest neighbor communication (single-hop communication).

4.3.2 Intersection Graph Based on Sensor Coverage

Similarly, for intersection graphs based on the sensor range the case of a sensor range greater than the radio range is considered separately from the case of a sensor range smaller than the radio range.

Case 1(b). *For s_i, the range of the sensors for application k, $R_s((s_i)_k)$, is greater than the radio range, $R_r(s_i)$.* As shown in Figure 4.10, the use of an intersection graph based on sensor range

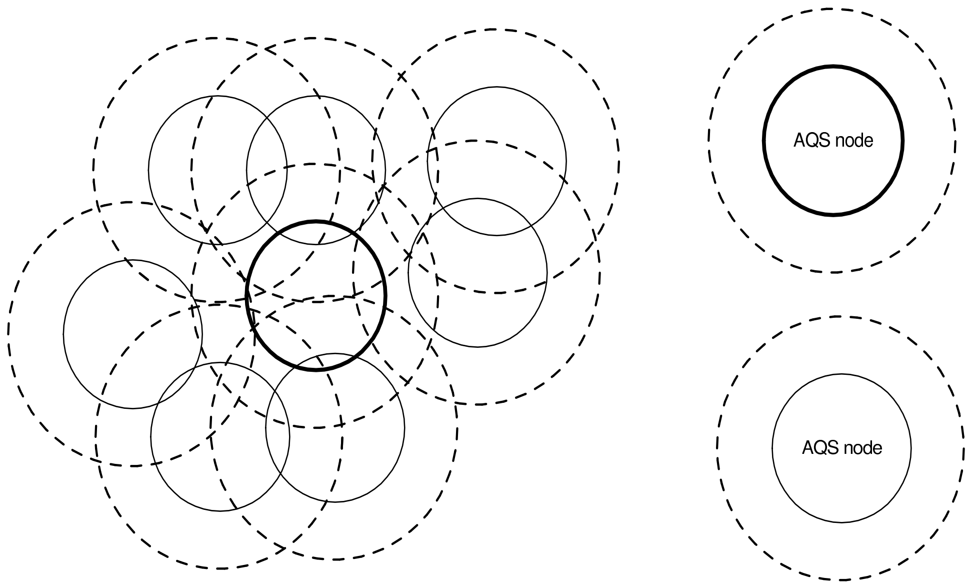


Figure 4.10: Intersection graph based on sensor coverage, Case 1(b): $R_s((s_i)_k) > R_r(s_i)$. The circle with the broken line represents the sensor range whereas the circle with the unbroken line represents the radio range.

rather than radio range leads to a more efficient selection of AQSs. However, this requires the use of multi-hop communication between nodes.

Case 2(b). *For s_i, the range of the sensors for application k, $R_s((s_i)_k)$, is smaller than the radio range, $R_r(s_i)$.* This case is somewhat redundant because typically when $R_s((s_i)_k) < R_r(s_i)$ one would select AQSs based on the radio range. However, it is analyzed for completeness. As shown in Figure 4.11, too many AQSs may be elected for a node density comparable to Case 2(a). This happens because there may not be enough overlap in sensor coverage between nodes. However, if the density of nodes in a region is high enough to introduce substantial overlap in sensor coverage, this case may yield an acceptable number of AQSs.

4.3.3 Conclusions

From an analysis of intersection graphs based on radio and sensor coverage as they relate to task assignment, it is concluded that neither type of graph offers a clear advantage over the other. The

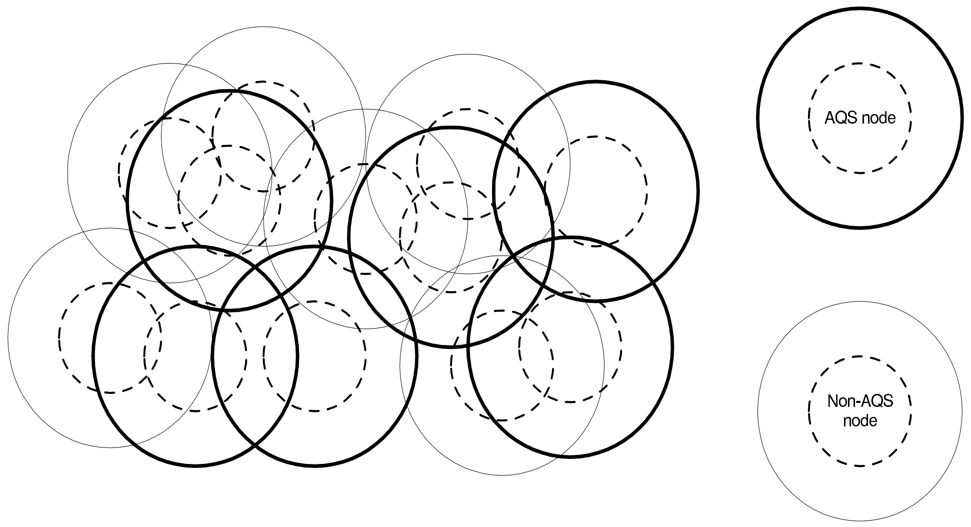


Figure 4.11: Intersection graph based on sensor coverage, Case 2(b): $R_s((s_i)_k) < R_r(s_i)$. The circle with the broken line represents the sensor range whereas the circle with the unbroken line represents the radio range.

decision on which graph to use is dictated by the relative difference between the sensor range and the radio range of a node. Additionally, if a node has multiple sensing elements with widely differing sensor ranges different types of graphs for different sensing elements may be required. The eventual objective is to minimize the number of AQSs and the potential for boundary conflicts.

In this thesis, simulations are conducted by considering both types of graphs. While the intersection graph based on the radio range yields to a simple implementation, the intersection graph based on sensor range requires the implementation of a multi-hop scheme to route messages between two nodes. Details of these implementation are presented in Chapter 5

4.4 Periodic Re-Election of Application Query Servers

New AQS must be chosen periodically to prevent the nodes chosen as the original AQSs from dying due to excessive load caused by the communication and co-ordination of tasks. Due to these reasons nodes that are functioning as AQSs will typically deplete their batteries faster than other nodes. When a re-election is conducted all nodes exchange their remaining battery life values; the nodes with the longest remaining battery lives are assigned more preference in the election (this is explained in Section 3.3). This usually leads to the selection of nodes that were not previously AQSs.

The re-election of new AQSs poses some problems. The primary issue is one of succession; the successor/s to the original AQSs must be made aware of the all the tasks and queries existing in the network just before the re-election. In this section the succession issue is analyzed for the three mutually exclusive cases. Prior to this discussion, the following nomenclature is introduced. A re-election begins at time t. A node that is an AQS *just* before a re-election, i.e., at time $t-\delta$, is represented as $AQS_{(t-\delta)}$. The nodes chosen as the new application query servers that are adjacent to the previous application query server, $AQS_{(t-\delta)}$, are represented as $AQS_{(t+\delta)_1}, AQS_{(t+\delta)_2} \ldots AQS_{(t+\delta)_n}$. X is the set of these newly elected AQSs.

Case 1. $|X| = 1,\ X \cap \{AQS_{(t-\delta)}\} = \{AQS_{(t-\delta)}\}$. In this case, the same node is chosen to be the AQS after the re-election, therefore there are no succession issues. All tasks handled by node $AQS_{(t-\delta)}$ before the re-election are automatically transferred to the new AQS, $AQS_{(t+\delta)}$.

Case 2. $|X| = 1,\ X \cap \{AQS_{(t-\delta)}\} = \emptyset$. In this case, the re-election results in the selection of a single adjacent AQS that is distinct from $AQS_{(t-\delta)}$. After the completion of the re-election, $AQS_{(t-\delta)}$ begins to transfer all its task information to the new AQS. For the most part, the new AQS accepts to take over the co-ordination for the set of currently executing tasks. However, the new AQS may be asked to manage a task for a node that is adjacent to $AQS_{(t-\delta)}$ but is not adjacent to itself. Figure 4.12 proposes a scheme that can be used to address this eventuality. Figure 4.12 (a) shows the AQS before the re-election, $AQS_{(t-\delta)}$, and the new AQS after the election, $AQS_{(t+\delta)}$. Consider the node s_i, which intersects with $AQS_{(t-\delta)}$ but does not intersect with $AQS_{(t+\delta)}$. In Figure 4.12 (b), $AQS_{(t-\delta)}$ broadcasts the tasking information pertaining to node s_i. $AQS_{(t+\delta)}$ cannot accept to co-ordinate this task because it does not intersect with s_i. Therefore, it sends a negative reply back to $AQS_{(t-\delta)}$ (Figure 4.12 (c)). At this point, $AQS_{(t-\delta)}$ knows the new AQS cannot be delegated this task so it re-directs this tasking information to the node that it pertains to, namely, node s_i (Figure 4.12 (d)). Finally, node s_i broadcasts this tasking information and

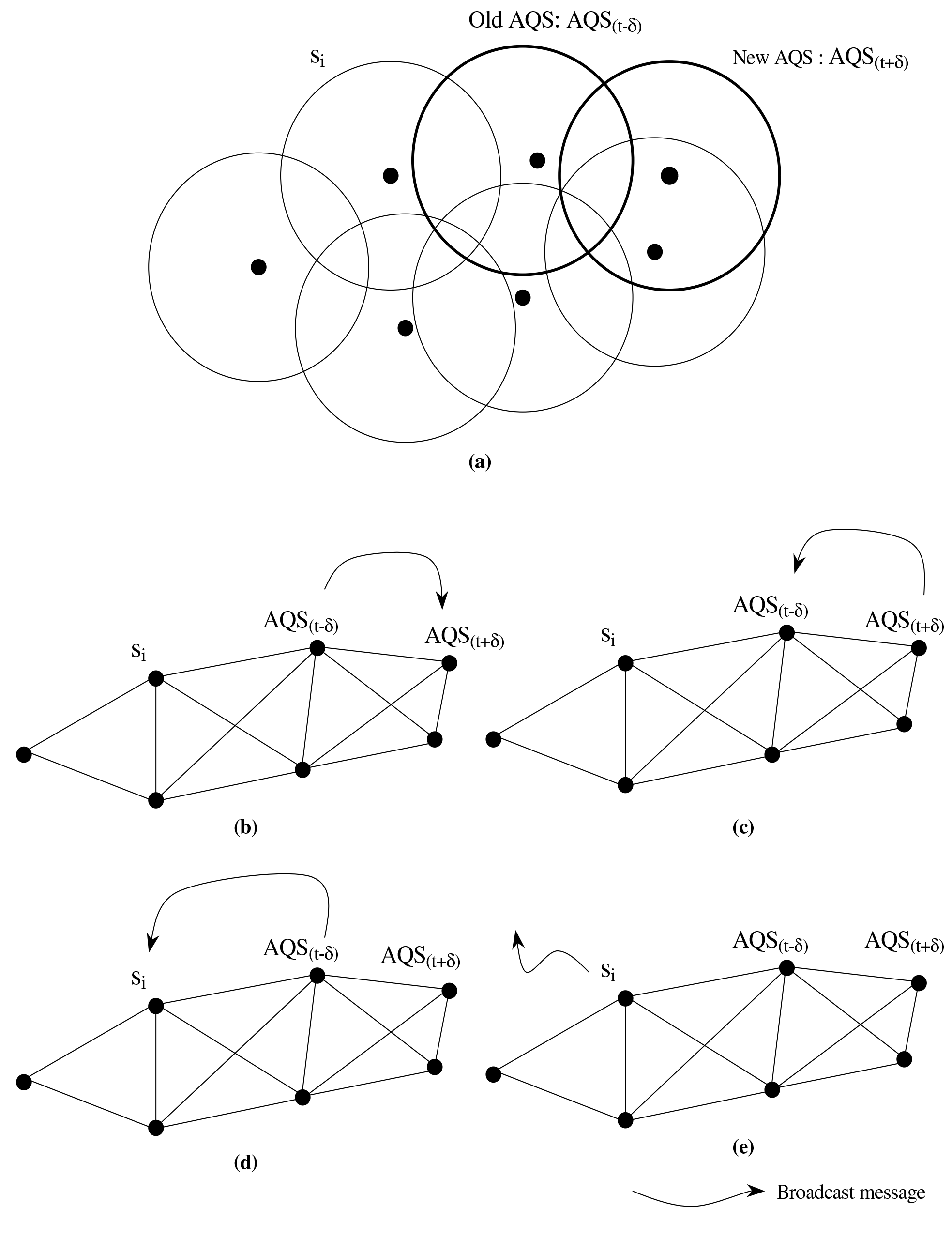


Figure 4.12: Transferring information to a new AQS following a re-election.

waits for *any* neighboring AQS to agree to manage this task (Figure 4.12 (e)). It is guaranteed that at least one such neighboring AQS exists; if not, then node s_i is necessarily an AQS and must co-ordinate its own task. In the worst case scenario the number of extra messages generated for this scheme is $4 \times |Q|$, where Q is the set of queries/tasks overseen by $AQS_{(t-\delta)}$. Note that all the communication in this case is nearest neighbor, and assumes a broadcast medium.

Case 3. $|X| > 1$. In this case, $AQS_{(t-\delta)}$ is replaced by at least two new AQSs that are adjacent to it. The selection of these multiple AQSs does not pose any new succession problems. If there exists a node that was tasked by $AQS_{(t-\delta)}$ but does not lie in the region of the new AQSs, a scheme similar to the one proposed for Case 2 may be employed. In contrast, if multiple AQSs try to manage the same task at a node, the node can send refusals to all those AQSs except one.

4.5 Summary

This chapter introduces the second stage of the distributed tasking algorithm, i.e., the assignment of tasks to individual nodes by the elected AQSs. This discussion is prefaced with a definition of the various operating states of a node. An algorithm is proposed for AQSs to assign tasks by constructing by a *grid map* of their region. For each user query, the AQS tries to task a minimal subset of nodes in its region by using this grid map. The algorithm is also analyzed in the context of the factors affecting the quality of sensor coverage for user queries. Potential inefficiencies in task assignment arising mainly due to boundary conflicts are discussed; methods are proposed to address them using single-hop and multi-hop communication. Finally, the chapter addresses the succession issue arising out of re-electing a new set of AQSs to replace the original AQS for a region.

Chapter 5

Simulation Framework

Analytical methods for investigating these distributed tasking algorithms provide a starting point in characterizing some properties such as the theoretical execution time of an election. However, other properties such as network lifetime and energy consumption do not yield to simple mathematical formulations. This is because the scope and complexity of such a sensor network is fairly large - a large number of nodes interact with each other in the presence of multiple queries and intermittent faults. Simulation methods are required to effectively study such a system and gauge the performance of the tasking algorithms.

This chapter provides an overview of the simulation environment and the models used to study the behavior of the distributed tasking algorithms. It begins with a description of the simulation tool used to construct and execute these simulations. Subsequent sections expand on the design and development of simulation models of the sensor network.

5.1 Simulation Tool - Opnet

Opnet is a commercial network simulation tool for modeling and analyzing various aspects of communication systems [53]. The choice of Opnet as a simulation tool is basically due to its modular design methodology, its support for modeling radio communication, and its extensive

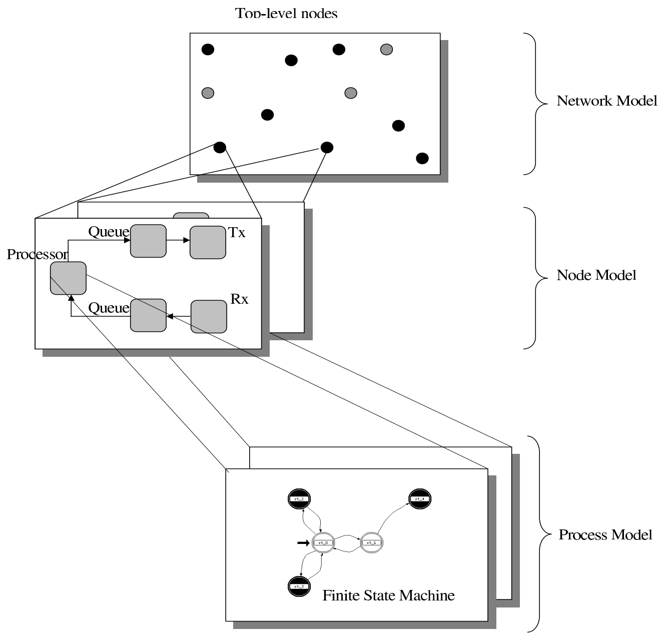


Figure 5.1: The three level hierarchy for an Opnet simulation.

documentation and technical support. A simulation in Opnet consists of a hierarchical structure. This hierarchy is depicted in Figure 5.1 as described in the Opnet documentation [53]. At the top most level, a *network* model describes the positions of the network components and the linkages between them. For a sensor network, these components are the sensor nodes. The next level in the hierarchy is the *node* model. This node model is essentially a description of how an individual node is constructed and of its capabilities. It consists of elements such as processors, queues, packet streams, and radio transceivers. The lowest level of modeling abstraction is the *process* model. A process model is a finite state machine that can be embedded inside a processor in the node model. It specifies the operations of a node as a collection of finite states. These states are entered and exited in response to various types of input, for example, user-specified interrupts, timers, and packet arrivals. These models are constructed using Opnet Radio Modeler v6.0.L, which includes the capability to realistically model radio communication between nodes in the network model.

In simulation terminology, an *event* is the change in the state of the system being modeled [54]. For a *discrete event* simulation, the system state is not defined continuously but only at each event. Opnet is a discrete event simulator. It consists of a *simulation kernel* that generates events for the simulation to progress on the simulation timeline. One of the drawbacks of Opnet is that a packet transmission at a node generates an event at all other nodes in the network [30]. This causes the simulation to slow down considerably.

Figure 5.2 presents a high-level block diagram of the model-development and simulation methodology used to create and study the sensor network. The node positions and TDMA time slots for nodes are generated using Matlab v5.3.1 (this is discussed further in Section 5.2). These are provided as inputs to the simulation kernel. The set of user queries and the simulation models (network, node, and process models) are the other inputs provided to the simulation kernel, which combines these resources and executes the simulation. Finally, results are extracted using the External Model Access (EMA) feature of Opnet and analyzed using Matlab. The following section details the construction of the simulation models and the other components shown in Figure 5.2 .

5.2 Simulation Models

5.2.1 Network Model

As explained earlier, the network model contains a top-level description of all the nodes in the network. This includes their positions and the node models associated with them. The placement of nodes in the network is a topic that has been briefly touched upon in Chapter 2. In determining these placements, one must ensure that enough redundant nodes exist in the network. For these experiments, different sets of positions are generated to gauge the effect of node density and placement on the performance of the tasking algorithms. Figure 5.3(a) shows the placement of nodes in a 2-D hexagonal grid structure such that all nodes are equidistant from each other, thereby creating equal sensor coverage overlap in all directions. Simulations are also conducted by placing nodes randomly in a given area, this is shown in Figure 5.3(b)

A Time Division Multiple Access (TDMA) scheme is chosen for controlling access to the radio medium [6]. This scheme has the advantage of avoiding collisions and contention for the medium. A localized scheme for assigning TDMA-like time slots for sensor nodes is described in [29]. For these

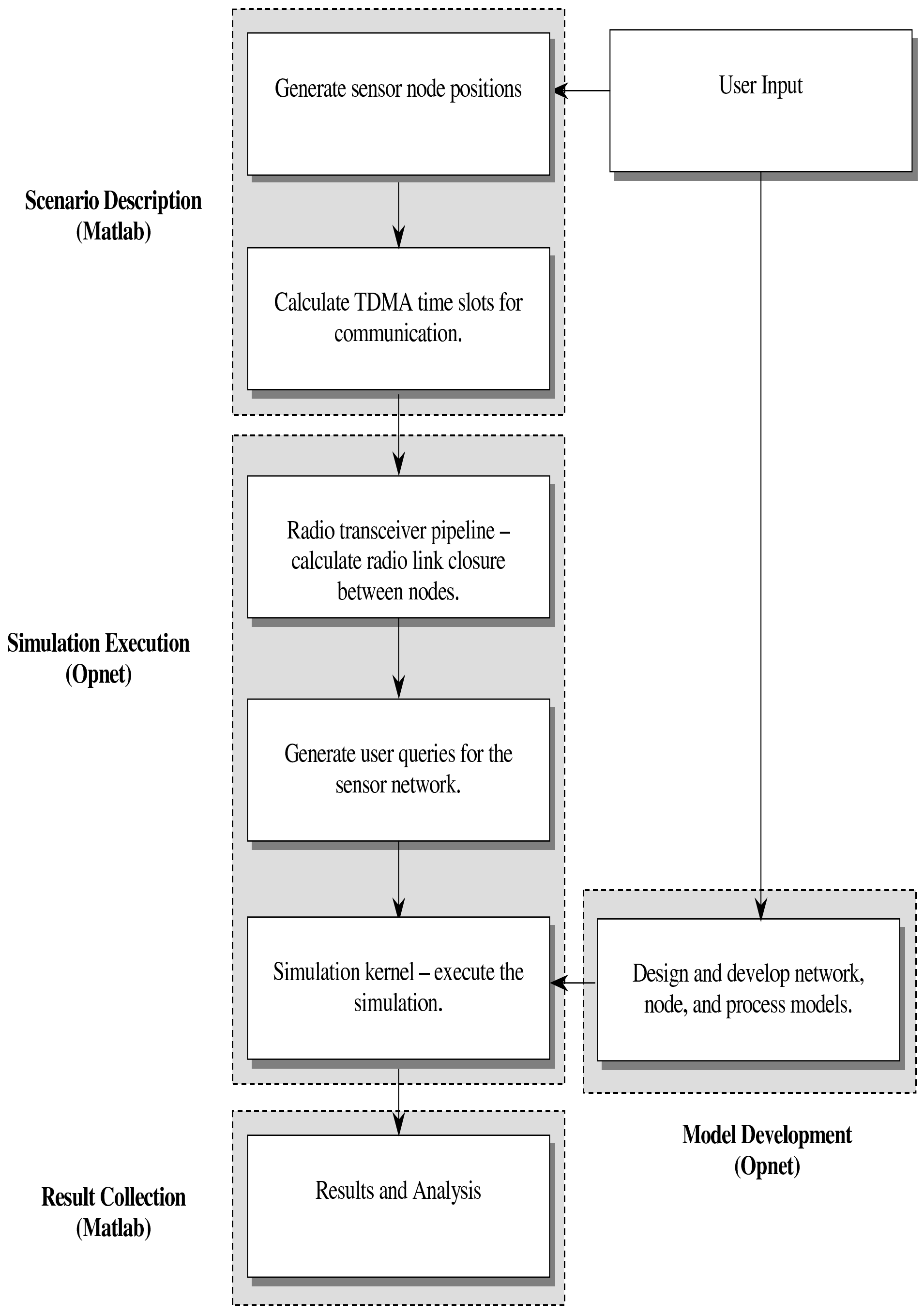
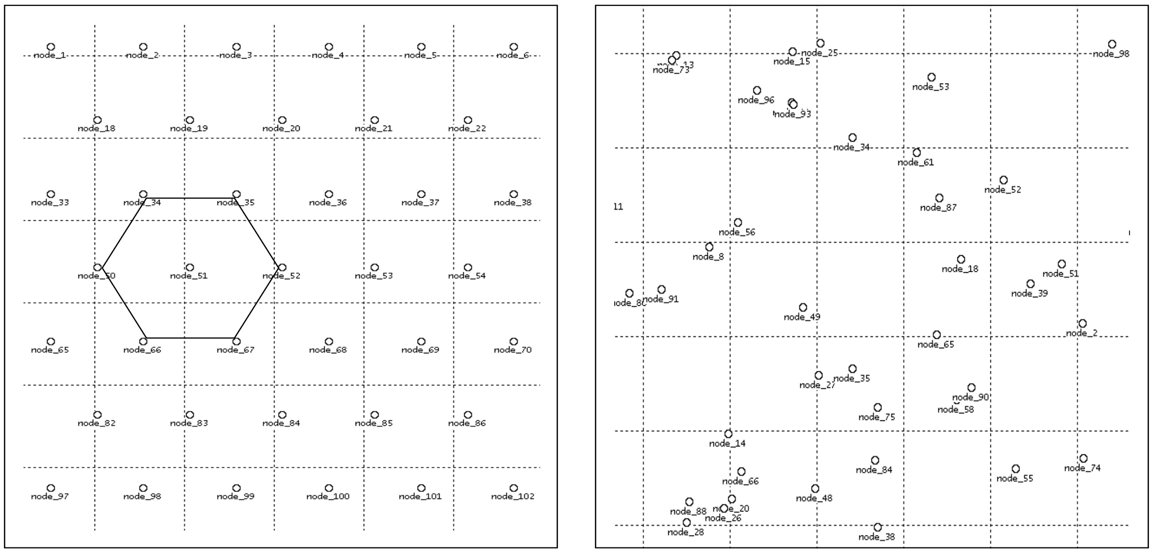

Scenario Description
(Matlab)
Generate sensor node positions
User Input
Calculate TDMA time slots for communication.
Simulation Execution
(Opnet)
Radio transceiver pipeline – calculate radio link closure between nodes.
Generate user queries for the sensor network.
Simulation kernel – execute the simulation.
Design and develop network, node, and process models.
Model Development
(Opnet)
Result Collection
(Matlab)
Results and Analysis

Figure 5.2: High-level block diagram of the model development and simulation methodology.



(a) Hexagonal placement of nodes

(b) Random Placement of nodes

Figure 5.3: Network model of a distributed sensor network.

experiments time slots are assigned centrally. The radio connectivity graph is used to calculate the set of nodes that are at a distance of more than 2 hops from each other. These nodes can be colored with the same color (or have the same time slot) because their transmissions will not interfere with each other. This can be seen from Figure 5.4 where two nodes s_i and s_j that are more than two radio hops apart can be assigned the same time slot. This is because a transmission from node s_i cannot be heard by node s_h, and a transmission from node s_j cannot be heard by node s_g. A vertex coloring obtained using this scheme yields the time slot schedule for the nodes. This schedule is provided as an input to the nodes in the network model.

5.2.2 Node Model

The node model of a sensor node is shown in Figure 5.5. A node consists of a processor, a queue, and transmitter/receiver modules. The processor contains the finite state machine used to control all the operations of the node. The queue is used to synchronize and schedule packet transmissions for the TDMA scheme. The radio transmitter/receiver modules support a bit-rate of 2.4 Kbps

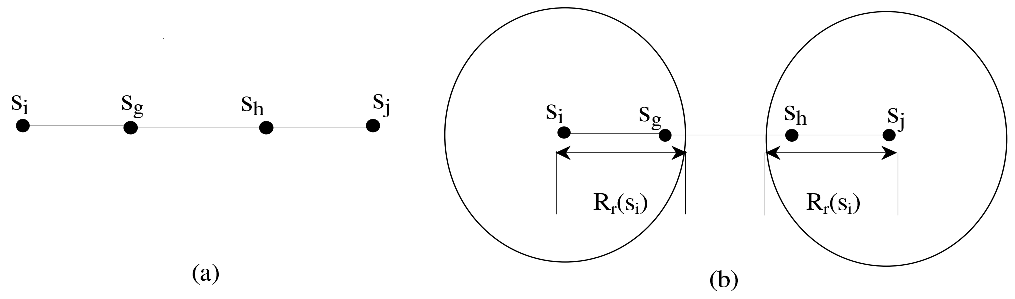


Figure 5.4: Assignment of time slots to two nodes that are more than 2 hops apart. (a) Radio connectivity graph for a set of nodes. (b) Nodes s_i and s_j can be assigned the same time slot. $R_r(s_i)$ represents the radio range of a node.

over a 25m range and operate in the 915 MHz range. The radio has two modes of operation; it can be configured such that communication between nodes is purely nearest neighbor or single hop. Alternatively, it can be used such that packets are forwarded over multiple hops. Multi-hop communication is described in more detail in Section 5.5. In addition to its radio capabilities, each node has two sensing elements; a seismic sensor capable of detecting activity at 10m, and a acoustic sensor with a range of 30m. Note that the range of the seismic sensor is smaller than the radio range whereas the range of the acoustic sensor is larger than the radio range. The seismic and acoustic sensing applications are used to demonstrate the effect of using connectivity graphs based on radio and sensor ranges respectively.

For nodes equipped with radio transmitters and receivers, Opnet executes the initial stage of a *radio transceiver pipeline* to determine the set of transmitters and receivers in the network that are capable of communicating with each other. The initial stage of the pipeline is executed only once at the beginning of each simulation run. Subsequent stages of the pipeline are executed once for each transmission to ascertain properties such as SNR, transmission delay, bit error rate, and other statistics.

The node energy model is based on the work of Stemm and Katz [23] and Srivastava, *et. al.* [24]. Nodes contain a battery source that is modeled as a bucket of energy. At the start of the simulation each node begins with a fixed energy quota of 1 Joule. Energy is drawn from this source based on the current node state, i.e. *asleep, awake, transmit*, or *receive*. Define T_{bl} to be the lifetime of an *awake*

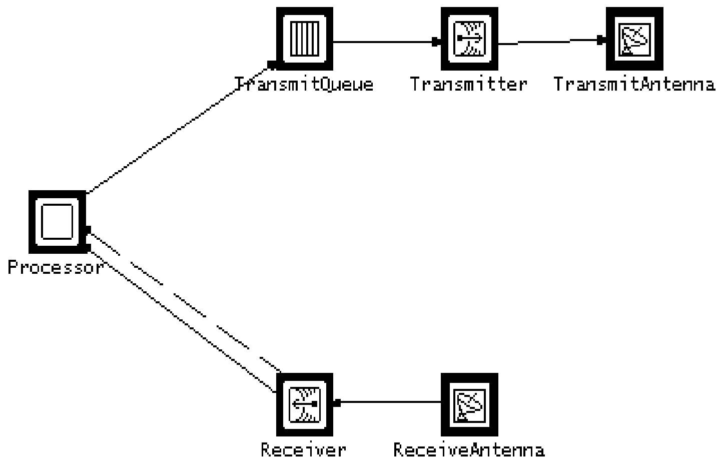


Figure 5.5: Node model of an individual node in the sensor network.

node that does not engage in any sensing or communication task. This value is calculated to be 24 hours based on the measurements conducted by Srivastava, *et. al.* [24] to gauge the power consumption rates of prototype nodes. The energy depleted in the other operational states is then computed according to the following ratios (*asleep : awake : receive : transmit*) (0 : 1 : 1.034 : 1.531). Additional energy is consumed by a node if it is engaged in sensing tasks; this extra energy is modeled as a linearly increasing function of the number of currently active tasks. The rates at which energy is consumed for each of these individual operations are combined into one equation that is used to determine the total energy expended by a node over a given period. Some terminology is introduced prior to introducing this equation.

s = Current operational state of a node, i.e., *asleep, awake, transmit*, or *receive.*
n_t = Current number of tasks at a node.
r_{total} = The rate at which energy is consumed at a node.
$r_{state}(s)$ = The rate at which energy is consumed in state s.
r_{task} = The rate at which energy is consumed for one task.

$$r_{total}(s, n_t) = r_{state}(s) + (r_{task} \times n_t) \tag{5.1}$$

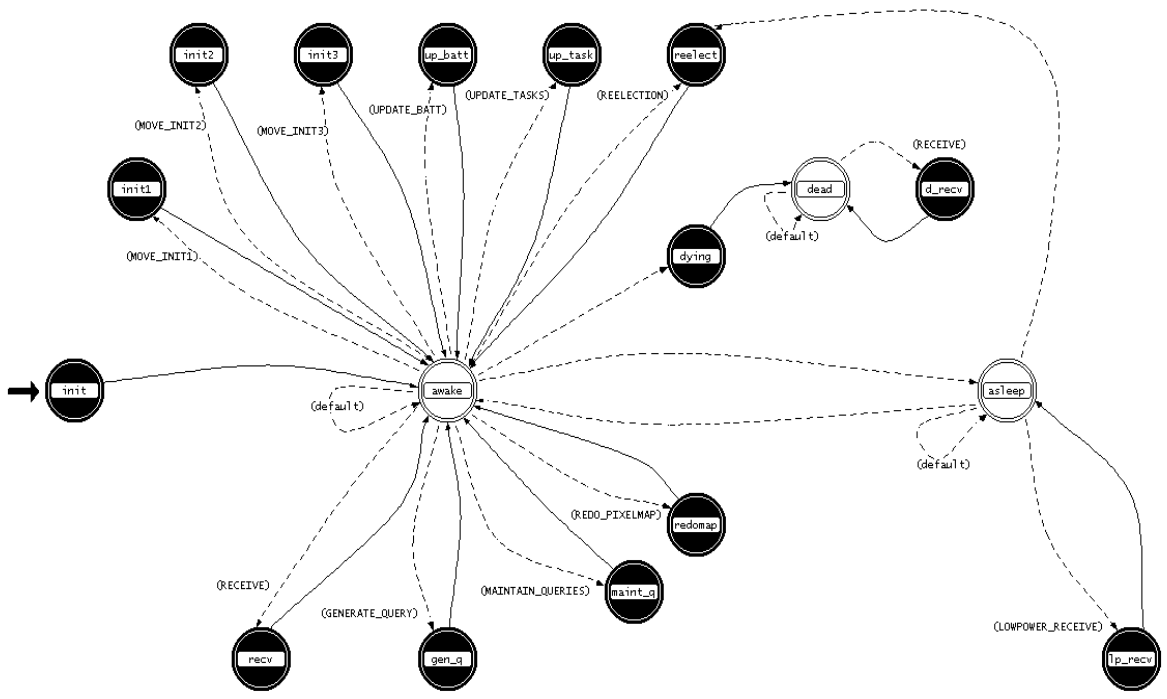


Figure 5.6: Process model of an individual node in the sensor network.

5.2.3 Process Model

The process model is the finite state machine that is used to represent the actual functioning of a node in the network. The basic process model used to model the operation of a sensor node is shown in Figure 5.6. Each state is associated with some operation performed by a node. These operations are implemented in the C programming language using the Opnet Application Programming Interface (API). Opnet provides extensive libraries to model common communication and processing operations.

The dark colored states in the state machine are *forced* states. After a transition to a forced state, the state machine returns automatically to the state from which the transition was invoked. The light colored states are *unforced* states. A node requires an explicit transition to exit from an unforced state. The unforced states in this state machine are similar to the operational states of

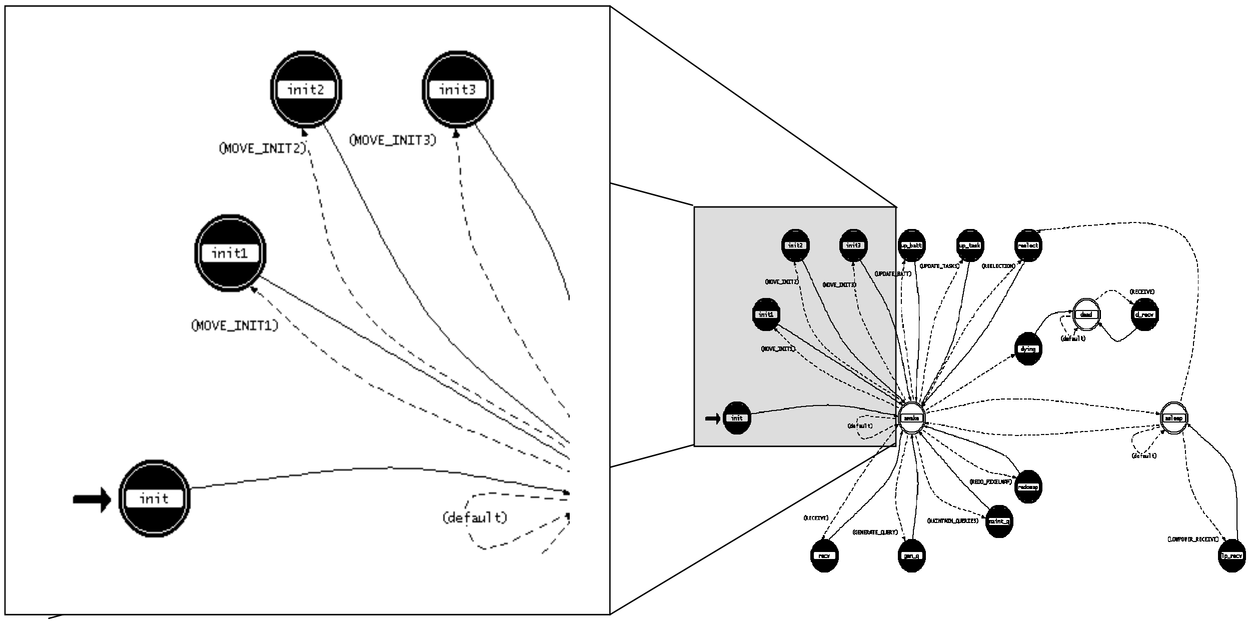


Figure 5.7: Process model: states that contain logic for the election of AQSs.

a node shown in Figure 4.1. This state machine has two distinct parts that correspond to the two stages of the distributed tasking algorithm.

The first part contains logic for the election of AQSs using the maximal independent set algorithm. Figure 5.7 shows the states associated with the election of AQSs. The distributed election algorithm proceeds as follows. In state `init`, nodes broadcast an initial message that contains their location and their $r(i)$ value. This initial message is used to build a list of neighbors at each node. In state `init1`, a node compares its $r(i)$ values with each of its neighbors' to determine if its $r(i)$ is the largest. If so, it assumes the responsibility of being an AQS; if not, it broadcasts a message claiming its lack of knowledge about an AQS. Any node that hears a message from a neighbor claiming to be an AQS knows that it is now managed by that AQS. Therefore, it broadcasts a message indicating that it does not wish to be an AQS. Nodes enter state `init2` periodically at the expiration of a timer. Each time state `init2` is entered, a node checks to see if its $r(i)$ value is the largest amongst all its neighbors that *do not* have any knowledge of an AQS; if so, it elects itself as an AQS and sends out a broadcast message to that effect. As the algorithm iterates, more and more nodes hear about their neighbors' inclusion in the set of AQSs, or if the neighbors have decided not be AQSs. This causes these nodes to take a decision on whether they wish to continue being involved in the election process (if they still do not know of an AQS) or wish to drop out (if a neighbor has advertised itself as an AQS). This iterative process continues until the expiration of a final timer that causes each node to transition to state `init3`. In `init3`, each node checks to see if it is aware of *any* neighboring AQS. If not, it elects itself to the set of AQSs. This last step guarantees that each and every node is aware of at least one AQS.

The second part of the state machine contains logic for AQS nodes to manage and distribute user queries. It also contains logic for non-AQS nodes to perform sensing tasks or sleep until woken up by the AQS. Though not strictly required by the tasking approach described in Chapter 4, in this implementation AQSs remains in the `awake` state at all times to manage existing tasks and to assign new tasks. These operations are performed in states `maint_q` and `gen_q` respectively (Figure 5.6). Each AQS also periodically re-computes the grid map used to assign tasks; this operation is performed in state `redomap`. A non-AQS node transitions immediately to the `asleep` state after an election if it has not been assigned any tasks. A node transitions from the `asleep` state back to the `awake` state at the receipt of a low-power wakeup signal (described in Section 4.1). This wakeup signal precedes a request to task that node. A node that has active tasks frequently updates its task status in state `up_task` and reports its updated battery values to the AQS in state `up_batt`. All nodes periodically enter the `reelect` state to begin another round of elections. Finally, when the battery of a node falls below 10% of the original energy value it transitions to the `dead` state.

5.3 Simulation Scenarios

As part of these experiments, realistic scenarios are constructed to represent situations in which distributed sensor networks may be employed. A scenario contains a set of parameters that define the purpose of the network, the positions where nodes are placed, and the queries posed to the network. The following two scenarios are constructed to evaluate performance criteria pertaining to energy, effective coverage, and network lifetime.

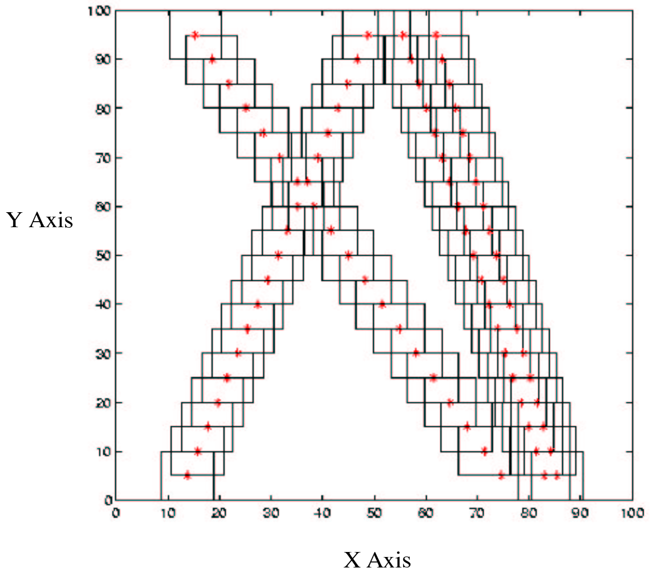


Figure 5.8: Scenario I: Users traversing a sensor field. The rectangular areas represent the queries dispatched by users to the sensor network.

5.3.1 Scenario I: User Traversing a Sensor Field

Large sensor networks can monitor vast expanses of territory and be used for providing situational awareness to users traversing through a region. Such a system may be employed for both military and civilian applications. In Scenario I, a set of nodes is deployed in a rectangular region. Users walk through the region along a straight line at a constant speed. The points of entry and egress into the region are randomly generated for each user. As the user walks through the sensor field, he generates a query for the area immediately surrounding him such that the nodes in that region are tasked to return information about the environment. This scenario is represented in Figure 5.8. The duration of a query is chosen such that multiple queries may be active at any given time along the straight line path.

Nodes are placed in a 2-D hexagonal grid structure for this scenario. This configuration ensures constant inter-node separation and sensor coverage overlap for all nodes. Experiments are conducted using this hexagonal structure and also a random placement of nodes. Results presented in Chapter 6 indicate the former placement scheme to be more efficient.

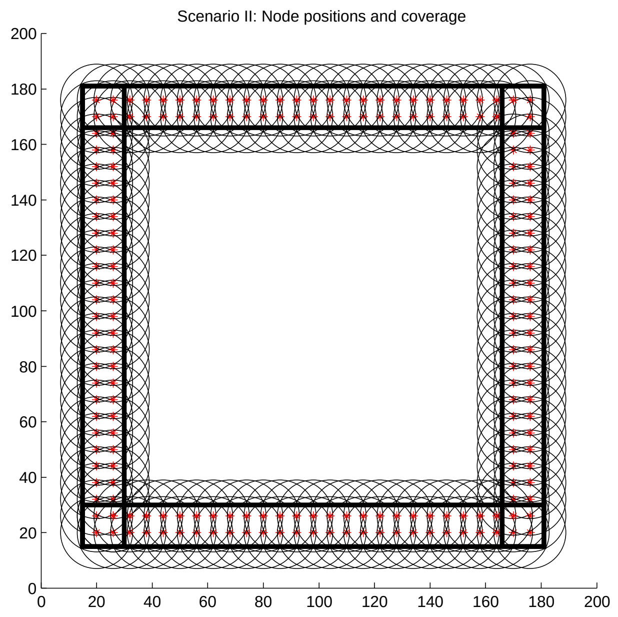


Figure 5.9: Scenario II: Monitoring a fixed perimeter. A circle represents the coverage radius of a sensor. The rectangles represent the four queries along the sides of the perimeter.

5.3.2 Scenario II: Perimeter Monitoring

In Scenario II, the sensor network is employed to detect intrusions across a fixed perimeter. The sensor network is continuously tasked with the responsibility of detecting intrusions across the perimeter. The perimeter is configured such that it contains an outer and an inner perimeter. A *breach* occurs when the sensor coverage between the outer and inner perimeter degrades to the point of allowing an entity to cross undetected.

In this case, the sensor nodes are evenly distributed along the sides of the perimeter as shown in Figure 5.9. Scenario II differs from Scenario I in the volume and nature of queries; it attempts to load the network with large, continuously operating queries.

5.4 Communication Faults

One of the goals of these distributed algorithms is to provide a robust, fault-tolerant method for the election of AQSs. In these simulations, the operation of the algorithm is studied for unreliable links. This section presents the experimental scheme used to generate and study these communication faults.

A packet transmitted over a communication link is prone to many types of transmission errors at various layers in the network protocol stack [55]. For this study, the interest is primarily in the fact that such errors may cause a packet not to be delivered to its intended recipient. As discussed in Section 3.2, the election algorithm is structured such that more AQSs may be chosen if the communication between nodes is unreliable. The election of excess AQSs can be intuitively seen to impact overall energy savings, and is therefore not a desired result. However, if the link is unreliable, then electing these extra AQSs may be the only available option. To study the effect of unreliable links on the election algorithm, packet errors are introduced by dropping the packets received at a node with a given probability ρ. The decision to drop received packets instead of transmitted packets allows for at least some nodes in the broadcast neighborhood to receive those packets. A point to be noted is that packets are dropped only for the stage of the algorithm that elects AQSs. Experimental results for $\rho = 0.05\%$, 0.5%, 5% , and 50% are given in Chapter 6.

5.5 Multi-Hop Communication Scheme

Radio communication between nodes is only a single hop for the basic mode of operation. In this section the extension of this single hop scheme to a multi-hop scheme is discussed. This may be required, for example, to build an intersection graph based on the sensor range vs. an intersection graph based on radio range (for a related discussion see Section 4.3).

Given that sensor nodes have limited processing capabilities, a scheme to route messages between nodes needs to be simple and light-weight [56]. One of the ways of allowing messages to propagate over multiple hops is by *flooding* packets to neighboring nodes [56, 27]. A major problem with routing messages using a flooding scheme that employs broadcasting is that it results in many redundant transmissions [57]. These redundant transmissions are expensive for sensor nodes that have constrained energy resources, and may lead to excessive time spent forwarding third-party messages. In this thesis, a simple forwarding scheme is implemented that operates by selectively re-broadcasting the messages received at intermediate nodes. A set of rules are devised to suppress redundant broadcasts at each node and allow data to propagate in the direction of the destination node. This implementation of this scheme is motivated by and similar to a diffusion routing approach [28] that allows data to propagate in the direction of greatest gradient.

In these simulations nodes basically send out two types of messages. The first type of message specifies a destination node for the message, for example, a message sent by a node indicating its battery status to an AQS. The second type of message does not have a specific destination and is intended for all nodes, for example, a message sent by a node that claims to be an AQS. While the first type of message is easy to suppress, the second type is more problematic because it needs to propagate to all nodes within a given distance of the originating node. The rules to suppress these redundant broadcasts are now presented.

Rule 5.1 *Suppression based on the distance travelled by a message.* For building intersection graphs based on the sensor range it is required that two nodes with overlapping sensor coverage communicate with each other. In Figure 5.10, consider the node s_i that broadcasts a message. If the distance of an intermediate node s_j from node s_i is less than twice the effective coverage range of sensor, $2 \times R_s((s_i)_k)$, then the message is re-broadcast otherwise it is dropped. It can be seen from Figure 5.10 that re-broadcasting this message for a node that is beyond a distance of $2 \times R_s((s_i)_k)$ from the original sender serves no purpose. This may not always be the case. A message may need to reach its destination by being re-broadcast from a node that is at a distance greater than $2 \times R_s((s_i)_k)$. However, this case may not occur very often given the density of such a network.

Rule 5.2 *Suppression based on the number of hops travelled by a message.* A limit on the number of hops can be used to aggressively limit the actual number of re-transmissions. This is so because enforcing Rule 5.1 may still allow a message bound for all nodes to propagate over a fairly large

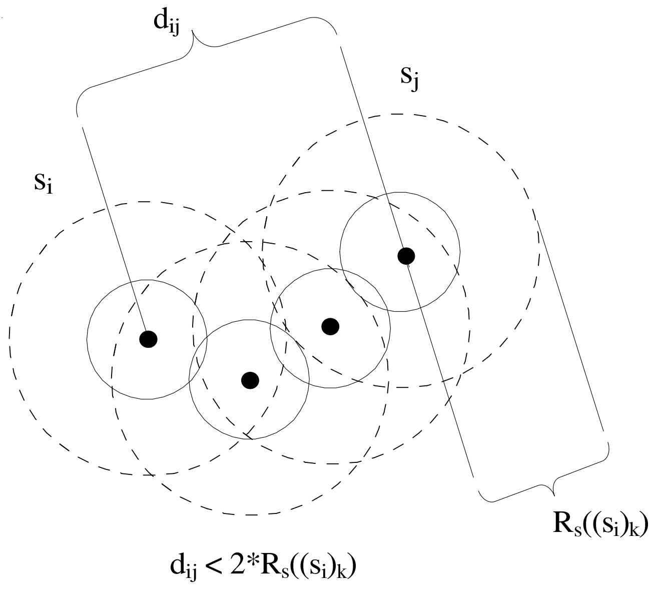


Figure 5.10: Suppression of broadcasts based on the distance travelled by a message. The circle with the broken line represents the sensor range whereas the circle with the unbroken line represents the radio range.

distance, thereby causing an unacceptable number of re-transmissions. The number of hops a message can travel is a parameter for these simulation experiments. The performance of the AQS election algorithm is evaluated as a function of this parameter and is presented in Chapter 6.

Rule 5.3 *Suppression based on the sequence numbers.* The use of sequence numbers allows the detection of duplicate messages. Each message carries the identifier of the node that originated the message and a unique sequence number for that particular message. Each node that receives this message updates the {*source identifier, sequence number*} tuple it maintains for the node that sent the message. Any copy of this message received subsequently, i.e., any message that carries a sequence number lower than the one currently stored is dropped.

Rule 5.4 *Suppression of a message to the north/south plane of propagation.* Consider Figure 5.11(a). The north plane of propagation with respect to a node s_i is defined to be quadrants I and II. Similarly, the south plane of propagation is defined to be quadrants III and IV. In Figure 5.11(b),

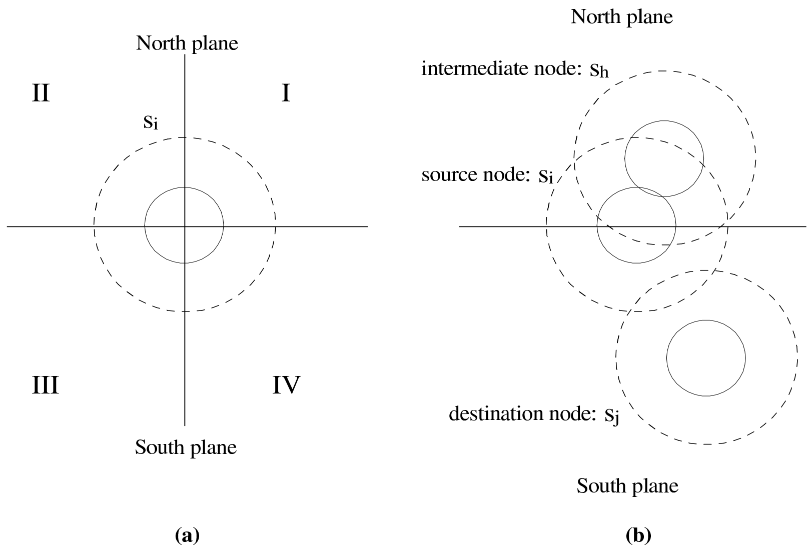


Figure 5.11: Suppression of a message to the north/south plane of propagation. The circle with the broken line represents the sensor range whereas the circle with the unbroken line represents the radio range.

a node s_i broadcasts a message that is intended for node s_j in its south plane. A node s_h may re-broadcast this message because it can hear the original transmission from node s_i. However, for this configuration this transmission by s_h serves no purpose because it is in the north plane. Therefore, node s_h drops the message unless it is in the same plane of propagation. It is possible, though rare given the density of such a network, that a message bound for a node in one plane *needs* to travel to its destination via the other plane. This potential drawback is accepted given that this rule offers the opportunity to reduce the re-transmissions by nearly half (assuming an equal number of nodes in the north and south planes).

Rule 5.5 *Suppression of a message if the distance between the source node and the destination node is less than the distance between source node and the intermediate node.* Consider the situation illustrated in Figure 5.12. A node s_i is sending a message to node s_j. This message also propagates to node s_h because it does not contradict any of the other rules. A re-broadcast by node s_h is not required because its distance from the source, d_{ih}, is larger than the distance of the source from the destination, d_{ij}. This rules makes the assumption that if the destination is closer to the

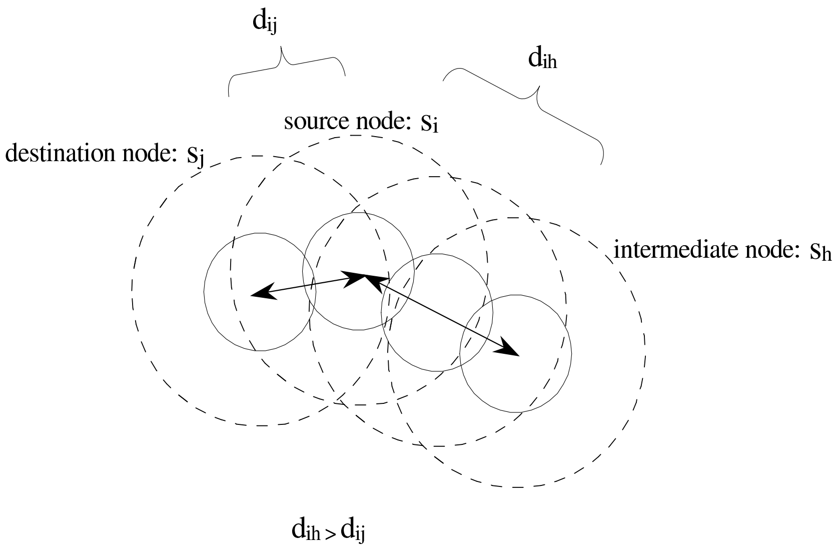


Figure 5.12: Suppression of a message if $d_{ih} > d_{ij}$. The circle with the broken line represents the sensor range whereas the circle with the unbroken line represents the radio range.

source than the intermediate node then the destination has already received the message via an alternative route. Enforcing this rule may lead to a situation where an intermediate node that is essential to the forwarding process drops the message. Nevertheless, as for Rules 5.4 and 5.1 this may not occur often for a redundant sensor field and is accepted as a potential drawback.

A point to be noted is that the rules described above are not meant to replace or provide an alternative routing scheme for sensor networks; they are heuristics meant to suppress the excess traffic in the network for a scenario in which messages must propagate over multiple hops. They are implemented primarily to study the effect of constructing different types of intersection graphs on the performance of the election algorithm.

5.6 Summary

This chapter discusses the simulation environment and the models that are used to study the operation of the sensor network. An overview of the simulation tool, Opnet, is presented. A hierarchical model of the sensor network is constructed by using network, node, and process models. A description of two scenarios where such networks may be utilized is presented. Simulation models are constructed based on these scenarios; the results obtained from these models are presented in Chapter 6. Communication faults such as packet losses and link failures are introduced in the models to understand the effect of such faults on network lifetime and energy consumption. Finally, the implementation of multi-hop communication between nodes is discussed. As a part of this discussion, rules to suppress redundant message transmissions are formulated.

Chapter 6

Results

This chapter presents the results obtained by simulating the sensor network for the two scenarios previously described in Chapter 5. These simulations are executed on a machine with a 1 GHz Pentium III processor and 1536 MB of RAM. The operating system used is Windows Advanced Server 2000. The simulation experiments are organized as follows. A group of individual simulations that share a common characteristic but differ in one simulation parameter are logically grouped together as a simulation set for ease of presentation. Such a set could be, for example, the set of all simulations with n nodes placed in an area of size $(l_x \times l_y)\ m^2$, where each simulation differs in the value of the redundancy index, RI.

Prior to presenting the results for these simulation sets, a group of metrics for energy usage and quality of sensor coverage are defined. Simulation sets are evaluated individually with respect to these metrics. Different simulation sets are also compared with each other to demonstrate the sensitivity of a simulation parameter for a particular configuration of the network.

6.1 Definition of Metrics

6.1.1 Metrics for Energy Usage and Characterization

Metric 1 *Average energy value of all the nodes in the network.* This measure is a simple expression of the total energy content of the network at a given time. When plotted against time it gives an indication of the lifetime of the network.

Metric 2 *Minimum energy of all the nodes in the network.* One of the attempts of the tasking algorithm is to balance power consumption across all nodes. The minimum energy value indicates how well that objective is achieved. If the tasking algorithm is able to appreciably prolong the life of the node with the least energy, then this objective is attained.

Metric 3 *Total energy consumed in the awake, transmit and receive states for each node.* The breakup of the energy consumed in each of these states provides insight into the modes of operation of individual nodes. Efforts to further reduce the energy consumption need to be aware of these statistics to target states with the greatest potential for energy savings.

Metric 4 *Maximum time spent by any node in electing a new AQS.* As the network size increases, the scalability and running time of the distributed election algorithm is tested by calculating how much time is spent in choosing AQSs. This value of this statistic is computed to be the maximum time taken by any node to determine an AQS. This is because the election terminates only when each and every node knows of an AQS.

Metric 5 *Number of elected AQSs.* As explained in Chapter 3, extra elected AQSs may be inefficient but essential if there are too many errors in the packets being transmitted or received. This metric is used to gauge how many extra AQSs are elected for one election if the links between nodes are unreliable, thereby causing packet loss.

Besides the basic metrics defined above, additional metrics for measuring the performance of a multi-hop communication scheme are also defined (multi-hop communication is employed to construct an intersection graph based on sensor range as explained in Chapter 4). As the number of hops that messages can travel increases more nodes hear of each others' messages. This leads to the creation of a denser intersection graph. Though this may be beneficial in selecting fewer AQSs, it enlarges the neighbor set thereby increasing inter-node communication. These additional metrics attempt to evaluate the operation of the distributed algorithms as a function the maximum number of hops that a message can travel.

Metric 6 *Maximum number of packets received at any node for one round of AQS elections.* The number of packets received by a node that relate to an election is an indicator of the time and energy spent by the node in the election process. Simulations are evaluated using this metric to understand the energy consumption pattern for elections as a function of the number of hops.

Metric 7 *Maximum ratio of (the network traffic at any node exclusive to one election) and (the total network traffic at any node over a fixed time interval).* This ratio indicates the relative amount of energy spent by a node for an election vs. the amount of energy consumed for other types of communication, such as tasking requests and periodic battery updates. This metric is significant because the network traffic for elections may increase at a faster rate than the network traffic for other types of activities as the number of hops increases. This may negatively impact the overall life of the network if excessive energy is being spent in electing AQSs.

6.1.2 Metrics for Quality of Sensor Coverage

Along with extending the lifetime of the network, it is important to ensure that the network is delivering on its overall objective, i.e. good sensor coverage throughout the lifetime of the network. Two measures are defined to gauge the current sensor coverage in the network.

Metric 8 *% of uncovered area in the region.* This metric is defined as the percentage of area over the entire region not covered by any sensor at a given time. When plotted against time, it gives an assessment of how long the network is able to achieve acceptable levels of coverage.

Metric 9 *Time at which the first breach occurs.* For Scenario II, an obvious quality measure is how long the integrity of the perimeter being sensed is maintained. When enough nodes have died so as to enable an entity to cross the perimeter without detection, a breach occurs and the sensor network effectively fails.

6.2 Scenario I

6.2.1 Experiments with Constant Node Density

Tables 6.1 and 6.2 list the parameters that define simulation sets #1 and #2 respectively. In Set #1, different numbers of nodes are placed in a regular 2-D hexagonal grid structure. These nodes are placed such that the node density or the inter-node separation is constant. In contrast, in Set #2 different numbers of nodes are placed randomly in a rectangular area. Because nodes are placed randomly for Set #2, the node density is defined as the ratio of the number of nodes and the total area in which these nodes are placed. For each Simulation ID in a set, simulations are conducted using Approach I and Approach II. As explained earlier in Chapter 4, Approach I is the simplistic tasking approach that is used to benchmark the distributed tasking approach, i.e., Approach II.

Table 6.1: Set #1 (Scenario I, **C**onstant Node **D**ensity, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*	Area
CDH100	100	0.90	100m x 100m
CDH256	256	0.90	160m x 150m
CDH400	400	0.90	200m x 185m
CDH900	900	0.90	300m x 275m

Results for Metric 1 (average energy) and Metric 2 (minimum energy) are shown in Figure 6.1 and Figure 6.2 respectively. The metrics are plotted as follows; the time taken in hours for the average/minimum energy to fall to 50% of the original is plotted on the Y axis vs. the Simulation ID that is plotted on the X axis. From Figure 6.1 it can be seen that Approach II shows a marked

Table 6.2: Set #2 (Scenario I, **C**onstant Node **D**ensity, **R**andom Node Placements)

Sim ID	# of Nodes	*RI*	Area
CDR100	100	0.90	100m x 100m
CDR256	256	0.90	160m x 150m
CDR400	400	0.90	200m x 185m
CDR900	900	0.90	300m x 275m

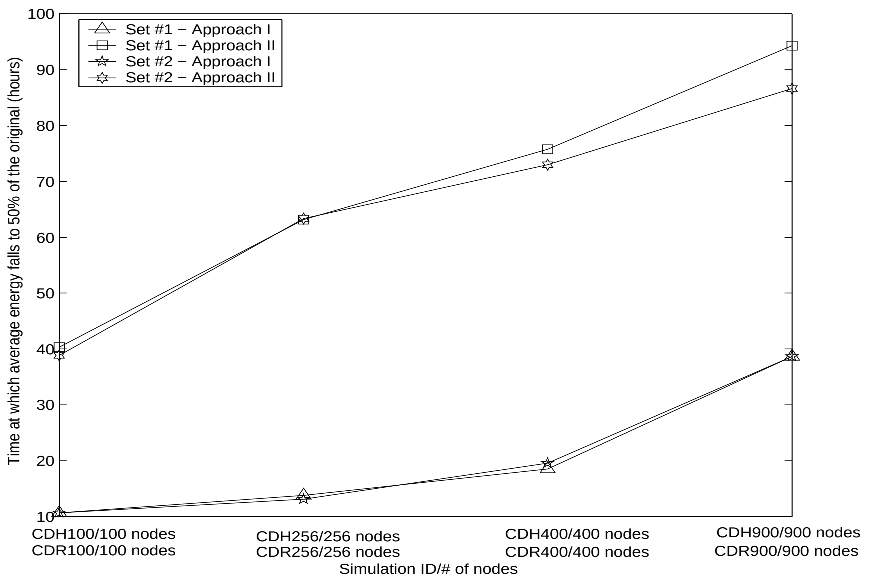


Figure 6.1: Sets #1 and #2: Metric 1, Time taken for the average energy of all nodes to fall to 50% of its original value.

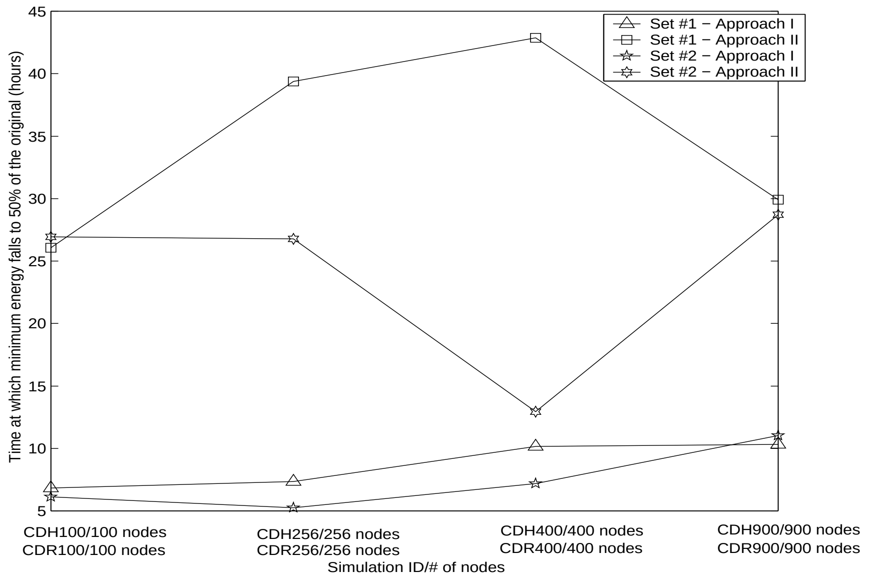


Figure 6.2: Sets #1 and #2: Metric 2, Time taken for the minimum energy of all the nodes to fall to 50% of its original value.

improvement over Approach I in extending the lifetime of the network. Similarly, the minimum energy plot shown in Figure 6.2 indicates the extent to which battery of *weakest* node in the network is protected. This is also shown to improve for Approach II over Approach I.

From these figures it is observed that a comparison of Set #1 and Set #2 with respect to Metric 1 indicates very similar results. However, the hexagonal placement (Set #1) yields better results if we consider Metric 2. This leads to the conclusion that whereas the average energy is largely unaffected by the random placement (as opposed to the hexagonal placement), the effect on the minimum energy is distinctly visible. Notwithstanding this observation, observe that the minimum energy statistic for the random placement does not show a well defined trend. This may be due to areas in the network that are covered only by a single node. If this node is tasked excessively its energy is depleted rapidly. In contrast, if it is not tasked as often the minimum energy statistic may actually improve as can be seen from Figure 6.2. Therefore, it may not be possible to adequately predict the minimum energy value for different random node placements. This a potential drawback for a random placement because the minimum energy value represents the first failure in the network.

6.2.2 Experiments with Varying Node Density

Tables 6.3 and 6.4 list the parameters that define simulation sets #3 and #4 respectively. In Set #3, a fixed number of nodes is placed in different size areas. In contrast, Set #4 contains a group of simulations where different numbers of nodes are placed in a fixed area size. The purpose of these experiments is to evaluate the performance of the system as a function of the node density.

Table 6.3: Set #3 (Scenario I, **V**arying Node **D**ensity, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*	Area
VDH256a	256	0.90	100m x 100m
VDH256b	256	0.90	160m x 150m
VDH256c	256	0.90	200m x 185m
VDH256d	256	0.90	300m x 275m

Table 6.4: Set #4 (Scenario I, **V**arying Node **D**ensity, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*	Area
VDH100e	100	0.90	100m x 100m
VDH256f	256	0.90	100m x 100m
VDH400g	400	0.90	100m x 100m
VDH900h	900	0.90	100m x 100m

Figure 6.3 shows the performance of both sets with respect to Metric 1. For Set #3, consider the average energy plots for Approach II and Approach I. As the distribution of nodes in an area becomes more sparse, the performance of Approach II begins to approach that of Approach I. In contrast, the average energy plots for Set #4 indicate that an increase in node density leads to an appreciable increase in the life of the network.

Figure 6.4 shows the performance of both sets with respect to Metric 2. The trend observed is

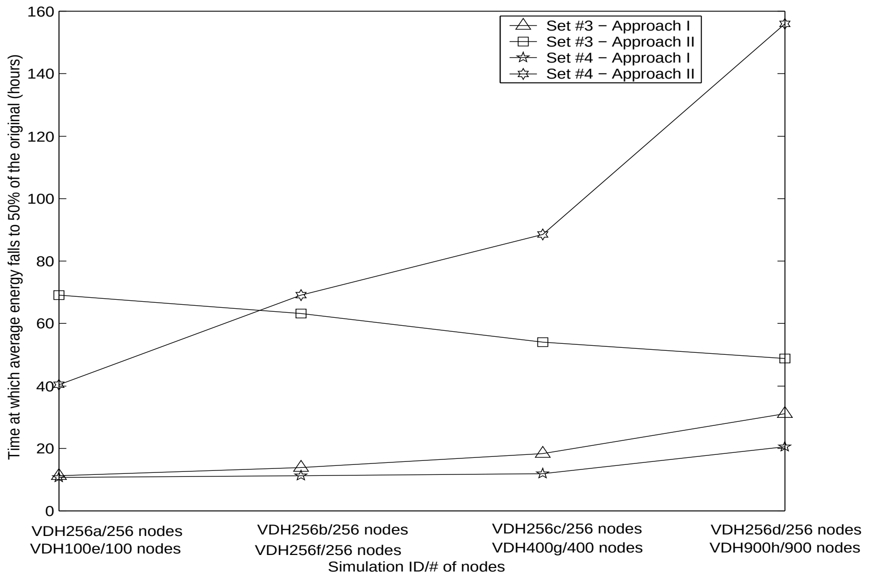


Figure 6.3: Sets #3 and #4: Metric 1, Time taken for the average energy of all nodes to fall to 50% of its original value.

nearly identical to that for Metric 1. A decrease in the node density for the simulations in Set #3 leads to decreased performance. On the contrary, an increase in node density for simulations in Set #4 leads to a marked improvement for the minimum energy metric.

The results obtained from these experiments with node density are expected because Approach II inherently relies on node redundancy to achieve energy savings. As the node density decreases, the tasking algorithm cannot utilize redundancy to rotate task assignments between nodes. On the other hand, as the node density increases substantial energy savings are achieved.

6.2.3 Experiments with Varying Redundancy Index

Tables 6.5 and 6.6 list the parameters that define simulation sets #5 and #6 respectively. The effect of varying the redundancy index (RI) is studied in these sets. In Set #5, nodes are placed in a hexagonal formation; the density and area of the network are constants. In Set #6, nodes are placed randomly; the density and area of the network are constants. For both sets the only variable parameter is the RI.

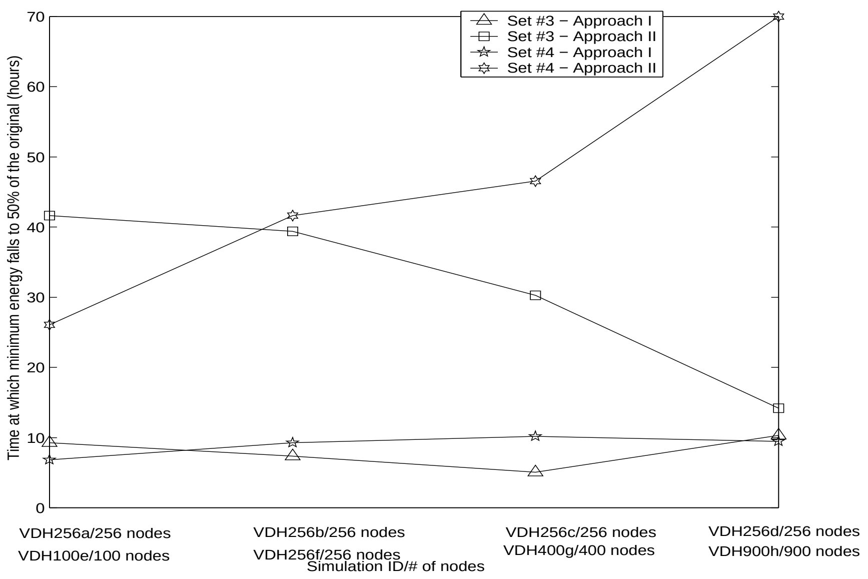


Figure 6.4: Sets #3 and #4: Metric 2, Time taken for the minimum energy of all the nodes to fall to 50% of its original value.

Figure 6.5 shows the performance of both sets with respect to Metric 1. The life of the network does not show any appreciable connection with the *RI* for either the hexagonal (Set #5) or the random (Set #6) placement of nodes. Likewise, the results for Metric 2 shown in Figure 6.6 also do not display any significant trend. Hence, one can conclude that varying the *RI* does not lead to significant increase in performance. This result is somewhat counter-intuitive. It is expected that lower values of *RI* should lead to appreciable increases in the network life.

A possible explanation for this result may be the following. Consider the operation of the tasking algorithm in Figure 4.4. A low value of *RI* may lead to a larger number of redundant nodes. Each time the grid map is redrawn these nodes are replaced with each other. Therefore, nodes may be placed and removed from the grid map much faster for lower values of *RI* as compared to higher values. Depending on how quickly the grid map is recomputed this may cause a larger number of nodes overall to transition to the *awake* state and then back to the *asleep* state. Because the energy consumption is dominated by the *awake* state this may lead to a faster decrease in the energy level.

The above hypothesis is tested using Set #5a and Set #5b which are summarized in Tables 6.7 and

Table 6.5: Set #5 (Scenario I, **V**arying **R**edundancy Index, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*	Area
VRH256a	256	0.90	160m x 150m
VRH256b	256	0.70	160m x 150m
VRH256c	256	0.50	160m x 150m
VRH256d	256	0.30	160m x 150m

Table 6.6: Set #6 (Scenario I, **V**arying **R**edundancy Index, **R**andom Node Placements)

Sim ID	# of Nodes	*RI*	Area
VRR256a	256	0.90	160m x 150m
VRR256b	256	0.70	160m x 150m
VRR256c	256	0.50	160m x 150m
VRR256d	256	0.30	160m x 150m

6.8. In Set #5a, both the RI and re-draw period of the grid map are increased. In Set #5b, the RI is constant and the redraw period is increased; this ensures that any performance gains are not derived solely from increasing the redraw period. Figures 6.7 and 6.8 depict the average energy value plots for these sets. The results indicate that increasing the grid map re-draw period and the RI have a positive effect on the average battery life as was expected.

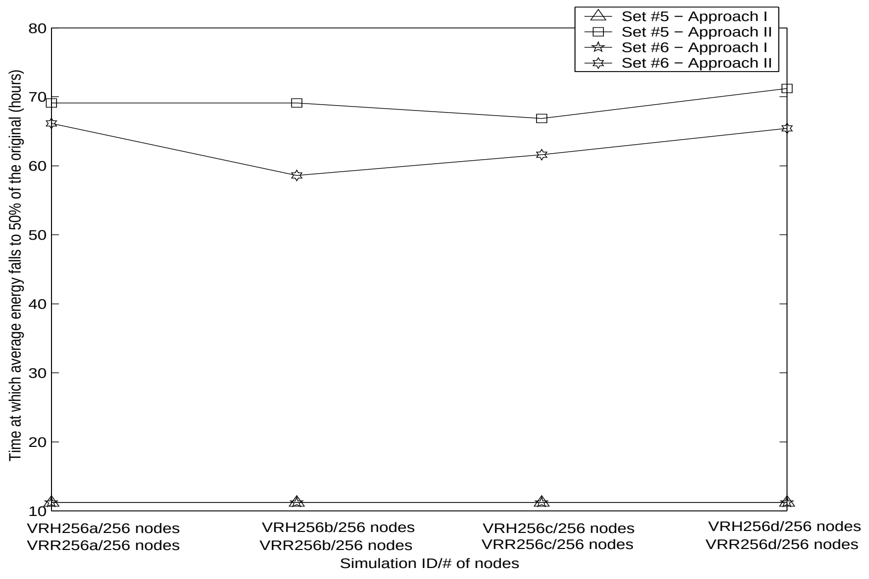


Figure 6.5: Sets #5 and #6: Metric 1, Time taken for the average energy of all nodes to fall to 50% of its original value.

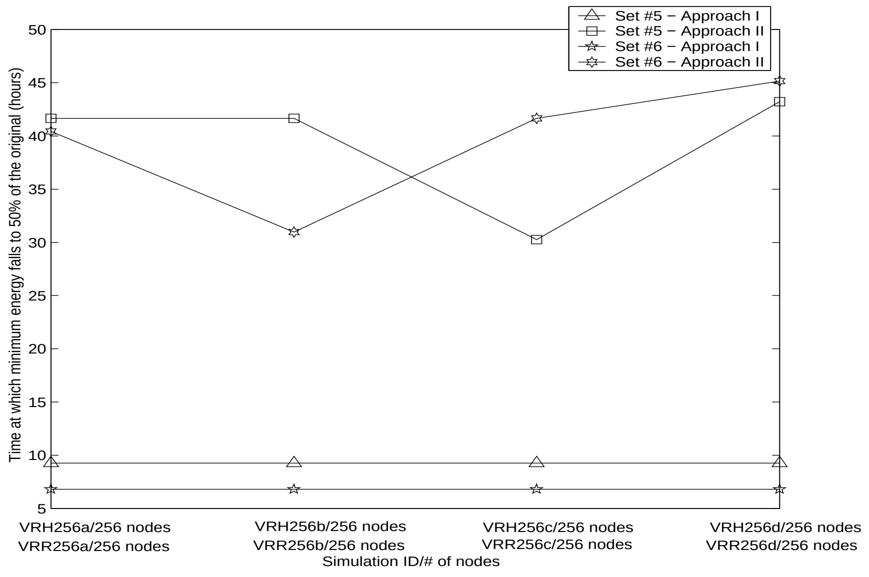


Figure 6.6: Sets #5 and #6: Metric 2, Time taken for the minimum energy of all the nodes to fall to 50% of its original value.

Table 6.7: Set #5a (Scenario I, **V**arying **R**edundancy Index, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*, Re-draw time(mins)	Area
VRH100e	100	0.90, 150	100m x 100m
VRH100f	100	0.70, 300	100m x 100m
VRH100g	100	0.50, 450	100m x 100m
VRH100h	100	0.30, 600	100m x 100m

Table 6.8: Set #5b (Scenario I, **V**arying **R**edundancy Index, **H**exagonal Node Placements)

Sim ID	# of Nodes	*RI*, Re-draw time(mins)	Area
VRH100i	100	0.50, 150	100m x 100m
VRH100j	100	0.50, 300	100m x 100m
VRH100k	100	0.50, 450	100m x 100m
VRH100ℓ	100	0.50, 600	100m x 100m

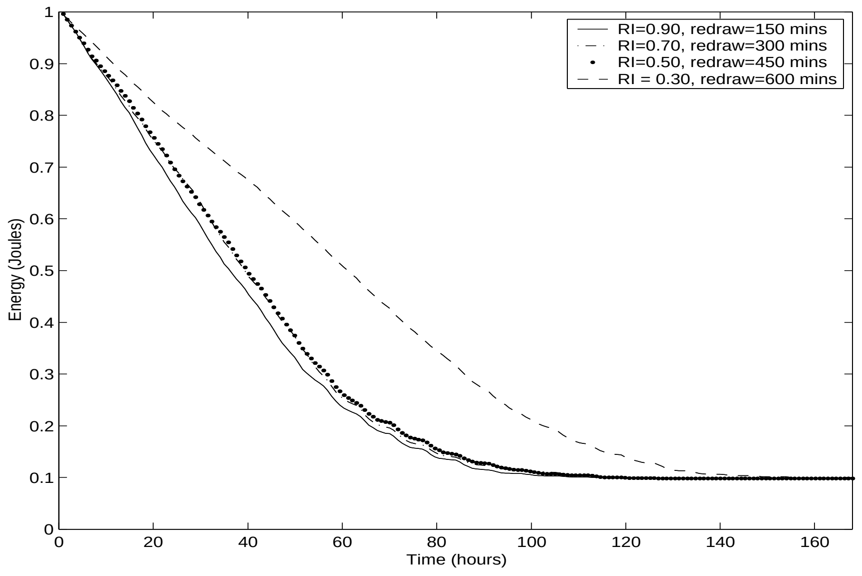


Figure 6.7: Set #5a, average energy values vs. time.

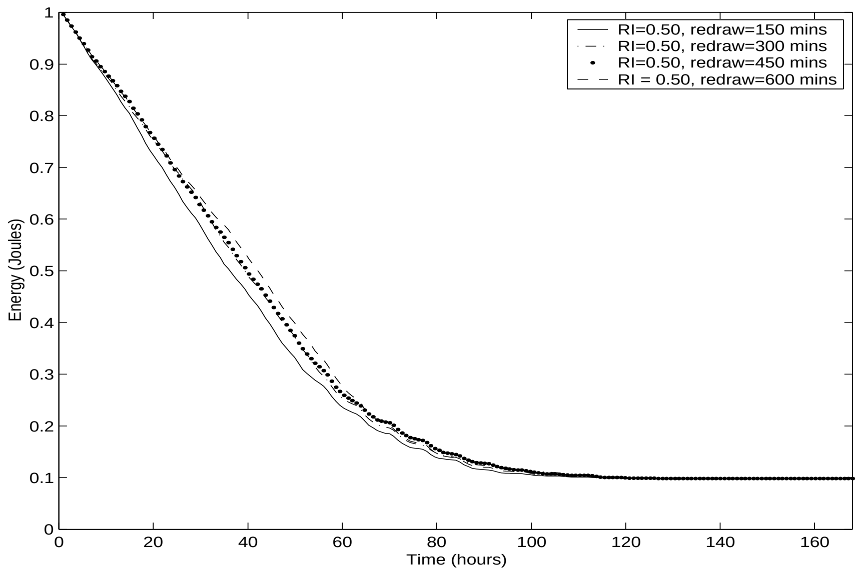


Figure 6.8: Set #5b, average energy values vs. time.

6.2.4 Energy Consumption for a Single Node

Table 6.9 displays the results for the individual energy consumption of a node in the *awake*, *transmit*, and *receive* states for Set #1. Only a small fraction of the total energy is consumed in communication with the balance being consumed in the awake state. Observe that nodes spend more time receiving messages in Approach I than in Approach II. This is an expected result, primarily because in Approach I all nodes in a region are woken up for each task thereby causing them to listen over the radio channel for longer periods.

Table 6.9: Average % energy consumed in the *awake*, *transmit*, and *receive* states

Set #1: Approach I				Set #1: Approach II			
Sim ID	*awake*	*transmit*	*receive*	Sim ID	*awake*	*transmit*	*receive*
CDH100	99.939	0.013	0.048	CDH100	99.506	0.081	0.413
CDH256	98.765	0.039	1.196	CDH256	99.437	0.087	0.477
CDH400	98.806	0.039	1.155	CDH400	99.416	0.086	0.499
CDH100	98.803	0.037	1.161	CDH900	99.404	0.082	0.515

6.2.5 Scalability of the AQS Election Algorithm

The time taken by nodes to elect AQSs is collected to demonstrate that the execution time of the algorithm is scalable as the number of nodes increases as well as to show the effect of node density on the execution time. Figure 6.9 shows the times taken to elect AQS for Set #1 and Set #4. For Set #1, the election time remains nearly constant as the number of nodes increases. This is because the density of nodes is constant. The effect of increasing the node density (Set #4) is to increase the time taken to elect AQSs. This is because the running time of the algorithm is affected by the number of neighbors for each node.

As explained in Section 3.3, the $r(i)$ values used in the election of nodes can be defined as $r(i) = a$, where a is a random number between 0 and 1. Alternatively, the algorithm can be modified to

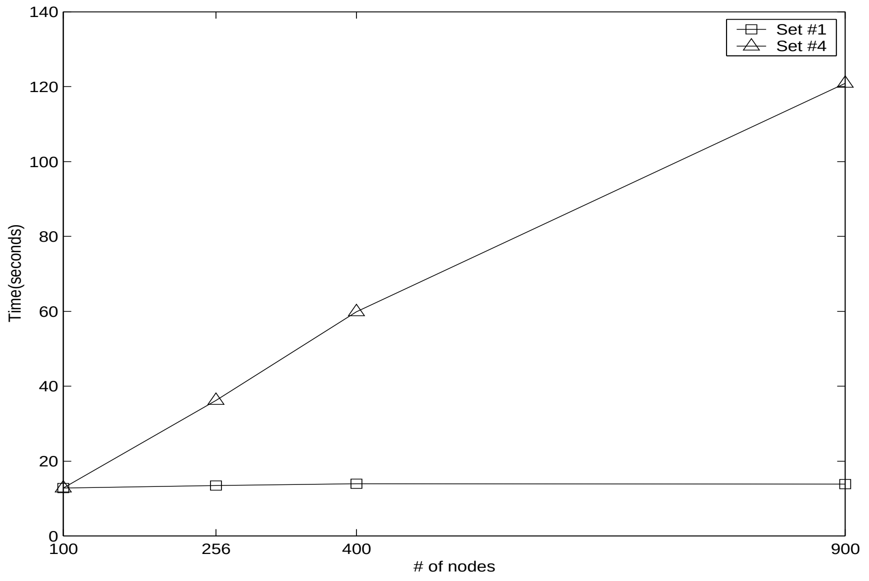


Figure 6.9: Sets #1 and #4: Metric 4, Maximum time spent by any node in electing a new AQS.

use the following definition: $r(i) = x + a$, where x is the remaining battery life and a is a random number between 0 and 1. The latter case allows AQSs to be chosen based on the remaining battery life of a node. The times taken for the AQS election algorithm are compared for both these cases to show that execution times remain bounded in each case. Figure 6.10 presents these results for different numbers of nodes.

6.2.6 Experiments with Varying Packet Error Rates

Tables 6.10, 6.11, 6.12, and 6.13 list the parameters for simulation sets #7, #8, #9 and #10 respectively. These experiments explore the effect of varying the Packet Error Rate (PER) on the performance of the election algorithm as also the overall energy consumption pattern. For each of the sets #7, #8, #9 and #10, experiments are conducted using PER = 0.05 %, 0.5 %, 5.0 %, and 50.0 %. The nodes are placed in a regular hexagonal formation for all the sets.

Figure 6.11 shows the performance of sets #7, #8, #9 and #10 with respect to Metric 5, i.e., the number of AQSs chosen globally for an election. The PER is plotted on a logarithmic scale on the Y axis, whereas the number of elected AQSs is plotted on the X axis. It is observed that the

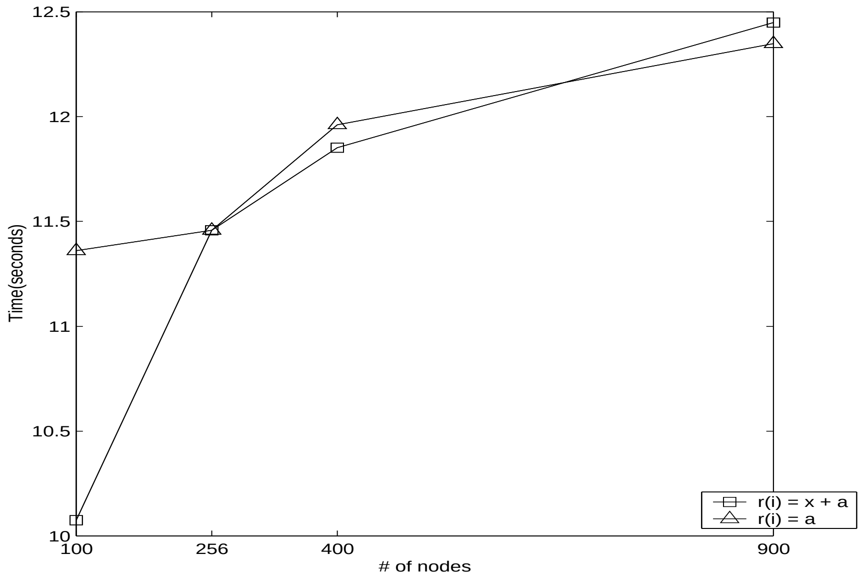


Figure 6.10: Extensions to the basic election algorithm, $r(i) = x + a$: Metric 4, Maximum time spent by any node in electing a new AQS.

number of AQS increases very slowly with increasing packet losses. Only when the packet error rate climbs to 50 % does the number of elected AQSs show a significant increase.

Figure 6.12 shows a comparison of average energy for the simulations with 256 nodes by varying PER (Set #8) vs. a simulation for 256 nodes with no packet errors (Set #1: VDH256). The average energy of all the nodes is plotted on the X axis vs. time on the Y axis. It can be seen that the average energy characteristic deteriorates slowly for increasing values of PER.

Figure 6.13 shows a comparison of minimum energy for the simulations with 256 nodes by varying PER (Set #8) vs. a simulation for 256 nodes with no packet errors (Set #1: VDH256). The minimum energy of all the nodes is plotted on the X axis vs. time on the Y axis. The minimum energy metric shows a trend similar to the average energy metric.

The results obtained by varying the PER indicate that the AQS election algorithm is tolerant to packet losses. Its performance is shown to degrade gracefully in the presence of such losses. As mentioned in Chapter 5, packet losses are introduced only for the election phase of the algorithm. Packets that contain tasking information and battery updates are not dropped.

Table 6.10: Set #7 (Scenario I, **V**arying **P**acket **E**rror Rates)

Sim ID	# of Nodes	PER	Area
VPE100a	100	0.05 %	100m x 100m
VPE100b	100	0.5 %	100m x 100m
VPE100c	100	5.0 %	100m x 100m
VPE100d	100	50.0 %	100m x 100m

Table 6.11: Set #8 (Scenario I, **V**arying **P**acket **E**rror Rates)

Sim ID	# of Nodes	PER	Area
VPE256a	256	0.05 %	160m x 150m
VPE256b	256	0.5 %	160m x 150m
VPE256c	256	5.0 %	160m x 150m
VPE256d	256	50.0 %	160m x 150m

Table 6.12: Set #9 (Scenario I, **V**arying **P**acket **E**rror Rates)

Sim ID	# of Nodes	PER	Area
VPE400a	400	0.05 %	200m x 185m
VPE400b	400	0.5 %	200m x 185m
VPE400c	400	5.0 %	200m x 185m
VPE400d	400	50.0 %	200m x 185m

Table 6.13: Set #10 (Scenario I, **V**arying **P**acket **E**rror Rates)

Sim ID	# of Nodes	PER	Area
VPE900a	900	0.05 %	300m x 275m
VPE900b	900	0.5 %	300m x 275m
VPE900c	900	5.0 %	300m x 275m
VPE900d	900	50.0 %	300m x 275m

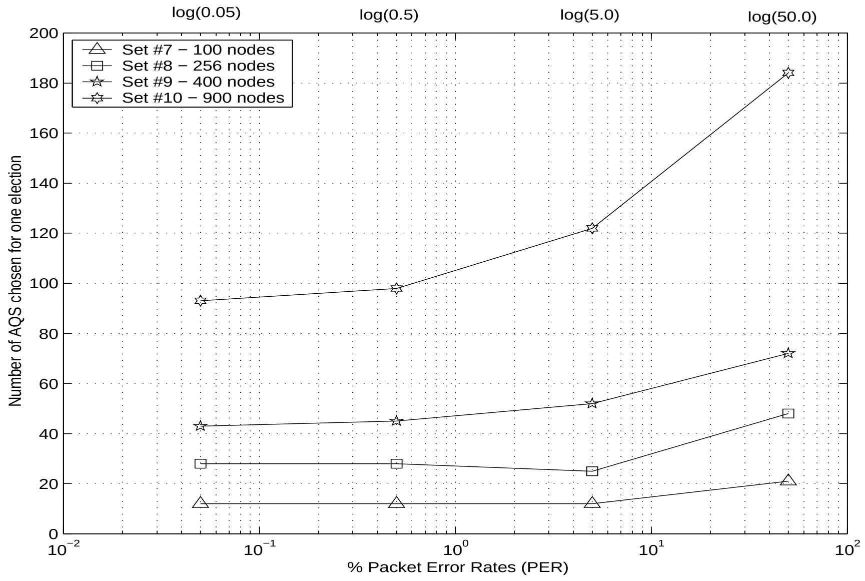


Figure 6.11: Sets #7, #8, #9 and #10: Metric 5, Number of elected AQSs.

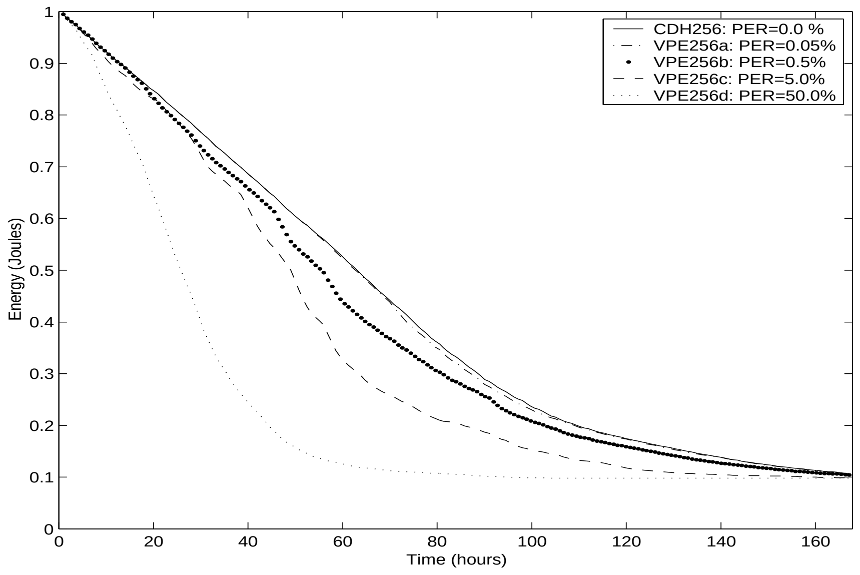


Figure 6.12: Set #8: Metric 1, average energy of all the nodes vs. the simulated time.

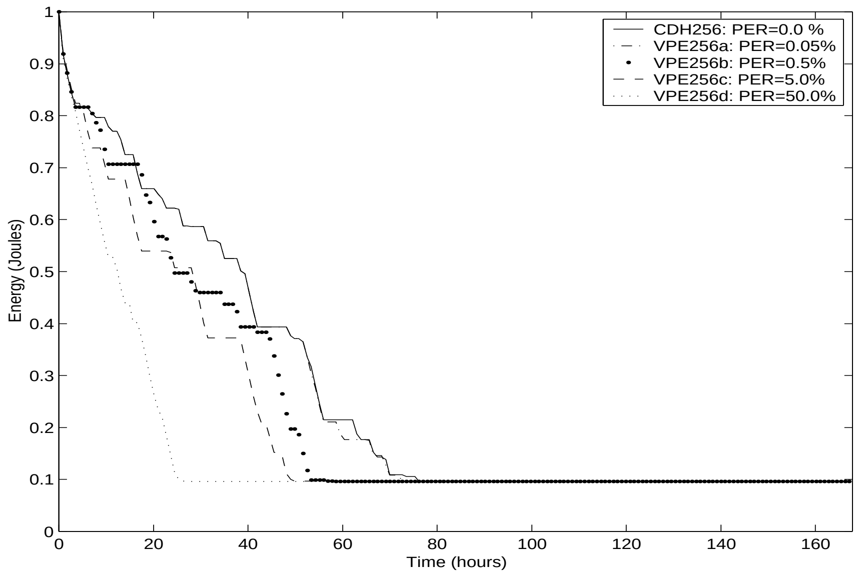


Figure 6.13: Set #8: Metric 2, minimum energy of all the nodes vs. the simulated time.

6.2.7 Experiments with a Multi-Hop Communication Scheme

Tables 6.14, 6.15, 6.16, and 6.17 list the parameters for simulation sets #11, #12, #13 and #14 respectively. These sets are used to study the effect of limiting the number of hops on the election algorithm and the overall energy consumption. For each of the sets #11, #12, #13 and #14, experiments are conducted by limiting the number of hops to 1, 2, 3, and 4. The nodes are placed in a regular hexagonal formation. The results obtained from these simulations are evaluated using metrics 4, 5, 6, and 7.

Table 6.14: Set #11 (Scenario I, **V**arying **N**umber of **H**ops)

Sim ID	# of Nodes	# of Hops	Area
VNH100a	100	1	100m x 100m
VNH100b	100	2	100m x 100m
VNH100c	100	3	100m x 100m
VNH100d	100	4	100m x 100m

Table 6.15: Set #12 (Scenario I, **V**arying **N**umber of **H**ops)

Sim ID	# of Nodes	# of Hops	Area
VNH256a	256	1	160m x 150m
VNH256b	256	2	160m x 150m
VNH256c	256	3	160m x 150m
VNH256d	256	4	160m x 150m

Figure 6.14 presents the results for sets #11, #12, #13 and #14 with respect to Metric 5, i.e., Number of elected AQSs. As the limit on the number of hops travelled by messages is increased the number of AQSs chosen decreases substantially. This is because, as explained in Chapter 5, a larger number of hops leads to the creation of a denser intersection graph. A smaller number of AQSs is beneficial because it decreases the chances for boundary conflicts and inefficient task

Table 6.16: Set #13 (Scenario I, **V**arying **N**umber of **H**ops)

Sim ID	# of Nodes	# of Hops	Area
VNH400a	400	1	200m x 185m
VNH400b	400	2	200m x 185m
VNH400c	400	3	200m x 185m
VNH400d	400	4	200m x 185m

Table 6.17: Set #14 (Scenario I, **V**arying **N**umber of **H**ops)

Sim ID	# of Nodes	# of Hops	Area
VNH900a	900	1	300m x 275m
VNH900b	900	2	300m x 275m
VNH900c	900	3	300m x 275m
VNH900d	900	4	300m x 275m

assignments (discussed in Chapter 4).

Figure 6.15 presents the results for sets #11, #12, #13 and #14 with respect to Metric 6, i.e., Maximum number of packets received at any node for one round of AQS elections. For all sets, the number of packets received during the course of an election increases by several orders of magnitude as the number of hops are increased. Because communication costs for sensor nodes are high, an excessive number of messages transmitted and received during an election may negatively impact the overall life of a node. As an example, consider that increasing the number of hops from 3 to 4 results in nearly doubling of the election traffic but only a small decrease in the number of AQSs chosen. This may not be an acceptable tradeoff. The results obtained for this metric indicate that the network traffic and energy consumption are sensitive to the limit on the number of hops.

Figure 6.16 presents the results for sets #11, #12, #13 and #14 with respect to Metric 7, i.e., Maximum ratio of (the network traffic at a node exclusive to one election) and (the total network

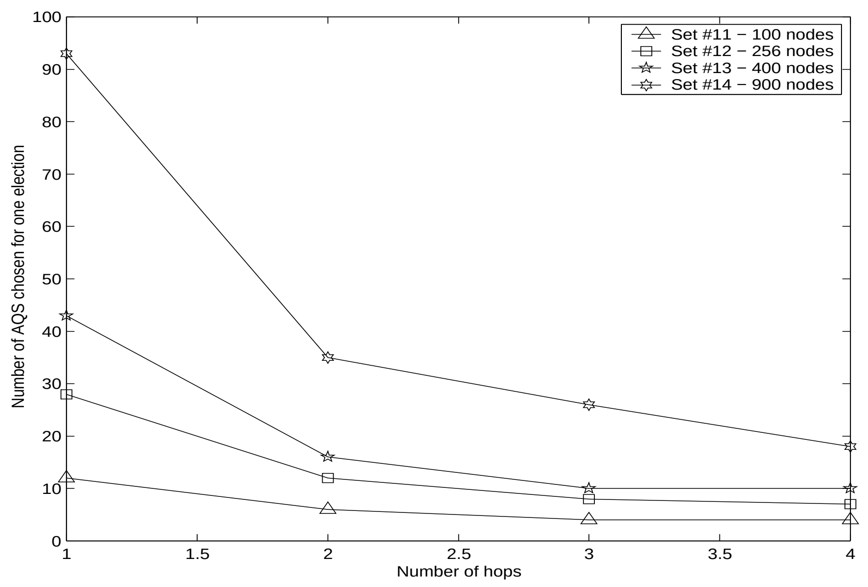


Figure 6.14: Sets #11, #12, #13 and #14: Metric 5, Number of elected AQSs vs. Number of hops.

traffic at a node over a fixed time interval). This metric essentially conveys the tradeoffs between a larger limit on the number of hops and the energy consumption for an election relative to the total energy consumption. With an increase in the number of hops the portion of the network traffic related to an election dominates the total network traffic. This leads to the conclusion that variables such as the frequency of re-elections, the frequency of other types of communication (for example, periodic battery updates), and the number of hops are factors that impact the overall energy consumption. The relationship between these quantities can be empirically expressed by an equation. Before doing so, the following nomenclature is introduced.

r = rate at which energy is consumed in the network

n_h = limit on the number of hops travelled by a message

f_e = frequency of elections

f_c = frequency of other types of periodic communications

k, p = constants

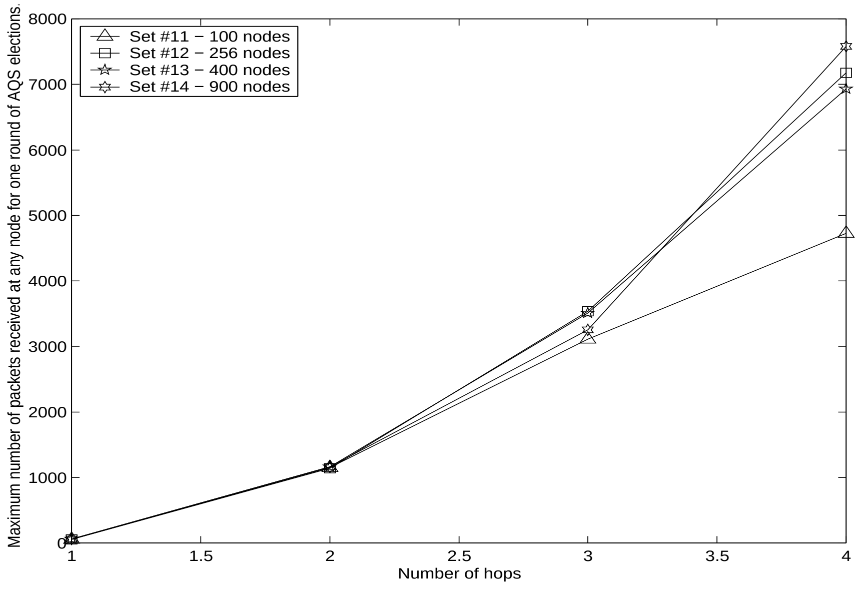


Figure 6.15: Sets #11, #12, #13 and #14: Metric 5, Maximum number of packets received at any node for one round of AQS elections. vs. Number of hops.

$$r = k \times (n_h)^p \times f_e \times f_c \tag{6.1}$$

This equation basically expresses the proportionality relationships between r and the variables n_h, f_e, and f_c. These relationships are briefly outlined. It can be seen from Figure 6.15 that rate of energy consumption, r, increases non-linearly with the number of hops, n_h. Because the number of neighbors is fixed for given number of hops, the rate of energy consumption increases linearly with the frequency of elections (f_e) or other periodic communications (f_c).

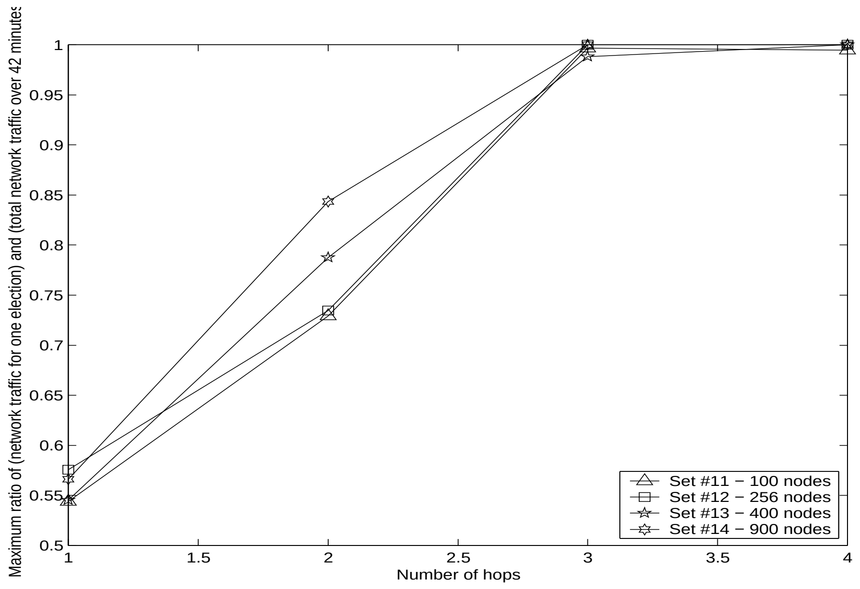


Figure 6.16: Sets #11, #12, #13 and #14: Metric 7, Maximum ratio of (the network traffic at any node exclusive to one election) and (the total network traffic at any node over a fixed time interval)

6.2.8 Experiments to Determine Quality of Sensor Coverage

Results are collected for Metric 8 and Metric 9 to determine the quality of sensor coverage for a user of the network. Figure 6.17 shows the plot of the time at which the first uncovered area is recorded for Set #1 using Approach II and Approach I. It is observed that Approach II achieves significantly better performance over Approach I.

Figure 6.18 displays a plot of the % uncovered area vs. time for 256 and 900 nodes placed in a hexagonal formation (Set #1) using Approach II and Approach I. This plot shows the coverage characteristic over the entire simulated time. Approach II is seen to be more effective in prolonging the time for which a user is guaranteed good quality sensor coverage. Note that both Approach II and Approach I start with 0 % uncovered area.

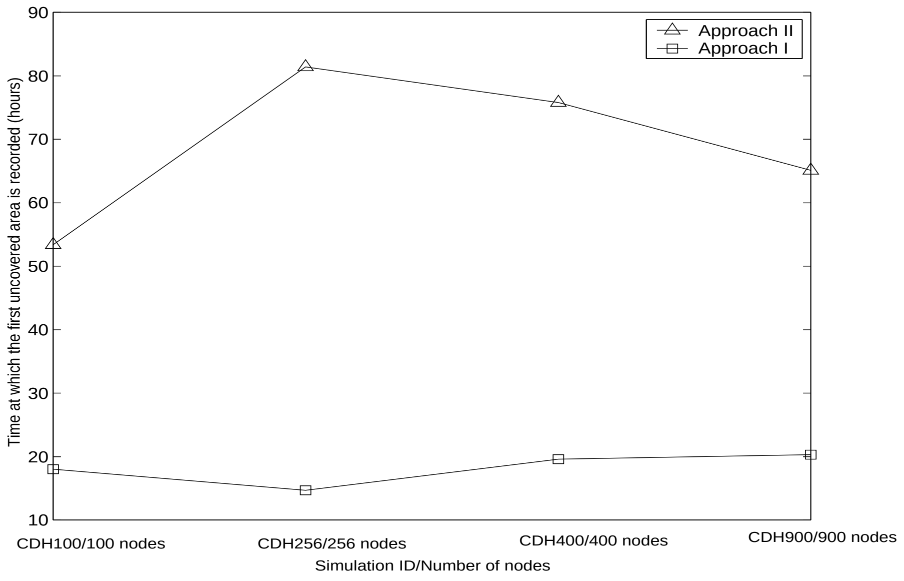


Figure 6.17: Set #1: Metric 8, Time at which first uncovered area is recorded.

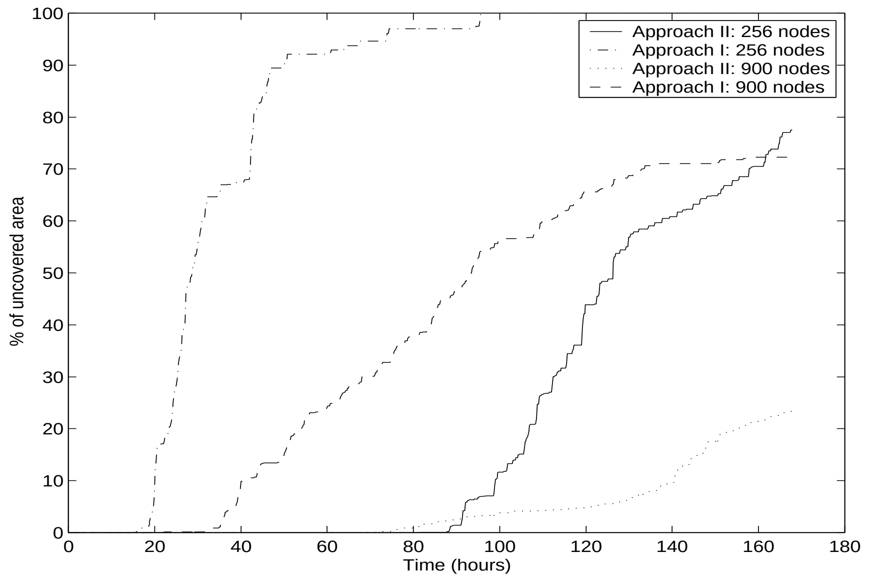


Figure 6.18: Set #1: Metric 8, Plot of % uncovered area vs. time.

6.3 Scenario II

6.3.1 Experiments with Varying Redundancy Index

Table 6.18 lists the parameters that define Set #15. For this set, nodes are placed around a rectangular perimeter for the perimeter monitoring application described in Chapter 5. The energy consumption behavior for this scenario is analyzed using Metric 1 (average energy) and Metric 2 (minimum energy). Experiments conducted for this scenario assume that the placement and density of nodes around the perimeter is fixed. The value of the RI is varied to gauge its effect on these metrics.

Figure 6.19 shows the performance with respect to the average energy metric. Approach II is seen to achieve significant energy savings over Approach I. Figure 6.20 shows the performance with respect to the minimum energy metric. Observe that decreasing the RI impacts the lifetime of the network negatively whereas the minimum energy characteristic shows some improvement with lower values of RI. This can be explained using an argument similar to that used for the hexagonal and random placement of nodes. The negative impact of the RI on the average energy is most likely due to

Table 6.18: Set #15 (Scenario II, **V**arying **R**edundancy Index, **P**erimeter Node Placements)

Sim ID	# of Nodes	*RI*	Area
VRP200a	200	0.90	200m x 200m
VRP200b	200	0.70	200m x 200m
VRP200c	200	0.50	200m x 200m
VRP200d	200	0.30	200m x 200m

excessive replacements of redundant nodes in the grid map. This effect is more pronounced for Scenario II because the inter-node separation in this case is smaller than for the experiment with varying *RI* for Scenario I.

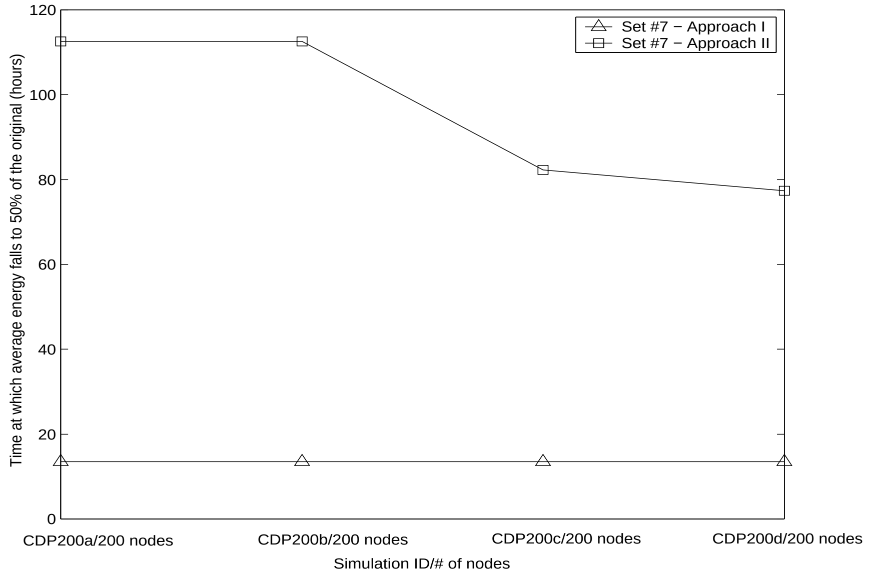


Figure 6.19: Set #15: Metric 1, Time taken for the average energy of all nodes to fall to 50% of its original value.

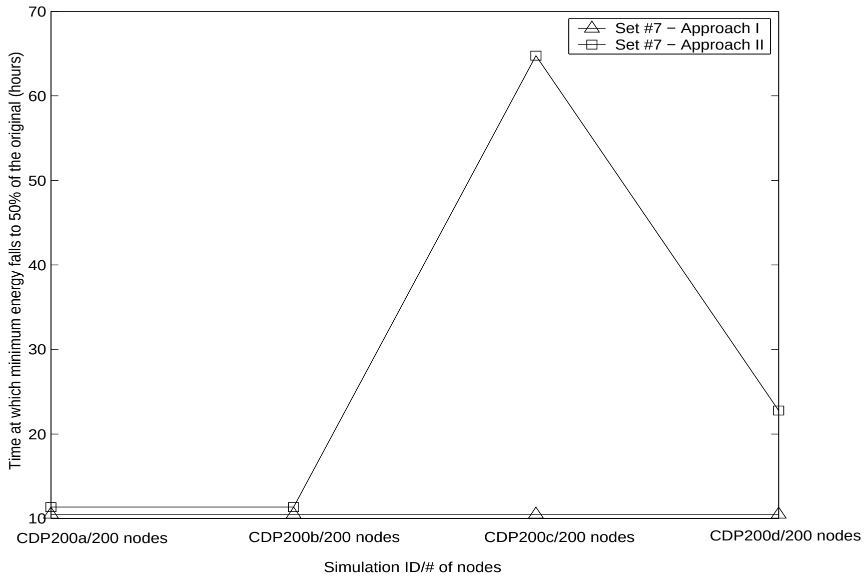


Figure 6.20: Set #15: Metric 2, Time taken for the minimum energy of all nodes to fall to 50% of its original value.

6.3.2 Experiments to Determine Quality of Sensor Coverage

Figure 6.21 shows a plot of the time at which the first uncovered area is recorded for Set #15 using Approach II and Approach I. Figure 6.22 shows a plot of the % uncovered area vs. time for RI values of 0.70 and 0.30. A comparison of these two figures leads to the following conclusion. Though the first uncovered area is recorded at a later time for lower values of RI, the overall coverage deteriorates much more rapidly that for higher values of RI.

Figure 6.23 shows a plot of the time at which the first breach across the perimeter is recorded (Metric 9). For lower values of RI, a breach is observed earlier in the network.

The evaluation of these metrics for quality of sensor coverage indicates the following. While lower values of RI cause the node with minimum energy to be protected, the overall coverage characteristic may not necessarily improve over time. However, in all cases Approach II exhibits performance superior to that of Approach I.

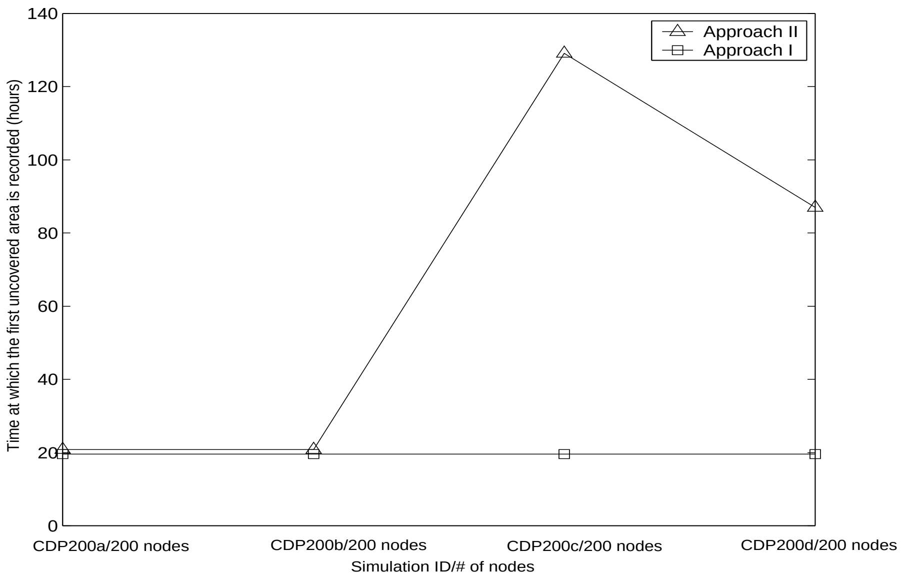


Figure 6.21: Set #15: Metric 8, Time at which first uncovered area is recorded.

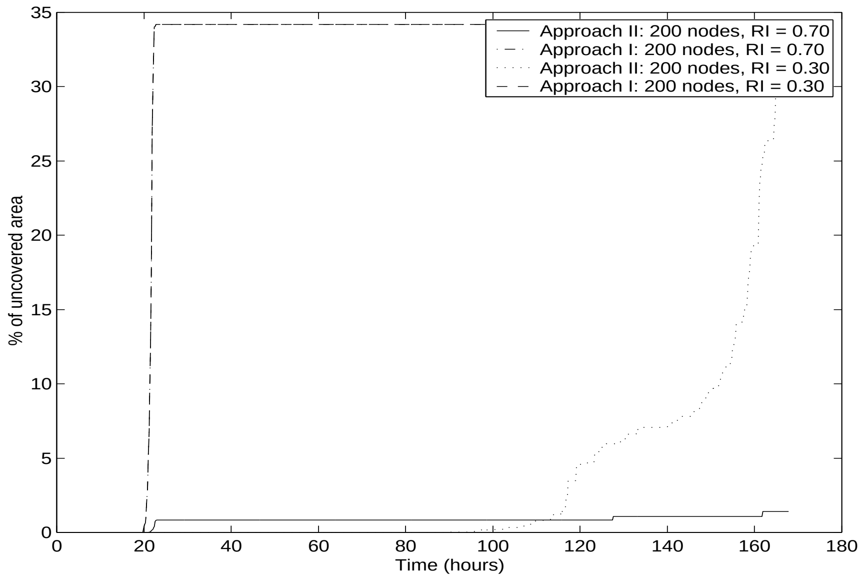


Figure 6.22: Set #15: Metric 8, Plot of % uncovered area vs. time.

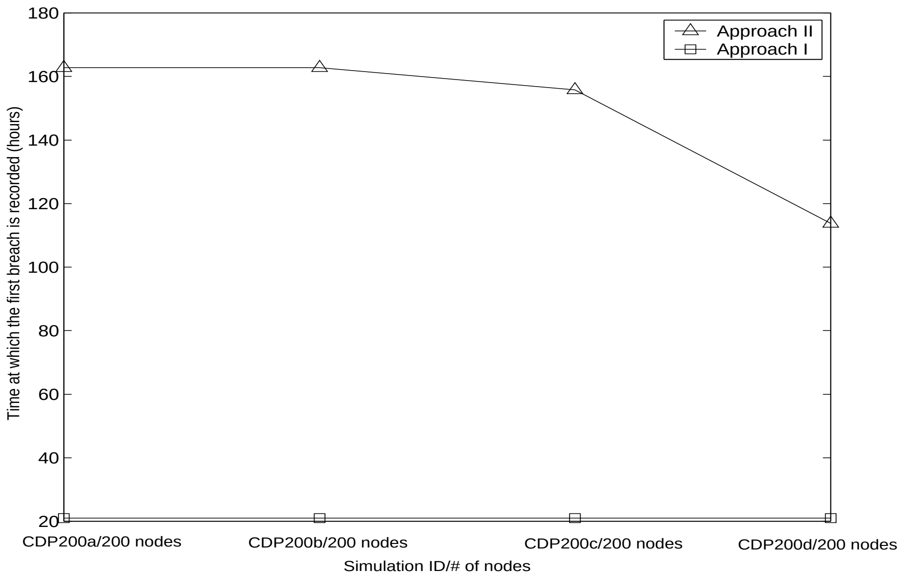


Figure 6.23: Set #15: Metric 9, Time at which the first breach is recorded.

6.4 Summary

This chapter presents and discusses the results collected from simulations of the sensor network. Various metrics pertaining to energy consumption of individual nodes and of the entire network are introduced. Additional metrics are presented to characterize the effect of using a multi-hop communication scheme, and to ascertain the quality of sensor coverage for users of the network. Simulations are analyzed using all these metrics.

One of the major findings from this analysis is that Approach II achieves significantly better performance over Approach I for a redundant sensor network. The algorithms are also shown to scale efficiently to a large network size. Another aspect explored is the effect of packet losses on the execution of the AQS election algorithm; the algorithm is shown to be tolerant to packet errors. Finally, the multi-hop communication scheme is shown to achieve its objective of creating intersection graphs based on sensor coverage leading to the election of fewer AQSs. However, this approach leads to rapidly increasing communication costs as the number of hops is increased.

Chapter 7

Validation and Verification

Simulation models that are built to represent any system must be credible, i.e., there must be a degree of confidence in the results and conclusions that can be drawn from them [54]. Achieving this level of confidence is a two step process that consists of *validating* and *verifying* the simulation model. Jain defines validation as the process of determining if the simulation model is a correct and reasonable representation of the real system under study [54]. In contrast, verification is defined as the process of determining if the model implements those assumptions correctly.

This chapter outlines the approaches used to validate and verify the model of a sensor network. Some representative results are presented to demonstrate the credibility of the constructed models.

7.1 Validation

Validation can be performed by comparing the models with one or more of the following - results obtained from intuition or a common sense approach, real system measurements, and theoretically generated results. The model of a distributed sensor network is validated with respect to the assumptions used in building the models and the outputs obtained from the simulations.

7.1.1 Assumptions in Model Construction

The assumptions used in building the model of a node are based primarily on the real system measurements obtained from studies of prototype sensor nodes (research in this area is summarized in Section 2.1).

1. *Energy model of a node.* The battery source for a node is modeled using power consumption measurements conducted by Srivastava, *et. al.* [24]. The battery is considered to be a bucket of energy from which energy is drawn linearly by various sources, such as the processor, radio, and sensing elements. The relative energy drawn by the battery in various states, i.e., *asleep, awake, transmit*, or *receive* is computed using measurements conducted by Stemm and Katz [23]. The effective life of the battery for a node that is continuously awake (T_{bl}) is modeled using measurements obtained for prototype nodes [24].

2. *Radio model of a node.* The radio capabilities of a node are modeled based on the specifications of a prototype sensor node developed by the University of California at Berkeley [14] that is studied by Srivastava, *et. al.* [24]. Other nodes studied as part of the model building process include the sensor node built by Rockwell [22], and the WINS node [13] developed by Sensoria Corporation. The radio propagation models used are part of the Opnet simulation environment.

3. *Sensing elements of a node.* The sensing capabilities of a node are modeled based on the specifications of the acoustic and seismic sensors that are part of the WINS node [13, 58]. The seismic sensor is capable of detecting personnel at 10m while the acoustic sensor has a range of 30m for detecting vehicles.

The scenarios or network models for these simulations are based primarily on the expectation of how such a network should operate and the situations in which is employed. The two scenarios constructed as part of this process, i.e. Scenario I (a user traversing a sensor field) and Scenario II (monitoring a perimeter), are constructed based on this intuition.

Simulation parameters related to user queries, for example the volume and duration of queries are also intuitively chosen to reflect the function of the network. As an example consider Scenario II; the user queries to the system consist of long running queries around the monitored perimeter. The position and the number of nodes for these scenarios are parameters that are varied to study the behavior of the network for different configurations.

7.1.2 Output and Results

The outputs obtained from simulation experiments are analyzed using the metrics defined in Section 6.1. Real system measurements or theoretical results for such networks are not available for most of these metrics. The only exception is the AQS election for which a theoretical expected execution time exists.

The expected execution time of the AQS algorithm is shown to be $EO(\log(|S|)/\log\log(|S|))$ (Section 3.2) for a case when the number of neighbors of each node is bounded. Results obtained using Metric 4, which is the maximum time spent by any node in electing an AQS indicate that AQS election algorithm scales to different numbers of nodes. These results are outlined in Figure 6.9. These experimental results show a behavior consistent with the theoretical result.

7.2 Verification

Verification is fundamentally the process of analyzing the models to see if they conform to the assumptions that were made in building them. This process can also be visualized as one of debugging the models so that they accurately represent the system under study. The methods and techniques used to verify these models of the sensor network are those proposed by Jain [54].

7.2.1 Simulation Traces

A trace is a listing of the state variables associated with a model at a given point in the simulation. Trace outputs at various levels of detail are used to analyze the operation of various aspects of the simulation as it progresses on the simulation timeline. Opnet provides functionality for configuring traces to estimate the states of various simulation objects such as nodes, links, packets, and processes. This capability is used to debug the models. Traces of memory statistics for packet allocations (Appendix A.1) and packet de-allocations (Appendix A.2) were used to isolate bugs in the finite state machine that lead to a mismatch in the number of packets created and packets destroyed. Traces of individual packet contents (Appendix A.3) were employed to verify the transmission and receipt of packets at the appropriate nodes for different message types, for example, periodic battery updates and task assignments. In conclusion, the trace capability offered by the Opnet Simulation Debugger (ODB) was seen to be an effective tool to verify the proper functioning of the simulation models.

7.2.2 Simplified Test Cases

Another method to determine the correctness of a simulation model is to verify its working by using trivial test cases that can be manually verified. A simple test case is chosen to verify the operation of the AQS election algorithm using the distributed algorithm described in Section 5.2.

Figures 7.1 shows a simple network configuration of 6 nodes that is used to verify the proper functioning of the election algorithm. For this configuration all the actual message transmissions between nodes are recorded and verified with the expected message transmissions.

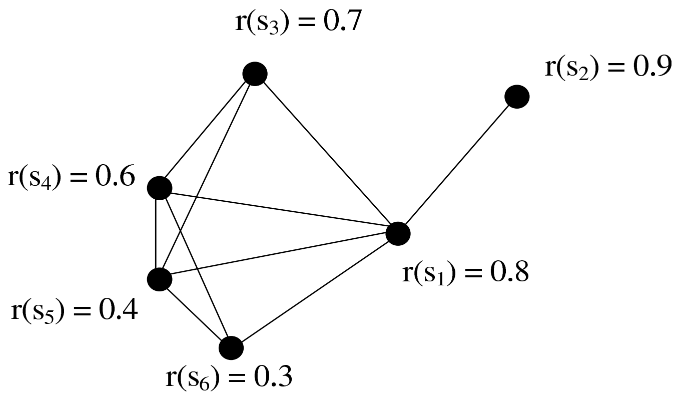


Figure 7.1: Test case for verifying the operation of the AQS election algorithm.

In the first step of the algorithm all neighboring nodes communicate their $r(s_i)$ values to each other. In the second step, node s_2 elects itself as an AQS because it has the largest $r(s_i)$ values of any of its neighbors. None of the other nodes can determine their AQS at the second iteration, therefore they broadcast a message expressing the lack of knowledge about an AQS. In the third step, Node s_1 broadcasts a message declaring its intent *not* to be an AQS because it has heard from an AQS (node s_2). In the fourth iteration, node s_3 elects itself as an AQS because its $r(i)$ value is the largest of all the nodes that do not know of an AQS (nodes s_4 and s_5). Note that even though the $r(s_i)$ value of node s_1 is larger than node s_3 it is not considered because it has already declared its intent not to be an AQS. In the fifth iteration, nodes s_4 and s_5 broadcast messages indicating that they have dropped out of the election process because they know of an AQS (node s_3). In the sixth and final iteration, node s_6 elects itself as an AQS because all its neighbors have indicated that they are not AQSs. Therefore, at the end of the election nodes s_2, s_3 and s_6 are elected as AQS. It can be easily verified that these nodes constitute a maximal independent set for this network configuration. The progression of the election as outlined above is verified by accounting for the message transmissions at each iteration.

A slightly different approach is used to verify that the election algorithm actually produce a MIS of nodes that constitute the AQSs for the network. Figure 7.2 shows the intersection graph for 100 nodes placed in a hexagonal configuration and the elected AQSs for this configuration. This graph is analyzed using Matlab to verify that these AQSs do indeed constitute a maximal independent set.

7.2.3 Continuity Tests

The sensor network is subjected to continuity tests that attempt to vary an input parameter by small increments. As a consequence of this small variation, the system should ideally produce only a small variation in the output value being monitored. Sudden changes in the output may be indicative of modeling errors.

A continuity test is performed to determine the effect of varying the volume of queries on the average energy (Metric 1) and the minimum energy (Metric 2). Table 7.1 lists the parameters for simulation set #16 that constitute the test suite for a continuity test. Nodes are placed in the 2-D hexagonal formation for this set. The query volume parameter refers to the time period between two new queries dispatched to the network.

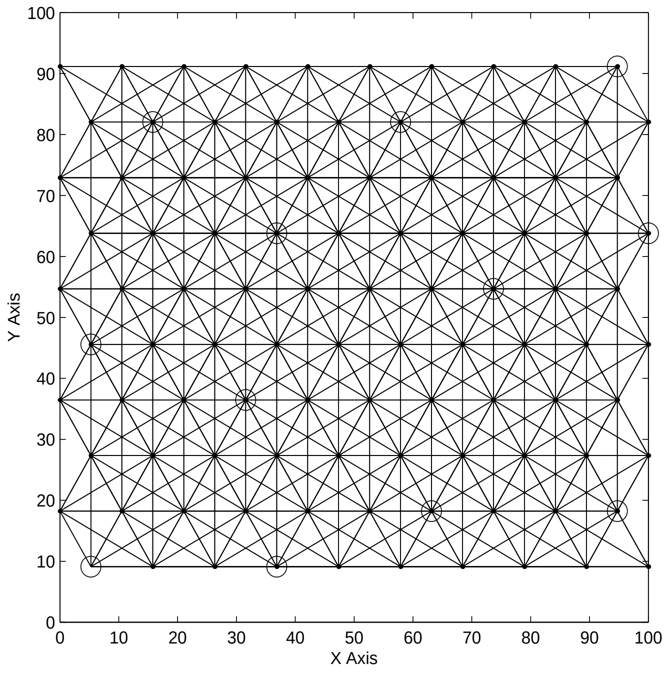


Figure 7.2: Test case for verifying that the election algorithm produces a MIS. The circles around the node positions represent AQSs. The lines represent the intersection graph.

Figure 7.3 shows the plot of average energy vs. time for Set #16. The average energy characteristic shows only minor deviations for the variation of the input parameter. Figure 7.4 shows the plot of minimum energy vs. time for Set #16. This plot shows slightly more variation because it plots the energy life of a single node (as opposed to Figure 7.3 which plots the average energy of all nodes).

Table 7.1: Set #16 Scenario I, (**Q**uery Volume **C**ontinuity **T**est)

Sim ID	# of Nodes	Query Volume (minutes)	Area
QCT100a	100	6	100m x 100m
QCT100b	100	7.5	100m x 100m
QCT100c	100	9	100m x 100m
QCT100d	100	10.5	100m x 100m

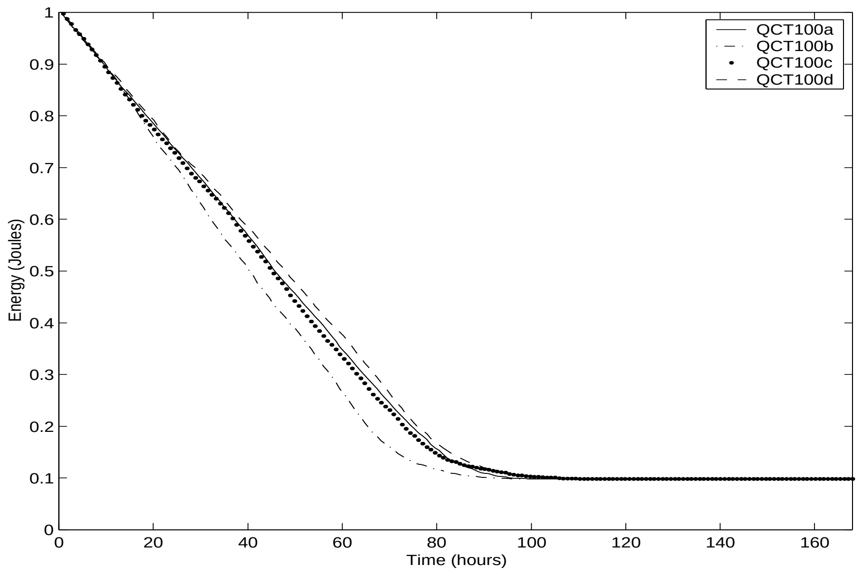


Figure 7.3: Set #16: Metric 1, average energy of all the nodes vs. the simulated time.

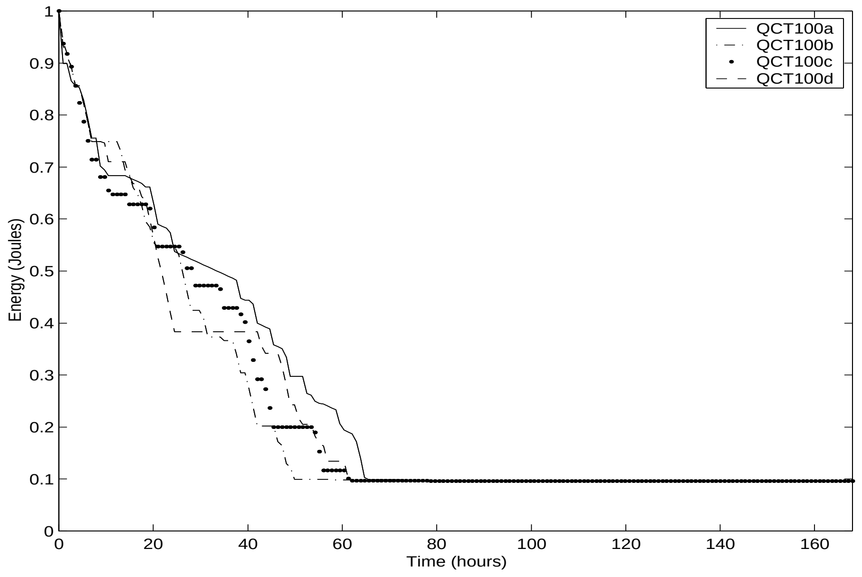


Figure 7.4: Set #16: Metric 2, minimum energy of all the nodes vs. the simulated time.

7.2.4 Degeneracy Tests

As opposed to a continuity test, a degeneracy test consists of inspecting the simulation model for extreme values of the input parameters. Though these extreme cases may not represent typical inputs to the system, the response of the system for these parameters is useful in determining if the behavior of the system obeys boundary conditions.

Table 7.2 lists the parameters for simulation set #17 that constitutes the test suite for a degeneracy test using query volumes as the input parameter. Nodes are placed in the 2-D hexagonal formation for this set. Simulation ID: QDT100a corresponds to a case with no user queries whereas Simulation ID: QDT100b corresponds to a case where a query lasts for ever once it is dispatched to the network; new queries arrive once every 6 minutes.

Table 7.2: Set #17 Scenario I, (**Q**uery Volume **D**egeneracy **T**est)

Sim ID	# of Nodes	Query Duration (minutes)	Area
QDT100a	100	0	100m x 100m
QDT100b	100	∞	100m x 100m

Figure 7.3 shows the plot of average energy vs. time for Set #17. The average energy of all nodes decreases slowly for QDT100a while it drops fairly quickly for QDT100b. This is an expected result given the volume and duration of user queries for each case. In fact, the only energy consumption in the absence of any queries would be for periodic elections of AQS. An increase in the time period between elections will lead to even slower energy consumption as indicated by Equation 5.1. Figure 7.4 shows the plot of minimum energy vs. time for Set #17. The trend observed is similar to that observed for Figure 7.3.

From both these figures we conclude that the implementation of the tasking algorithm lies within the expected bounds of operation.

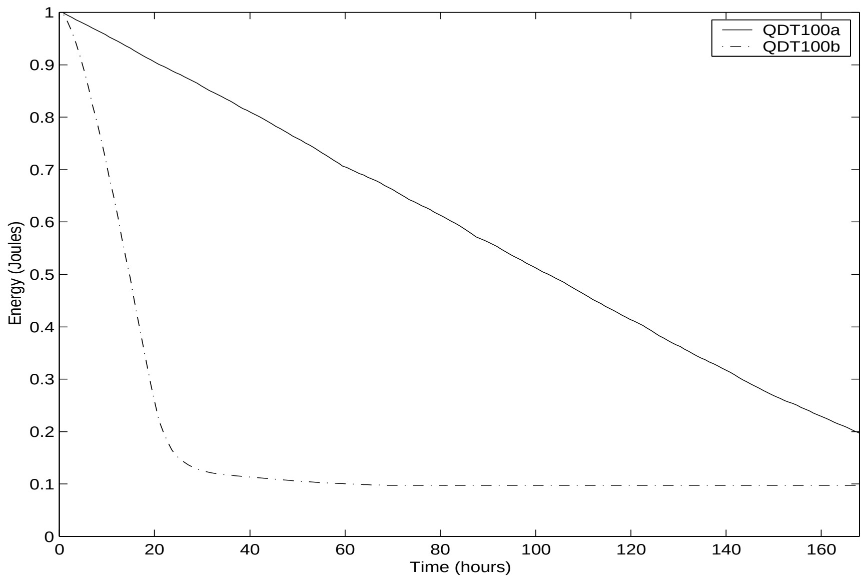


Figure 7.5: Set #17: Metric 1, average energy of all the nodes vs. the simulated time.

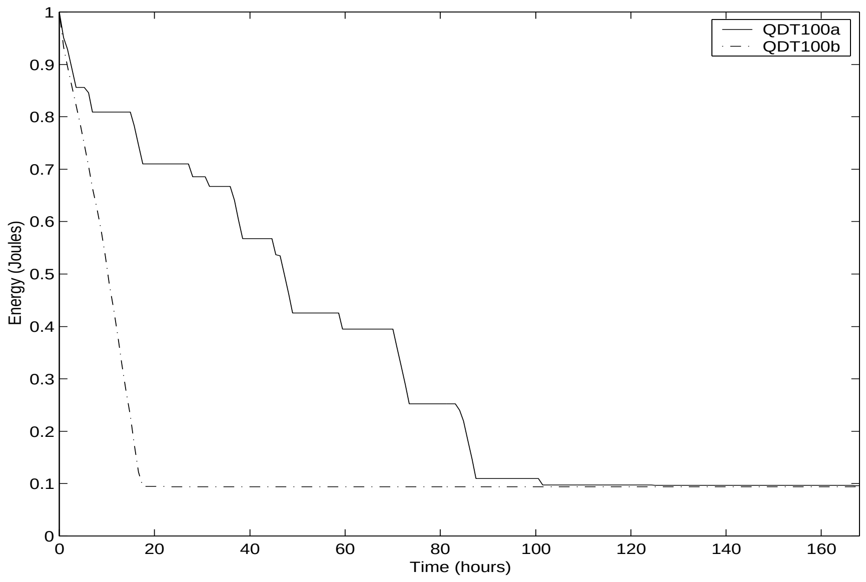


Figure 7.6: Set #17: Metric 2, minimum energy of all the nodes vs. the simulated time.

7.3 Summary

This chapter introduces the techniques used to perform to validation and verification of the model of the sensor network. These checks are necessary to ensure that the model is an accurate and precise representation of the system under study. Validation is performed by testing the assumptions used in creating the model with real system measurements, theoretical analysis or intuition. Verification is performed by examining the model for a series of test cases and confirming that the output conforms to expected norms. Representative results are presented to demonstrate these approaches.

Chapter 8

Conclusions

This chapter presents a summary of the efforts undertaken as regards the design, implementation, and verification of the distributed algorithms by outlining the various chapters of this document. Conclusions and observations from this study are presented. A discussion of future work to enhance these algorithms is also presented along with a discussion of lateral areas of research.

8.1 Summary

Chapter 1 introduced the concept of a distributed sensor network - a system composed of multiple sensor nodes that combine sensing and communication capabilities on a single platform and collaborate their efforts towards a common goal. The defining characteristics of such a network were discussed and contrasted with other types of communication networks. An example was presented for the operation of the network where a user sends a request for information, or query, to a geographical region in the network. A distributed approach for assigning work to individual nodes was shown to be more desirable than the use of a centralized approach.

Related work on distributed sensor networks was presented in Chapter 2. Efforts towards building compact and energy efficient sensor nodes were summarized. Other work reviewed consisted of the communication protocols for inter-node communication, algorithms and architectures for collaborative processing of sensor data, and the problem of optimal sensor node location for such networks. A key factor in the development of designs for sensor networks is energy efficiency; nodes are typically endowed with only a small and limited energy source that is not easily replenished. Many of the ideas and measurements obtained from these related studies were used in this thesis to construct simulation models and develop intuition regarding the operation of sensor networks.

Chapter 3 presented the first stage of the distributed tasking algorithms, i.e. the distributed election of an application query server (AQS) from among the nodes in the network. An AQS is a node in the network that oversees the assignment and management of user queries for other nodes. The set of AQSs can be visualized as a maximal independent set (MIS) of the nodes in the network. Existing sequential and parallel algorithms for the selection of a MIS were characterized. One such algorithm for selecting a MIS in a distributed fashion was extended to an algorithm for the selection of AQSs for the network. This algorithm was theoretically shown to be fault-tolerant and scalable; this was confirmed by results presented in Chapter 6

The second stage of the distributed algorithm consists of AQSs allocating tasks to nodes in the network. This concept was developed in Chapter 4. The allocation of tasks to the network must be performed in a manner that conserves energy by rotating tasks among nodes. Additionally, the selection of nodes that are tasked for a particular query must ensure that a desired level of sensor coverage is available to the query. These requirements motivated an approach that employs a rectangular grid map to represent sensor coverage. This grid map is used by AQSs to assign and manage node-level tasks. Algorithms that construct such a grid map and use it for assigning tasks were presented. Also included was a discussion of how the algorithms may use a connectivity graph based on sensor coverage overlap rather than radio coverage to enable more efficient tasking of nodes. This modification, however, leads to the additional requirement that nodes be able to communicate messages over multiple hops.

The simulation environment and models constructed to study the execution of these algorithms were presented in Chapter 5. These models were built using the Opnet simulation tool. A description of these models and the interactions between them were presented. A scheme for multi-hop communication between nodes was also developed. This scheme relies on the re-broadcasting of messages at intermediate nodes along the path between the sender and destination nodes. Rules were proposed to suppress redundant broadcasts in this scheme. Two scenarios were constructed to demonstrate the operation of the sensor network; these scenarios were used in simulations. Each scenario consists of parameters that define the placement of nodes, the purpose of the network, and the volume and nature of user queries.

Simulation results for the two scenarios introduced in Chapter 5 were presented in Chapter 6. A discussion of these results was prefaced with the definition of various metrics. These metrics characterize aspects such as the life of the network, energy consumption of nodes, scalability of the algorithms and quality of sensor coverage provided to a user. Using these metrics, the algorithms were shown to achieve significant energy savings and balance the total energy consumption among nodes. The performance of the distributed election algorithm was seen to scale to a large number of nodes as expected by the theoretical bounds. The algorithm was also seen to be fairly robust to faults introduced by packet losses.

The simulation models constructed as part of the experimental process were validated and verified to ensure that they produced reliable and credible results. These efforts were summarized in Chapter 7. The assumptions on which the models are constructed were summarized. Verification of the simulations was performed by subjecting the models to various tests. These tests verified that the models behave as expected for different work loads. It was also verified that the AQS election algorithm does indeed result in the creation of a MIS of nodes in the network.

This thesis introduced distributed algorithms for large, ad-hoc sensor networks that allocate work based on user requests to the individual sensor nodes that compose the network. The primary purpose of these algorithms is to provide robust, scalable, and distributed methods to assign user defined tasks to nodes in an energy aware fashion. Extensive simulation experiments indicated that the performance of these algorithms met the objectives defined for the study. They were shown to achieve a significant increase the effective life of the network. Additionally, these algorithms ensured a good quality of sensor coverage for user queries that remained nearly constant throughout the lifetime of the network.

8.2 Future Work

8.2.1 Enhancements to Existing Models

The algorithms and simulations in this thesis do not involve significant interactions with the underlying network routing scheme. This allows these algorithms to be studied and characterized independently from any particular routing scheme. However, it may be beneficial to extend these models such that they interact with the routing protocols. This is a logical next step in the development of these algorithms because the routing scheme will have a significant impact on the energy efficiency. This observation is also motivated by the results obtained for the multi-hop communication scheme described in Section 5.5. Excessive message transmissions lead to a quick degradation in the energy levels of nodes. Additionally, routing schemes may incorporate methods to efficiently route messages for nodes that are critical to the function of the network. An example of such a node may be one equipped with a long range radio used to communicate with the end user of the network. It may be desirable for this node not be tasked like any other node.

In this thesis, simple models are used represent the sensor coverage of the sensing elements of a node. The sensor coverage associated with a sensing element may be a more complex function that the circular area assumed throughout in this work. Alternatively, the coverage characteristic may not be a function of only distance. For example, consider the motion detection sensor that forms a part of the WINS node [13, 58]. This sensor functions like a *trip-wire*, i.e., it produces a binary output indicating the absence of presence of movement in a particular direction. The extension of the simulation models to incorporate different definitions for sensor coverage may provide a more accurate representation of real world systems. It is believed that the basic algorithms described in this thesis will continue to be effective in tasking sensors with different coverage characteristics. However, this needs to be verified in simulation.

Besides sensor coverage, the energy consumption models for individual nodes need to be improved so that they are more accurate representations of energy usage for sensing tasks. Models used for these simulations assume a simple linear energy consumption model as described in Equation 5.1.

8.2.2 Lateral Areas of Research

Though the effect of placement on energy consumption or longevity of the network has not been appreciably studied this is an important direction for future research. The placement of nodes is briefly discussed in Section 2.5. The deployment of sensor nodes is fundamentally an optimization problem. Some optimization strategies for the optimal placement of nodes are outlined in Section 2.5. Sensor node placement can have a significant impact on the effectiveness of the tasking algorithms that rely inherently on node redundancy to achieve energy savings. The placement problem can essentially be visualized for the following three scenarios: (a) placement with random variance, (b) exact placement, and (c) mixed placement. For all these cases, placement methods that take tasking into account could possibly achieve further energy savings.

The end users of a sensor network are typically concerned with information regarding a region of the network as opposed to exact information about a set of nodes [7]. This information could be, for example, ambient temperature in a given region. Schemes such as directed diffusion [28] have been proposed to achieve this goal of aggregating and transporting data from a set of nodes to the user. Similar approaches may be employed for managing and collecting information regarding the status or health of the overall network. To illustrate this consider the following scenario. Some nodes in a region of the network die prematurely due to excessive battery usage; this leads to the creation of a *hole* in the sensor coverage over this region. The entity managing the network is not particularly concerned with the exact details of which nodes died and their locations. Rather, it is in the interest of the managing entity to know of the size and region of this hole so that more nodes may be deployed to cover that area. Methods employing data aggregation and diffusion of messages can be developed to provide an efficient solution to such network management and monitoring problems.

Bibliography

[1] W. Daugherty, "The growth of wireless mobile," *Business 2.0*, December 12, 2000.

[2] M. Zeng, A. Annamalai, and V. Bhargava, "Recent advances in cellular wireless communications," *IEEE Communications Magazine*, vol. 37 (9), pp. 128–138, September 1999.

[3] R. Min, M. Bhardwaj, S.-H. Cho, A. Sinha, E. Shih, A. Wang, and A. P. Chandrakasan, "An architecture for a power-aware distributed microsensor node," *IEEE Workshop on Signal Processing Systems (SiPS '00)*, October 2000.

[4] K. Bult, A. Burstein, D. Chang, M. Dong, and W. Kaiser, "Wireless integrated microsensors," *Proceedings of Conference on Sensors and Systems (Sensors Expo). Anaheim, CA, USA*, pp. 33–38, April 16-18 1996.

[5] G. Pottie, "Wireless sensor networks," *1998 Information Theory Workshop, Killarney, Ireland*, pp. 139–40, 22-26 June 1998.

[6] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," *ACM MobiCom 99*, 1999.

[7] R. Min, M. Bhardwaj, S.-H. Cho, A. Sinha, E. Shih, A. Wang, and A. Chandrakasan, "Low-power wireless sensor networks," *VLSI Design 2001*, January 2001.

[8] R. Brooks and S. Iyengar, "Maximizing multi-sensor system dependability," *Multisensor Fusion and Integration for Intelligent Systems, 1996. IEEE/SICE/RSJ International Conference on*, pp. 1–8, 1996.

[9] B. C. Arrue, A. Ollero, and J. R. M. de Dios, "An intelligent system for false alarm reduction in infrared forest-fire detection," *IEEE Intelligent Systems*, vol. 15, pp. 64–73, 2000.

[10] Wireless Integrated Network Sensors Project, "http://www.janet.ucla.edu/wins."

[11] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 551–8, May 2000.

[12] Sensoria Corporation, "http://www.sensoria.com."

[13] Sensoria Corporation, *WINS NG 2.0 Hardware Platform Specification.*

[14] Smart-Dust Project, "http://robotics.eecs.berkeley.edu/˜pister/smartdust."

[15] J. Kahn, R. Katz, and K. Pister, "Next century challenges: mobile networking for smart dust," *In Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pp. 271–278, 1999.

[16] K. Pister, J. Kahn, and B. Boser, "Smart dust: Wireless networks of millimeter-scale sensor nodes," *Highlight Article in 1999 Electronics Research Laboratory Research Summary*, 1999.

[17] μAMPS Project, "http://www-mtl.mit.edu/research/icsystems/uamps."

[18] A. Sinha and A. Chandrakasan, "Operating system and algorithmic techniques for energy scalable wireless sensor networks," *Proceedings of the 2nd International Conference on Mobile Data Management*, January 2001.

[19] PicoRadio Project, "http://bwrc.eecs.berkeley.edu/research/pico_radio."

[20] J. L. da Silva Jr., J. Shamberger, M. J. Ammer, C. Guo, S. Li, R. Shah, T. Tuan, M. Sheets, J. M. Rabaey, B. Nikolic, A. Sangiovanni-Vincentelli, and P. Wright, "Design methodologies for picoradio networks," *Proceedings of the Design, Automation, and Test in Europe conference. Munich, Germany*, March 2001.

[21] Rockwell Science Center, "http://wins.rsc.rockwell.com."

[22] J. R. Agre, L. P. Clare, G. J. Pottie, and N. P. Romanov, "Development platform for self-organizing wireless sensor networks," *Proceedings SPIE, Unattended Ground Sensor Technologies and Applications*, vol. 3713, pp. 257–268, 1999.

[23] M. Stemm and R. Katz, "Measuring and reducing energy consumpation of network interfaces in hand-held devices," *Transactions on Communications, Special Issue on Mobile Computing*, vol. 8, pp. 1125–1131, 1997.

[24] A. Savvides, S. Park, and M. B. Srivastava, "On modeling networks of wireless microsensors," Technical Report TM-UCLA-NESL-2000-11-001, University of California, Los Angeles, November 2000.

[25] S. Park, A. Savvides, and M. B. Srivastava, "Sensorsim: A simulation framework for sensor networks," *Proceedings of MSWiM 2000, Boston, MA, August 11, 2000*, 2000.

[26] M. Bhardwaj, T. Garnett, and A. P. Chandrakasan, "Upper bounds on the lifetime of sensor networks," *Proceedings. ICC 2001*, June 2001.

[27] Y. Xu, J. Heidemann, and D. Estrin, "Adaptive energy-conserving routing for multihop ad hoc networks," Research Report 527, USC/Information Sciences Institute, October 2000.

[28] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000)*, 2000.

[29] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "A self-organizing wireless sensor network," *37th Allerton Conference on Communication, Control, and Computing*, 1999.

[30] D. D. Patel, "Energy in ad-hoc networking for the picoradio," Master's thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 2000.

[31] J. Rabaey, J. Ammer, J. da Silva Jr., and D. Patel, "Picoradio:ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes," *Proceedings of the IEEE Computer Society Workshop on WVLSI 2000*, pp. 9–12, 2000.

[32] A. Wang, W. Heinzelman, and A. Chandrakasan, "Energy-scalable protocols for battery-operated microsensor networks," *Proc. SiPS '99*, pp. 483–492, 1999.

[33] K. Yao, R. Hudson, C. Reed, and F. L. D. Chen, "Blind beamforming on a randomly distributed sensor array system," *IEEE Journal on selected Topics in Communications*, vol. 16, no. 8, pp. 1555–67, October 1998.

[34] J. H. Yoo and C. Chien, "Distributed wireless microsensor for target tracking," *ASC Proceedings, ARL Federated Laboratory 5th Annual Symposium, College Park, MD*, March 20-22, 2001.

[35] D. Jayasimha, D. Nadig, and S. Iyengar, "A versatile architecture for the distributed sensor integration problem," *Computers, IEEE Transactions on*, vol. 43, no. 2, pp. 175–185, February 1994.

[36] P. Sadegh and J. Spall, "Optimal sensor configuration for complex systems with application to signal detection in structures," *Instrumentation and Measurement Technology Conference, 2000. IMTC2000. Proceedings of the 17th IEEE*, vol. 1, pp. 330–334, 2000.

[37] Y. Lee and S. Pillai, "An algorithm for optimal placement of sensor elements," *1988 International Conference on Acoustics, Speech, and Signal Processing. ICASSP-88.*, vol. 5, pp. 2674–2677, 1988.

[38] D. Brown and J. Schamburg, "A simulation-optimization methodology for sensor placement," *1997 IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation*, vol. 1, pp. 439–443, 1997.

[39] H. Sherali, C. Pendyala, and T.S.Rappaport, "Optimal location of transmitters for microcellular radio communication system design," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 662–673, May 1996.

[40] S. Fortune, D. Gay, B. Kernighan, O. Landron, R. Valenzuela, and M. Wright, "Wise design of indoor wireless systems: practical computation and optimization," *IEEE Computational Science and Engineering*, vol. 2, pp. 58–68, 1995.

[41] M. Wright, "Optimization methods for base station placement in wireless applications," *48th IEEE Vehicular Technology Conference. VTC 98*, vol. 1, pp. 387–391, 1998.

[42] I. Howitt and H. Seung-Yong, "Base station location optimization," *IEEE VTS 50th Vehicular Technology Conference. VTC 1999*, vol. 4, pp. 2067–2071, Fall 1999.

[43] J. A. bondy and U. S. R. Murty, *Graph Theory with Applications.* Elsevier Science Publishing Co., Inc., 1976.

[44] M. Garey and D. Johnson, *Computers and Intractability.* W.H. Freeman, 1979.

[45] O. G. Gómez, *Efficient Parallel Algorithms for Combinatorial Problems.* PhD thesis, Department of Computer Science, Lund University, Sweden, January 1996.

[46] R. M. Karp and A. Wigderson, "A fast parallel algorithm for the maximal independent set problem," *Journal of the ACM*, vol. 32, no. 4, pp. 762–773, 1985.

[47] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *SIAM Journal of Computing*, vol. 15, pp. 1036–1053, November 1986.

[48] M. T. Jones and P. E. Plassmann, "A parallel graph coloring heuristic," *SIAM Journal on Scientific Computing*, vol. 14, no. 3, pp. 654–669, 1993.

[49] R. Gjertsen, M. T. Jones, and P. E. Plassmann, "Parallel heuristics for improved, balanced graph colorings," *Journal of Parallel and Distributed Computing*, vol. 37, pp. 171–186, 1996.

[50] B. Schott, "Personal communication," 2000.

[51] A. Savvides, A. Boulis, F. Koushanfar, M. Potkonjak, V. N. Karavas, and M. B. Srivastava, "Dynamic location discovery in ad-hoc wireless networks," Technical Report NESL TM-UCLA-NESL-2000-07-001, University of California, Los Angeles, July 2000.

[52] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, October 2000.

[53] Opnet, *Opnet v6.0.L Reference Manual.*

[54] R. Jain, *The Art of Computer Systems Performance Analysis.* John Wiley and Sons, Inc., 1991.

[55] L. Peterson and B. Davie, *Computer Networks: A Systems Approach, 2nd edition.* Morgan Kaufmann Publishers, 1999.

[56] W. R. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," *In proceedings MOBICOM '99, Seattle, USA*, pp. 174–185, August 1999.

[57] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *In Proceedings ACM/IEEE MobiCom*, pp. 151–162, 1999.

[58] W. Kaiser, "Personal communication," 2000.

Appendix A

Simulation Traces

A.1 Memory Allocation Sources for Packets

```
Stack 1 of 1 - Total object count (1894), call count (1894), assoc call count (0),
               time (0.000000)
------------------------------------------------------------------------------------
m3_main (argc, argv)
sim_main (prog_name, argc, argv, dynamic_sim, def_net_name, num_procs, proc_set_ptr,
          num_pk_meths, pk_set_ptr, num_pk_fd_meths, pk_fd_set_ptr)
sim_ev_loop ()
sim_obj_rarxch_complete (ch_ptr, pkptr)
sim_obj_rarxch_end_valid (ch_ptr, pkptr, num_errors, accepted)
sim_strm_send (src_mptr, strm_index, pkptr, delay, method, evhndl_ptr)
sim_ev_force (modptr, pdptr, handler, code, type, exec_proc, state_ptr, value, src_obid)
sim_strm_insert ()
sim_obj_qps_intrpt (simev_ptr)
SensorNode () [recv enter execs]
op_pk_destroy (pkptr)
sim_pk_free (pkptr)
sim_pk_dealloc (pkptr)
Vos_Dealloc_Object (ob_hndl, ob_ptr)
```

A.2 Memory De-Allocation Sinks for Packets

```
Stack 1 of 4 - Total object count (1769), call count (1769), assoc call count (0),
               time (0.000000)
------------------------------------------------------------------------------------
m3_main (argc, argv)
sim_main (prog_name, argc, argv, dynamic_sim, def_net_name, num_procs, proc_set_ptr,
      num_pk_meths, pk_set_ptr, num_pk_fd_meths, pk_fd_set_ptr)
sim_ev_loop ()
sim_obj_qps_intrpt (simev_ptr)
SensorNode_acb_fifo () [svc_compl enter execs]
op_pk_send_forced (pkptr, outstrm_index)
sim_strm_send (src_mptr, strm_index, pkptr, delay, method, evhndl_ptr)
sim_ev_force (modptr, pdptr, handler, code, type, exec_proc, state_ptr, value, src_obid)
sim_strm_insert ()
sim_obj_ratx_intrpt (simev_ptr)
sim_obj_ratx_start (ch_ptr, pkptr)
sim_pk_clone (pkptr)
sim_pk_alloc ()
Vos_Alloc_Object (ob_hndl)


Stack 2 of 4 - Total object count (100), call count (100), assoc call count (0),
               time (0.000000)
------------------------------------------------------------------------------------
m3_main (argc, argv)
sim_main (prog_name, argc, argv, dynamic_sim, def_net_name, num_procs, proc_set_ptr,
      num_pk_meths, pk_set_ptr, num_pk_fd_meths, pk_fd_set_ptr)
sim_ev_loop ()
sim_obj_qps_intrpt (simev_ptr)
SensorNode () [init enter execs]
op_pk_create_fmt (format_name)
sim_pk_create_named (module_ptr, format_name, bulk_size)
sim_pk_create_formatted (module_ptr, format_ptr, bulk_size)
sim_pk_create_common (owner_id)
sim_pk_alloc ()
Vos_Alloc_Object (ob_hndl)


Stack 3 of 4 - Total object count (100), call count (100), assoc call count (0),
               time (0.000000)
------------------------------------------------------------------------------------
m3_main (argc, argv)
sim_main (prog_name, argc, argv, dynamic_sim, def_net_name, num_procs, proc_set_ptr,
      num_pk_meths, pk_set_ptr, num_pk_fd_meths, pk_fd_set_ptr)
sim_ev_loop ()
sim_obj_qps_intrpt (simev_ptr)
SensorNode () [init2 enter execs]
op_pk_create_fmt (format_name)
sim_pk_create_named (module_ptr, format_name, bulk_size)
sim_pk_create_formatted (module_ptr, format_ptr, bulk_size)
sim_pk_create_common (owner_id)
sim_pk_alloc ()
Vos_Alloc_Object (ob_hndl)


Stack 4 of 4 - Total object count (32), call count (32), assoc call count (0),
               time (0.000000)
------------------------------------------------------------------------------------
m3_main (argc, argv)
sim_main (prog_name, argc, argv, dynamic_sim, def_net_name, num_procs, proc_set_ptr,
      num_pk_meths, pk_set_ptr, num_pk_fd_meths, pk_fd_set_ptr)
```

```
sim_ev_loop ()
sim_obj_rarxch_complete (ch_ptr, pkptr)
sim_obj_rarxch_end_valid (ch_ptr, pkptr, num_errors, accepted)
sim_strm_send (src_mptr, strm_index, pkptr, delay, method, evhndl_ptr)
sim_ev_force (modptr, pdptr, handler, code, type, exec_proc, state_ptr, value, src_obid)
sim_strm_insert ()
sim_obj_qps_intrpt (simev_ptr)
SensorNode () [recv enter execs]
op_pk_create_fmt (format_name)
sim_pk_create_named (module_ptr, format_name, bulk_size)
sim_pk_create_formatted (module_ptr, format_ptr, bulk_size)
sim_pk_create_common (owner_id)
sim_pk_alloc ()
Vos_Alloc_Object (ob_hndl)
```

A.3 Packet Trace

```
* packet contents:
    ID              :  1996
    tree ID         :  190
    address         :  0x328E808
    format          :  SensorNode_packet
    creation module :  top.Campus Network.node_91.Processor
    creation time   :  0.127008
    stamp module    :  top.Campus Network.node_91.Processor
    stamp time      :  0.127008
    bulk size       :  0
    total size      :  224
    owner           :  top.Campus Network.node_91.Transmitter
    ICI ID          :  NONE
    ID trace        :  off
    tree ID trace   :  off
    encap flags     :  NONE

    Index       Name                Type        Value               Size

    0           senderID            integer     91                  16
    1           destID              integer     -1                  16
    2           msgType             integer     11                  16
    3           qID                 integer     0                   16
    4           qBoundary           integer     0                   32
    5           qType               double      0.0                 32
    6           battery             double      0.963649108804544   32
    7           xPos                double      0.0                 32
    8           yPos                double      91.161003112793     32
```

Vita

Shashank Mehrotra was born on September 13, 1976 in New Delhi, India. He schooled at Bhartiya Vidya Bhawan, New Delhi; Ecole Française de Boumerdes, Algérie; and the Delhi Public School, New Delhi, graduating in 1994. Upon graduation he spent a year at St. Stephens College, New Delhi, studying physics. He subsequently enrolled in the Delhi Institute of Technology, Delhi, in 1995. Shashank received a Bachelor's degree in Instrumentation and Controls engineering in 1999.

In August 1999 he joined the Bradley department of Electrical and Computer Engineering at Virginia Tech to begin work on a Master's degree in Computer Engineering. In the fall of 2001 he will join Hughes Network Systems in Germantown, Maryland, to work with their mobile satellite systems team.