Energy-Efficient Measurement of Coverage in Distributed Sensor Networks

Ravi AnilKumar

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
Electrical Engineering

Dr. Mark T. Jones, Chair
Dr. Peter M. Athanas
Dr. Jeffrey H. Reed

January 30, 2004
Blacksburg, Virginia

Keywords: Distributed Sensor Networks, Energy-Efficient, B-Splines, Coverage

Energy-Efficient Measurement Of Coverage in Distributed Sensor Networks

R. Anilkumar

Dr. Mark T. Jones, Chair

The Bradley Department of Electrical and Computer Engineering

(Abstract)

Large-scale sensor networks have become a reality due to recent developments in sensor node hardware and algorithms. Sensor networks can provide real-time information based on detection and tracking. This information cannot be reliable if little is known about the sensor coverage of the network, which can be defined as the total sensing range of the network due to contributions from each sensor node. Knowledge about coverage can also be useful in determining if there is any gap in coverage in the region of interest as well as improving the algorithm that determines the placement of nodes. Although coverage estimation is this thesis's central concern, other factors such as energy-efficiency and network lifespan that affect the network performance are investigated. Energy-effciency and network lifespan depend on the communication model used for obtaining coverage information from each sensor node. This thesis proposes the use of B-splines for describing coverage efficiently. The properties of B-splines also enable communication models such as directed diffusion and hierarchical clustering to provide better performance as compared to a centralized scheme. Results obtained from simulation experiments indicate that hierarchical clustering and directed diffusion can be used effectively for coverage measurement. The hierarchical clustering model, however, exhibited some drawbacks such as a dependency on the routing scheme and poor node-failure recovery.

Acknowledgments

It is a pleasure to thank many people who made this thesis possible. This thesis would not have been possible without the help of my advisor, Dr. Mark Jones. I thank him for all the motivation and the helpful comments that he has provided throughout. He has helped me learn how to organize my thoughts into writing, which I believe would help me in my career.

I want to thank Dr. Peter Athanas for all his encouragement and support. I want to express my appreciation to Dr. Jeffrey Reed for serving on my committee and providing helpful comments. I am indebted to Dr. Jae Park for helping me throughout my research work. His assistance provided the foundation for many of the ideas that were used in this thesis.

The people at Configurable Computing Lab have helped immensely during various stages of my thesis work and I express my gratitude to them. Lastly, I want to thank my family and friends for being there for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Although large-scale sensor networks have been proven to be useful in military and industrial applications, monitoring the performance has not always been shown to be an easy task [4, 5, 6, 7]. Given the many important uses to which sensor networks may be put, and the fact that accurate knowledge of *sensor coverage* [6, 8] is fundamental to all sensor network applications, regardless of their specific focus, it is clear that a more productive way of estimating coverage in sensor networks is needed if this critical field is to advance. This thesis proposes a productive way of estimating sensor network coverage.

1.1 Problem

Figure 1.1 illustrates a typical sensor network used in a military setting, along with a visualization of the network's sensor coverage, which can be defined as the total sensing range of the network due to contributions from each sensor node. In order to ensure that no trespassing occurs across the field covered by the sensing network, sensor nodes are placed such that the cumulative sensing range of the network covers the entire field. Information on coverage, which is critical in such an application, is required in several different situations. Some of these situations are: validating the results of a detection [9], determining if there is any lapse in coverage in the region of interest,

improving the algorithm that determines the placement of nodes [10], and checking for a breach path[1].

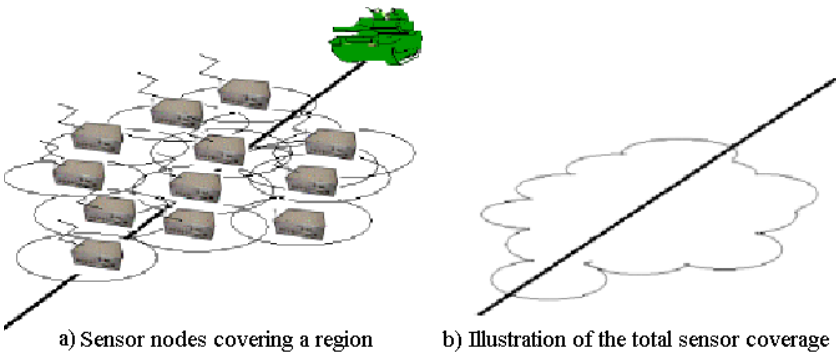

Figure 1.1: WINS NG 2.0 nodes (Chapter 9) deployed over a region

This thesis investigates and describes coverage so that the information is helpful in the aforementioned applications. Both quantitative and visual representations of coverage data are key aspects of the estimation-algorithm and its design.

It should be noted that although coverage estimation is this thesis's central concern, other factors such as energy efficiency and network lifespan affect network performance and must be taken into account when estimating network coverage [11, 12]. Energy efficiency and network lifespan depend on the communication model used for obtaining coverage information from each sensor node [11].

In a centralized scheme [13], each sensor node can send its coverage data to the gateway by using multi-hop communication [11, 14, 15]. However, due to the large amount of data required to describe coverage information (Chapter 3), a centralized scheme is too energy expensive. Moreover, a centralized algorithm would result in a single point of failure, which would reduce the network's lifespan. Therefore, a suitable localized algorithm for communication is required to obtain an energy-efficient method of coverage estimation. Such an algorithm is proposed within this thesis.

[1] A path in the sensor field that can go undetected by the sensors.

1.2 Approach

A sensor node's coverage can be described as every point in space that lies within its sensing range. Aggregation of sensor-coverage from each sensor node provides an estimate of the total sensor coverage [16]. In such a multi-hop communication model, the size of the data has a direct effect on how efficient the network may be in conserving energy. The main steps for energy-efficient coverage estimation include reducing the amount of data required to describe individual sensor-coverage, modeling communication schemes that efficiently use properties of data, and distributing the task load among the nodes.

B-splines [17], a mathematical concept used extensively in computer-graphics, can help reduce the amount of data points that are required to represent the network's coverage (Chapter 3). B-spline based algorithms are useful for determining a smooth curve that approximates a set of points in the plane. Such algorithms can be easily incorporated into a basic processor. Moreover, many communication models can be modeled to use B-spline based coverage data in order to reduce network energy consumption. In this thesis, multi-level hierarchical clustering (Chapter 4) and directed diffusion-based communication models (Chapter 5) are used for coverage estimation.

1.3 Thesis Statement

This thesis explores a B-spline based approach for efficient and accurate quantification and visualization of coverage. Energy efficient implementation schemes are also drawn upon to achieve this end.

The thesis makes the following contributions to research in this field.

- It explores the use of B-splines for modeling coverage data.
- It describes the implementations of the proposed application in two communication models.
- It compares the models to the centralized scheme with regard to energy efficiency and network lifetime.

1.4 Thesis Organization

The rest of the thesis is organized in the following manner. Chapter 2 gives an overview of ongoing research in distributed sensor networks. Various aspects such as sensor nodes, network issues, organization, directed diffusion and algorithms, are described. Work related to B-splines is also included. Chapter 3 formulates the central research problem and evaluates B-splines and computations that are used for coverage determination. Chapter 4 introduces hierarchical clustering and related algorithms. An analytical comparison of hierarchical clustering and centralized schemes is also provided. Chapter 5 provides an extensive look at directed diffusion routing and aggregation methods. An analytical comparison of directed diffusion and hierarchical clustering is also given.

Chapter 6 describes the tool used for simulation, and also provides a synopsis of the design of the current framework and how the proposed algorithms are designed to work. Chapter 7 introduces metrics to evaluate the data collected in simulations. These metrics are applied to the scenarios described in the thesis. Results are presented, along with a discussion of observed trends. Analysis of the results provides an insight into the advantages of hierarchical and diffusion-based routing. In Chapter 8, assumptions involved with the models and algorithms that are used are verified by comparison with analytical values and experimental values. Certain aspects of verification have been performed based on brute force analysis of typical node configurations.

Chapter 9 describes porting of the simulation code to WINS NG 2.0 nodes. It also describes the experimental set up for hierarchical clustering-based coverage estimation. Chapter 10 includes concluding remarks; it also outlines some possible future work in this area. This chapter includes suggested improvements that may be made on the models and the tools used in the thesis.

Chapter 2

Overview of Related Work

This chapter begins by highlighting research that emphasizes the importance of self-organization and multi-hop communication in sensor networks. The chapter then outlines important aspects of clustering techniques, data-centric routing, directed diffusion, and the publish-subscribe paradigm. Towards the end of the chapter, works related to coverage and B-splines are enumerated.

2.1 Self-Organization and Multi-Hop Communication

The production of a large number of miniature and economical sensor nodes became a reality with the advent of cost-effective hardware, such as *Smart Dust*, which are an example of the dramatic advances in miniaturizing sensor nodes. Kahn *et al.* provide details of smart dust nodes that use MEMS technology-based corner-cube retro-reflectors (CCRs) to maintain optical communication links between nodes [18]. Due to the large number of such sensors deployed, the semi-autonomous form of a network is found to be ideal, as contrasted with having control over each individual node.

Studies show that multi-hop communication is advantageous when large numbers of nodes are deployed and when this type of communication dominates protocol overhead [11]. However, in order to maintain communication and the proper integration of data, there is a need for node organization [19, 20]. One of the earliest examples of literature on self-organization is the Linked

Cluster Algorithm by Baker and Ephremides [21], which uses a TDMA-based slot system in which each node is allocated a slot. In this system, the number of nodes is kept constant in order to simplify the algorithm. However, this scheme has inherent problems with regard to scalability.

R. Katz and L. Subramanian enumerate certain decision policies for a self-configurable system [22], including support of heterogeneous nodes, scalability, paradigm of data dissemination, data discovery and power constraints. The authors also introduce an application-specific concept called data-centric networking. A data-centric networking architecture uses many different sensors, such as router sensors, specialized sensors, sink nodes and aggregator nodes. A large number of router sensors are capable of increasing fault tolerance in their system. This algorithm essentially consists of four phases: the discovery phase, the organizational phase, the maintenance phase and the self-reorganization phase. One of the most important parts of the algorithm is the formation of a hierarchy. However, although the system has many advantages, end-to-end latency is a potential drawback.

Alberto Cerpa and Deborah Estrin introduced a new algorithm called ASCENT that self-configures a network to provide better communication topology [23]. Each node ascertains its need to join a multi-hop path. Like most distributed and localized algorithms, ASCENT has different phases. In this method, the dynamic state information is not repeatedly sent across the network, thereby reducing message loss and energy consumption.

2.2 Clustering-based Algorithms

Clustering is another mode of self-organization of nodes in a sensor network so as to localize algorithms and improve energy-efficiency [19, 20]. The advantages of a multi-hop architecture help in the design of clustered networks. A few clustering-based algorithms are described in this section.

Wendi Heinzelman *et al.* describe a clustering-based protocol that minimizes energy dissipation in sensor networks; this protocol is known as a Low-Energy Adaptive Clustering Hierarchy or LEACH [24]. LEACH is one of the earliest localization algorithms that use clustering along with data compression. Clusters transmit data to the base station when required. The algorithm uses

random number generation, along with certain threshold values to elect a cluster-leader. Usually, multi-hop transmissions can cause the death of certain sets of nodes that are near to the base-station much earlier than other nodes [11]. LEACH solves this problem by dynamically clustering and distributing energy consumption among the nodes.

S. Lindsey and C.S Raghavendra propose an algorithm called Power Efficient GAthering in Sensor Information Systems, or PEGASIS [25], which can improve the performance of LEACH. This algorithm works like an optimal chain-based protocol [26]; it achieves between 100 and 300% improvement in network lifetime. In this algorithm, the nodes that form chains take turns transmitting to the base station. Building a chain of nodes that minimizes total length is similar to the traveling salesman problem [27].

Another algorithm that uses clustering is hierarchical clustering [28, 29, 30, 31], which uses a hierarchical routing system [32] between gateway nodes to reduce data communication. This technique, which is based on the spanning tree of a graph [33], reduces the state information needed at the nodes. Applications based on hierarchical clustering are used in military-based packet radio networks such as the survivable packet radio network (SURAN) and the DARPA packet radio protocol (PRNET) [34].

Clustering overhead can challenge the advantages of hierarchical schemes. The hierarchical state routing protocol proposed by Gerla *et al.* describes how a multi-level scheme can reduce overhead [35]. The same authors describe a fisheye state routing protocol that can reduce routing table overheads [36]. In another work, John Sucec and Ivan Marsec analyze overhead due to routing table and level formations [30].

2.3 Research on Data Centric Routing and Directed Diffusion

The data-centric protocol is a new approach for sensor networks as opposed to the address-centric protocol [37]. In an address-centric protocol, each source independently sends data along the shortest path to a sink based on the route that the queries took. A data-centric protocol involves finding routes from multiple sources to a single destination such that intermediate nodes between

the sender and the receiver perform aggregation functions on the data [16]. The advent of the publish-subscribe scheme for distributed systems provides a boost to the sensor network data-centric model [38]. This system consists of a routing scheme that delivers information from publishers[1] to interested subscribers[2]. This approach is the basis for a new application-specific routing system in sensor networks, which is known as directed diffusion [1].

2.3.1 Directed Diffusion

Diffusion routing is a data-centric approach based on the publish-subscribe concept [37]. An attribute-key pair is used to describe the data of a sensor node [39]. A node requests data by sending an interest message that describes the attributes of the data. These nodes are known as the sink nodes. The following shows an example of attributes in an interest message.

Type = Coverage
Distance = 200m
Location=[-100, 100,200,400]

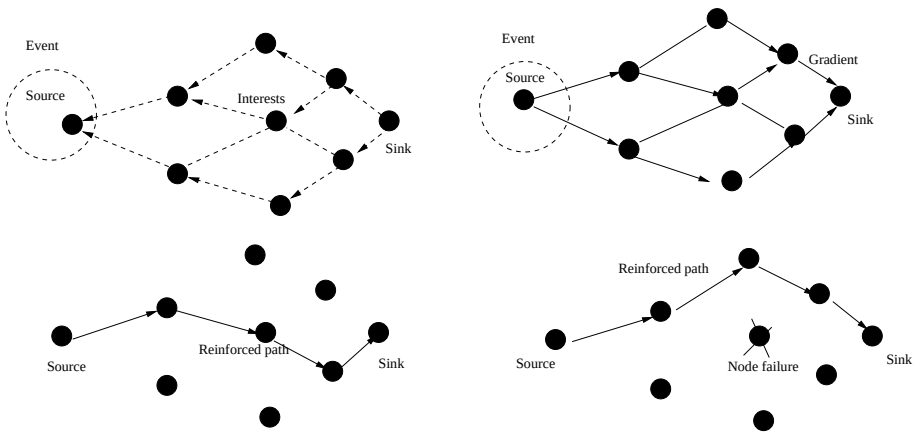

Figure 2.1: Directed Diffusion Routing [1]

Initially, the interest message is flooded throughout the network. Every node has an interest cache

[1]Publishers provide information without needing to know which applications are interested in the information.
[2]Subscribers are consumers of specific information.

that is compared to the current message. If a node receives an interest message that matches its attributes, data is sent to the network. A node which gets data message from a neighbor attempts to find a matching interest in its cache. If no matching entry is found, then the message is dropped. If a match exists, then the node checks the data cache associated with the matching interest entry. If the there is no match found in the data cache, then the message gets forwarded. Each node also keeps track of the nodes from which it receives the interest message. Sets of such nodes, known as gradient nodes, are reinforced depending on delay metrics.

2.3.2 The Publish-Subscribe Paradigm

The directed diffusion scheme follows the publish-subscribe paradigm, in which publishers are the data generating source nodes and subscribers are the sink nodes. The publish-subscribe scheme has many other applications in distributed environments. Given the fact that individual point-to-point and synchronous communications lead to rigid and static applications that make scalability a cumbersome task, and that there are a large number of people using the Internet, there is a need for a more flexible communication model [40]. A publish-subscribe approach is well-suited to the loosely coupled nature of distributed interaction in large-scale applications over the Internet [40].

Another application of publish-subscribe is its use as an overlay network in a GPRS system. This form of event notification system [41], named SIENA [42], supports a variety of applications that involve multiple clients. It operates using servers that act as access points along with store-and-forward network routers. Publishers can use these access points to advertise information about notifications that they generate. The publish-subscribe system has many other uses in the area of peer-to-peer networks [43], such as software update notifications from major software vendors.

Eugster *et al.* describe the many faces of the publish-subscribe paradigm, and how it has influenced different subsystems [38]. The authors also describe different variants of the publish-subscribe scheme (topic-based, context-based, and type-based) that have shown strengths in different fields.

2.4 Data Aggregation

Data Aggregation helps reduce the amount of data transmitted by different nodes. Although aggregation is application specific, the mechanism for dissemination of information to the sink remains the same [39]. Experiments on a test-bed have shown that aggregation can improve energy consumption by a factor of three [39]. Directed diffusion constructs low-latency paths from sources that form trees rooted at the sinks. This form of aggregation, in which the virtual nodes of the trees act as aggregator nodes, is called opportunistic aggregation [16]. A possible downside of aggregation is latency, although avoiding the store-and-forward approach can reduce that latency.

The greedy approach is another form of aggregation that is energy-efficient (in contrast to opportunistic methods) [16]. In this scheme, path establishment is made more stable by using energy metrics. This data-centric reinforcement model performs similarly to an opportunistic scheme in a low-density network, although significant energy saving is observed in a high-density network.

In another work related to monitoring of residual energy, Estrin *et al.* propose an approach that scans the network for energy-related information and applies in-network aggregation [5]. This concept of in-network aggregation is used in the directed diffusion-based model (see Chapter 5) in this thesis.

2.5 Coverage Issues

Not many researchers have explored ways to estimate the coverage of a sensor network. Mani *et al.* describe coverage in terms of two scenarios: a worst-case coverage and a best-case coverage. In the worst case coverage, attempts were made to find areas of low observability from sensor nodes and therefore, estimate the breach path region. A best-case coverage refers to the areas of high observability from sensors. This information helps in judging the location of new nodes that can improve the coverage of the area under consideration. However, this model does not provide a visualization of the actual coverage. Moreover, the implementation was based on centralized scheme. Also, in their model, there is no provision for reduction of data, which can reduce energy

consumption. The coverage-estimation model described in this thesis helps in visualization of the complete coverage. Using few number of data points, information such as breach path and areas of coverage can be easily determined. The algorithms can be used over a period of time for diagnostic purposes. The coverage estimation method described in thesis makes use of B-spline for both coverage estimation as well as reduction of energy consumption.

2.5.1 Sensor Coverage and B-splines

Sensor coverage provides information that can help determine faults, network lifetime, and statistics that improve the node-placement scheme. In this thesis, coverage is described using a closed curve that represents the cumulative sensing range of all sensor nodes. A closed curve can be computed using various methods such as chain codes and Fourier descriptors. Chain codes use numerous sequential segments to describe a curve; they require large amounts of data [44]. Fourier descriptors [45] are used for 2D object characterization. They are derived from a Fourier transform of contour coordinates and are considered computationally intensive.

The term *spline* refers to strips of metal placed on the surface of airplanes, cars and ships. *B* stands for *basis functions*, as splines can be represented as weighted sum of polynomial basis functions (Chapter 3). Schoenberg, the "father of splines", has shown that any uniform spline can be represented as the sum of shifted basis functions [46].

The B-Spline is an efficient way of determining a smooth curve that can pass through a group of points in a plane. B-splines are not overly sensitive to round-off errors, and can be easily implemented using elementary arithmetic operations. The most compelling reason for using B-splines is that the fitting process can be efficiently accomplished by digital filtering without any need for matrix manipulation.

Chapter 3

Coverage: Model and Computations

This chapter provides the concepts and algorithms used for modeling the network's sensor coverage based on B-splines. The chapter starts with a detailed look at the approach used for visualizing the problem of sensor coverage. The rest of the chapter provides a brief outline of the computations involved in coverage estimation using concepts of B-splines.

3.1 Problem Formulation

In this section, a brief description of the problem of the estimation of sensor coverage is provided. The initial assumptions used in coverage estimation are as follows.

- A sensor network is represented by a simple connected graph [33].
- All of the nodes know their absolute position.
- A node has a circular sensor coverage of radius r.
- Communication occurs through a reliable channel without any dropping of packets.

Figure 3.1 shows an example of two nodes with overlapping sensing area. If $S(u)$ and $S(v)$ represent the sensing area of nodes u and v, then the total coverage can be written as

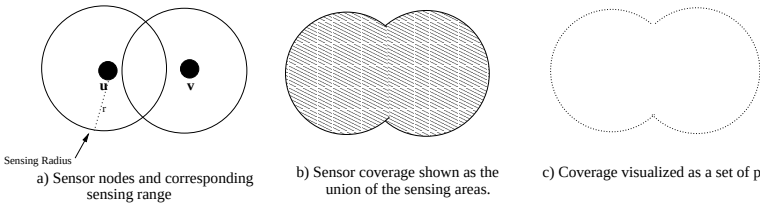

Figure 3.1: Illustration of sensor coverage

$$S(u) \cup S(v). \tag{3.1}$$

To formulate the problem, consider the unit disk graph (Figure 3.2), in which the nodes are equally spaced (constant node density configuration) [47]. This unit disk graph represents a simple communication and coverage model in which the network's communication range (tx_{range}) equals the sensing radius (r) of each node. In order to define unit disk graph, assume that all the nodes are placed in the Euclidean plane $\mathbf{R}^2$. Based on the paper on geometric ad-hoc routing for disk graphs and general cost models, an Unit Disk Graph can be defined as follows [48].

Definition 1. (Unit Disk Graph) *Let $V \subset \mathbf{R}^2$ be the set of nodes and let $E \subseteq V^2$ be the set of edges such that $(u, v) \in E$ if and only if the Euclidean distance between u and v is at most 1 unit. Then the Euclidean graph $G = (V, E)$ is called the unit disk graph of the nodes in V.*

In a typical sensor network, the Euclidean distance of one unit refers to the normalized value of the diameter of the sensor range. For the application of coverage estimation, two nodes are considered to be neighbors if they have a Euclidean distance less than tx_{range} (transmission range) and r (sensing range). The set of neighboring nodes, $N(v)$, can be expressed as

$$N(v) = \bigcup_{v' \in V, v' \neq v} v' | dist(v, v') < tx_{range}, r, \tag{3.2}$$

where $tx_{range} = r$ due to the assumptions made about an Unit Disk Graph.

As observed from the figure (Figure 3.2), the total sensor coverage is given by union of individual sensing areas, which can be considered to be a *merging* or *aggregation* process. If the sensing area of i^{th} node is given by S_i, then the total sensor coverage, C, can be expressed as

$$C \equiv \bigcup_i S_i. \tag{3.3}$$

S_i, the sensing area of the i^{th} node, can be described as every point, $p(i_x, i_y)$, in space that lies within the node's sensing range, r (i_x and i_y are position coordinates of node i). If $f(i_x, i_y)$ is the equation of the circle that represents the sensing range, S_i can be described as

$$S_i \equiv f(i_x, i_y) \leq r. \tag{3.4}$$

Therefore, in order to represent S_i, it is sufficient that the locus of points (represented by P_i) that is r units from the coordinates (i_x, i_y) of the i^{th} node be known. For the coverage application, it is convenient to express P_i in the parametric form, where each point is represented as $P_i(i_x + r\cos\theta, i_y + r\sin\theta)$. It should be noted that 'θ' ($0 \leq \theta \leq 2\pi$) represents the *parameter* that can be varied in order to obtain *samples* of P_i.

From the above observations, the total sensor coverage can be expressed as

$$C \equiv \bigcup_i P_i \bigcap (\bigcup_{j=N(v)} S_j). \tag{3.5}$$

In order to estimate the total coverage, the gateway node has to obtain the set of data points represented by C. However, with an increase in the area of the sensor field, there is an increase in data points that are required. To reduce the amount of data, the sensor coverage has to be represented by a small number of points, which should describe the coverage with minimum error. For this purpose, the following section examines a B-spline-based coverage estimation model.

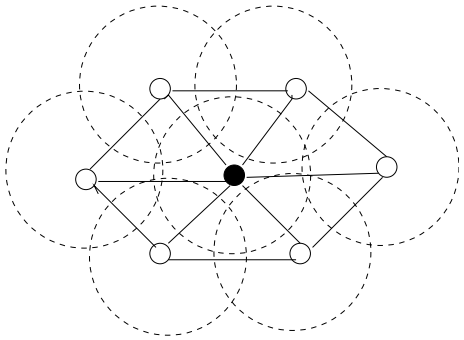

Figure 3.2: Unit disk graph for a constant node density configuration

3.2 B-Splines: Concepts

B-splines can describe sensor coverage with fewer points than would be needed to represent coverage area with data points (represented by C in previous section). This section provides a brief description of the concepts of B-splines [17].

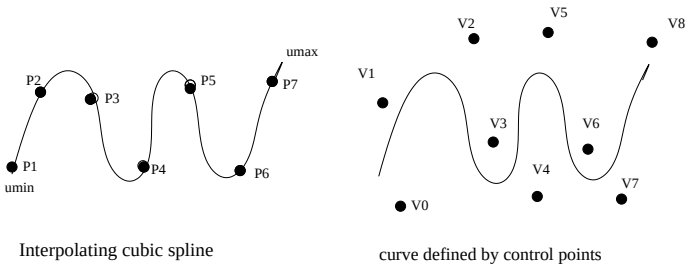

Figure 3.3: Illustration of interpolating and approximation splines

Consider that a curve is to be fitted through $m+1$ data points, $P_0, P_1 ..., P_m$. It is convenient to represent a two-dimensional curve in parametric form given by

$$\mathbf{Q}(\bar{u}) = (X(\bar{u}), Y(\bar{u})), \tag{3.6}$$

where $X(\bar{u})$ and $Y(\bar{u})$ are each single-valued functions of $\bar{u}$; $\bar{u}$ refers to a single parameterization of the entire curve [17]. Usually, the entire curve is not represented by a single polynomial for $X(\bar{u})$

and $Y(\bar{u})$. The curve is often broken into some number of pieces called *segments*, each defined by separate polynomials. The segments can be joined together to form a *piecewise polynomial* curve.

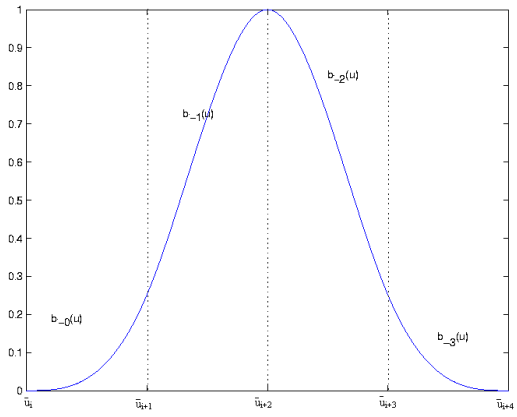

Figure 3.4: Cubic spline basis-function

B-splines are useful for generating a single piecewise polynomial curve through any number of *control points*, which are defined as vertices or control points that can alter the curve *locally* or at that segment. For the purpose of coverage, cubic B-splines are used[1]. Cubic B-splines can be written in terms of linear combinations of unit functions called *basis-functions*. Hence the parametric form of the curve can be written as

$$\begin{aligned} \mathbf{Q}(\bar{u}) &= \sum_i \mathbf{V_i} B_i(\bar{u}) \\ &= \sum_i (X_i B_i(\bar{u}), Y_i B_i(\bar{u})), \end{aligned} \tag{3.7}$$

where $\mathbf{V_i}$ represents the control points and B_i represents the basis function [17]. Within each knot

[1]Cubic refers to the degree of the piecewise polynomial.

interval, a basis-function can be defined by a cubic polynomial of the form

$$a_j + b_j u + c_j u^2 + d_j u^3, i - 3 \leq j \leq i \tag{3.8}$$

.

Cubic splines are considered *uniform* if their knots are equidistant. Uniform cubic B-splines are used in the coverage estimation algorithm.

$B_i(\bar{u})$ (Figure 3.4), is a cubic basis-function centered at $\bar{u}_{i+2}$. It is zero for $\bar{u} \leq \bar{u}_i$ and for $\bar{u} \geq \bar{u}_{i+4}$. The non-zero portion of $B_i(\bar{u})$ is composed of the four polynomial segments $b_{-0}(u)$, $b_{-1}(u)$, $b_{-2}(u)$, and $b_{-3}(u)$. Segments of a curve can be described as an un-scaled shifted version of the basis-function B_i. From calculations it is found that (Figure 3.4),

$$\begin{aligned}
b_{-0}(u) &= \frac{1}{4}u^3, \text{ if u} \geq 0 \text{ and u} < 1, \\
b_{-1}(u) &= -(u-1)^3 + \frac{1}{4}u^3, \text{if u} \geq 1 \text{ and u} < 2, \\
b_{-2}(u) &= \frac{6}{4}(u-2)^3 - (u-1)^3 + \frac{1}{4}u^3, \text{if u} \geq 2 \text{ and u} < 3, \text{ and} \\
b_{-3}(u) &= -(u-3)^3 + \frac{6}{4}(u-2)^3 - (u-1)^3 + \frac{1}{4}u^3, \text{ if u} \geq 3 \text{ and u} < 4.
\end{aligned} \tag{3.9}$$

Spline representation can be divided into an **interpolation method** and an **approximation method** [49]. When the number of data points is small, the interpolation method is convenient; however, when the number of points is large the approximation method is applied. The spline approximation method replaces the actual set of points by a spline function [17]. In short, the problem of least square approximation can be written as follows.

it Given M data points sampled (x_i, y_i) and N basis-functions $B_i(x)$, obtain a linear combination of the basis-functions $f(x)$ that minimizes the error between the data points and $f(x)$. The expression that is to be minimized is given by

$$E = \sum_{i=1}^{M}(M_i - \sum_{k=1}^{N}(V_k B_k(x)))^2 = R, \tag{3.10}$$

where V_K represents the control points. As Equation 3.10 is quadratic, its minimum occurs for those values of X_i and Y_i such that

$$\begin{aligned} \frac{\partial}{\partial X_l} R &= 0 \\ \frac{\partial}{\partial Y_l} R &= 0, \end{aligned} \tag{3.11}$$

where l ranges between 0 and N. In order to find the solution to the above equations in the least square sense, consider the matrix A, which is known as the *design matrix* [50]. This matrix consists of $M \times N$ elements: the rows representing the linear system of equations for each data point and the column representing the basis-function. This is an *over-determined* system of linear equations, since $M > N$ [50].

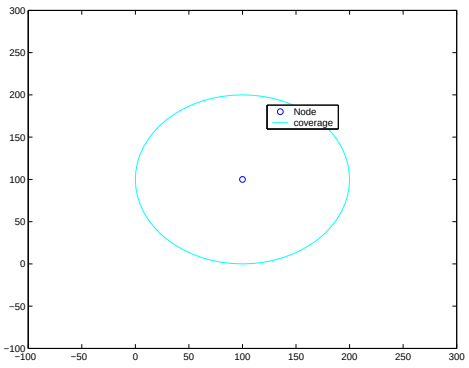

Figure 3.5: A node with its coverage shown

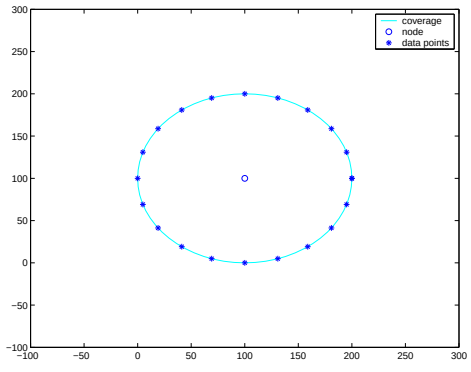

Figure 3.6: Data points sampled from the coverage

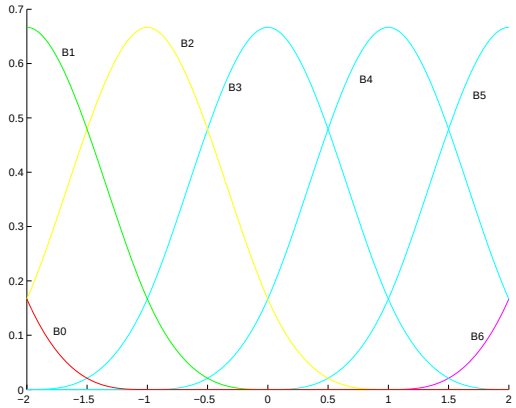

Figure 3.7: The circular coverage described using scaled sum of basis-functions

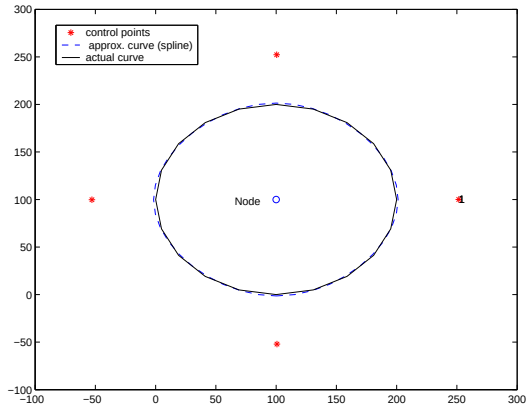

Figure 3.8: Spline approximated coverage and the actual coverage (using four control points)

The linear system can be written in the form $Ax = b$. In order to solve the system, it should be reduced to the form $N \times N$. This result is achieved by multiplying the matrix by the transpose of matrix A, which leads to the *normal equation* that can be written as [50]

$$\begin{aligned} (A^T.A).x &= A^T.b \\ x = \; & ((A^T.A))^{-1}.A^T.b \leftrightarrow x = A^{-1}.b \end{aligned} \tag{3.12}$$

The solution of the linear system of equations, x, provides the control points or coefficients that minimize the error between the data points and the spline-based function. However, finding the inverse of a matrix is computationally intensive. A Gaussian elimination method is used in this algorithm to reduce the amount of computation [51].

Coverage of a node using B-Splines: An Illustration

For the application of coverage computation, closed curves are to be considered. It is known that, for a closed curve, the first three control-points are identical to last three control-points [17]. For $m+1$ segments in a closed curve there are $m+4$ vertices or control points, which can be represented as $V_0, V_1, V_2, ..., V_{m-1}, V_0, V_1, V_2$.

As described earlier, a group of data points are used to represent coverage of a single node. Figure 3.5 shows a node with a sensing radius of 100m. In this illustration, B-spline coverage is calculated using four control-points. B-spline-based control-points are obtained using using least square approximation method [50].

In order to define the entire circular coverage, a total of seven basis functions over four parametric intervals are chosen [17]. As shown in Figure 3.7, the left-most basis-function, B_0, extends three additional intervals to the left, while the right-most basis function extends three additional intervals to the right. The basis functions can be considered to be wrapped around due to the assumption that the coverage is represented by a closed curve [17]. Once the control points are found , an approximate B-spline curve of the coverage can be drawn [17].

3.3 Application of B-Splines in aggregation

The previous section described how the sensing range of a single node is computed by using B-splines. After aggregating the sensor coverage of all the nodes, a complete picture of network coverage can be obtained. However, the process of aggregation depends on the communication model used. Chapter 4 and 5 give a detailed description of the communication models used; the chapters also explain how B-spline-based algorithms are used for coverage estimation.

Chapter 4

Communication Model: Hierarchical Clustering

This chapter begins with a brief overview of multi-hop communication and the two communication models used in this thesis. A detailed description of the hierarchical clustering algorithm that is used, along with analysis, are also included.

4.1 Multi-Hop Communication

The communication models in this thesis use multi-hop communication for transportation of data between the nodes or between the nodes and the gateway. Research on multi-hop communication have shown that it is energy efficient when the distances between the trasmitting and receiving nodes are large [11]. In order to examine the effect of multi-hop communication on a sensor network application, consider the scenario in which data is directly transmitted over a distance of R (R is the distance between the source and the destination). Also consider another transmission that occurs in three hops, each $R/3$ in distance. It is known that the *propagation loss* L and distance between the nodes, R, are related by the expression [52]

$$L = KR^n \text{ (K and n are constants; R is the distance between nodes).} \tag{4.1}$$

In order to compare the two scenarios, consider the ratio of the propagation losses between the two, which can be written as follows.

$$\frac{K(R)^n}{3K(R/3)^n} = \frac{3^{n-1}}{1}. \tag{4.2}$$

The above equation shows that the ratio of the propagation loss of direct transmission to multi-hop transmission increases with the number of hops. This implies that, for a fixed distance between the source and the destination, there is a reduction in energy loss when the number of hops are increased. It should also be noted that the size of data transmitted plays an important part in determining the amount energy consumed during transmission or reception [11]. Taking these factors into consideration, a metric of the form *hopcount* $\times$ *data-payload* [1] (also termed *hopcount* $\times$ *data-byte* in this thesis) is useful for analysis of the system in terms of energy consumption during communication.

To illustrate a simple data communication model using multi-hop communication, consider a group of linear, equally spaced nodes that are capable of communicating with their immediate neighbors (Figure 4.1). In a centralized scheme, the number of hops to the gateway node can be expressed as $N \times (N-1)/2$ hops. Energy consumption is very high in this case. If the nodes are grouped into clusters (static clustering) [12], with a predetermined set of cluster-heads that can combine data by using lossy compressions [2], a considerable reduction in the *hopcount* $\times$ *data-byte* metric (Figure 4.1) results.

[1] Data-payload or data-byte refers to the size of the message that is to be transmitted.

4.2 Communication Models

Although a static clustering scheme looks promising initially, this scheme also involves an uneven distribution of load on the nodes, which can lead to poor network lifetime [53]. Localization of tasks among a subset of the total number of nodes along with dynamic cluster formation can reduce overloading a single node and, hence, increase network lifetime [24]. To address this issue, this thesis estimates the network coverage by using application level algorithms that minimize the *hopcount* $\times$ *data-byte* metric and improve network lifetime. A modified form of multi-level hierarchical clustering and directed diffusion-based communication schemes are used for implementing the coverage estimation application. These schemes are also termed as communication models in this thesis.

Multi-level hierarchical clustering model or hierarchical clustering consists of an energy-efficient election algorithm that uses maximal independent set (MIS) at multiple levels [2]. A diffusion-based model relies on a directed diffusion routing scheme and opportunistic aggregation [16]. In the following sections, a detailed look at the hierarchical clustering model is provided.

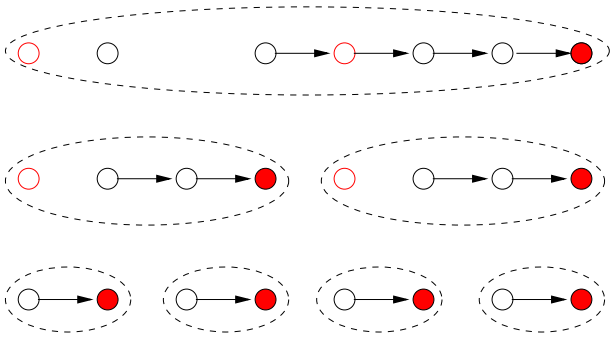

Figure 4.1: Clustering at multiple levels in linear, equally spaced, 8-node network with static clustering method

4.3 Hierarchical Clustering: Algorithm and Modeling

As illustrated in Chapter 3, a sensor network can be modeled using unit disk graph, where each node has its own set of neighboring nodes. A single node can be used to compute the coverage of its neighboring nodes (defined in Chapter 3) and itself. This form of localized coverage information from different nodes can be aggregated to obtain the complete network coverage. This motivates the use of clustering, which can be defined in simple terms as partitioning of the nodes (into *clusters*) based on certain properties that minimize the amount of data exchanged. The main properties of clustering with regard to the coverage application are: all the nodes should belong to one cluster, all the clusters should have certain limit on its size, and all the nodes in a cluster should be able to talk using nodes in the same cluster. It has been shown that properties of maximal independent set are favorable for clustering [2, 30, 54, 55].

Given that a link between two nodes exists if there is an overlap of the sensing area (S_i) and the commmunication range (tx_{range}), maximal indepedent set can be defined as follows.

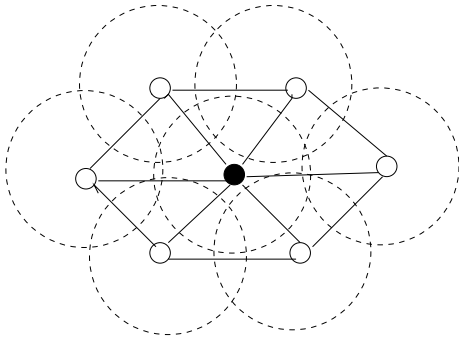

Figure 4.2: Unit graph for a constant node density configuration

Definition 2. Let $G = (V, E)$, where V represents the set of nodes $(v1, v2..vn)$ and E represents the set of links $(e1, e2..en)$. A maximal independent set is defined as a set of vertices, i.e a subset V' $\subset$ V such that no two vertices in V' are joined by an edge in E.

A simple algorithm for obtaining the maximal independent set is presented in the following section.

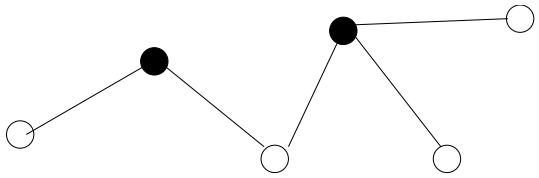

Figure 4.3: Example of leader nodes after using MIS algorithm

4.3.1 Basic MIS-based Election Algorithm

The main assumptions used in MIS election algorithm are as follows.

- The communication is reliable and bi-directional.
- There is a high degree of connectivity and node density.
- The random number used by one of the nodes is not reused by any of the other nodes.

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

Figure 4.4: Luby's Monte Carlo algorithm to determine a maximal independent set [2]

Figure 4.4 shows a basic algorithm for finding the *leader nodes*, which is based on Luby's Monte Carlo method for finding MIS (I is the independent set and $N(I)$ is the neighbor set of I). However, algorithms in sensor networks have to run in parallel and be asynchronous in nature. The paper by M. Jones and P. Plassmann [54] describes an asynchronous algorithm for vertex coloring a graph in

parallel. In this thesis, an extension of that algorithm is used for implementing the election stages of hierarchical clustering [2].

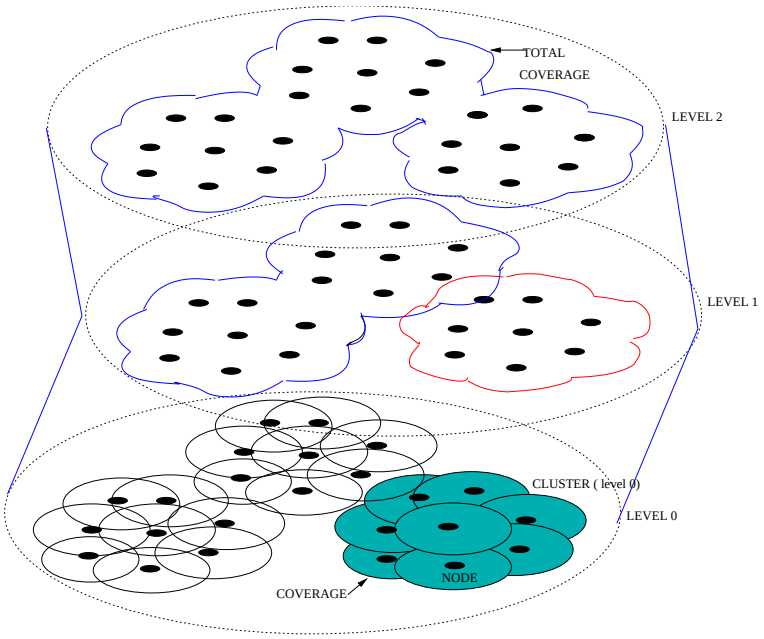

Figure 4.5: An illustration of various stages in hierarchical clustering

4.3.2 Algorithm for Implementation of Multi-level Hierarchical Clustering

Hierarchical organization of a network, which is a well-known and studied problem of distributed computing, has been proven effective in the solution of several problems such as the minimization of the amount of storage for the communication of information, optimization of the usage of network bandwidth, and distribution of resources throughout the network. In this thesis, multi-level hierarchical clustering is used as one of the communication models for coverage application.

In multi-level hierarchical clustering, the election algorithm is run at different stages. At each stage (or termed as *level* in this thesis), leader nodes from the previous stage take part in another round of MIS-based election process. Also, at each level the neighboring nodes are considered to be one hop further than it was from the previous level. Figure 4.7 shows an example of hierarchical clustering with three levels in which the complete coverage is obtained by the leader node at the

final level. The algorithm for hierarchical clustering is shown in Figure 4.6 [54, 2]. Some of the symbolic notations used in the algorithm are

S = sensor nodes,

$|S|$ = cardinality of S,

$r_n(i)$ = a random number $(0 < r(i) < 65566)$,

s_i = sensor node i,

$r_c(s_i)$ = effective sensor coverage radius of the sensor node i,

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

G_r = the intersection graph where s_i is a vertex in the graph, and

e_{ij}= an edge which exist if and only if $r_c(s_i)$ intersects with $r_c(s_j)$ and $tx(s_i)$ intersects with $tx(s_j)$.

1: $H = G_r$
2: do
3: $s_i \leftarrow r_n(i)$
4: $P = S$
5: $I = \emptyset$
6: $H = G_r$
7: while $P \neq \emptyset$ do
8: $V = \{s_i \in H : r_n(i) > r_n(j)\ for\ each\ e_{ij}\}$
9: $P = P \setminus V$
10: $P = P \setminus$ adjoining (V)
11: $I = I \cup V$
12: H is induced by P
13: end while
12: $G_r = H$
14: $S = I$
15: while $S \neq \emptyset$

Figure 4.6: Basic algorithm for multiple level MIS-based hierarchical leader election (based on work by S. Mehrotra [2])

The algorithm iterates as long as $S \neq \emptyset$. The inner iteration loop handles the part where the set of nodes V, with a random number greater than its neighbors' random number, is chosen from the

remaining graph H. The set of V is added to the maximal independent set, and is subtracted from H. The outer loop updates the vertices (nodes) based on Independent set of the previous iteration. Assuming that the number of neighbors for a node in G_r remains bounded as $|S|$ grows, the inner loop would execute $EO(log(|S|)/loglog(|S|))$ iterations [54].

4.4 Coverage Computation in Hierarchical Clustering

After each level of clustering, the leader nodes take part in the computation of coverage of the newly formed cluster. In order to compute the coverage of the new cluster, a leader-node obtains position information and coverage information (control points) from the smaller clusters (set K represent these smaller clusters) that belong to the newly formed cluster. Using this information, the leader-node determines a minimum set from K whose sensing areas cover the entire cluster (let Z represent this set (Figure 4.7)).

As described in Chapter 3, the computation algorithm using B-splines requires a set of data points, C. Data points are obtained by sampling the coverage of each cluster in set Z in such a way that the data points do not fall in the coverage of its neighboring clusters. The data points that describe the coverage of a cluster can be written as

$$C(l_z) = \bigcup_{i}^{Z} \overline{[(P_i \cap (\bigcup_{j=1;j\neq i}^{N} S_j))]}, \tag{4.3}$$

where S_j represents the sensing area of any cluster j, N represents the set of neighboring nodes as defined in Chapter 3, l_z represents the leader node, and P_i represents the data points that describe the sensor coverage of i^{th} node in the set Z. It should be also noted that the order of the data points have to be maintained while determining the set $C(l_z)$. Using the set $C(l_z)$, coefficients or control points are computed using the B-spline algorithm based on least square approximation.

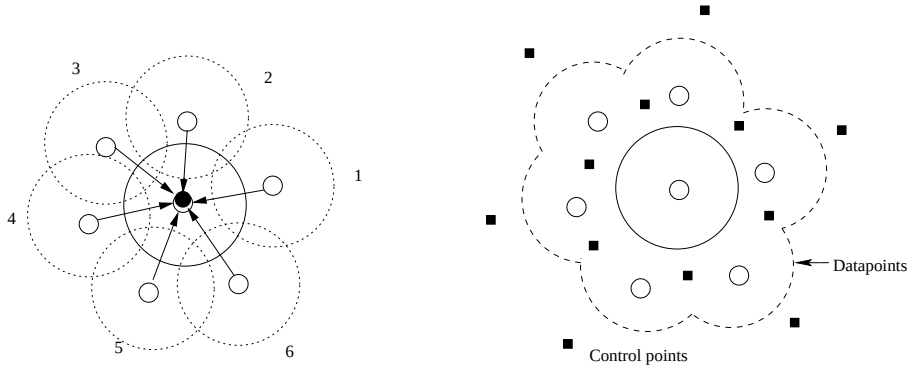

Figure 4.7: Generation of data points and control points (coefficients)

4.4.1 Algorithm for determining the set Z

A modified version of the *gift-wrapping* [27] algorithm is used for obtaining the set Z. As a first step, the algorithm finds the node having the largest x-coordinate among the nodes in the cluster. A horizontal line is swept in an anti-clockwise direction, using this node as the center. The node that has the least angle, and that is within the sensing range, is taken as the next node. These steps are repeated until the next node is the starting node. Using this simple algorithm the set Z can be determined.

4.5 Analysis of Clustering and Centralized Model

Analysis of hierarchical clustering and the centralized model provides a comparison of their theoretical performance. As described earlier, a *hopcount* $\times$ *data-byte* metric is a good measure for evaluating the efficiency of a communication model used in the coverage-estimation application.

A constant node density configuration, in which the distances between the nodes and their neighbors are equal, is used for an approximate analysis of the clustering algorithm. Figure 4.8 shows an example of a constant node configuration with 133 nodes and Figure 4.9 shows groups of hexagonal

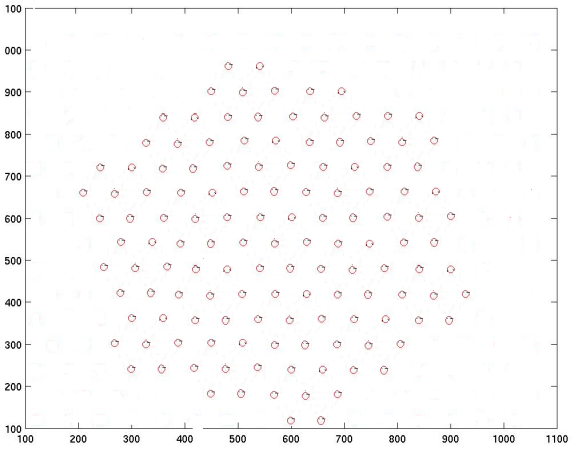

Figure 4.8: An example of constant node density configuration

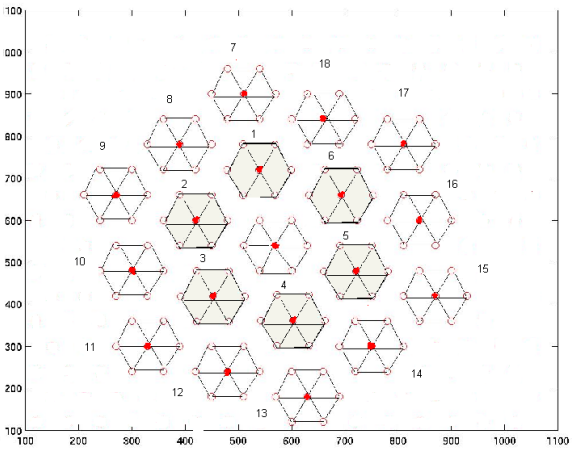

Figure 4.9: Hexagonal clusters after first level of elections

clusters formed after one level of election process. In order to simplify the analysis, these groups of hexagonal clusters are considered to form *cluster rings*. An example of one such *cluster ring* is shown in Figure 4.9 (clusters numbered 1 through 6 form one cluster ring). The total number of nodes participating in the process can be expressed in terms of the number of *cluster rings*. If there are v *cluster rings* after the first level of elections, the number of nodes in the network can be written as

$$n = (1 + 6\sum_{i=0}^{v} i) \times 7. \tag{4.4}$$

In an ideal case, clustering occurs in optimal number of levels with clusters having optimal size. Such optimal values play an important part in reducing the amount of energy spent during communication. It can be observed from Figure 4.9 that after the first level of clustering, a hexagonal cluster has at most six neighboring clusters. In an ideal case, the election algorithm would find the minimum maximal independent set [33], which would correspond to the election of minimum number of leader nodes that can cover the most neighboring nodes. However, the hierarchical clustering algorithm finds one of the independent sets as a solution. Therefore, certain assumptions

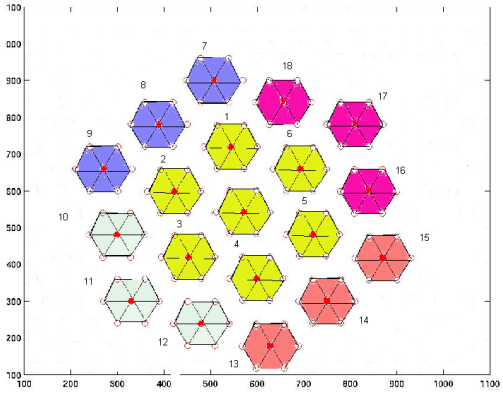

Figure 4.10: Clusters formed after the second level of clustering

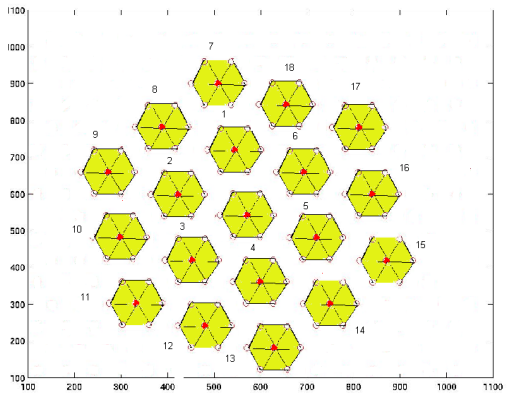

Figure 4.11: Complete coverage information obtained after the final level

are to be made in order to analyze the algorithm.

- A node located at the center is the gateway node and is considered to generate the highest random number.
- After the first level of clustering, there are, at most, three hops between the leaders.
- At any level, neighboring nodes are considered to be three hops further than the previous level.
- Message transmission and reception occur using an ideal routing scheme.

Based on the above assumptions a typical scenario that uses hierarchical clustering can be considered, one in which the load due to election process is distributed among the nodes. Figure 4.10 and Figure 4.11 shows the two levels of clustering. This scenario is scalable and similar trends are observed for larger number of nodes.

Using the above scenario, an estimate of the *hopcount* $\times$ *data-byte* metric can be obtained by using a theoretical formula. In order to analyze, the hierarchical clustering algorithm can be divided into three phases: the random number transmission, status information exchange, and information gathering by the leader nodes. In order to obtain the total hop-count, the number of

hops contributed by each phase is considered. The amount of data at each phase is also included in the theoretical formula. The parameters used in the theoretical calculation of the metric are

n - sensor nodes,
v - the number of rings around the center node (A ring is illustrated in Figure 4.9),
L - number of levels, and
B_1 - the size of the data in the random number transmission phase of the algorithm,
B_2 - the size of the data in the status information exchange phase of the algorithm, and
B_3 - the size of the data in the information gathering phase of the algorithm.

In order to determine the theoretical formula, one has to estimate the number of iterations and number of nodes that can participate in the election process at each level. The number of iterations can be associated with the status information exchange phase. It is known that if the number of neighbors of any node in G_r remains bounded as $|n|$ grows, the loop undergoes $EOlog(|n|)/log(log(|n|)$ iterations [54]. For the purpose of analysis, consider I_L to be the number of iterations at level L. For the purpose of calculation, values of I_L were based on actual simulation experiments on the constant node density configuration.

Another parameter required for the theoretical formula is the number of nodes participating in the elections at each level. At the first level, the number of nodes can be assumed to be $n/7$ due to the initial configuration. At higher levels, the number of leader nodes that participate in the election process can be assumed to reduce by a third of the previous level. This is based on the assumptions and the nature of clustering observed in the typical case (at a certain level, each cluster-head has two *followers* or neighbors). From the above observations, the theoretical formula for obtaining the *hopcount* $\times$ *data-byte* metric can be expressed as

$$hopcount \times databyte = H_0 + \sum_{j=1}^{L} H_j, \tag{4.5}$$

where H_j represents the value of the metric at j^{th} level. At a level L, the value of H_L can be expressed as

$$H_L(n) \equiv \begin{cases} 6nB_1 + I_1 6nB_2 + 6nB_3 & \text{if L = 0} \\ 18M_L LB_1 + I_L 18M_L LB_2 + 18M_L LB_3 & \text{L>0,} \end{cases} \tag{4.6}$$

where M_L represents the expression $n/(7 \times 3^{(L-1)})$. The number of levels can be described as that value of L (where $L > 0$ and $n > 7$) for which the expression $\lfloor \frac{n}{7 \times 3^{(L-1)}} \rfloor > 1$ is true. The above equation can also be written as

$$H_L(n) \equiv \begin{cases} k_1 n + I_1 n k_2 + k_3 n & \text{if}L{=}0 \\ M_L L k_4 + I_L M_L L k_5 + M_L L k_6 & L{>}0, \end{cases} \tag{4.7}$$

where k_1, k_2, k_3, k_4, k_5, and k_6 are constants. It can be observed that, for $L > 0$, the value of LM_L reduces with increasing number of nodes. Applying iterative methods and simplifications on the expression, it can be found that the bound on the metric can be given by $O(n)$.

For the purpose of obtaining a quantitative measure of the metric, the size of the application level messages is considered. The messages in the first phase of the algorithm consists of a random number, hop count, and a phase-identification number. The second phase consists of hop count, a phase-identification number, the node status (leader node or otherwise), and location coordinates. The third phase consists of the control points, hop information, phase-identification number, and the location coordinates. Based on the data messages, typical values of data-payload are: $B_1 = 12$ bytes, $B_2 = 20$ bytes, and $B_3 = 104$ bytes.

In a centralized model [24], data can be sent by multiple hops to the central gateway node. Figure 4.13 shows the nodes in the configuration divided into sets. Nodes that require equal number of hops to transport data to the gateway node are placed under the same set. The main assumptions made are as follows.

- B represents the total amount of data (in bytes) sent by each node, including individual coverage information.
- There is an ideal routing scheme in place.
- The coverage data from each individual node is required to obtain the complete coverage

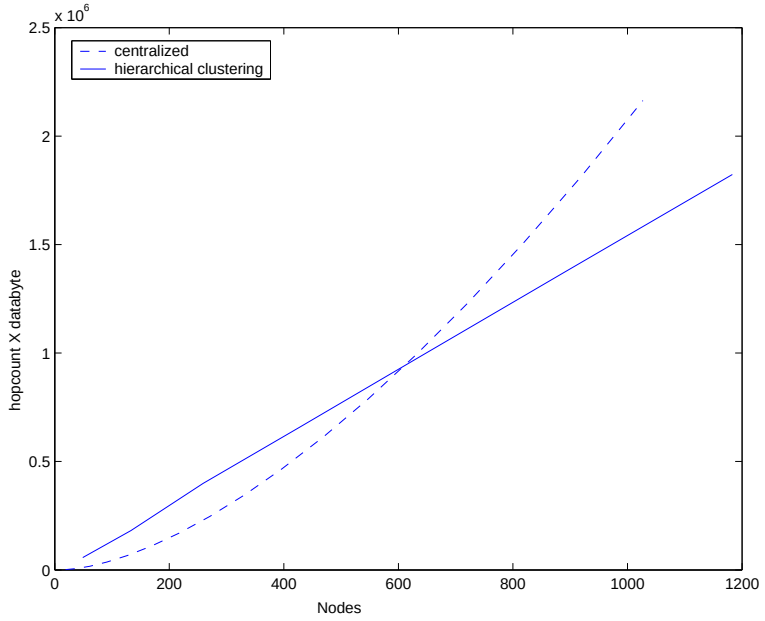

Figure 4.12: Hierarchical clustering Vs Centralized scheme - Theoretical results (based on the scenario described above)

information.

Based on the assumptions, the hopcount $\times$ databyte metric (for centralized scheme) can be expressed as

$$\begin{aligned} hopcount \times databyte &\equiv B(6 \times 1 \times 1 + 6 \times 2 \times 2 \cdots 6 \times L \times L) \\ &\equiv \sum_{i=1}^{S} 6i^2 B, \end{aligned} \tag{4.8}$$

where S is the number of sets (as shown in Figure 4.13), which is related to the number of nodes in the configuration. In order to obtain a comparison of the performance, the above expression can be simplified as follows.

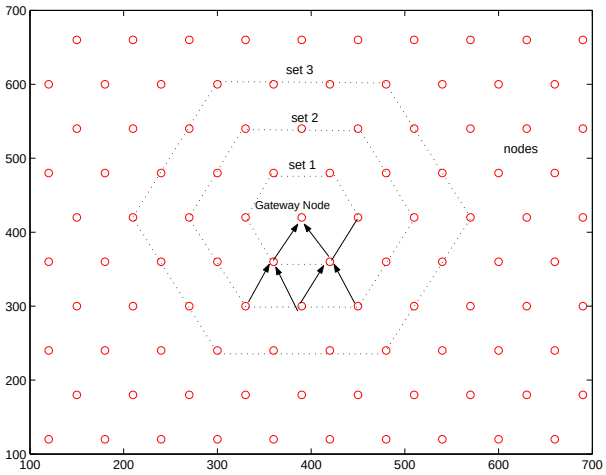

Figure 4.13: Constant node density configuration with nodes grouped into sets

$$hopcount \times databyte \equiv S(S+1)(2S+1)B. \tag{4.9}$$

It can be seen that the total number of nodes (excluding the gateway node), n, can be written as

$$n = 6\sum_{i=1}^{S} i = 3S(S+1). \tag{4.10}$$

From the above expressions, the metric can be written as

$$hopcount \times databyte \equiv \frac{n(2S+1)}{3}B, \tag{4.11}$$

where the value of $2S$ is less than that of n. However with the increase in the values of S and n, the metric follows a non-linear trend and therefore can be considered to be $O(n^2)$.

For obtaining a quantitative measure, B is assumed to be 192 bytes. The above results are useful for validation purposes, which is described in Chapter 8.

Chapter 5

Communication model: Directed Diffusion

This chapter provides an overview of the directed diffusion algorithm and the aggregation process involved in estimation of coverage. An analysis of the directed diffusion scheme is also included.

5.1 Basic Algorithm

The directed diffusion framework consists of four elements: interest propagation, data propagation, data caching, and reinforcement. Interest propagation is the initial phase in directed diffusion routing in which *interest* messages are broadcast by the gateway node. The attributes of the *interest* message are such that the gateway node receives information from all the nodes. These *interest* messages help in the creation of a *gradient* in a direction reverse of the interest propagation. The gradient is reinforced by the *exploratory* messages [1]. The *exploratory* messages, which are data messages, also enable multiple paths to be discovered. The paths can be considered to be a union of the shortest path trees rooted at the source nodes[1] [56]. Figure 5.1 shows a simple example of two sources with a set of shortest paths.

[1]Source nodes are the nodes in the network excluding gateway node.

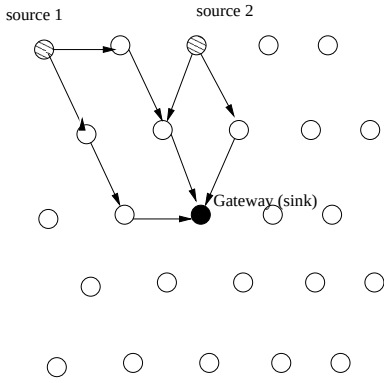

Figure 5.1: Illustration of shortest paths with a common link

The shortest path tree can be defined as a directed subgraph $G' = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$, such that: V' is the set of vertices reachable from source s in G, G' forms a tree rooted at s, and for all $v \in V'$, the unique path from s to v in G' is a shortest path from s to v in G [33].

If $p = (v_0, \cdots, v_k)$ be a path from $s = v_0$ to $v = v_k$, the weight of p is defined as

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v) \text{ [33]}, \tag{5.1}$$

where $w(u_1, u_2)$ represents the weight between the nodes u_1 and u_2. The shortest path is the one that has the least total weight, which is defined as the delay or latency between the nodes.

5.1.1 Collection of Coverage Information

Determination of local coverage information: At each node, coverage information is calculated using the algorithm in Chapter 3. The coverage information is stored as control points.

Dissemination of coverage: After local coverage information is available, it is disseminated across the network to compute the network's composite coverage. To obtain this information, the gateway node expresses a special *interest* message for the network. This message propagates throughout the network via flooding. Upon receiving an *interest* message, each node sets the sender node as a

parent node, which leads to formation of an aggregation tree with the root of the tree at the gateway node. This aggregation tree is updated periodically so that node failure recovery time is reduced. In the case of a node failure, different paths are set up using gradient nodes that reduce latency. A detailed description of the algorithm governing directed diffusion is provided in Chapter 2.

Aggregation of Coverage: As a message is passed along the aggregation tree, local coverage information gets aggregated. The message from a node, with coverage information about a certain area, can aggregate at a particular node if there is an overlap of the datapoints of the coverage. Owing to the use of B-splines, the size of the data remains the same throughout the aggregation process. Details of the implementation of the algorithm is provided in Chapter 6.

Figure 5.2 and 5.3 illustrates an example of the aggregation process. This kind of aggregation is called opportunistic aggregation [16].

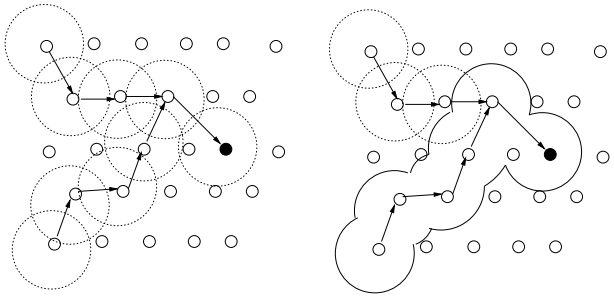

Figure 5.2: Example showing two different reinforced paths that are formed

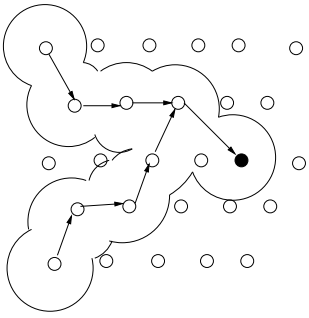

Figure 5.3: Shows aggregation among branches

5.2 Analysis of Directed Diffusion

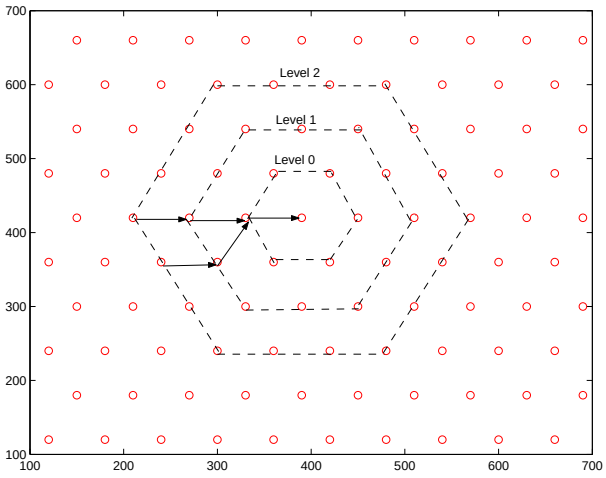

Figure 5.4: Constant node density configuration with hexagonal patterns representing a level

Figure 5.4 shows a constant node density configuration, which is used for the purpose of analysis. In this scenario, gateway nodes act as the sink node, while the other nodes act as the source nodes. In order to simplify the analysis, the nodes are assumed to lie on different *levels*[2]. The nodes in a certain level have equal number of hops (shortest number) to the gateway node. In order to analyze the model, certain assumptions are made, which are as follows.

- The source nodes encircle the *gateway node*.
- The probability of branches aggregating at a certain *level* is equal to that happening at any other level (Figure 5.4).
- All the nodes at a particular level send data to the gateway node in equal number of hops.
- Data delivery cost is the union of the shortest path tree rooted at each source [56].

The trees formed with different sources share certain common links that can be eliminated to form

[2]Levels represent hexagonally-placed nodes concentric about the gateway node.

a unified path. Intuitively, aggregation of branches at an early stage gives better performance. For the purpose of analysis, it is assumed that aggregation of branches occur at a later stage (*level 0*).

Based on the assumptions and observations, a theoretical formula for the *hopcount* $\times$ *data-byte* metric can be obtained. The metric can be computed by taking into account the contributions due to interest propagation and data propagation. Due to flooding, the contribution of interest propagation to the *hopcount* $\times$ *databyte* metric can be written as

$$\begin{aligned} I &= B_1(6 \times 1 \times 1 + 6 \times 2 \times 2 \cdots 6 \times L \times L) \\ &= B_1 \times \sum_{i=1}^{L} 6 \times i^2, \end{aligned} \tag{5.2}$$

where L represents the number of levels of nodes and B_1 is the size (number of bytes) of the interest message.

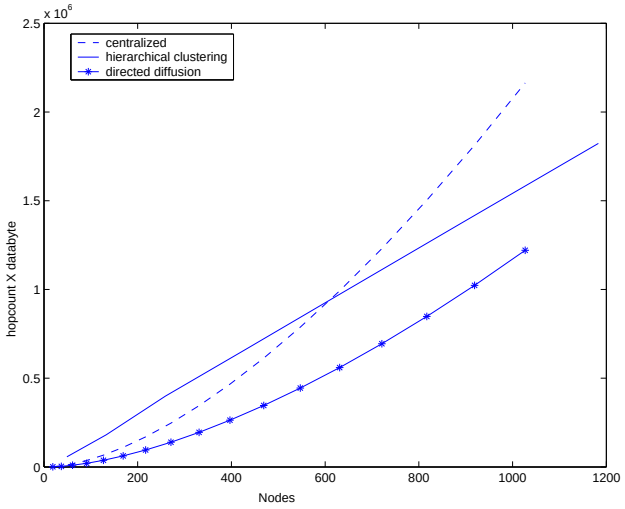

Figure 5.5: Theoretical comparison of the communication models

Based on the assumptions, it can be observed that the nodes at the edge of a *level* require $L \times (L - 1)/2$ hops to transport the data to gateway node. The number of hops contributed by other nodes

in that level is equal to $6L$. If B_2 represents the size of the data, the cost of data delivery can be written as

$$D = B_2 \sum_{i=1}^{L} (6i(i-1)/2) + 6i. \tag{5.3}$$

The total contribution to the *hopcount* $\times$ *databyte* metric is equal to $I + D$. For obtaining a quantitative value of the metric, typical size of the messages are considered (assume B_1 =12 bytes and B_2 = 104 bytes). Figure 5.5 shows the theoretical results of the three communication models based on typical values of data payload.

Chapter 6

Simulation Framework

The actual performance of the coverage-estimation application depends on various parameters such as interactions at different layers of the protocol, routing methodology, and issues related to delay. With an increase in the size of the network, analysis of performance becomes difficult. Simulation of the system helps in obtaining reliable performance-statistics for large-scale scenarios. This chapter provides an overview of the simulation framework used and outlines the design and implementation of coverage estimation schemes (the hierarchical clustering, the directed diffusion, and the centralized scheme). The chapter begins with a description of the simulation tool used to construct and execute the simulations. Subsequent sections give an overview of the framework and modeling of the implementation schemes.

6.1 Simulation Tool: ns-2

The *ns-2* (network simulator) is a research-oriented network simulation tool for modeling and analyzing wired and wireless communication networks. The availability of directed diffusion framework (in ns-2.26) and the ease of porting the application on to a WINS NG 2.0 sensor node were the main reasons for choosing *ns-2* over other simulators.

The *ns-2* is a discrete event simulator in which the system state is not defined continuously but only

at each event [3]. The *ns* is written in C++ (object-oriented language) with an Otcl interpreter functioning as the front-end. The Otcl acts as a scripting tool for defining events, creating instance of objects needed, and providing initialization parameters. The basic *ns* architecture is shown in Figure 6.1 .

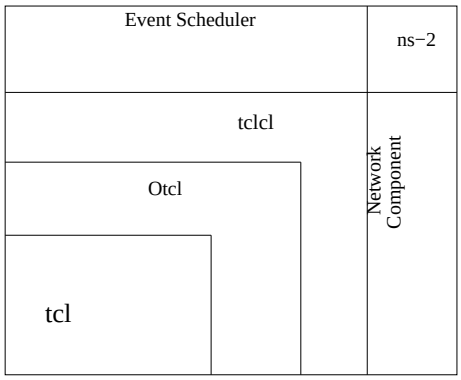

Figure 6.1: ns-2 Smiluation framework [3]

Figure 6.2 shows the wireless infrastructure used in the simulator [3]. Wireless support is provided by a *mobile-node* that is derived from the base class of a simple *node* [3]. The complete protocol stack consists of a link layer, interface queue, MAC layer, and a network interface with an antenna and a propagation model. These layers are interconnected and have access to a wireless channel. A diffusion core forms the routing agent as the directed diffusion framework is used for simulation purposes [57].

The link layer of the protocol stack (LL) simulates the data link protocols. The link layer also sets the MAC destination address in the MAC header of the packet. The ARP module resolves all IP to MAC address conversions. All outgoing packets from the routing agent get handed down to the LL layer. The LL hands down the packets to the Interface Queue. The in-coming packets enter the LL from the MAC layer through the *entry point* in the node. The ARP has a buffer for one packet that has an unknown destination address. The next hop must be known before it is injected into the interface queue [3].

The interface queue is usually a *priority queue* that gives priority to the routing protocol packets; it starts by removing those packets with a specified destination address. The MAC layer is the

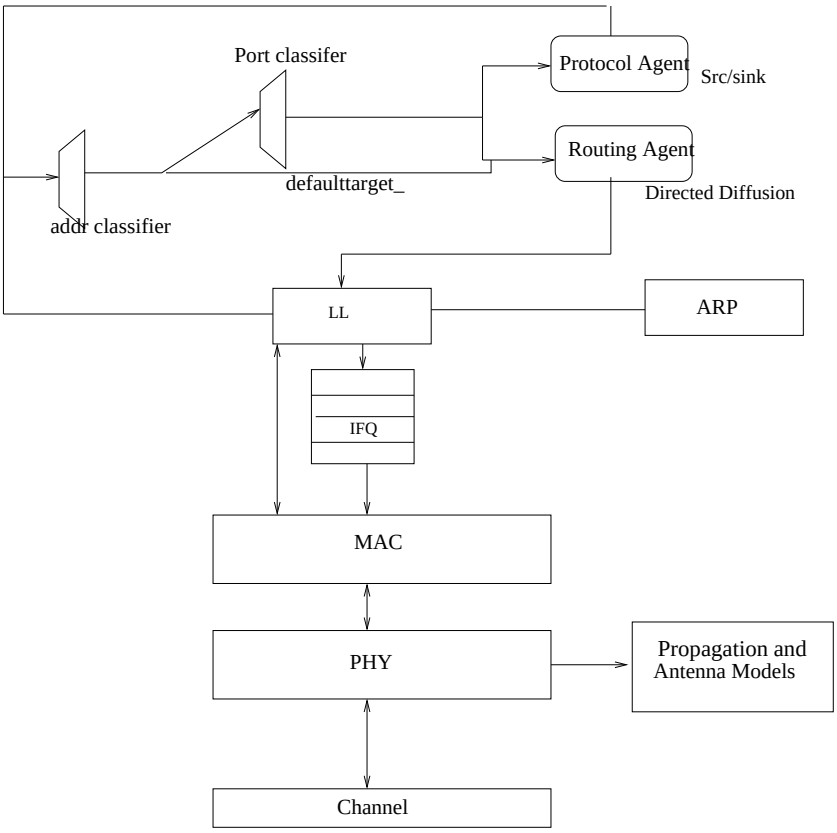

Figure 6.2: Protocol stack of the sensor node [3]

IEEE 802.11 distributed coordination function (DCF) implementation distributed by CMU [3].

The wireless shared media interface is the PHY. The propagation model receives the packets transmitted by other node interfaces to the channel. The interface stamps each transmitted packet with data related to the transmitting interface, such as power [3]. The propagation model uses the data in the packet header to determine if the packet has enough strength to be received. The current implementation of the propagation model is based on the DSSS radio interface [3].

Directed diffusion routing is implemented in ns-2 as the *diffusion-core* [3, 57] with libraries that provide an API for application-level programming. The API for diffusion provides the mechanism for creation of attributes, subscribing, and publishing [57].

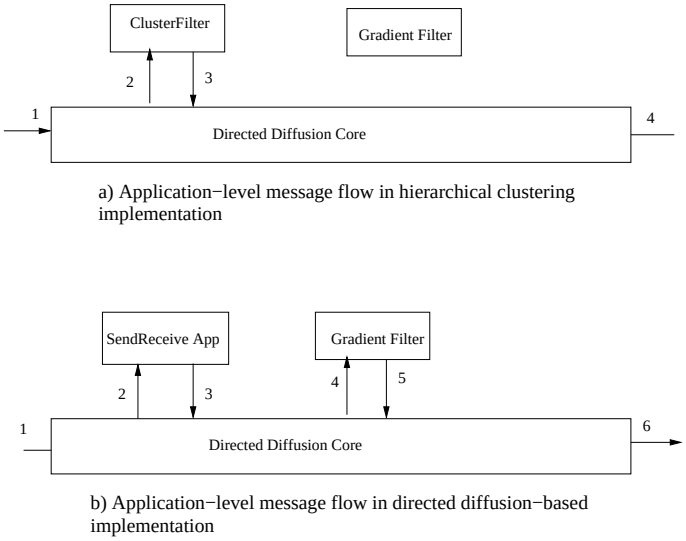

Figure 6.3: Interaction between the diffusioncore, API and filters

6.2 Simulation Models

6.2.1 Hierarchical Clustering

The maximal independent set algorithm is the basis for clustering at each level (see Chapter 4). With the passage of simulation time, the nodes learn if it is a *leader node* or not (non-leader nodes are termed follower nodes in this thesis). A *follower node* knows its leader; it interacts with the leader for certain tasks. A node that becomes a follower node in one level helps in forwarding messages in succeeding levels.

Figure 6.3 shows how the *diffusion core* (routing agent) in ns-2 works [3, 57]. The incoming message packets are sent across to the filters[1] registered with the *diffusion core*, in the order of the filter's priority. For hierarchical clustering, broadcast-based routing is used. To enable broadcasting of messages, the priority of the filter is set so that the *gradient filter* module is bypassed. The input priority is set so that the clusterfilter[2] receives packets directly from the diffusion core.

[1] Application-specific codes that run processes like aggregation and clustering are known as filters.

[2] The clusterfilter module is a new application level program that carries out the algorithm. A copy of the module resides in all the nodes.

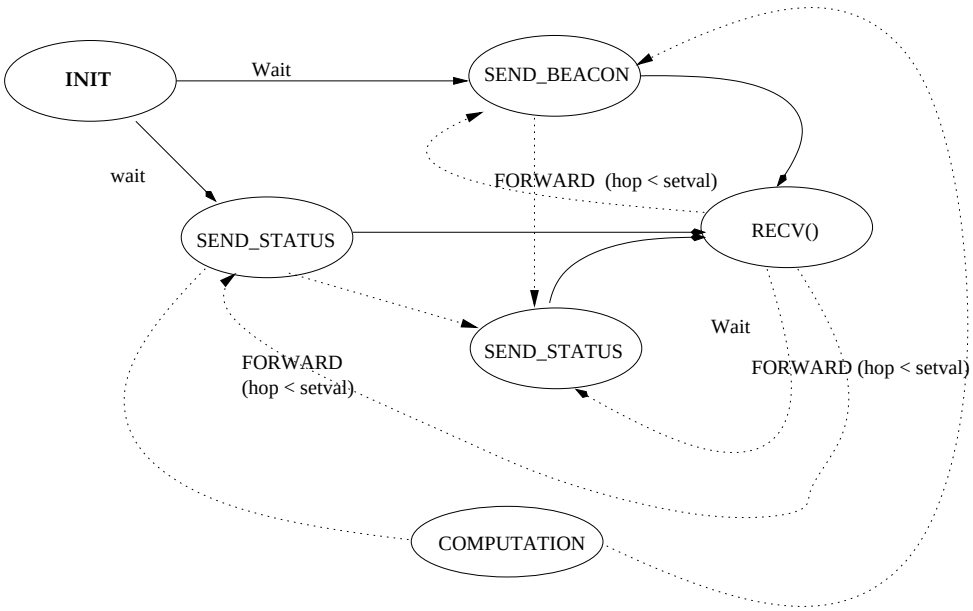

Figure 6.4: Simplified flow diagram for the clustering algorithm (ClusterFilter)

Figure 6.4 shows the flow diagram of the implementation. A filter can obtain incoming messages based on certain rules. If the matching rules are very generic, then the filter will receive messages from other nodes [57]. In the filter used in this implementation, the attribute specifies for all interests to be forwarded to a callback function[3]. The *recv()* function is a callback function in this filter. Initially, all nodes are set to undetermined node status (i.e., the node does not know whether it is a leader node or a follower node). Let *aqs-status* denote the state of the nodes, which is represented by a certain value as shown below.

$aqs\text{-}status = 1 \rightarrow$ Leader node

$aqs\text{-}status = 2 \rightarrow$ Follower node

$aqs\text{-}status = 3 \rightarrow$ undetermined node

After initializations, all nodes send a beacon message (which consists of random number and hop-count information) to its neighbors. After a certain timeout value has expired, some of the nodes can determine their status. As per the algorithm (Chapter 4), a node that has the highest random

[3]A callback function is a function that can read messages asynchronously from a queue, which can happen after a timeout provided that a message is available in the local queue.

number among its neighbors elects itself as a *leader*. These leader nodes send a status-information message to their neighbors. On receiving this message some of the neighbors can confirm that they cannot be leader nodes and become *follower nodes*. The *undetermined nodes* can use this information in the next iteration to resolve their status. The above steps (except sending of beacon messages) are carried out until all nodes are either leaders or followers.

After the first round of election, each leader calculates the coverage of its cluster. Details of this computation are provided in Chapter 3 and Chapter 4. This coverage information, which consists of data points, is used to calculate a set of control points, which are used to represent the clusters in the subsequent levels.

After the first level, the current set of *leader nodes* participates in another round of election. In this level, neighbors are two hops away; thus, at a level L, the neighbors are assumed to be $L + 1$ hops away.

Messages involved in the communication among nodes are composed of attributes that describe the data. Each piece of the attribute is described using a key-value-operator triplet, as shown in the example below. In order to ease attribute creation and manipulation, the API provides factories to create attribute [57]. Below is a sample code that was used to construct an attribute for a random number.

```
//Create a factory for the RAND_KEY attribute
NRSimpleAttributeFactory <int> RandomVal(RAND_KEY, NRAttribute::INT32_TYPE)

//Create random number attribute with operator as IS and value obtained from rand()
function NRAttribute *random = RandomVal.make(NRAttribute::IS,rand())
```

`RAND_KEY` denotes the key for the random number and`INT32_TYPE` indicates the primitive type of the key.

6.2.2 Redundant Message Suppresion in Hierarchical clustering

Loop formations and redundant messages are two problems that can affect the performance of a broadcast-based algorithm. The hierarchical clustering-based model uses various methods to reduce the amount of loop formation and redundant messages. The first level of clustering involves a single-hop broadcast of messages that can be received by nodes within communication range. For higher levels, nodes that do not participate in the clustering act as intermediate nodes. Intermediate nodes broadcast messages to all neighbors except the source node.

Hop-count information forms one of the attribute fields in the message that gets incremented, as the message gets retransmitted by each intermediate node. A local cache is included in the nodes to confirm that the message has been received. The message is not retransmitted once it reaches any participating node.

6.2.3 Directed Diffusion

Chapter 2 and Chapter 5 provide an overview of the basic directed diffusion scheme. As shown in Figure 6.3, a *sendreceive* (new module) application, as well as a *gradient-filter* (existing module), interact with the diffusion API [3, 57]. The current implementation of a gradient-filter in ns-2 uses delay as the metric for reinforcement purposes. Figure 6.5 shows the interaction between the *sendreceive* application, the routing API and the routing agent (diffusion core). The main function of the *sendreceive* application is to help each node calculate its own coverage. This application module acts a *coverage-data* publisher for all nodes except the gateway node. The gateway node uses the module to obtain aggregate coverage information.

The interest message (transmitted by the gateway node) consists of a *node ID* and *beacon message ID*. The application module has a callback function that can receive the interest message. On receiving an interest message, a node sends its local coverage information. The diffusion core accepts the messages from the API and processes them. When diffusion finds that another filter (i.e. gradient-filter) exists and is registered, the message is send to the gradient filter. The gradient-filter carries out certain operations before sending the message back to the *diffusion core*.

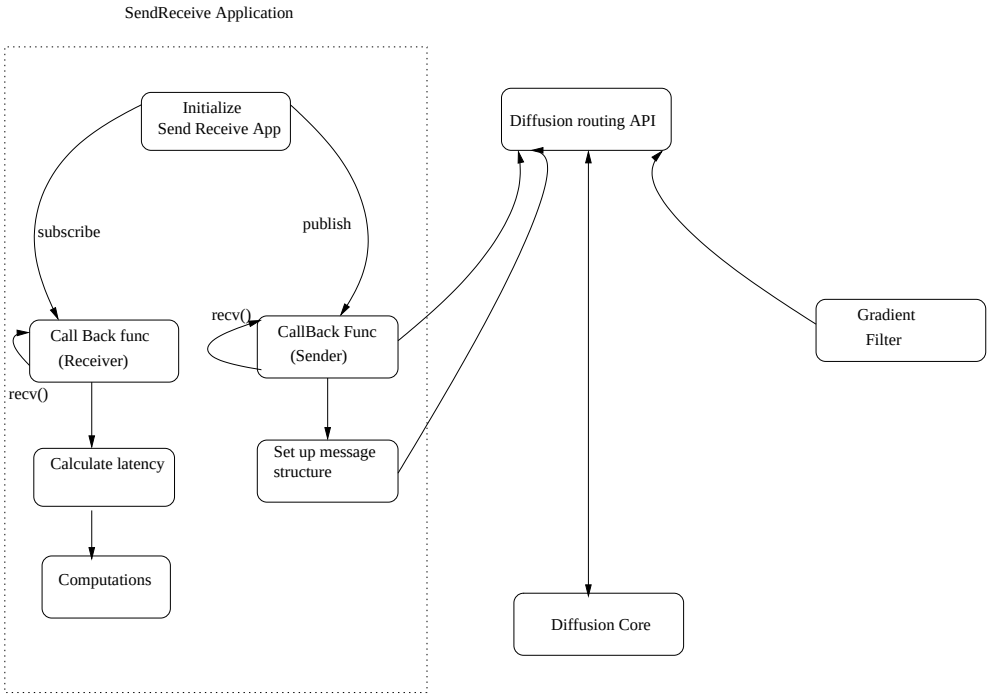

Figure 6.5: *SendReceive* application and interactions with other components

Figure 6.6 shows a simplified flow diagram of the *gradient-filter*. The current implementation of the gradient-filter uses delay as the metric for reinforcing the nodes. The gradient-filter has a cache for keeping track of messages and preventing loop formations. To enable coverage computation, a local coverage cache is also included. This local cache is used to check if computations can be carried out or not. If the message is from a new node, a check is carried out to see if the local coverage can be merged with the coverage information in the incoming message. If merging of coverage information is possible, aggregation is carried out and a new set of control points are computed. The local cache is updated with the new set of control points and the out-going message is also modified, so that it now carries the new set of control points. The gateway obtains the information from the nodes and merges the information.

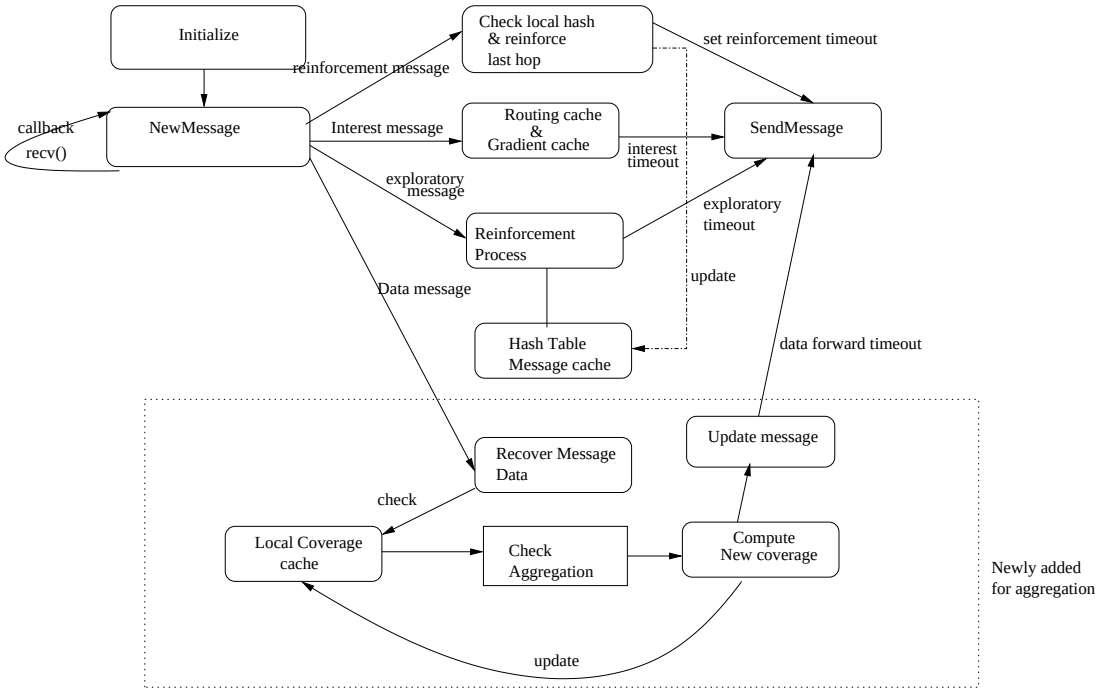

Figure 6.6: Gradient Filter with modifications for aggregation

6.2.4 Centralized Model

The centralized model is used to compare the two communication models in terms of energy efficiency metric, *hopcount* × *databyte*. The implementation is similar to the *cluster-filter* module, although different phases are not involved. Each node broadcasts information such as coordinates, hop information and data points of its coverage. Redundant transmissions are avoided by using message suppression techniques, which were described earlier. Geographical information and local caching are also used to avoid loop formation.

After the verification process (see Chapter 8), simulation using various node configurations (Appendix A) were carried out. The results and analysis of the experiments are provided in Chapter 7.

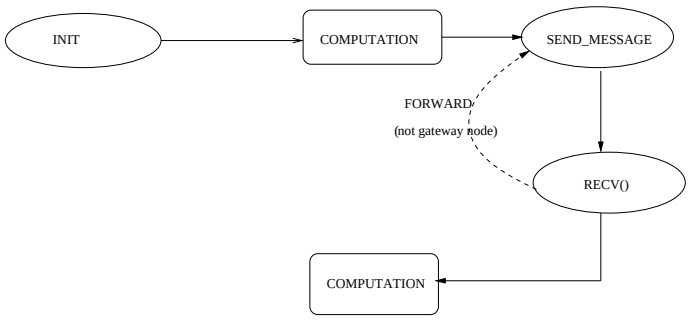

Figure 6.7: A basic implementation of centralized model

Chapter 7

Results

Chapter 3 provided a detailed look of the B-spline based coverage computation that describes the network's sensor coverage with minimal amount of data. This form of data-compression in the computation brings out the advantages of the communication models, which were discussed in Chapter 4 and Chapter 5. However, studies have shown that B-spline based computations are prone to error due to factors such as the use of a small number of control points, the use of uniform knot sequences [17, 49], and the possibility of sharp edges in the coverage of a topology. The computation of coverage is also dependent on the communication models, which can affect the overall performance of the system.

This chapter illustrates the trade-off between the accuracy of the B-spline based coverage and the reduction in energy consumption. Various advantages of the proposed coverage estimation method such as scalability and reliability are discussed using a number of experiments. Analysis of the communication models has shown that both the hierarchical clustering and the directed diffusion based approach are efficient when compared to centralized scheme (see Chapter 4 and Chapter 5). This chapter supports the theoretical analysis and shows how directed diffusion is more energy efficient as compared to the hierarchical clustering. Directed diffusion based scheme is also known to show properties like efficient node-failure recovery system and low latency, which are supported by various experiments in this chapter. Realistic scenarios were used to conduct simulation studies of the actual implementation.

The simulation experiments can be broadly classified into two groups, namely, experiments on the computation algorithm and experiments on communication models. The communication models include: the centralized scheme, hierarchical clustering (Chapter 4), and the directed diffusion-based model (Chapter 5). This chapter presents the results of the simulations on various scenarios and configurations of the sensor network. The experiments were executed on a machine with a Pentium III processor and 1GB of RAM. The operating system used was Linux. Experiments were carried out on three different configurations or scenarios, which are described briefly in the next section.

7.1 Simulation Scenario

Three groups of node configurations: constant node density configuration, boundary node configuration, and a geographical feature-based node placement scheme were used for the purpose of simulation experiments. Table 7.1 enumerates the different node configurations. A constant node density configuration (Configuration A) refers to the scenario in which nodes are placed in such a way that there is equal distance between the neighbors. Experiments on this configuration are useful for comparing the analytical results to the simulation results. As a boundary case experiment (Configuration B), the nodes were placed along the perimeter of a square field, which provides a 2-dimensional version of the scenario in which the nodes are placed linearly. This configuration can be considered to be a worst-case scenario as far as hierarchical clustering is concerned due to the large number of levels that may be required for the complete estimation.

The map shown in Figure 7.1 is from a region in Roanoke, Virginia. GIS (Geographical Information System) data was used as the input for generating the placement of nodes based on geographical features [58]. Three node topologies that displayed increasing node densities (with increasing number of nodes) were chosen for the experiments (Configuration C, D and E). Configuration C had the least node density when compared to the other two topologies. Figure 7.1 shows one of the regions that were considered for the placement of the nodes, along with 1024 nodes placed (Configuration D with 1024 nodes or D.1024). Examples of the other test cases are provided in Appendix A.

Configuration	Number of nodes	Representation	Area
A	64	A.64	500 x 450
A	100	A.128	600 x 600
A	256	A.256	1000 x 1000
A	529	A.529	1400 x1400
A	1024	A.1024	2000 x 2000
B	126	B.126	3000 x 3500m
B	252	B.252	3000 x 3500m
B	510	B.510	3000 x 3500m
C	64...1024 nodes	C.64... C.1024	600m x 900m
D	64...1024 nodes	D.64... D.1024	1500m x 1200m
E	64...1024 nodes	E.64... E.1024	1300m x 1400m

Table 7.1: Node configuration parameters

7.2 Definition of Metrics

Prior to presenting the results of the simulation sets, a group of metrics related to energy consumption is listed as follows.

Metric 1: A considerable amount of energy may be consumed by the gateway node during the process of coverage estimation. The energy remaining at the gateway node after one complete run of the simulation can illustrate the efficiency of the algorithm.

Metric 2: The time spent for the election of the *leader nodes* is a metric that gives the time taken at each level of hierarchical clustering.

Metric 3: The cluster size is yet another metric that provides information about how efficient the algorithm is in effectively spreading the load. Cluster size is the average number of nodes covered by the clusters at a particular level.

Metric 4: The hop-count $\times$ data-payload (or hopcount $\times$ databyte) product is a metric that is useful for measuring the effectiveness of the system. This metric has a direct relation to the energy consumed by the algorithm and provides a measure of the energy consumption due to message transmissions.

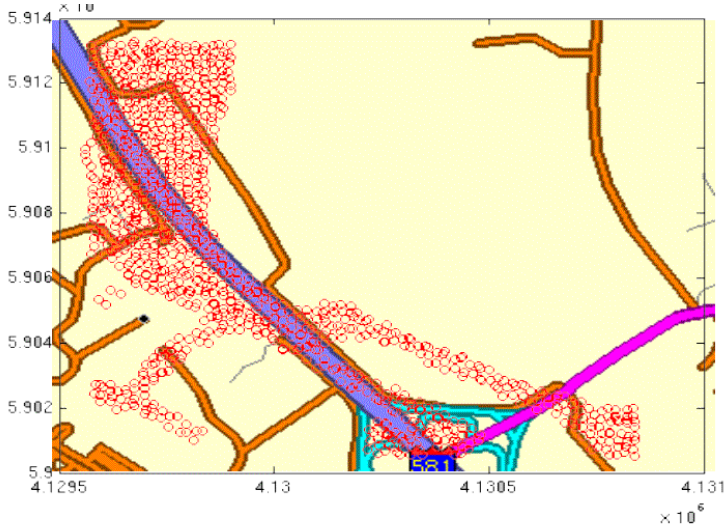

Figure 7.1: Configuration D with 1024 nodes placed over a region (D.1024)

Metric 5: The average delay/latency can be considered as the amount of time spent for a *send-receive* operation. This is a helpful metric to study the effectiveness of having a cached system such as directed diffusion routing.

Metric 6: The average compression ratio illustrates the efficiency of using B-spline geometry for compressing the number of data points. Average compression ratio gives the ratio between the data points compressed at the final stage (the final level in the case of hierarchical clustering) to the individual node coverage-data (before clustering or aggregation).

Metric 7: The mean square error (MSE) illustrates how much error is present between the B-spline curve generated by the control points and the actual data points. This metric can help in determining the trade-off between the number of control points to be used and the allowable error. The MSE can be considered to be the average of the squares of the distance between every actual datapoint and the point generated using B-spline. Mathematically, if $p_i(x, y)$ represents the i^{th} data point and $s_i(x, y)$ represents the i^{th} point generated using B-spline, then the mean square error (MSE) can be written as

$$MSE = \frac{\sum_i^n (|p_i(x,y) - s_i(x,y)|)^2}{n} \tag{7.1}$$

The following sections provide an outline of the experiments and the observations that were made. The common parameters used in the experiments are given in Table 7.2 [3].

7.3 Experiments on the Coverage Aspects

To illustrate the coverage estimation using B-spline geometry, consider Figure 7.2 and Figure 7.3 that show the visualization of the coverage based on an experiment on Configuration D. Figure 7.2 shows the data-points that were obtained at the final stage of clustering, while Figure 7.3 shows the corresponding control points (16 control points). These figures indicate that the data-points are large in number at the final stage. The use of *lossy* compression helps in bringing out the advantages of clustering and aggregation. Experiments were conducted to study the reliability and the factors affecting B-splines with regard to the application of coverage estimation.

Parameter Name	Parameter Used
MAC	802.11
Initial energy	10000J
Transmission Power	0.660W
Receiving Power	0.395W
Idle Time Power	0.035W
Transmission range	100m
Sensing range	100m
Antenna	Omni-Directional
Gain	Unity
Air Interface	914MHz Lucent WaveLAN DSSS Radio Interface
Number of control points	12

Table 7.2: Parameters used in the experimental setup [3]

7.3.1 Experiments for Finding the Compression Ratio

The average compression ratio (metric 6) provides a quantitative measure of the data compressed at the final stage. This is a good measure for judging the performance of B-spline geometry with regard to the compression of data. The compression ratio also indirectly provides an estimate of the reduction in communication overhead. In this experiment, the average compression ratio was calculated for an increasing number of nodes.

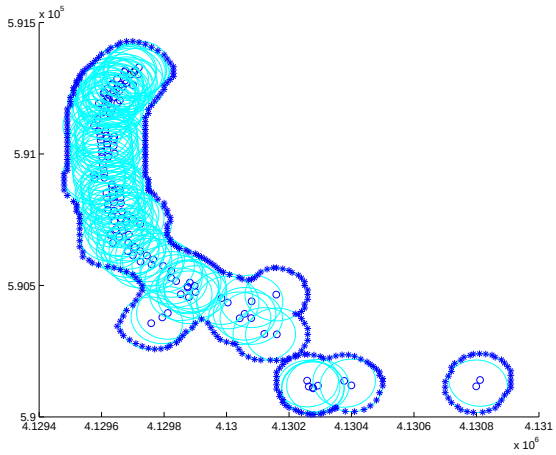

Figure 7.2: Data points describing the complete sensor coverage of D.128

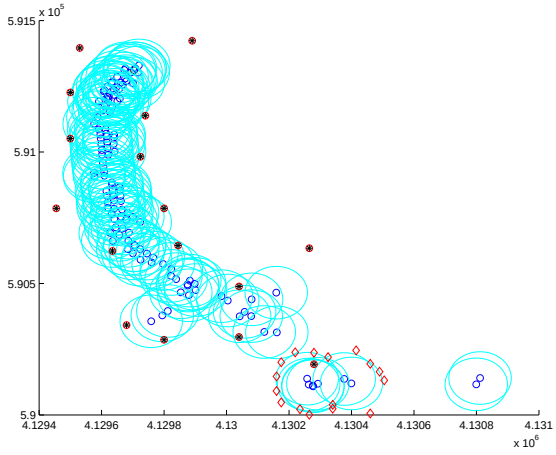

Figure 7.3: Control points representing the sensor coverage of D.128

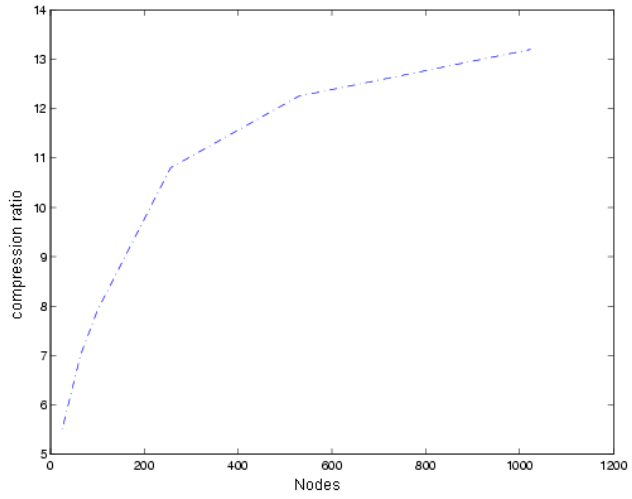

Figure 7.4: Results of the experiment for finding compression ratio - Configuration A (metric 6)

Figure 7.4 shows the results of the average compression ratio (metric 6) for an experiment carried out using Configuration A (hierarchical clustering as the communication model). The graph shows that the compression ratio increases almost linearly with the increase in the number of nodes. This increase is due to the fact that coverage area increased along with the increase in number of nodes, though the number of control points remained the same. This shows the effectiveness of the B-spline geometry in reducing the number of number of data points required to represent the coverage. However, when a large number of nodes are used, a larger number of samples of the data points (data points representing the coverage of each sensor node) are required. This is evident from the reduction in the slope of the plot for large numbers of nodes.

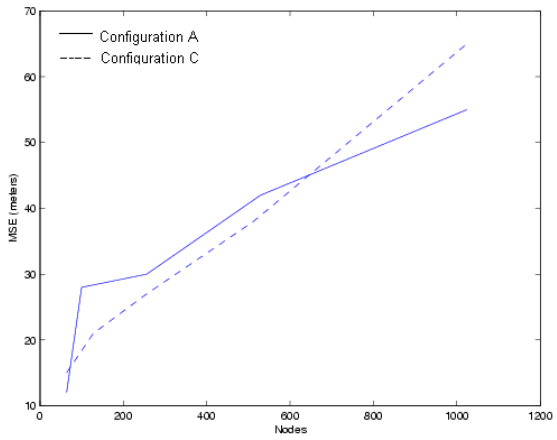

Figure 7.5: Results of Mean Square Error (metric 7)

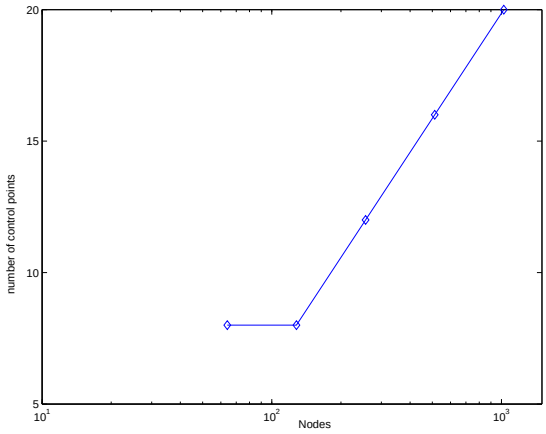

Figure 7.6: Number of nodes versus number of control points (MSE_{max}=40)

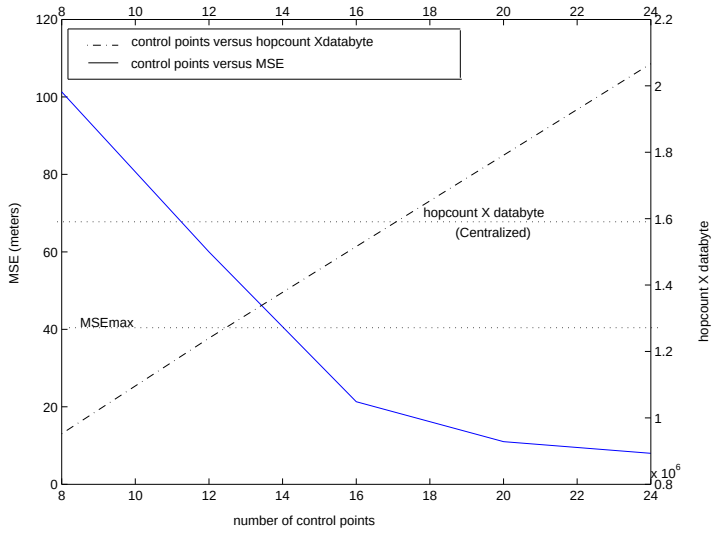

Figure 7.7: Control points versus MSE and control points versus hopcount × databyte using C.512

7.3.2 Experiments for Finding the Mean Square Error

Mean square error (metric 7) shows the reliability of the coverage information obtained (number of control points is kept constant). This is an important metric that gives an estimate of the error when B-splines are used. The experiments were conducted on Configuration A and Configuration C, using hierarchical clustering as the communication model.

The mean square error (MSE) values were obtained from the final level and are shown in Figure 7.5. It can be observed that both of the configurations show increasing values of MSE for an increasing number of nodes. From the experiments it was noted that larger number of levels of clustering were required with the increase in the number of nodes, which explains why there is an increase in the MSE. Yet another reason for the increase in MSE, in the case of Configuration C, was the sharp

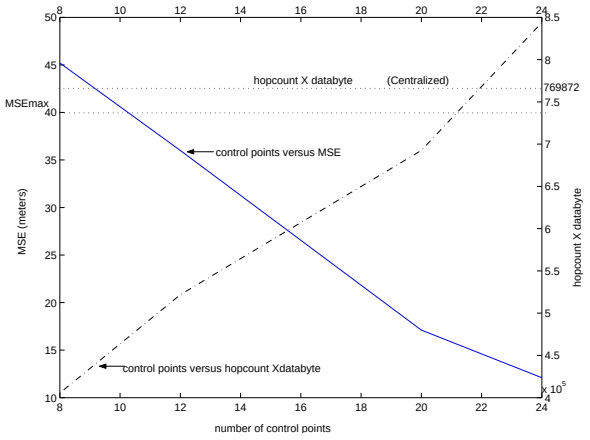

Figure 7.8: Control points versus MSE and control points versus hopcount × databyte using C.256

edges in the topology. Topologies that have coverage with sharp bends can induce large errors in the B-spline computation due to use of fixed knot sequances and limited number of control points [17, 49]. These experiments show that with the increase in the number of nodes and node density, the error in the coverage can increase to considerably large values.

In order to study the relationship between the number of nodes and the number of control points, consider a maximum value of MSE below which the coverage result can be assumed to be reliable. For the purpose of the experiment, assume the maximum value of MSE (MSE_{max}) to be 40. A large number of control points can be used to make the MSE to have a value within the limits. However, a large number of control points increases the energy consumption. An experiment was conducted on configuration C to determine the MSE for a group of control points (the control points being 8, 12, 16, 20, and 24). Figure 7.6 shows a graph plotted using the minimum value of control points required to have the MSE within the maximum limit. The plot shows that for a large number of nodes the required number of control points follows a logarithmic trend. Therefore, it can be concluded that the number of control points required for an increasing number of nodes

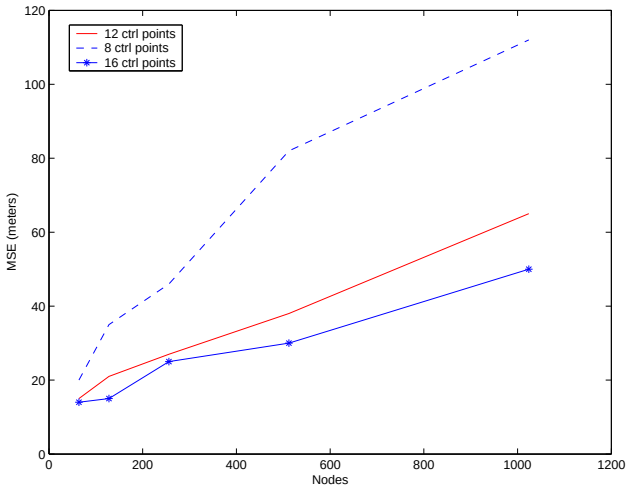

Figure 7.9: Mean square error for varying number of control points (Configuration E)

does not increase abruptly. Therefore this method of coverage estimation can be scalable to large number of nodes.

7.3.3 Effect of Control Points on MSE

Previous experiment showed that the MSE increases with the increase in node density and the number of nodes. An experiment was conducted using hierarchical clustering on Configuration E to check if increasing the number of control points can reduce the error (MSE) value. Figure 7.9 shows the results obtained for mean square error (metric 7) for varying values of control points. As expected, the increase in control points improves the performance. However, this improvement diminishes as the number of control points increases. This result can be attributed to the number of data points used to describe the individual sensor coverage. Larger number of samples of the individual node's sensor-coverage can improve the performance when a large number of control points are used. However, the use of a larger number of control points reduces the compression ratio and decreases performance in terms of metric 4 ($hopcount \times databyte$). Therefore, a trade-off exists between the compression ratio and the number of control points with regard to energy efficiency.

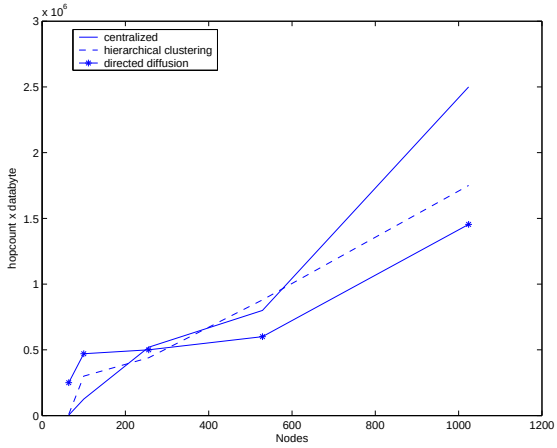

Figure 7.10: Results of comparison between the different communication models (metric 4)-Configuration A

Figure 7.7 and Figure 7.8 show example of the trade-off that exists between the number of control points and energy consumption (both directed diffusion-based and hierarchical clustering). The results are based on an experiment carried out on Configuration C.256 (using directed diffusion) and C.512 (using hierarchical clustering). The results of the MSE and the metric 4 (*hopcount* $\times$ *databyte*) are included in the same graph. Two kinds of limits can be applied for selecting the number of control points: the limit introduced by the value of metric 4 (*hopcount* $\times$ *databyte*) when centralized scheme is used and the MSE_{max}. As described earlier, MSE_{max} is assumed to be 40 for the purpose of the experiment. Using the constraints, a range of values for the number of control points can be obtained. From Figure 7.7 it can be observed that the control points in the range of 14-16 can be a perfect fit. Similarly, Figure 7.8 shows how the control points in the range of 10-20 can be a perfect fit. The experiments conducted show that the range of control points that can be used for this application reduces along with the increase in number of nodes. Therefore, such trade-off plots can help deduce the possible range of control points that can be used for a larger number of nodes.

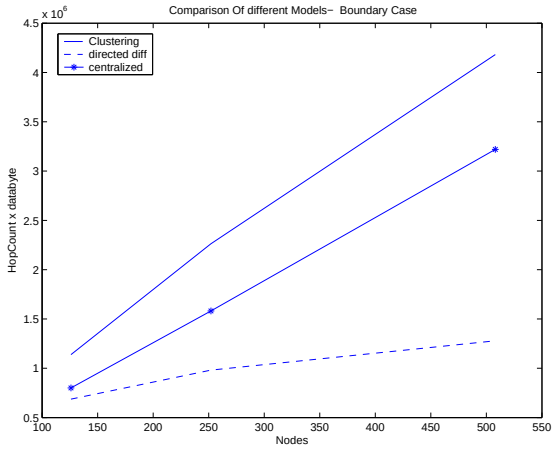

Figure 7.11: Results of the comparison between the different communication models (metric 4)-Configuration B

7.4 Experiments on Communication Models

The following sections provide a brief overview of the experiments that were conducted to study the performance aspects of the communication models. Comparison of the models with regard to the performance metrics is also provided.

7.4.1 Experiment on Configuration A and Configuration B (finding metric 4)

Experiments on Configuration A were carried out as a part of the validation and verification process (see Chapter 8). The results were compared to the theoretical analysis, which was discussed in Chapter 4 and Chapter 5. Figure 7.10 shows the results of the experiment where it can be seen that when a larger number of nodes are used, the directed diffusion-based model and hierarchical clustering perform better than the centralized scheme.

Experiments on Configuration B help check how the models perform in this scenario in which the nodes are placed along the perimeter of a square (the perimeter remains a constant). Figure 7.11 shows the results of the experiment, which indicates that the hierarchical clustering model does not perform well as compared to the centralized scheme. The amount of overhead due to cluster-

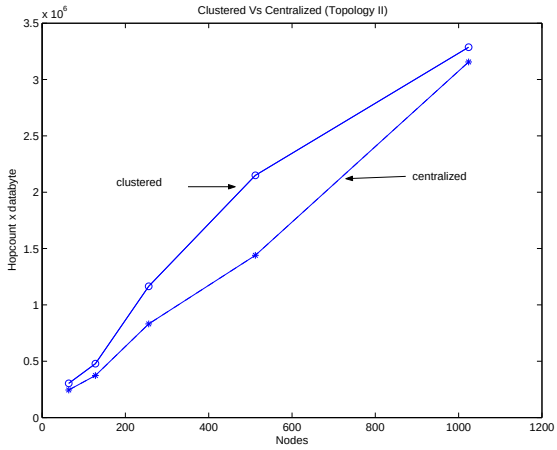

Figure 7.12: Centralized scheme versus hierarchical clustering for metric 4 (Configuration D)

ing was high as a large number of levels were required. However, directed diffusion-based model provided better performance as compared to the centralized scheme. From the observations it can be concluded that the hierarchical clustering is indeed dependent on the number of levels required, which hampers its performance.

7.4.2 Experiments on Geographic Feature-based Node Configuration (for finding metric 4)

The experiments were conducted using hierarchical clustering on all the three topologies to assess the behavior of the algorithm in terms of *hopcount* $\times$ *data-byte*. Figure 7.12 shows the comparison between the models with regard to Configuration D. Figure 7.13 show that the use of directed diffusion at a fixed level (for the purpose of this experiment a level that is two levels before the final) improves the performance of hierarchical clustering. This result is attributed to the reduction in the number of levels and the energy-efficient properties of directed diffusion. In practice, the fixed levels were chosen after determing the actual number of levels needed by the hierarchical clustering model. Figure 7.14 shows a comparison of the centralized model and the hierarchical clustering model in terms of metric 4 based on experiments on Configuration E. This graph indicates that the hierarchical clustering can show poor performance in certain topologies.

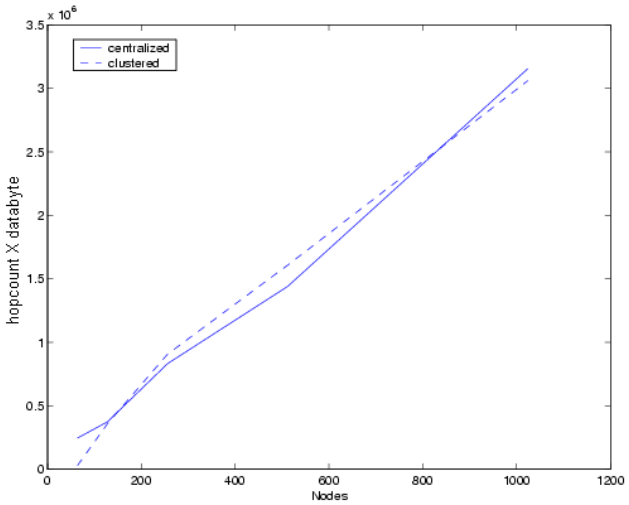

Figure 7.13: Centralized versus hierarchical clustering for metric 4 (directed diffusion as the final level (Configuration D)

High node densities can increase the time taken for the election of a leader (Figure 7.15). This is one of the reasons for the poorer performance of hierarchical clustering. Figure 7.15 show that the constant node density configuration takes lesser time for election as compared to the configuration C. Another reason for the poor performance is the rudimentary routing scheme that was used. At higher levels, the routing scheme plays an important part in reducing the communication overhead. Figure 7.16 shows the comparison between directed diffusion-based and the hierarchical clustering. The observations show that directed diffusion based model provides better performance in terms of metric 4.

7.4.3 Experiments for finding Metric 5

The delay that can occur during the aggregation process is an important metric with regard to the directed diffusion model. An experiment was conducted on Configuration D to determine the delay when using the aggregation method, which was described in Chapter 5 and Chapter 6. The coverage estimation algorithm using directed diffusion was run twice in the same simulation run. The second run was carried out after a certain time interval, in order to study the effects of caching on delay. Results obtained are shown in Figure 7.17.

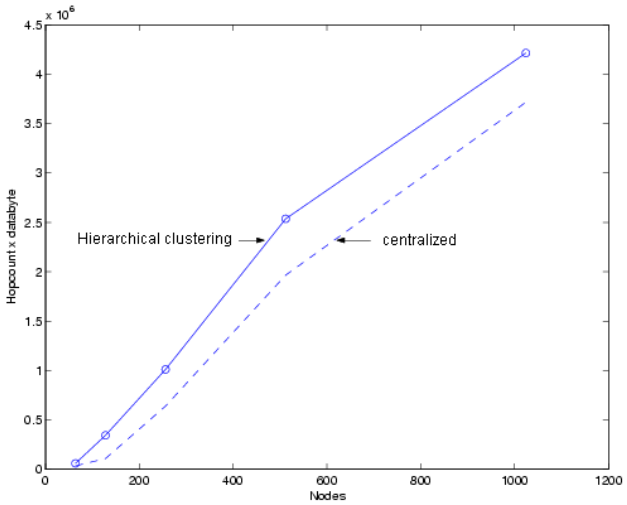

Figure 7.14: Centralized Scheme versus hierarchical clustering for metric 4 (Configuration E)

The results of the average delay metric show that there is a reduction in latency when a second run of the simulation is carried out. The reduction in delay is limited due to the fact that reinforced paths were different in both runs. Use of greedy-based aggregation can help improve reduce latency in the second run [16].

7.4.4 Effects of Node Failure

The hierarchical clustering approach, which does not use caching, gives large delays when there is a node failure. This high delay is due to the fact that node failures at a particular level require the complete election process to be repeated.

An experiment was carried out to see the effect of node failures on directed diffusion. From the list of reinforced nodes, certain nodes that were critical for communication were marked as critical nodes. As a part of the experiment, a percentage of the critical nodes were turned off at two different times.

Table 7.3 shows the results of the experiment. There are no substantial changes in the average delay or in the number of packets dropped. Figure 7.18 shows the cumulative distribution of the

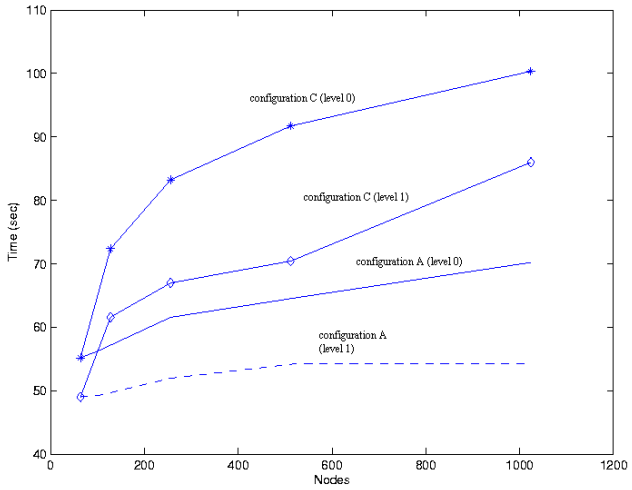

Figure 7.15: Average time taken for election (level 0 and level 1) (Configuration A and D)

Error Case	Average delay (sec)	Dropped bytes	Sent bytes
30 % critical node failure(time I)	0.421654	3437	358120
20 % critical Node failure (time I)	0.4108	3596	394520
20 % critical Node failure (time II)	0.4102	3616	426920
No node failure	0.3118	3355	414388

Table 7.3: Statistics of experiment on node failure failures

network's end-to-end delay. It can be seen that there is no considerable difference between the two plots. Certain discrepancies, such as values in the negative region, can be attributed to the initial phase, when the actual values of latency could not be obtained from the trace file.

In a typical case scenario, node failures can occur in the critical path. Directed diffusion can change the reinforced paths dynamically, during run-time, due to the set of gradients that are created during the initial phase. This aspect of directed diffusion helps reduce the time taken to recover from a node failure.

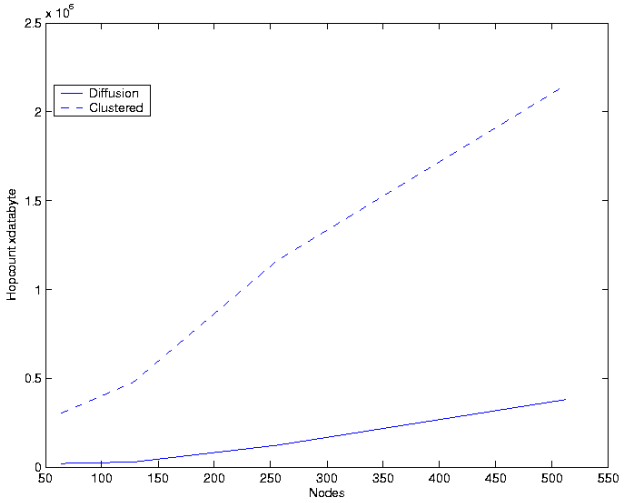

Figure 7.16: Hierarchical clustering versus directed diffusion based for metric 4 (Configuration D)

7.5 Summary Of Discussions

Simulation results were split into two different categories, namely, computation aspects and the communication aspects. Both sets of results have influence on each other beacuse the models use the properties of B-splines for improving performance. Experiments on the computation aspects indicated the advantages and constraints of using B-splines as a *lossy compression scheme*. B-splines effectively compress the amount of data transmitted, thereby improving the overall performance. A trade-off exists between the number of control-points and energy-efficiency. The error in coverage information, measured in terms of mean squared error, was analyzed to depend on coverage and number of aggregations. The results from the coverage computation aspects can help in designing the model according to given constraints. Experiments on communication models showed that they followed the theoretical trends. Directed diffusion showed consistent performance, while hierarchical clustering was shown to have limitations based on the number of levels. The limitations of hierarchical clustering were predominant in high node density case due to a large election time and an inefficient message suppression mechanism. Directed diffusion was also shown to have efficient node failure recovery technique.

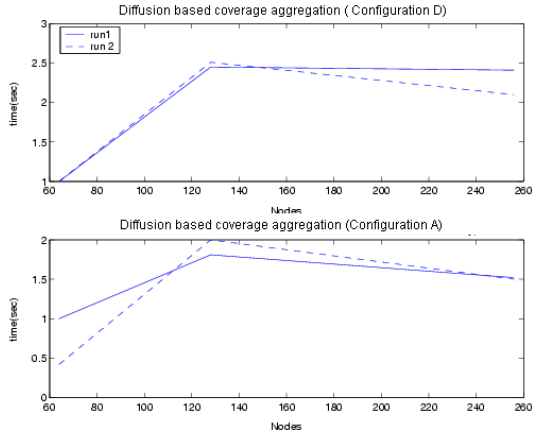

Figure 7.17: Results of Metric 5 - Directed diffusion-based model (Configuration A and Configuration D)

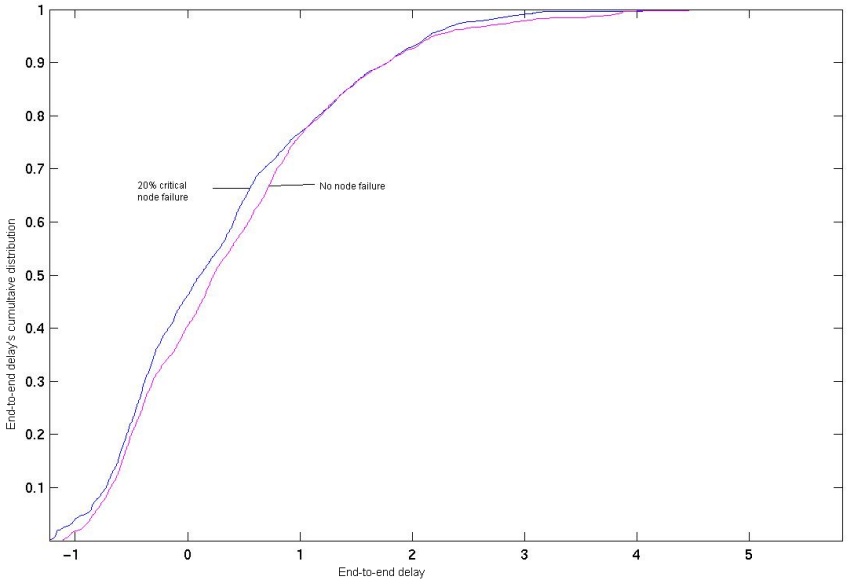

Figure 7.18: End-to-end delay and its cumulative distribution

Chapter 8

Validation and Verification

Validation and verification are significant elements in any simulation study. Confidence can be placed in the results of a study based on verification and validation. Model validation can be defined as "substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model" (Schlesinger *et al.* [59]). Schlesinger *et al.* also defined model verification as "ensuring that the computer program of the model and its implementation are correct". This chapter deals with the approaches taken to validate and verify the simulation models.

8.1 Validation

Validation is an integral part of model development and can be considered to be iterative in nature. Validation can be carried out by comparing the models with real system measurements or with theoretical results of the application. The configuration and the parameters (Table 7.2) used for the simulation are based on experimental studies [60]. For example, the energy model, the radio model, and the sensor node model used in the simulation were shown to produce reliable results [60]. Experimental results of the coverage-estimation application (using hierarchical clustering or directed diffusion-based model) are not available. However, it is possible to validate the simulation model

approximately, by comparing the results with an analytical model. Chapter 4 and Chapter 5 provide a brief description of the analytical models.

Chapter 7 described the experiment carried out on the configuration A, which helps compare the analytical results and the simulation results. In order to obtain reliable values, the results were taken as an average of multiple runs, where each run had a different initial random seed. The results from the multiple runs showed negligible difference, which verifies the reliability of the output. The following section describes the comparison between the simulation model and the analytical model.

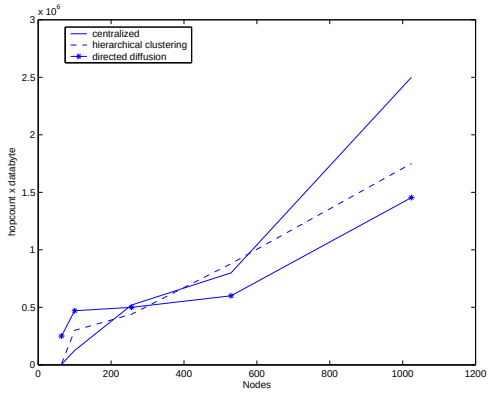

Figure 8.1: Results of comparison between the different communication models (metric 4)

8.1.1 Comparison of the Analytical model and Simulation model

To compare the analytical model and the simulation model there should conceptual-model validity for the analytical model. This means that a theoretical model can be used to validate the simulation model if certain assumptions of the theoretical model have been proved by measurements or by expert intuition [61]. In Chapter 4 and Chapter 5 certain assumptions were used to intuitively analyze the algorithms. For example, in the case of hierarchical clustering, the assumptions that influence the results are based on the cluster size, the number of levels, the cluster formation, and perfect routing. The cluster size at each level and the total number of levels affect the results of *hopcount* $\times$ *databyte* (metric 4). The assumptions made regarding these factors are based on a typical case (Configuration A) and the simulation experiments for the typical scenario have

shown that the assumptions are reliable. However, discrepancies do exist in the simulation models that were not considered in the analytical model. For example, it was assumed that after the first level of elections, the number of hops between the immediate leaders is three. From the experiment, it is seen that in certain cases the number of hops can be two. This result is due to the communication range of 100m that was used. Moreover, the hierarchical clustering and the centralized scheme are implemented using a rudimentary routing scheme that is primarily based on broadcasting of messages. Therefore, although an exact comparison is not possible with analytical methods, an approximate validation can be attempted. Figure 8.2 shows the comparison between the simulation results and the analytical results for the hierarchical clustering model. The result shows the similarity between the analytical trend and the simulation output.

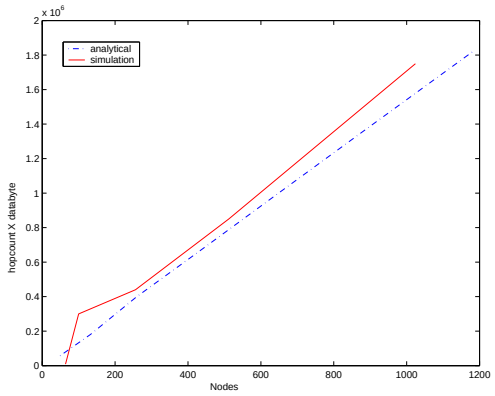

Figure 8.2: Comparison between simulation results and analytical results- hierarchical clustering

Figure 8.3 shows the comparison in the case of the centralized scheme, while Figure 8.4 shows that of directed diffusion. The results show how similar the trends are even though there are some differences.

In the case of directed diffusion, a more accurate numerical validation is possible using the theoretical equations that were derived by Intanagonwiwat *et al.* [56]. The authors derived the equations based on the assumption that the tree that diffusion scheme constructs is the *union* of the shortest path tree rooted at each source. Taking into consideration the assumptions used for deriving the equations, each node is made to send identical coverage information.

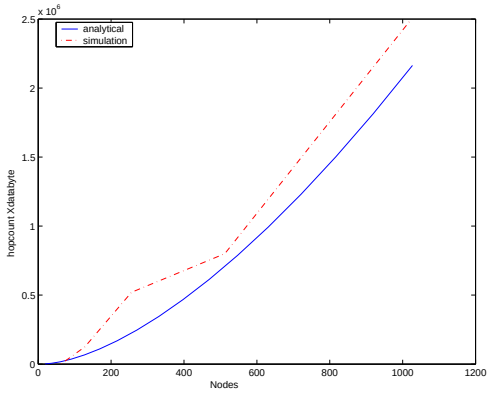

Figure 8.3: Comparison between simulation results and analytical results- centralized scheme

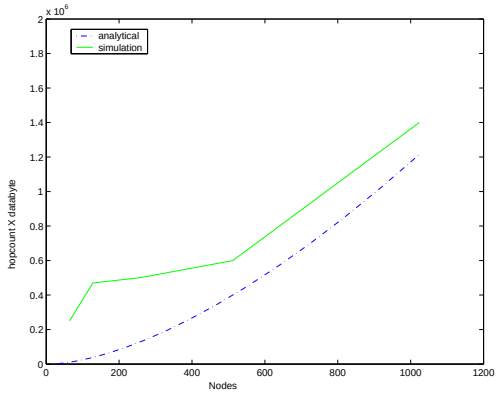

Figure 8.4: Comparison between simulation results and analytical results- directed diffusion

Figure 8.5 shows the scenario used for this analysis. Consider s1, s2, and s3 to be the sources that transmit identical coverage information. Let reception and transmission account for one unit of energy cost. G is the gateway node or the sink node that initiates the coverage determination. Data delivery cost can be obtained from the generalized equations provided by the authors (Intanagonwiwat*et al.* [56]).

$$C_d = C(\cup T_{1\rightarrow n}) = C(T_1) + \sum_{j=2}^{n} \left(H(T_j - \cup T_{1\rightarrow(j-1)}) = D(T_j - \cup T_{1\rightarrow(j-1)}) \right), \tag{8.1}$$

where

$$H(T_j - \cup T_{1\rightarrow(j-1)}) = H(T_j) = 2\left(\sqrt{N} - 1 - \left(\lfloor \frac{j}{2} \rfloor d_n - min(\lfloor \lfloor \frac{j}{2} \frac{d_n}{d_m} \rfloor, \lfloor \frac{m-(jmod2)}{2} \rfloor) d_m \right) \right)$$

$$D(T_j - \cup T_{1\rightarrow(j-1)}) = 2\left(\lceil\frac{m+(jmod2)}{2}\rceil d_n + \sum_{l=1}^{min(\lfloor\lfloor\frac{j}{2}\rfloor\frac{d_n}{d_m}\rfloor,\lfloor\frac{m-(jmod2)}{2}\rfloor)} min(d_n, d_n\lfloor\frac{j}{2}\rfloor - ld_m\right)$$

In the equations, the shortest path tree rooted at source j is denoted by T_j; m and n represents the number of sinks and sources in the system.

Using the above equations, the data delivery cost for this particular scenario is found to be 16 units. From observations, the cost due to interest propagation is found to be 22 units. Therefore, the total *hopcount* × *databyte* is 1152 bytes. The simulation model gives a result of 1456 bytes. The increase can be attributed to implementation issues such as broadcast-based interest propagation.

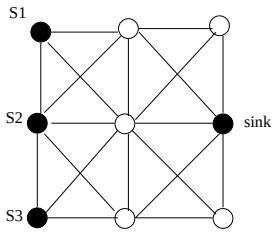

Figure 8.5: Simplified configuration for obtaining a theoretical result

8.2 Verification

Verification can be considered as a debugging process that tries to ensure that the model does what it is intended to do [61]. Some of the model verification techniques used are described here.

8.2.1 Simplified Test Case

Simple test cases can be easily compared to the simulation results. Figure 8.6(a) shows the test case that is used for verifying the hierarchical clustering model.

The nodes are given a pre-determined set of random numbers before the algorithm is run. Message transmissions are recorded and analyzed. In the first phase of the algorithm, all the neighboring nodes communicate their random number values. Node 1 is the only node that can determine itself

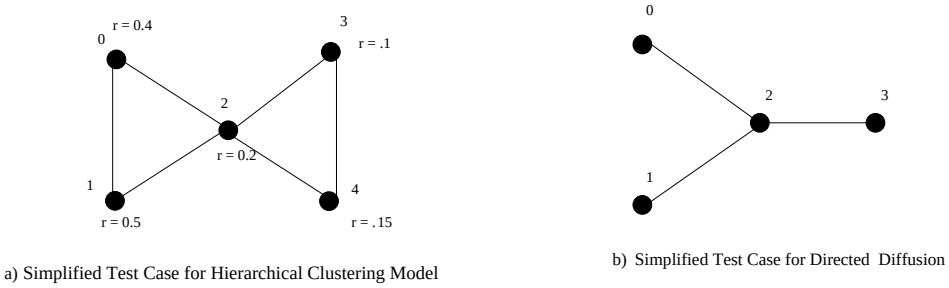

Figure 8.6: Simplified test cases

as its leader. Node 1 set its *aqs-status* to 1. Having heard from node 1, nodes 0 and 2 decide that they are follower nodes and sets *aqs-status* to 2. In the fourth phase, nodes 3 and 4 hear that node 2 is not participating any further. Node 4, which has the highest random number as compared to node 3, becomes a leader. Node 3 now drops out of the election and changes its *aqs-status* to 2. Therefore, at the end of level 0, nodes 1 and 4 are elected as leaders. The leaders then move on to level 1 after resetting their *aqs-status* and computing their respective coverage. At level 1, the number of hops to neighbor is set to two. Nodes 1 and 4 exchange their random number attribute and node 1 decides to be a leader.

Figure 8.7 shows a constant node density configuration with 100 nodes (D.100), which is used to verify the model. The solid-filled nodes represent the leaders after level 0. It is observed that the leaders are indeed part of a maximal independent set.

Figure 8.6(b) shows the configuration used for verifying the directed diffusion model. A logging functionality, which is added to the current implementation of *gradientfilter* (provided with the ns-2 framework), is used for tracking the messages and verifying the implementation. Nodes 1 and 2 send coverage information, while node 3 acts as the gateway node.

8.2.2 Continuity Test

The continuity test consists of running the simulation for slightly different values of input parameters. For a single parameter change, a slight change in the output can be observed. To test for

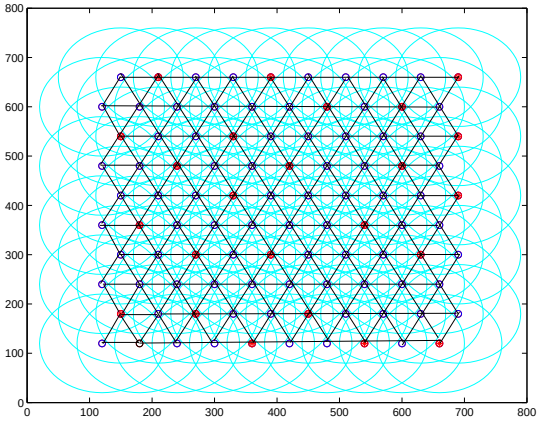

Figure 8.7: Configuration A.100 with leaders, after one level of election

continuity, an experiment was conducted on Configuration D.256. The number of control points was increased and the changes were observed.

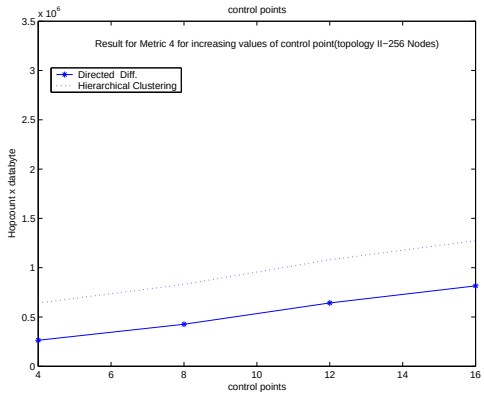

Figure 8.8: Continuity test (varying values of control points) *hopcount* × *databyte* Vs Control-points

Figure 8.8 show the result of the test on the hierarchical clustering model and the directed diffusion model. It can be seen from that the result complies with the continuity test.

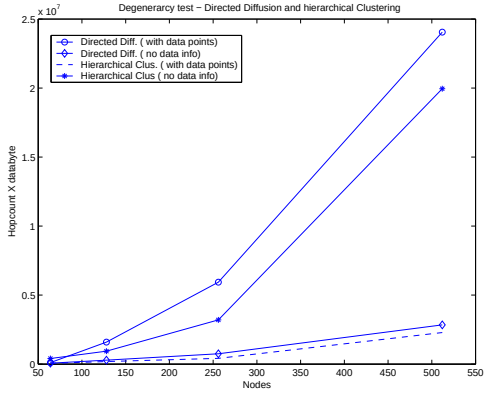

Figure 8.9: Dengeneracy test carried out using two extreme cases (Configuration D.512)

8.2.3 Degeneracy Test

Degeneracy tests consist of inspecting the simulation model for extreme values. Although these tests do not represent actual scenarios, they provide information about boundary conditions. Two tests were conducted: one with data points as coverage, the other, with no coverage information. Figure 8.9 shows the results of the degeneracy tests for the directed diffusion and the hierarchical clustering model. As expected, using data points to represent coverage leads to increase in energy consumption. This finding is exemplified by the sharp increase that occurs in both the models. Hierarchical clustering shows poorer performance for the extreme cases. The results of the actual B-spline based experiments lie within these boundary cases, and therefore, verify the model for the degeneracy test.

8.3 Summary of Discussions

In this chapter, the simulation outputs were validated using analytical models that conformed to a few valid assumptions. The results followed the analytical trends to a high degree of similarity (all the models). Proven sets of equations were also used to validate the simulation results of the directed diffusion-based model. Verification activities such as simplified test cases, continuity test, and degeneracy test illustrated the reliability of the simulation model that was implemented.

Chapter 9

Experimental Setup

Sensoria Corporation's WINS NG 2.0 (Wireless Integrated Network Sensor - Next Generation) is a distributed sensor development system that was developed for the SensIT (Sensor Information Technology) program. The WINS NG 2.0 node provides basic sensor node operations, with API interfaces to develop software applications for areas such as data storage, data dissemination, and sensor technology application demonstration (such as coverage estimation application). The simulation models in *ns-2* can be easily ported to the WINS NG 2.0 nodes with minimal changes. This chapter provides a brief description of the node, testbed setup, and an example of the actual experiment.

9.1 Sensor nodes and the Testbed

The SH-4 processor, running at 167MHz with a Linux 2.4.12 kernel operating system, forms the main functioning unit of the node [62]. The node also provides capabilities for digital signal processing with 4-channel 16-bit ADC. WINS NG 2.0 radios utilize a TDMA base-remote structure in which a node takes the role of either a slave or a base with respect to its neighbors. They also provide high-speed sampling, RF support, GPS support, and re-programmability. The development environment provides an API for configuring the radios to any required topology. Each node can

be assigned an IP address, thus enabling remote access to each individual node. Detailed hardware specifications are provided in Appendix C. Figure 9.1 shows the topology of the testbed that was used in conjunction with the SensIT[1] experiments.

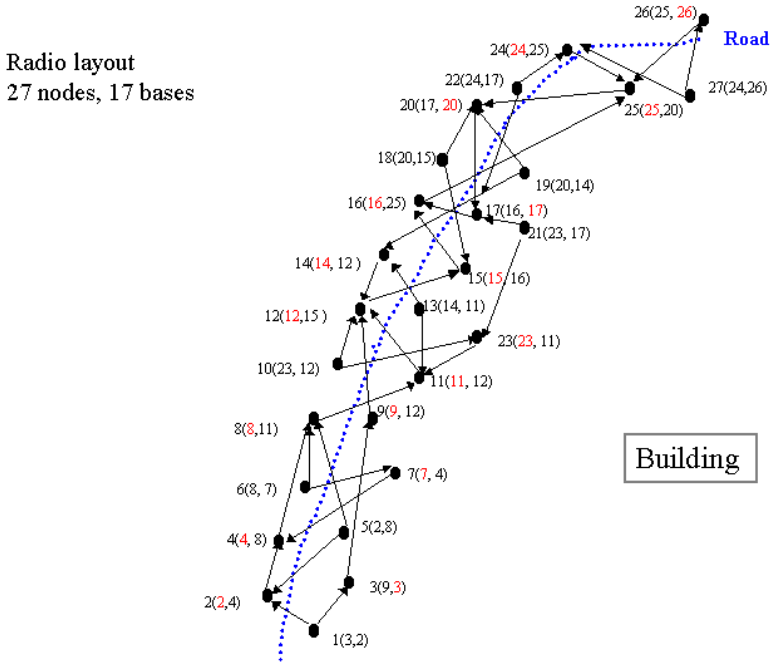

Figure 9.1: Radio layout of the testbed (figure taken from BBN Technologies)

Each base can support up to 16 remotes, however 7 or less is considered ideal. A boot-up script was used to configure the nodes automatically.

9.1.1 Coverage-estimation Experiment

In order to test the algorithms on the testbed, the *ns-2* based simulation model, which was almost entirely written at the application layer, was ported onto the sensor nodes. Changes made to the simulation model include capabilities to read certain parameters from the environment. These parameters include position coordinates, time-out values, and *node-id*. Time-out values were set based on repeated trails on the testbed.

[1]DARPA/ITO Sensor Information Technology Program

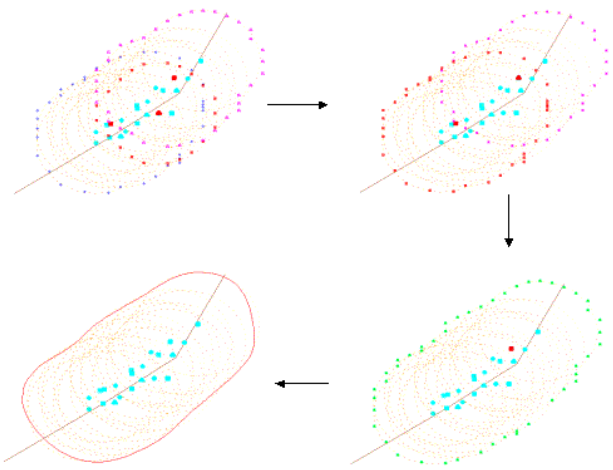

Figure 9.2: Various levels of the hierarchical clustering model (on the testbed)

The radio layout of the testbed is shown in Figure 9.1. A gateway computer was used to access the nodes from a remote location (each node had static IP address). An executable version of the application was copied onto each node using a simple Linux-based script. Coverage measurement was initiated by another script that triggered and killed a process after certain time interval. Figure 9.2 shows an example of the coverage of a group of nodes in the testbed (using hierarchical clustering model).

9.2 Conclusions

The above experiment showed the feasibility and validity of the coverage-estimation algorithm on an actual sensor testbed. Figure 9.2 provides a visualization of the coverage of a group of sensor nodes on which the algorithm was run. Though different values of *timeout* were required for the experiment, it illustrated how the model that was setup on a simulation environment was ported on to a real hardware with minimal changes.

Chapter 10

Conclusions

This chapter presents an outline of the previous chapters with regard to design, analysis, implementation, results, and verification activities involved in the distributed coverage-estimation algorithm. A brief discussion of the observations made in the study and possible future work is listed.

10.1 Summary

Although a large amount of work has been done in the field of tracking and detection in sensor networks, little has been done in the field of coverage measurement. This thesis illustrates the modeling of coverage using an energy-efficient scheme based on B-splines. It started with an introduction to various concepts of wireless sensor networks along with a definition of the problem of coverage measurement. Various reasons, such as validating results of detection, diagnosing the network, and improving the placement algorithm, were cited to show the importance of the research efforts. The determination of coverage with a reduction in energy consumption formed the basis of the thesis statement.

Both quantitative and visual representations of coverage data are key aspects of the estimation-algorithm and its design. A detailed review of the problem and an analysis of B-splines showed how the coverage of an individual sensor can be described using a minimal number of data points.

Using an approximation method, control points were generated to describe the coverage. These control points are much fewer in number as compared to the number of data points required to describe the sensing area. The cumulative sensing area or coverage of the complete sensor network can be obtained by aggregating the sensing area of individual nodes. This thesis proposes the use of communication models such as multi-level hierarchical clustering and directed diffusion for obtaining the aggregated coverage. Analysis of the above communication models showed that they performed better (in terms of energy efficiency) when compared to a centralized communication model. A metric of the form $hopcount \times databyte$ was introduced to analyze the algorithms with regard to energy efficiency. The use of B-spline based coverage data also played an important part in reducing the energy consumption.

The thesis also describes the implementation of the communication models on *ns-2* along with the various experiments that were carried out. Experiments on the computation algorithm indicated the advantages and constraints of B-splines. B-splines were shown to efficiently compress the amount of data that needed to be communicated between nodes. However, a trade-off exists between the number of control points used and the error in the estimation. The results from the coverage computation aspects can help in designing the model according to constraints. Simulation on communication models showed that they followed the theoretical trends. Directed diffusion showed consistent performance, while hierarchical clustering was shown to have limitations based on the number of levels and the routing scheme. The limitations of hierarchical clustering were predominant in high node density case due to a large election time and an inefficient suppression mechanism. Directed diffusion was shown to have better performance. Other aspects of directed diffusion, such as the effects of caching on latency and node failure recovery, were studied in detail. Validation and verification processes were also initiated to ensure credibility of the results. Towards the end of the thesis, an example of the experiment carried out on the actual sensor test-bed was illustrated to show the reliability of the algorithm in a real-case scenario.

This thesis presents one of the first instances where B-spline based computational geometry was used to measure coverage in a large-scale sensor network. The primary objective of the thesis was the design and implementation of an energy-efficient coverage measurement system. The simulations on computational aspects and communication models showed that the study's objectives were

achieved. The coverage model was shown to provide visualization and quantification of coverage with certain degree of error.

10.2 Future Work

A true evaluation of multi-level hierarchical clustering requires an efficient network level routing scheme. This thesis uses a broadcast scheme with rudimentary message suppression for multi-hop transmissions. Extending the current hierarchical clustering model with routing schemes used in ad-hoc networks could prove to be beneficial. This is a logical step after observations from the experiments conducted.

Also, this thesis uses an *opportunistic aggregation* [16] model of directed diffusion. The current model can be extended with a *greedy approach* [16] to improve energy efficiency.

The energy and communication models can be improved by obtaining field data from WINS NG 2.0. Future works can study other methods of data representation and compression, which can reduce the amount of data and adequately describe the sensor coverage.

Bibliography

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Mobile Computing and Networking*, pp. 56–67, 2000.

[2] S. Mehrotra, "Distributed algorithms for tasking large sensor networks," Master's thesis, Virginia Tech, July 2001.

[3] K. Fall and K. Varadhan, *The ns Manual*. UC Berkeley, LBL, USC/ISI, February 2001.

[4] P. Fuhr, "Industrial wireless technology for 21st century," Tech. Rep. U.S. Department of Energy, December 2002.

[5] Y. Zhao, R. Govindan, and D. Estrin, "Residual energy scans for monitoring wireless sensor networks," Tech. Rep. 01-745, May 2001.

[6] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *IEEE InfoCom*, vol. 3, pp. 1380–1387, 2001.

[7] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with sensor networks," *Proc. ICASSP'2001, Salt Lake City, UT*, pp. 2675–2678, May 2001.

[8] S. Shakkottai, R. Srikant, and N. Shroff, "Unreliable sensor grids: Coverage, connectivity and diameter," *Proceedings of IEEE Infocom, San Francisco, CA*, April 2003.

[9] J.-F. Chamberland and V. V. Veeravalli, "Decentralized detection in sensor networks," *IEEE Transaction on Signal Processing*, vol. 51, pp. 407–416, February 2003.

[10] E. Biagioni and G. Sasaki, "Wireless sensor placement for reliable and efficient data collection," in *Proceedings of the 36th Hawaii International Conference on Systems Sciences, HICSS 03, Big Island, Hawaii*, January 2003.

[11] E. Shih, S.-H. Cho, N. Ickes, A. Sinha, R. Min, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," *MOBICOM 2001, Rome, Italy*, pp. 272 – 287, 2001.

[12] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *IEEE Proceedings of the 33rd Hawaii International Conference on System Sciences*, pp. 3005–3014, January 2000.

[13] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *MobiCom*, pp. 263–270, August 1999.

[14] A. Iwata, Ching-Chiang, G. Pei, M. Gerla, and T. wei Chen, "Scalable routing strategies for multi-hop ad-hoc wireless network," *IEEE Journal of Selected Areas on Communications*, vol. 17, August 1999.

[15] E. Celebi, "Performance evaluation of wireless multi-hop ad-hoc network routing protocols," Master's thesis, Bogazici University, 2002.

[16] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of network density on data aggregation in wireless sensor networks," Tech. Rep. 01-750, University of Southern California Computer Science Department, November 2001.

[17] R. H. Bartel, J. C. Beatty, and B. A. Barsky., *An introduction to Splines for use in Computer graphics and Geometric modeling.* Morgan Kaufmann publishers, Inc., 1990.

[18] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next century challenges: Mobile networking for smart dust," in *International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 271–278, 1999.

[19] L. P. Clare, G. J. Pottie, and J. R. Agre, "Self-organsing distributed sensor networks," in *Proceedings of SPIE's 13th Annual International Symposium on Aerspace/Defense Sensing, Simulation, and Controls (AeroSense), Orlando*, April 1999.

[20] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J.Pottie, "Protocols for self-organisation of a wireless sensor network," *IEEE Personal Communications*, vol. 7, pp. 16–27, October 2000.

[21] D. Baker and A.Ephremides, "The architectural organization of a mobile radio network via a distributed algorithm," *IEEE Transactions on Communications*, vol. COM-29, pp. 1694–1701, November 1981.

[22] L. Subramanian and R. H.Katz, "An architecture for building self-configurable systems," in *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC 2000)*, pp. 63–73, Department of Electrical Engineering and Computer Sciences, U.C. Berkeley, August 2000.

[23] A. Cerpa and D. Estrin, "Ascent: Adaptive self-configuring sensor networks topologies," in *In Proceedings of the IEEE Infocom*, vol. 3, pp. 1278–1287, IEEE Computer Society, January 2002.

[24] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *the Proceedings of the Hawaii International Conference on System Sciences, January 4-7*, 2000.

[25] S. Lindsey and C. S.Raghavendra, "Pegasis: Power efficient gathering in sensor information systems," in *IEEE Aerospace Conference*, March 2002.

[26] K. Du, J. Wu, and D. Zhou, "Chain-based protocols for data broadcasting and gathering in sensor networks," in *International Parallel and Distributed Processing Symposium*, April 2003.

[27] G. S.Fishman, *Discrete-Event Simulation: Modeling, Programming and Analysis*. Springer-Verlag, Berlin, 2001.

[28] S. Banerjee and S. Khuller, "A clustering scheme for hierarchical control in multi-hop wireless networks," in *INFOCOM*, pp. 1028–1037, 2001.

[29] M. J. Handy, M. Haase, and D. Timmermann, "Low energy adaptive clustering hierarchy with deterministic cluster-head selection," in *IEEE International Conference on Mobile and Wireless Communications Networks, Stockholm*, 2002.

[30] J. Sucec and I. Marsic, "Clustering Overhead for Hierarchical Routing in Mobile Ad hoc Networks," in *IEEE INFOCOM 2002*, (New York, NY), June 23–27 2002.

[31] C. F. Olson, "Parallel algorithms for hierarchical clustering," *Parallel Computing*, vol. 21, no. 8, pp. 1313–1325, 1995.

[32] L. K. Farouk, *Hierarchical Routing for Large Networks.* North-Holland Publishing Company, 1977.

[33] D. B. West, *Introduction to Graph Theory.* Prentice Hall, 1996, 2001.

[34] J. Jubin and J. D. Tornow, "The darpa packet radio network protocols," in *Proceedings of the IEEE*, vol. 75, pp. 21–32, January 1987.

[35] G. Pei, M. Gerla, X. Hong, and C. Chiang, "Wireless hierarchical routing protocol with group mobility (WHIRL)," Tech. Rep. 990006, 25, 1999.

[36] G. Pei, M. Gerla, and T. Chen, "Fisheye state routing in mobile ad hoc networks," in *(ICDCS) Workshop on Wireless Networks and Mobile Computing*, pp. D71–D78, 2000.

[37] B. Krishnamachari, D. Estrin, and S. Wicker, "Modelling data-centric routing in wireless sensor networks," in *IEEE INFOCOM*, 2002.

[38] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, pp. 114–131, June 2003.

[39] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," in *Proceedings of the Symposium on Operating Systems Principles*, (Chateau Lake Louise, Banff, Alberta, Canada), pp. 146–159, ACM, October 2001.

[40] D. S. Rosenblum and A. L. Wolf, "A design framework for internet-scale event observation and notification," in *Proceedings of the Sixth European Software Engineering Conference (ESEC/FSE 97)* (M. Jazayeri and H. Schauer, eds.), pp. 344–360, Springer–Verlag, 1997.

[41] A. Carzaniga, D. S. Rosenblum, and A. Wolf, "Design and evaluation of a wide area event notification service," *ACM Transactions on Computer Systems*, vol. 19, pp. 332–383, August 2001.

[42] M. Castaldi, A. Carzaniga, P. Inverardi, and A. L. Wolf, "A lightweight infrastructure for reconfiguring applications," in *SCM 2001/2003* (B. Westfechtel and A. van der Hoek, eds.), no. 2649 in LNCS, (Portland, Oregon), pp. 231–244, Springer-Verlag, May 2003.

[43] D. Heimbigner, "Adapting publish/subscribe middleware to achieve gnutella-like functionality," in *Selected Areas in Cryptography*, pp. 176–181, 2001.

[44] J. Saghri and H. Freeman, "Analysis of the precision of generalized chain codes for the representation of planar curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, pp. 533–539, September 1981.

[45] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Transactions on Computers C-21*, pp. 269–281, 1972.

[46] I. Schoenberg, *Approximations with Special Emphasis on Spline Functions.* Academic Press, New York, 1969.

[47] R. Kershner, "The number of circles covering a set," *American Journal of Mathematics*, vol. 61, pp. 665–671, 1939.

[48] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Geometric ad-hoc routing for unit disk grpahs and general cost models," Tech. Rep. Technical report TR373, ETH Zurich.

[49] J. Kostkova and R. Halir, "A spline approximation of a large set of points," *in Proc. of the 8th International Conference in Central Europe on Comoputer Graphics, Visualization and Interactive Digital Media (WSCG2000), Czech Republic*, February 2000.

[50] W. H.Press, S. A.Teukolsky, W. T.Vetterling, and B. P.Flannery, *Numerical Recipes in C++: The Art of Scientific Computing.* Cambridge University Press, 1998.

[51] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis.* McGraw-Hill, New York, 2ed ., 1978.

[52] T. Rappaport, *Wireless Communications: Principles and Practice, 2nd Edition*. Prentice Hall, Dec 31, 2001.

[53] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, "Scalable coordination for wireless sensor networks:self-configuring localization systems," in *International Symposium on Communication Theory and Applications (ISCTA)*, July 2001.

[54] 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.

[55] M. Gerla, "Clustering and routing in large ad hoc wireless nets," *Final Report 1998-99 for MICRO project*, 1998.

[56] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking.," *(ACM/IEEE) Transactions on Networking*, vol. 11, pp. 2–16, February 2000.

[57] F. Silva, J. Heidemann, and R. Govindan, *Network Routing Application Programmer's Interface (API) and Walk Through 9.0.1*, December 2002.

[58] J. H. Park, G. Friedman, and M. Jones, "Geographical feature sensitive sensor placement," *Journal of Parallel and Distributed Computing*, November 2003.

[59] S.Schleisinger, "Terminology for model credibility," *Simulation*, vol. 32, no. 3, pp. 103–104, 1979.

[60] S. Park, A. Savvides, and M. B. Srivastava, "Sensorsim: A simulation framework for sensor networks," in *Proceedings of the 3rd ACM International workshop on Modeling, analysis and simulation of wireless and mobile systems*, (Boston, Massachusetts), pp. 104–111, August 2000.

[61] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley-Interscience, New York, NY, April 1991.

[62] Sensoria, *WINS 2.0 NG User Guide*. Sensoria.

Appendix A

Examples of Configurations C, D, and E

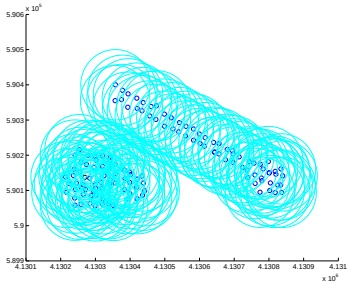

Figure A.1: Configuration C.128

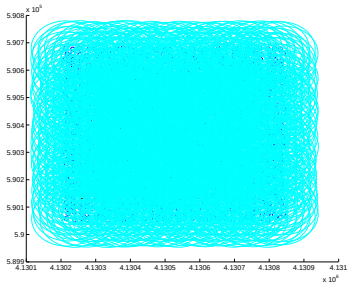

Figure A.2: Configuration C.1024

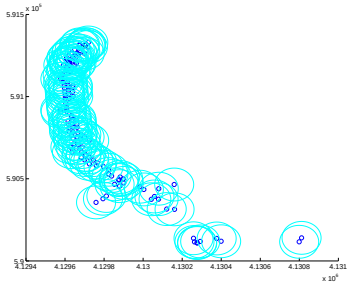

Figure A.3: Configuration D.128

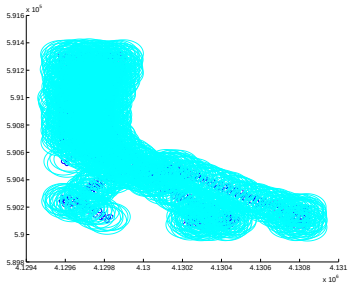

Figure A.4: Configuration D.1024

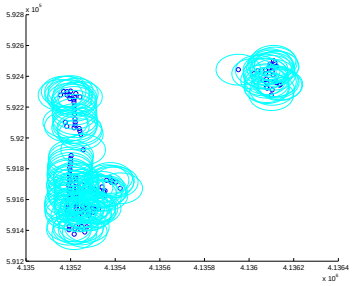

Figure A.5: Configuration E.128

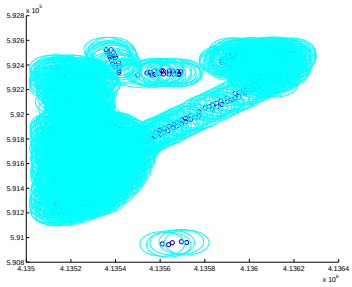

Figure A.6: Configuration E.1024

Appendix B

Coverage Estimation: Illustrations

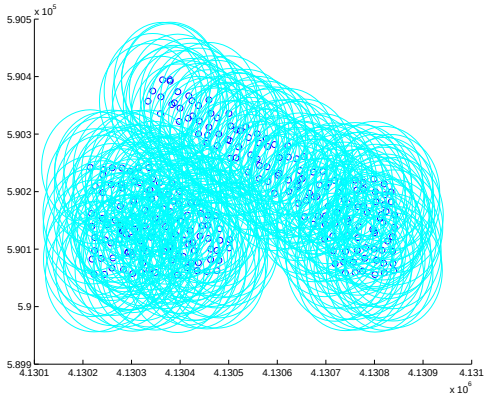

Figure B.1: Configuration C.256

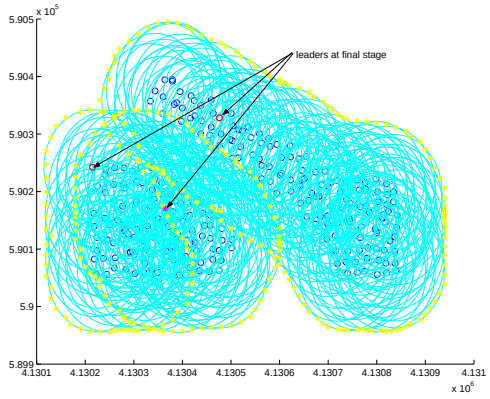

Figure B.2: Final stage of hierarchical clustering

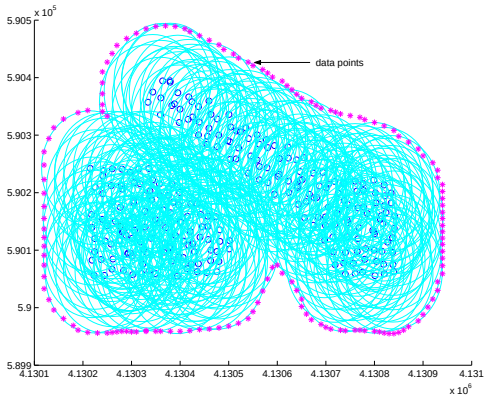

Figure B.3: Data points of the complete coverage, hierarhical clustering (C.256)

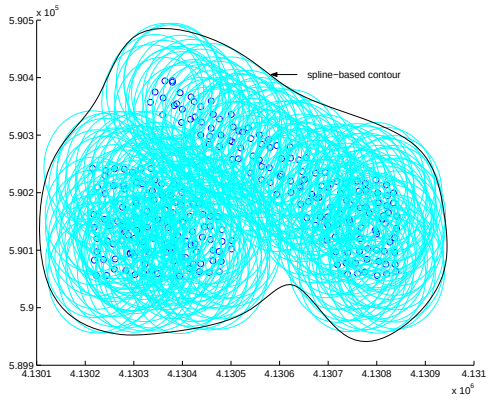

Figure B.4: Spline-based coverage representation, hierarhical clustering (C.256)

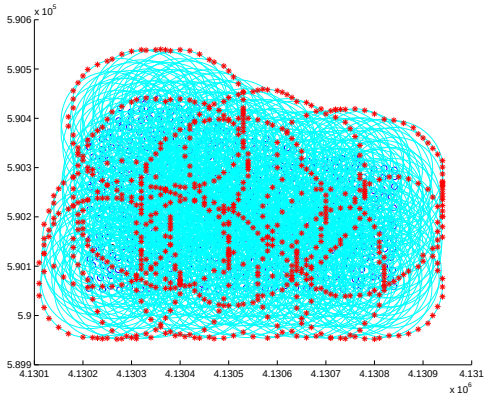

Figure B.5: Data points at level 1, hierarchical clustering (D.512)

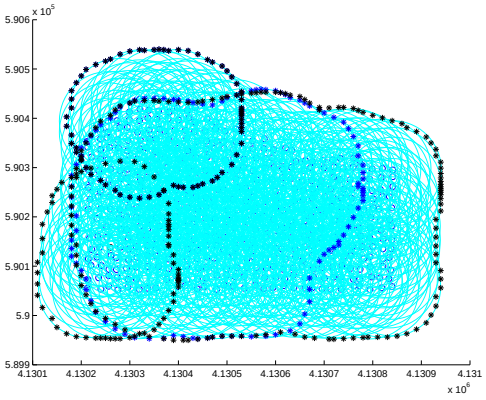

Figure B.6: Data points at level 3, hierarchical clustering (D.512)

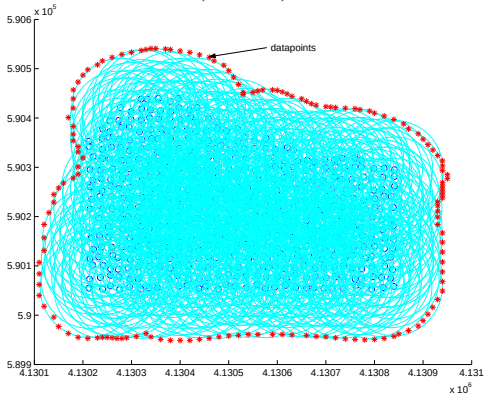

Figure B.7: Data points representing the complete coverage, hierarchical clustering (D.512)

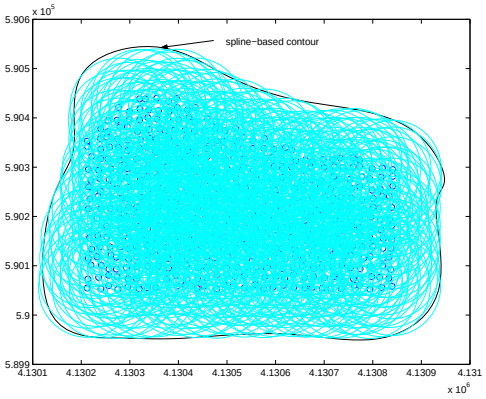

Figure B.8: Spline-based coverage of Configurations D.512

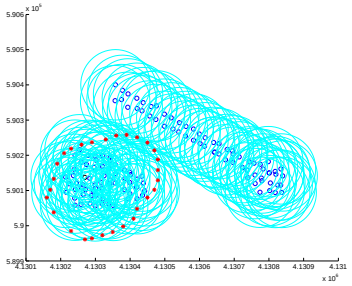

Figure B.9: Stages in directed diffusion (a) (C.128)

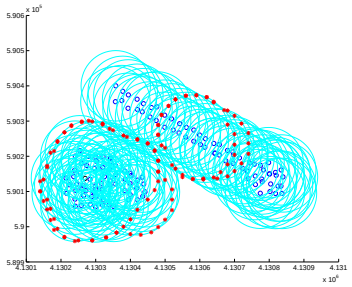

Figure B.10: Stages in directed diffusion (b) (C.128)

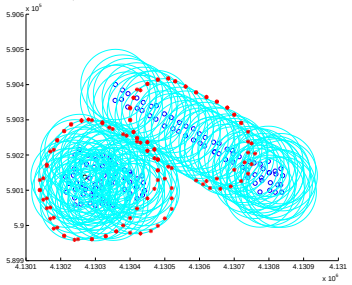

Figure B.11: Stages in directed diffusion (c) (C.128)

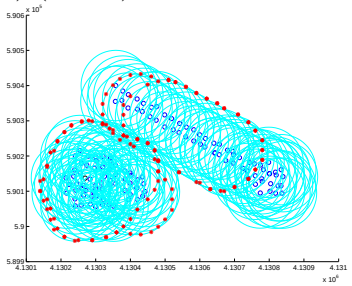

Figure B.12: Stages in directed diffusion (d) (C.128)

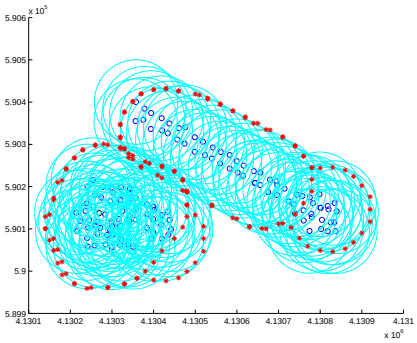

Figure B.13: Stages in directed diffusion (e) (C.128)

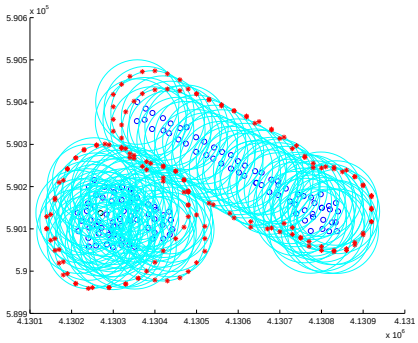

Figure B.14: Stages in directed diffusion (f) (C.128)

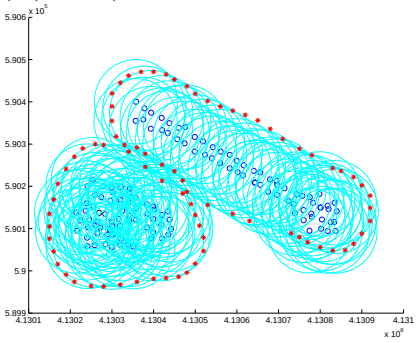

Figure B.15: Stages in directed diffusion (g) (C.128)

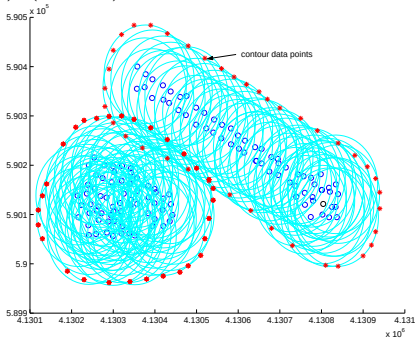

Figure B.16: Final stage with data points (C.128)

Appendix C

WINS NG 2.0 Nodes

Platform Processor Core	Hitachi SH-4, 32-bit RISC
Platform Processor Performance	300 MIPS CPU, 1.1 GFLOP FPU
Platform Memory	16 to 64MB RAM, 16 to 32MB Flash
Analog Input Channels	4
Sampling Frequency (samples/sec)	20 KHz per channel
A/D Resolution	16-bit
RF Modem	Integrated dual channel, 2.4GHz modem
FCC Certification	FCC Part 15.247 and ETS300-328 rules, license free
GPS Receiver	L1 frequency, C/A code
GPS Protocol	NMEA v2.2
GPS Antenna Cable Length	6
Digital I/O	15 configurable GPIO lines
Expansion Ports	Two PCMCIA Type II slots
Sensor Connectors	Polarized, 6-pin, differential input
Wired Interfaces	10 Mbps Ethernet, RS-232 Serial

Table C.1: Hardware Specifications of the WINS NG 2.0 development system (from Sensoria)

Vita

Ravi Anilkumar was born on October 22, 1976 in Kerala, India. He schooled at Christ Nagar High School and Indian School (Bahrain), graduating in 1994. In 1998 he completed his Bachelor's degree in Electronics Engineering at Cochin University, India. Upon graduation he worked as a Software Engineer for Robert Bosch, India (Bangalore, India) for about two years.

In August 2000 he joined the Bradley department of Electrical and Computer Engineering at Virginia Tech to begin work on a Master's degree in Electrical Engineering. Currently he works as a Network Engineer at Computer Networks & Software, Inc. (Springfield, VA).