MOVEMENT STRATEGIES FOR INTRUSION DETECTION IN INCOMPLETE SENSOR NETWORKS

THESIS

Presented to the Graduate Council
of Texas State University-San Marcos
In Partial Fulfillment
of the Requirements

for the Degree

Master of SCIENCE

by

Eric Daniel Reeves, B.A.

San Marcos, Texas
May 2006

ACKNOWLEDGEMENTS

I would like to extend the greatest gratitude possible to my mother and father. They have given me an immeasurable amount of inspiration and motivation. It is their sacrifices that have given me the resources and courage to see this through.

I would also like to give thanks to the members of my thesis committee for their guidance and motivation. Dr. Peng's willingness to take me under his wing and motivate me over the last ten months has been nothing less than invaluable. His knowledge of the subject matter presented herein was critical to the inception and completion of this project. He, Dr. Drissi and Dr. Haddix provided the level of detail necessary to help me refine and transform a collection of ideas into a thesis.

This manuscript was submitted on April 24, 2006.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

MOVEMENT STRATEGIES FOR INTRUSION DETECTION IN INCOMPLETE SENSOR NETWORKS

by

Eric Daniel Reeves, B.A.

Texas State University-San Marcos

May 2006

SUPERVISING PROFESSOR: WUXU PENG

Incomplete sensor networks are an area of wireless communications with broad application. The intention of this document is twofold. One goal of this research is to establish a relationship between the method by which a sensor network is deployed, the movement patterns employed by sensors in the network, and the rate at which the network detects infiltrations inside the area in which it is deployed. The rate of detection and the movement patterns employed by mobile sensors are examined through mathematical analysis and software simulation. Furthermore this document also seeks to relate sensor quantity, communication range, and the rate at which sensors are converged when data is broadcast throughout an incomplete sensor network. The second goal of this research is to determine if various combinations of sensor quantity and communication range are positively correlated with the amount of time required to converge a sensor network.

CHAPTER 1

INTRODUCTION

Wireless sensor networks are comprised of a number of tiny sensors possessing a limited energy supply [8]. In addition to the battery that serves as their power source, these tiny devices possess a circuit board, an antenna and the necessary electronics to connect those components. Recently the increasing quantity of cost-efficient wireless devices has resulted in the growth of wireless sensor networks. Sensor networks have broad implications in security, environment monitoring, manufacturing machinery, performance monitoring, structural safety and a multitude of military applications [22]. The challenges of creating and maintaining sensor networks are numerous. These challenges include but are not limited to scalability, bandwidth consumption, and energy conservation [25]. The devices in a sensor network are equipped with a limited amount of battery power, creating an imperative requirement of either an inexpensive replacement model or the efficient execution of a sensor network's directive.

Research in sensor networks has been frequent and increasingly extensive in the last several years [8, 19, 10, 15, 13], focused on a multitude of areas ranging from physical and media access layers to sophisticated routing and transport protocols [11]. Additionally, there has been increasing insight into area coverage, energy expenditure, and tracking. As an example, consider the presence of a sensor network in a habitat where the effects of human presence upon the area's ecosystem are monitored. Deployment of sensor networks in these areas present opportunities for long-term studies into human intrusion without the economic and administrative constraints associated with maintaining human personnel. The existence of sensors in these areas in lieu of human presence also allow for extensive monitoring while preserving the environment in which they are deployed [16].

The goal of this research is to provide insight into how the deployment of a sensor

network affects its resiliency to infiltrations. These networks are somewhat unique in that they are incomplete sensor networks. An incomplete sensor network is one for which the sensor density is insufficient to completely cover the space of interest. The (in)completeness of a sensor network is primarily dictated by the relationship between the network's purpose and the geometric size of the area of interests. While sensors are commonly regarded as inexpensive devices the assumption is based on a limited set of features. Namely, static sensors do not have the mechanisms to be mobile however as these mechanisms are included the cost will inevitably increase. Those who seek to populate a sensor network with mobile nodes may not be able to afford as many mobile nodes as static nodes; thus, the need for few mobile sensors to cover the area of many static nodes is a problem with many complexities.

As an example, consider the purpose of a network of four sensors inside a very large rectangular area. If the application of the sensor network only requires that each of the area's four corners are monitored then one can consider the network to be complete so long as all four corners are monitored. However, if the purpose of the network is to detect infiltrations inside the area then four sensors could not possibly provide the coverage necessary to perform this task. The latter case is one that this document seeks to analyze in detail.

Much of the research in sensor networks has directed its focus to completely covered networks. In comparison, the amount of research in incomplete networks has been relatively small. Several strategies for governing the deployment of sensors are examined. The movements of mobile sensors are also examined. We compare the effectiveness of static sensors to mobile sensors. Additionally we examine different infiltration strategies employed by intruders; the parameters involved are movement pattern, point of entry into the area of interest, and point of exit from the area of interest.

Broadcasting In Incomplete Sensor Networks

There exist a multitude of broadcasting protocols designed to accomodate the challenges presented in mobile ad hoc networks [25, 2]. Broadcasting is a critical element of incomplete sensor networks due to the difficulty of placing mobile sensors within the communication range of at least one neighbor. Habitat monitoring poses the objective of

reliably distributing information about detections to all defenders. As an example, consider a scenario where the coordinates of every detection are broadcast from the defender to all other sensors. As neighboring sensors receive this information they may be able to use mechanisms to record and predict the point of entry for subsequent intruders. This information can also be used to predict an intruder's next coordinates and dispense defenders accordingly.

This study propopses that the communication range and positioning of defending sensors dictate to some degree the percentage of defending sensors that receives a message. The effects of deployment and mobility upon the sensors' ability to disseminate information in an incomplete sensor network are examined. Specifically, the receive threshold, transmit threshold, and simulation time are increased to find which combinations of these parameters are most successful in distributing messages across the entire network when comprised of randomly moving sensors. We also propose that the communication range and positioning are directly linked to the amount of time it takes a sensor network to arrive at convergence once an infiltration is detected. Hence, the relationship between sensor quantities, communication range, and convergence time for deployments that yield a 100% broadcast rate is explored in detail.

Coupled with an effective deployment and patrol strategy, effective broadcasting provides an invaluable supplement to the three most critical requirements of monitoring via sensor networks: deployment, patrol strategy, and communication.

CHAPTER 2

RELATED LITERATURE

Mainwaring, Polastre et al. provided an in-depth study of employing wireless sensor networks to monitor habitats [16]. [19] introduces ideas and gathers information on simulating wireless sensor networks. The work examines architecture models for both sensor nodes and sensor networks. Savvides et al. introduce several models of sensor networks, including battery models, node models and power characterization [26].

Previous work in [3] introduced the notion of the unauthorized traversal problem as well as an approach to solve it. The work proposed an algorithm for sensor deployment, one which deploys sensors with the purpose of tracking a moving target. These sensors were deployed randomly and retained a fixed location. [24] introduces the exposure-based coverage model which the authors define as exposure. Exposure is used to evaluate the coverage of a region by a set of wireless sensors performing target detection. [4] builds upon both the notion of exposure and unauthorized traversal problem. This work also randomly deploys sensors to detect targets moving through a region. The minimum exposure measures the effectiveness of the deployment with the goal to maximize the exposure of the least exposed path in the region. [33] introduces a strategy to maximize coverage in sensor networks by way of adaptive sensing. However, this approach guarantees a 100% sensing coverage made possible by several assumptions. One such assumption is that all nodes are able to directly communicate within the distance of two times the sensor's radius. Gao et. al [8] also approaches two of the foremost challenges in sensor networks: maintaining sensor lifetime and providing sufficient coverage area. Their approach is similar to [3] in that sensors are deployed randomly. Additionally the sensing range of each sensor is a circular area centered at that sensor. In their simulations all sensors have the same sensing range and no two sensors are deployed at the same location in the two-dimensional area.

The work in [27] proposes solutions in achieving a balance between energy expenditure and the quality at which mobile targets are tracked within a network. The study proposes coverage, signal attenuation and sensor density are the main parameters by which the balance is measured. [34] offers the idea that a hybrid sensor network offers benefits of both mobile networks and networks populated with only static sensors. A hybrid sensor network is one in which both static and mobile sensors are deployed.

[29] explains the maximal exposure problem: Given a sensor network the maximal exposure path between a location of origin and a destination location is the path in the sensor network connecting the two coordinates such that the exposure yielded by traversing the path is maximal. The maximal exposure problem is reduced to the NP-complete longest path problem and offers several heuristics for solving it. In [1] mobile objects are tracked inside binary sensor networks. Their model employs sensors whose value is interpreted as one binary bit. This bit is used as information in determining if an object is moving toward or away from the sensor. Huang [11] proposes approaches to the k-problem - determining if every point in a monitored area is covered by at least k sensors where k is some predefined value. Incomplete sensor networks face the problem that points in a service area may not be covered by any sensors. Furthermore with a random distribution of sensor deployment coordinates it's possible for there to exist points with undesirably uneven concentrations of sensors. This research assumes that sensors' locations are fixed and thus in an incomplete sensor network there exists no opportunity to improve the sensors' geographic composition.

In [25] and [2] the many problems and challenges of topology control in sensor networks are discussed. The authors survey a collection of modern solutions for topology control, including location, direction and / or neighbor based approaches. [18] proposes a method for topology control and lifetime in wireless sensor networks, based on a two-tier system with sensor nodes, designated base stations and application nodes.

CHAPTER 3

DETECTABILITY

We define the term detectability as the probability at which an intrusion is detected. What follows is an overview of the discoveries in detectability for several scenarios. Four different scenarios are presented, as well as empirical data to support our preliminary findings.

Let R be a rectangular two-dimensional area with measurements $a \times b$. Assume that k sensors are deployed in R with identical hardware and software architectures. Also assume that each sensor can transmit and receive information such that it is at the center and the radius is λ [33]. With these assumption in mind a sensor can monitor an area of $\pi\lambda^2$. Let $\bar{R}$ indicate the combined area that these k sensors can possibly cover. Thus $\bar{R}$ is a $(a+2\lambda) \times (b+2\lambda)$ rectangle with rounded corners. Using algebra the area of $\bar{R}$ is derived to $|\bar{R}| = ab+\lambda^2+2\lambda(a+b)$. The area that k sensors can possibly cover is not necessarily equal to the actual area at the time of deployment. The maximum area covered by k sensors is equal to $k\pi\lambda^2$ when the sensing radius of any two sensors do not overlap.

One class of intruder in this experiment is one that moves in a straight line, parallel to the vertical edge of R. Let x be the distance between the intruder and the vertical edge on the left side of R. Let E represent the event at which an intruder is detected by at least one of the sensors [21]. Let F_1, F_2, and F_3 denote the events $0 \leq x \leq \lambda$, $\lambda \leq x \leq (a-\lambda)$ and $(a-\lambda) \leq x \leq a$. Therefore: $Pr[E|F_2] = 1-(1-\frac{2\lambda}{a})^k$ [21].

Going forward we calculate $Pr[E|F_1]$:

$$Pr[E|F_1] = \tfrac{1}{\lambda}\int_0^{\lambda}(1-(1-\tfrac{x+\lambda}{a})^k)dx$$

$$\Rightarrow Pr[E|F_1] = 1-\tfrac{a}{\lambda}\int_{\lambda}^{0}(1-\tfrac{x+\lambda}{a})^k\, d(1-\tfrac{x+\lambda}{a})$$

$$\Rightarrow Pr[E|F_1] = 1-\tfrac{a}{(k+1)\lambda}\,((1-\tfrac{\lambda}{a}^{k+1}-(1-\tfrac{2\lambda}{a})^{k+1})$$

Additionally we calculate $Pr[E|F_3]$ [21]:

$$Pr[E|F_3] = 1 - \frac{a}{(k+1)\lambda}\left((1-\frac{\lambda}{a})^{k+1} - (1-\frac{2\lambda}{a})^{k+1}\right)$$

Thus:

$$Pr[F_1] = \frac{\lambda}{a}$$
$$Pr[F_2] = \frac{a-2\lambda}{a}$$
$$Pr[F_3] = \frac{\lambda}{a}$$

Finally we calculate the probability that the intruder is detected by at least one sensor:

$$Pr[E] = \frac{\lambda}{a} \times Pr[E|F_1] + \frac{a-2\lambda}{a} \times Pr[E|F_2] + \frac{\lambda}{a} \times Pr[E|F_3]$$
$$\Rightarrow Pr[E] = 1 - \frac{2}{k+1}\left(1-\frac{\lambda}{a}\right)^{k+1} - \frac{k-1}{k+1}\left(1-\frac{2\lambda}{a}\right)^{k+1}$$

The simulated results are somewhat lower than the theoretical results, however the theoretical results do not take into consideration such factors as a predictable seed (e.g. one based on time), the speed of the intruder, or an extremely uneven composition of sensors. The simulated results as well as the experiment procedure is later discussed in more detail.

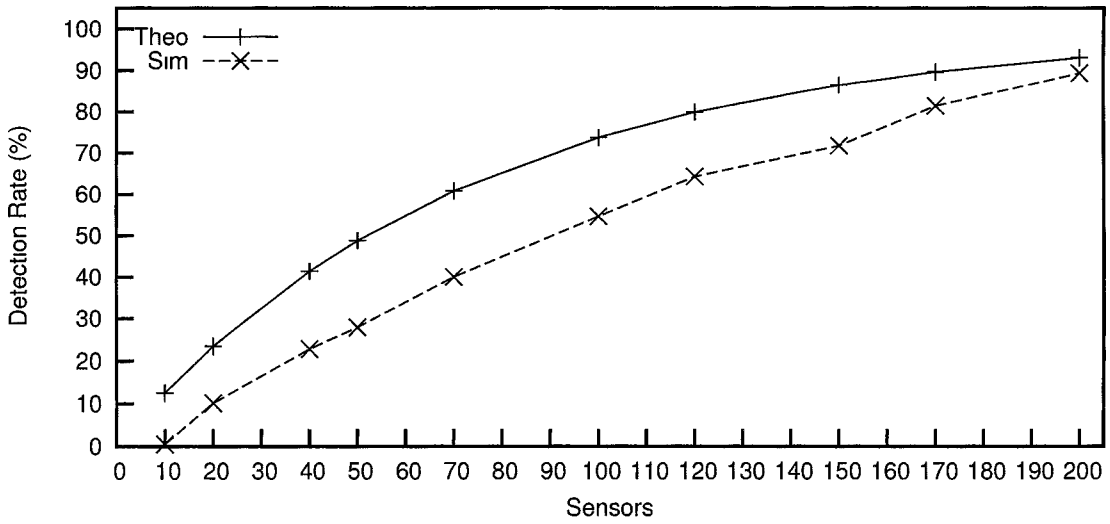

Figure 3.1: Theoretical vs. Simulated Results: Static Sensors

Random / Uniform Deployment of Static Sensors

The goal of this study is to correlate the rate of detection to whether nodes in a sensor network are randomly or uniformly deployed. The effectiveness of randomly deployed static sensors is jeopardized as intruders acquire knowledge about the defenders' locations. Thus, random deployments must be derived from a reliable and sufficiently complex seed. Likewise a pseudo-random function can position sensors advantageously while manipulating the intruder's perception of the density and positioning in the monitored environment. An ideal scenario in areas covered by pseudo-randomly deployed static sensors is one in which all nodes appear to have random placement, maximizing both coverage and the capacity at which defending sensors communicate.

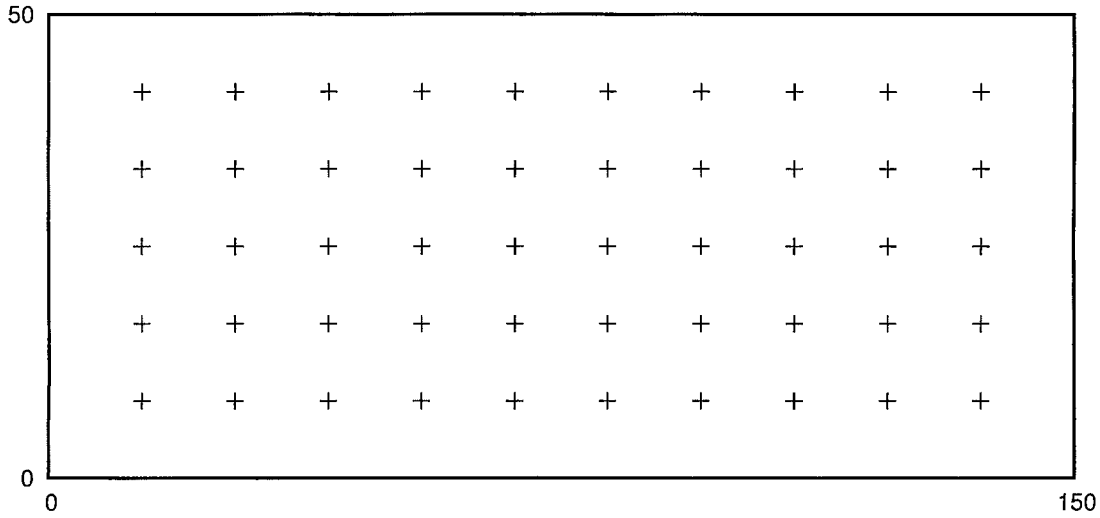

Figure 3.2: Uniform deployment of fifty protecting sensors

Sensors are also deployed via uniform distribution. Based on the number of sensors protecting the area, rows of sensors are deployed along the X axis of the two-dimensional plane in which our experiments are performed.

Random / Uniform Deployment of Mobile Sensors

While static sensors are advantageous for certain scenarios, mobile sensors provide to a scenario more features that may improve the rate of detection. Furthermore, given the requirements for an area the movement patterns which optimize effectiveness in that area

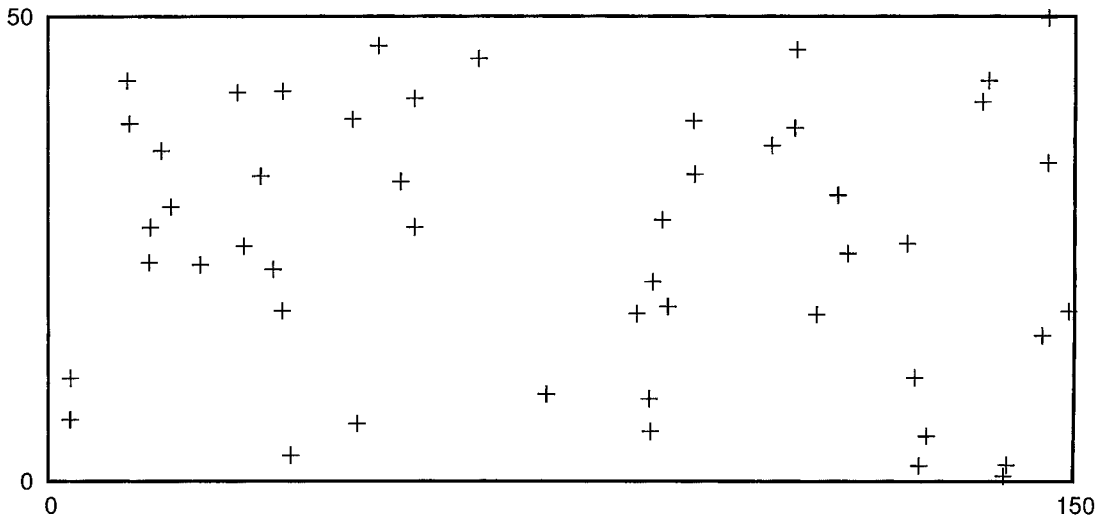

Figure 3.3: Random deployment of fifty protecting sensors

may be applied. Areas covered by sensors possessing movement patterns not befitting to the area can become expensive in terms of energy consumption, communication capacity, and intrusion detection. Thus, it is imperative that the strategies driving the movement patterns contain forethought and planning.

Just as static sensors can be randomly deployed, mobile sensors can be randomly deployed. They can also possess random movement. These capabilities prevent intruders from learning sensor patterns; pending the results of this research there may exist a method that can be applied to any area with some degree of success. Like statically deployed sensors, moving sensors must exhibit true randomness or pseudo-randomness. A multitude of patterns can be created from these pseudo-random properties; a small subset of these patterns will be examined in detail. The success of movement patterns are in part contingent upon the speed in which those patterns are traversed. While previous research has offered that higher movement speeds increase the probability of detection, the energy consumed by sensors was known to increase. The scope of this research does not include in-depth analysis of energy consumption, however the implications of sensor movement and speeds with respect to energy are observed. Performance is measured through expiriments by way of software simulation [19].

Random / Uniform Deployment of Intruders

In order to create a reliable comparison between each method of sensor deployment the intruder employs one of two types of behavior. One method deploys the sensor at Y = 0 in the two-dimensional area with a random X coordinate. The sensor moves in a straight line toward the maximum Y coordinate. The other method deploys the intruder according to two randomly generated variables: the boundary and coordinate from which the intruder enters the area. The intruder moves in a straight line toward its designated exit point. The exit point is also determined by randomly generating the boundary and the coordinate on the boundary. The intruder cannot have an entry and exit coordinate along the same boundary. The intruder is given two methods of mobility to further compare the effectiveness of protecting sensors' deployment and movement patterns.

CHAPTER 4

EXPERIMENT PROCEDURE

The sensors deployed in this simulation have the hardware and software architectures necessary to communicate via the IP transport protocol by way of wireless communication. The sensors which populate this area are of heterogeneous sensing capacity, energy consumption models, and means of communication [28]. Radio propagation is achieved using the two-ray ground reflection model. The two-ray ground model considers propagation by both line-of-sight and a ground reflection path [7]. At a distance of d, the received power is predicted by

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

where $L = 1$, h_t is the height of the transmitting antenna, h_r is the height of the receiving antenna, and G_t and G_r are the gains of the transmitting and receiving antenna, respectively. The simulation area is two dimensional. The length of the simulation area along the X axis is 150 meters. The length of the simulation area along the Y axis is 50 meters [21]. No geological or artificial obstacles exist such that the paths of protecting sensors are inhibited. It is assumed that any bodies that exist can otherwise be traveled through or over. All sensors have the exact same receive and transmit threshold. These thresholds are by majority 1 meter however some simulations will require an increase in these parameters. These circumstances are clearly indicated. Sensors are given a small range of communication so the effects of the mobility model are easier to discern. Both defending sensors and intruders move at a velocity of 1 m/s unless otherwise noted.

All protecting sensors are able to distinguish another protector from an intruder however those mechanisms are not discussed or evaluated in detail. The length of the simulation is determined by its scope; since the objective is to determine the rate at which intrusions are detected, the simulation ends when either the intruder has been detected or

when it has successfully reached the opposite boundary. Thus, the maximum time at which the simulation runs is equal to the distance divided by the velocity of the intruder. Since the velocity of the intruder is 1m/s the simulation runs for as many seconds as the length of the intruder's traversal. In some scenarios the distance traveled by an intruder is of variable length and thus the simulation time is based on the maximum distance a sensor could travel along its traversal path. Additionally, since these experiments only seek to discover the rate of detection they do not take into consideration what happens to an intruder after it has been discovered.

Each simulation places an intruder against k defending sensors, where $k = 10$, 20, 40, 50, 70, 100, 120, 150, 170, 200 sensors [21]. Each simulation is run by average one-thousand times [29] and at the end of the one-thousandth simulation the number of detections is divided by one-thousand to determine the rate of detection.

CHAPTER 5

VERTICAL INTRUDER, UNIFORM SENSOR DEPLOYMENT

Four different scenarios are examined in which a vertically-moving intruder faces various detector deployments. In each scenario the detecting sensors' positioning strategy is the same however the movement patterns differ. Detectors are deployed at ten sensors per row where the number of rows is determined by the total number of sensors in the environment. Furthermore, detectors are initially deployed horizontally from the middle of the area [24]. The formula which dictates these coordinates is as follows. Let maxY be the maximum Y coordinate of the two-dimensional area, and let maxX be the maximum X coordinate. $n$umberOfNodes indicates the number of nodes protecting the area. The coverage expands at ten per row from the middle of the area toward the boundaries Y = 0, Y = maxY as more sensors are deployed.

$$yDiv = maxY/((numberOfNodes/10) + 1)$$
$$xDiv = maxX/(10 + 1)$$

A single intruder begins at coordinates (rand,0), where rand is a random coodinate within the boundaries of the X axis. The intruder stays on this X coordinate while moving toward the maximum Y coordinate. Once both the intruder and the detecting sensors are deployed the simulation begins. Once the simulation begins all sensors simultaneously begin moving as dictated by their assigned behavior. The behaviors assigned to the sensors are as follows.

Static Sensors

Sensors do not move from the point at which they were deployed. The coordinates of k sensors will never change; their positions are constant throughout all one-thousand

simulations.

Randomly Moving Sensors (Rnd Move)

Sensors move in a random direction at a statically assigned interval. At every ten time units of the simulation the sensor moves in a new randomly chosen direction. The interval between assignments is intentionally short to inhibit the sensor from exhibiting predictable behavior.

Random Straight Line Sensors (Rnd Line)

Sensors move toward one of the area boundaries. The boundary and the coordinate on the boundary is chosen at random. Once a sensor has reached the boundary of the area a new and different boundary is randomly chosen and the sensor begins its movement along that path. Sensors under this mobility model travel boundary to boundary until the simulation is complete. We also refer to this as the *R*andom Boundary model.

Area Constrained Sensors (Area)

Sensors move at random however they do not move outside the area surrounding their point of deployment. The size of area is proportional to the number of detecting sensors in the area such that the entire environment is subject to the presence of a detector. The goal of this simulation is to create a medium between the Static and random movers scenarios above. Area constrained sensors are only deployed in simulations with uniform sensor distribution.

Sine Wave Movement Sensors

This simulation employs another scenario: defending sensors which move in a sine wave pattern. This pattern is performed along the X axis of the simulation area; in other words, defending sensors are moving from their deployment coordinates to $x = 0$ or the maximum x coordinate. A horizontal movement is chosen with the intention of intercepting an intruder's path. Given that our simulated habitat is rectangular we desire a sensor have

more space to cover before encountering an area boundary and patrol the same region more than once. The sine wave movements in this scenario are controlled by many parameters.

The two most common attributes of sine waves are their frequency (α) and amplitude (λ). Frequency determines how many times a cycle appears within a unit of time while amplitude is a nonnegative measure of the maximum disturbance during a wave cycle [31]. Both the amplitude and frequency should have a large influence on the results of the simulation due to the control each has on the behavior of the movements: the maximum y-distance from which defending sensors will travel and how frequently defending sensors will approach the maximum y-distance.

$$y = \alpha + sin(2 * \pi * \lambda)$$

Figure 5.1: Formula for plotting sensor sine wave movements

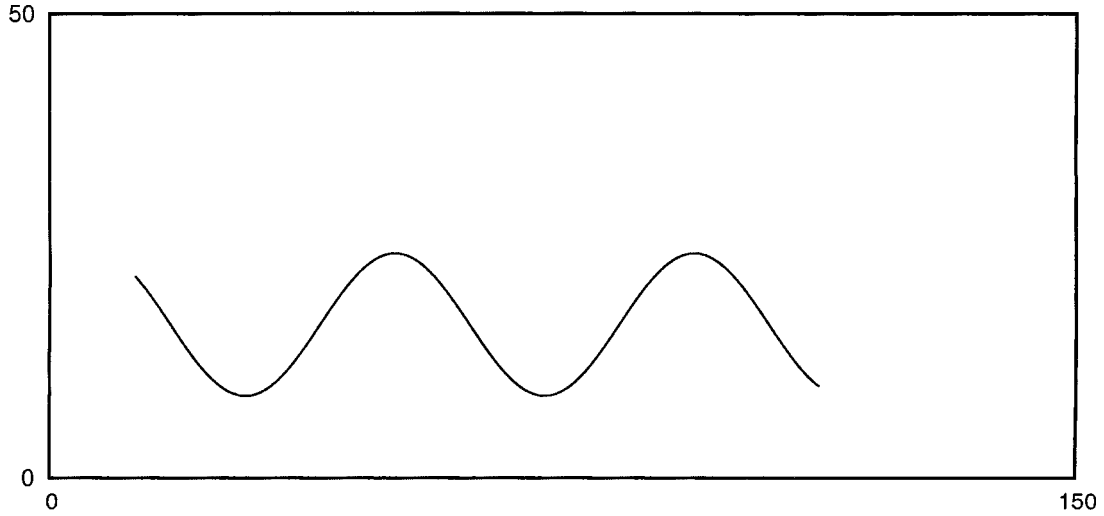

Figure 5.2: Sine Wave Movement Pattern by Amplitude 1.10, Frequency 80kHz

Constant Velocity Along Arc Length (SW (Arc Len))

The velocity of a defending sensor can be measured by one of two methods. A common way of measuring the velocity of traveling entities is by the length of the arc produced by the current location and the next location. We refer to this method as "Sine Wave (Arc Len)". When we assign sensors to the Sine Wave (Arc Len) model we designate their velocity as constant along the length of the arc between two plotted points. Hence, a

sensor with a velocity of 1 m/s travels along the arc at that rate. In order for the velocity to be properly and accurately measured the arc length must be constant.

Constant Velocity Along X Axis (SW (X-Axis))

Likewise, the velocity may also be measured relative to the simulation area's X axis. This method of measuring velocity is done by observing the speed along the X axis. We refer to this method as "Sine Wave (X Axis)". A sensor that has been assigned the Sine Wave (X Axis) movement model and a velocity of 1m/s will move in a sine wave pattern whose velocity along the X axis is constant at 1m/s. The velocity along the arc varies. Consider two sensors using the same mobility model, frequency and amplitude, travelling toward a destination with an X axis coordinate as their destination. The sensor with a constant x-axis velocity of 1 m/s will travel further along the X axis than a sensor with a constant arc length velocity of 1m/s. This observation is very important when comparing the effectiveness of one such velocity type to the other.

Simulation Results

Sensors which move in a sine wave pattern with a constant velocity along the X axis yielded the highest rate of detection for all scenarios, however sensors with a constant velocity along the sine wave arc performed at less than both static sensors and random boundary sensors. From the perspective of a single sensor moving in a sine wave pattern with constant X axis velocity the chances of detecting an intruder are more promising due to the movement along both the X and Y axes. Additionally as a sensor increases and decreases acceleration to maintain the constant dimensional velocity its speeds go above 1.0m/s, giving it somewhat of an advantage over the other designated movement patterns. Without this constant velocity sine wave patterns offer no additional utility against a vertically moving intruder.

As the number of detecting sensors in the area increases the benefit of statically deployed sensors grows at a higher rate than that of the other three scenarios. Simulation results show that deploying a low number of static sensors into the area is ineffective. The scenario in which sensors move in a straight line at a random direction yields the highest

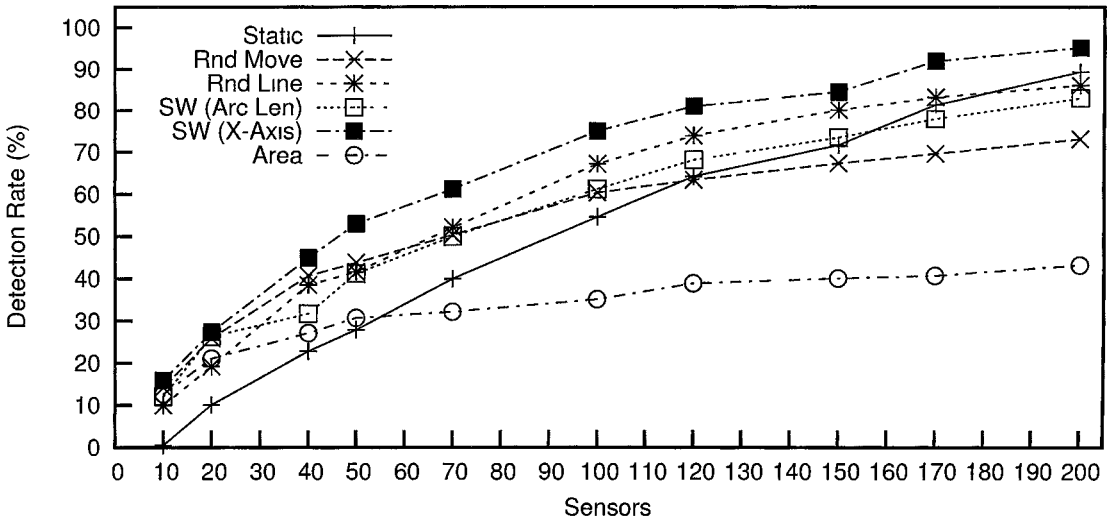

Figure 5.3: Detection Rates: Vertical Intruder, Uniform Sensor Deployment

rate of detection at the maximum number of sensors, yet at lower sensor counts is not as effective. Sensors constrained to an area were not effective for several reasons. First, the rate of detecting intruders was considerably lower than the other three scenarios. Also since the energy expenditure of this scenario is the same as the random mover scenario, the lower probability of detection affords no advantage to constraining the movements of a sensor to the area surrounding its point of deployment.

Environments containing less than 100 nodes may benefit more from employing random moving sensors to detect intrusions, yet environments with more than one-hundred nodes would yield a higher rate of detection by sensors dictated by random straight-line defenders. Monitored areas may actually fluctuate with regard to the number of active sensors; as the battery power in sensors deteriorate they are recharged or replaced [23]. When sensors cannot be replaced at the same rate at which they become inactive then the best overall method of movement is random straight-line defenders. It is worth noting that when the number of detectors surpasses some threshold (in this case, one-hundred sensors), static sensors yield a very competitive rate of detection. At 120 nodes and higher static sensors reach and surpass the detection rates of random moving intruders. As battery power is an ever-present concern in sensor networks, the energy saved by static deployments yields a better overall benefit when resources are scarce.

When sensors are not replaced at the rate at which they deplete their energy source,

random straight line defenders provide the best overall protection. Otherwise when the number of active sensors is guaranteed to never fall below a certain threshold and energy conservation is paramount static sensors provide a favorable compromise between effectiveness and power consumption.

CHAPTER 6

VERTICAL INTRUDER, RANDOM SENSOR DEPLOYMENT

Four different scenarios were examined in which a vertically-moving intruder faces various detector deployments. These scenarios differ from the previous simulations where a vertically moving intruder faces uniform deployments of detecting sensors. Defending sensors are deployed randomly then are either static or mobile, employing a particular pattern of movement.

Random deployments have the potential to create high concentrations of defending sensors in a particular area, highly increasing the probability of detecting intruders in those areas. This observation gives way to creating heuristics used to guess an intruder's point of deployment, exit point and movement patterns. As more intruders are detected the history of those parameters are stored; from those statistics can heuristics be created.

Simulation Results

The results for static defenders, as well as both random movers and random straight-line movers indicate that random deployments are more effective than uniform deployments when coupled with these movement patterns. Surprisingly, the rate of detection for random moving and random straight-line moving detectors are more similar than in simulations deploying according to uniform coordinates. Protecting the area with fixed sensors yields a slightly lower rate of detection when deployed with uniform coordinates.

In these simulations the detection rate for random movers and random straight-line movers are very similar yet are not as effective as when deployed in uniform distribution. Despite their decreased detection rates both patterns produce more detections than static sensors, who thus far have performed better in uniform deployments. Sine wave movements for which the velocity is measured by X axis were most successful, due largely in part to the

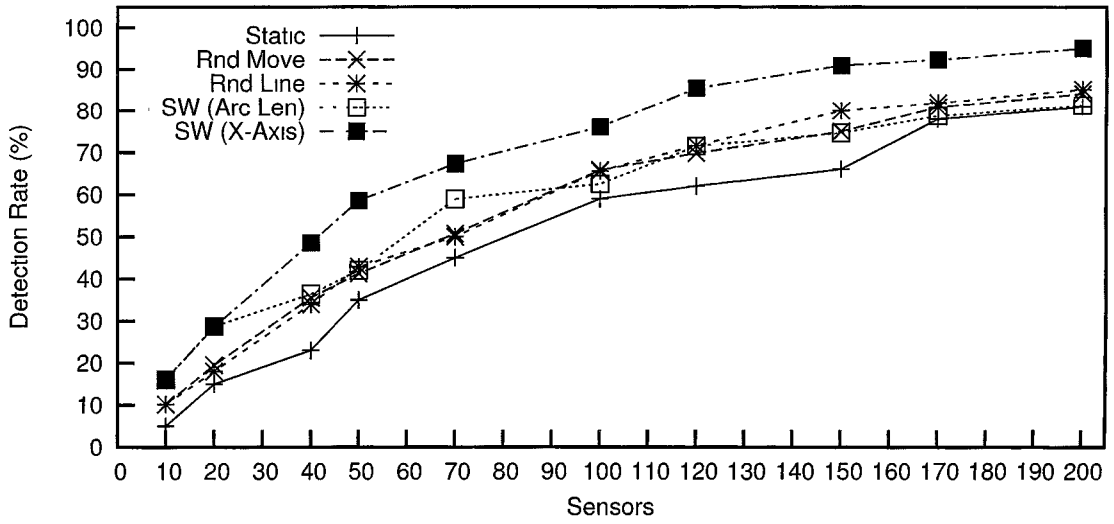

Figure 6.1: Detection Rates: Vertical Intruder, Random Sensor Deployment

sensors, coverage along the X axis and improved coverage along the Y axis. Additionally, the fluctuations in the y coordinate distance allow defending sensors to either detect intruders preemptively or detect intruders after they have crossed the y coordinate of a defender's point of deployment.

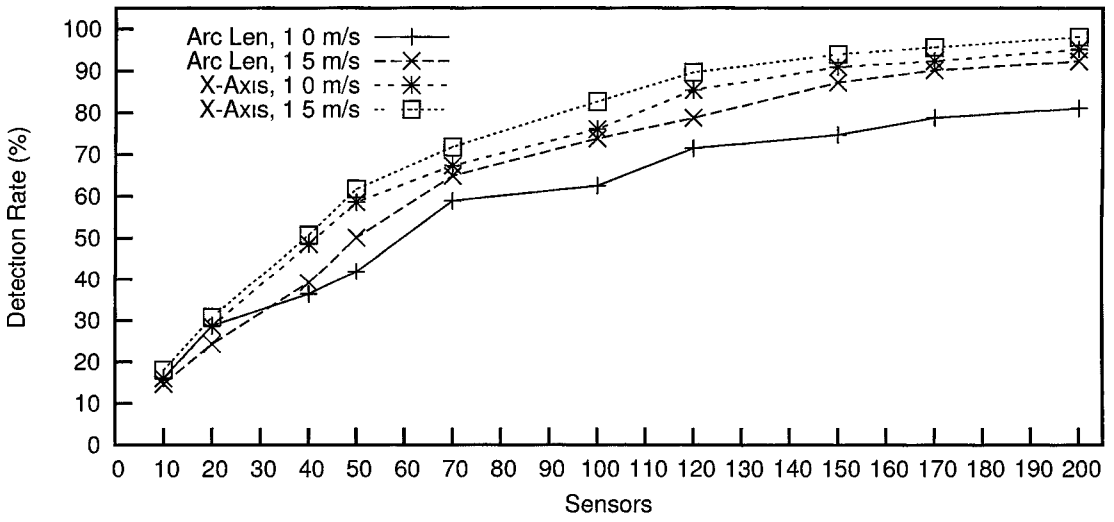

Figure 6.2: Detection Rates: Sine Wave Movements by Sensor Velocity

The random deployments of sensors cause some irregularities in the results. Figure 7 and Table 1 shows how the rate of detection increases in sporadic amounts as more sensors are added. Static sensors and random straight line sensors experience the highest frequency

of incidents where the detection rate is affected by this.

Figure 6 shows how the detection rate is affected by the velocity of defending sensors moving in a sine wave pattern. At an increase of 0.5 m/s the rate of detection is increased by anywhere from 5-12%. Obviously, the energy expenditure is increased by this however the cost to benefit ratio between energy depletion and detectability is affected. While sine wave / X axis velocity sensors have significantly higher rates of detection the benefit does not come without cost. Given that sensors have limited battery power, the positive and negative acceleration needed to maintain the constant velocity along the X axis requires more energy than what is required to enable the other proposed movement patterns. Furthermore sensors moving in a sine-wave pattern at a particular speed per constant arc length will consume less energy than sensors moving in a sine-wave pattern at the same speed per unit along the X axis. Thus, when the lifetime expectancy, simulation area and other circumstances permit liberal energy usage then the latter approach is most favorable. Otherwise, random deployment of random straight-line defenders proves to be effective and consistent across simulations of both random and uniform deployments.

CHAPTER 7

RANDOM INTRUDER, UNIFORM SENSOR DEPLOYMENT

Three different scenarios were examined in which a randomly deployed intruder faces static, random moving, random straight-line moving and sine wave movement patterns. The intruder is placed at any position at one of the four boundaries of the simulation area, and moves in a straight line to any boundary other than the one at which it was deployed. The intruder moves in a straight line to minimize the time it spends in monitored territory. Random entry and exit coordinates afford intruders the benefit of a variable length traversal through the protected area; this places the intruder at more of an advantage than in previous scenarios however this behavior is more realistic. For example, a sensor deployed in the northwest corner of the simulation area may choose to cross the area and exit at the nearby boundary on the west side (so long as the origin of the intruder is on the north boundary), or it may choose to exit at some point near the southeast corner. The possibility that an intruder is deployed at the bottom of the simulation area and moves vertically in a straight line does still exist albeit the chances are quite small.

Unlike the intruder, detecting nodes are deployed according to uniform positioning and begin their assigned movement behavior when the simulation begins. Given the nature of the three movement patterns assigned to the protectors, a randomly deployed intruder that is inside the monitored area for the same amount of time as a vertically moving intruder faces the same probability of it being discovered. The only advantage the former has over the latter is that the distance between origin and destination is of variable length. The opportunity for higher rates of detection exist when that variable length is longer than the length of traversing a straight line from Y = 0 to the maximum Y coordinate.

As intruders are deployed at one boundary and exits the area at another they are subject to increased risk of detection; If a vertically-moving intruder is within the commu-

nication range of a static defender the intruder will be detected by the first defender at that X coordinate. Likewise, diagonal movements across the area place the intruder closer to the communication ranges of more defenders. Furthermore while each mobility model and sensor quantity is tested 1000 times it should be noted that at each test the intruder is given a new set of entry and exit coordinates.

Simulation Results

Static defenders perform very well, yielding a higher rate of detecting randomly deployed intruders than random straight-line defenders and random moving defenders across simulations with all quantities of sensors.

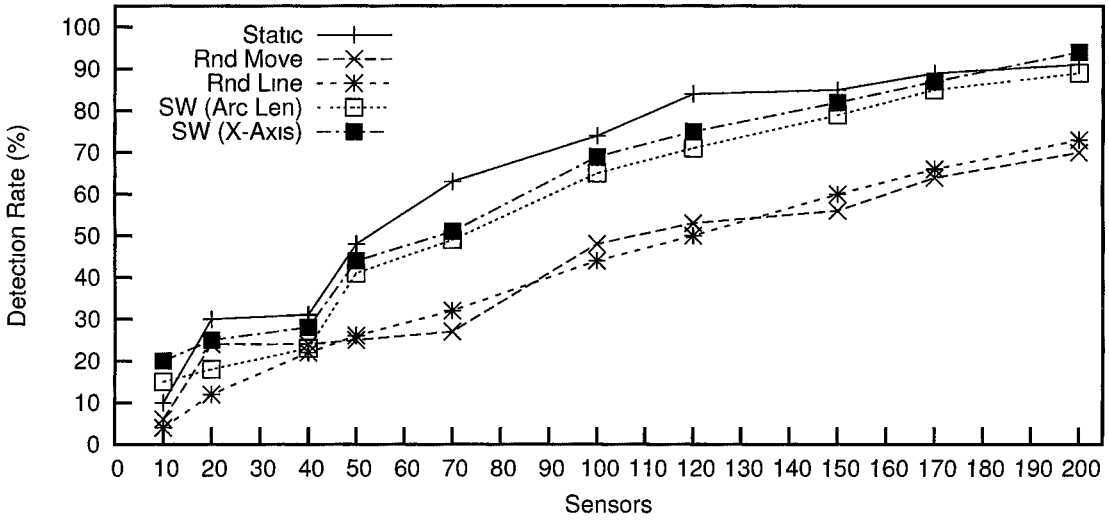

Figure 7.1: Detection Rates: Random Intruder, Uniform Sensor Deployment

The outcome of this experiment is similar to Chapter 6 due to the anomalies in the results. For instance, random moving sensors benefit from a 300% increase in detection rate when increasing their quantity from 10 to 20, however the benefit from 20 to 40 sensors is negligible. The increase in detection rate is quite sporadic because of the unpredictable entry and exit coordinates obtained by the intruder. While giving the intruder new random coordinates at each simulation causes some uncertainty about the amount of benefit from increased sensor quantities, the results still show that more sensors improves the probability of detection.

Despite the disparity in benefit, the results of this simulation indicate that defenders moving in a sine wave pattern with constant X axis velocity create the highest percentage of detection when the area is populated with two-hundred sensors. Yet, static defenders offer a higher rate of detection for all lesser node densities. Realistically the nature of a sensor network create difficulties in maintaining 100% node density, and as sensors become inoperable they will require a collective strategy that will account for variable numbers of defenders. Thus, static defenders are the obvious choice. Static deployments perform better than the other three strategies and their relatively small energy expenditure increases their life expectancy, ease of tracking and replacement. Therefore, static sensor deployments maintain higher node densities for longer periods of time.

CHAPTER 8

RANDOM INTRUDER, RANDOM SENSOR DEPLOYMENT

Three different scenarios were examined in which a randomly deployed intruder faces each of the five movement strategies. The intruder is placed at any position at one of the four boundaries of the simulation area, and moves in a straight line to any boundary other than the one at which it was deployed. As previously indicated an intruder will move in a straight line to minimize the time it spends in monitored territory. This behavior affords intruders the benefit of a variable length traversal through the protected area by providing the intruder more of an advantage than in previous scenarios.

Sensors protecting the monitored area are deployed at random locations throughout the area. Given the nature of the three movement patterns assigned to the protectors, a randomly deployed intruder that is inside the monitored area for the same amount of time as a vertically moving intruder faces the same probability of it being discovered. The only advantage the former has over the latter is that the distance between origin and destination is of variable length. The opportunity for higher rates of detection exist when that variable length is longer than the length of traversing a straight line from Y = 0 to the maximum Y coordinate.

Simulation Results

Intruders entering and leaving the monitored area at randomly chosen boundaries face the highest rate of detection against randomly generated, static defenders. Uniform deployment of static sensors still produces a higher detection rate against randomly deployed intruders however are still vulnerable to infiltration when a horizontally moving intruder maintains a constant Y coordinate or when a vertically moving intruder maintains a constant X coordinate. random moving and random straight-line defenders are much less effective

than both their static relatives and identically moving defenders in a uniformly deployed environment.

Most interesting is that the average detection rate for an area of 120 sensors is approximately 5% less than that of 100 nodes. This is attributed to the newly generated entry and exit coordinates at the beginning of each simulation. In general the random intruder and defending sensor deployments cause some irregularities in the results, yet overall it's still apparent that more sensors cause more detections.

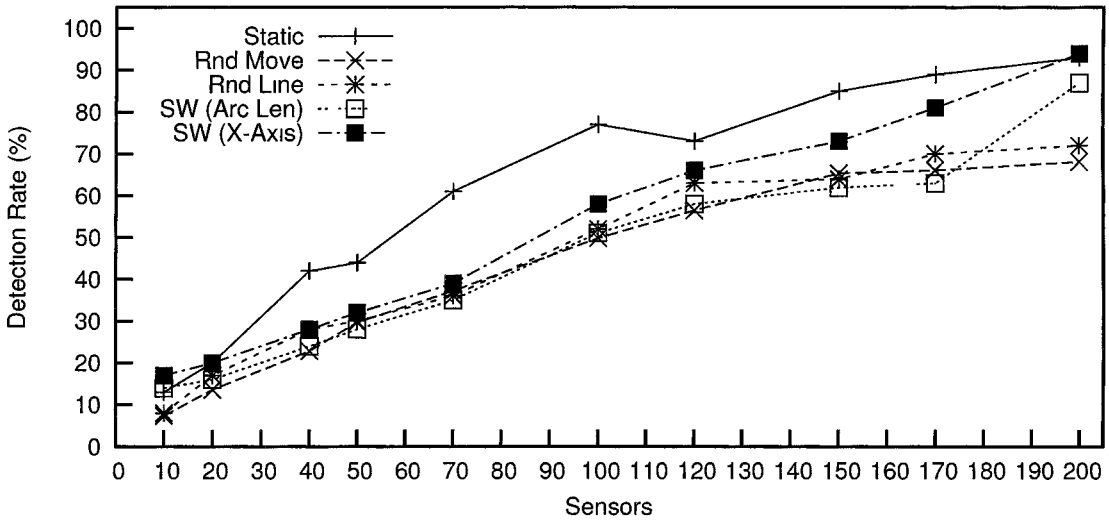

Figure 8.1: Detection Rates: Random Intruder, Random Sensor Deployment

Randomly deployed static sensors prove to be the ideal means of protecting areas where an intruder gains entrance to a monitored area by any boundary. Considering its obvious advantages with respect to energy consumption static sensors provide the highest overall benefit with its high detection rates at densities of forty nodes and above.

CHAPTER 9

COMPARISON OF MOBILITY MODELS

Thus far we have presented insight into which deployment and patrolling strategy is most effective against two different classes of intruders. Unfortunately financial, bureaucratic and technological constraints may prevent one from implementing the most ideal perceived solution. As such the effectiveness of each mobility model is evaluated by the expected intruder behavior and method of deployment. VI indicates the vertically moving intruder while RI indicates the randomly moving intruder. Likewise, we denote US as uniform sensor deployment and RS as random sensor deployment.

Static Sensors

Static sensors perform best when randomly deployed in areas where an intruder moves in a straight line from any one boundary to the other. They perform similarly well against randomly deployed intruders when positioned with uniform distribution. An apparent vulnerability is the probability of intrusion detection when low sensor densities are coupled with random distribution, as very large areas can be left completely unprotected. The only ways to cover unprotected areas are with more random sensor deployments or larger communication ranges. Static sensors provide a good solution to diagonally moving intruders because of their high detection rates and comparitively low energy consumption.

Randomly Moving Sensors

Randomly moving sensors yield the highest rate of detection when a vertically moving intruder infiltrates the bottom of the monitored area and exits through the top. Random distribution of these sensors against a vertically moving intruder yield higher percentages of detection. Simulation results suggest that intruders travelling through random entry and

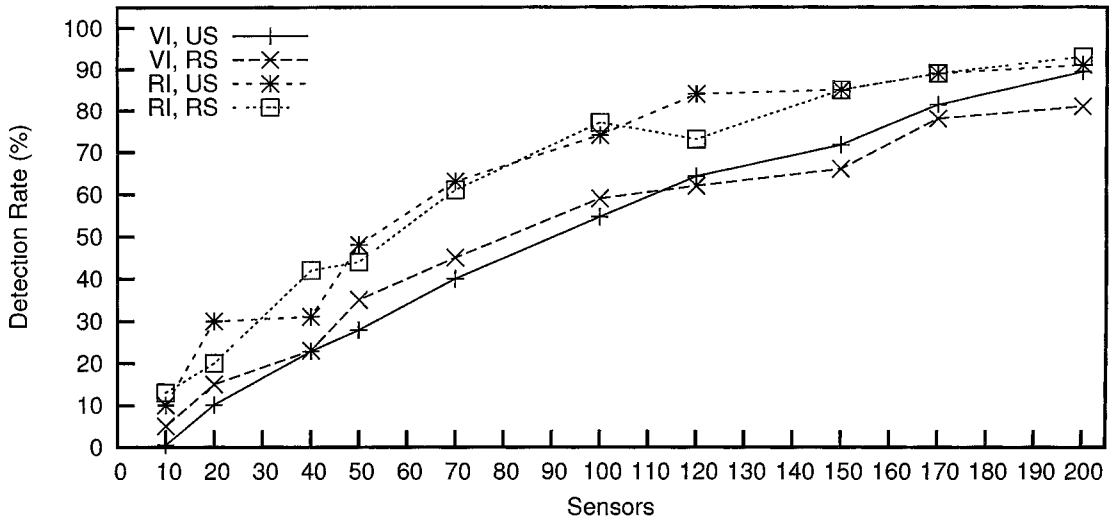

Figure 9.1: Static Sensor Performance

exit points have a significantly higher chance of traversing the area without being detected. As such, random moving sensors are not recommended against this class of intruder.

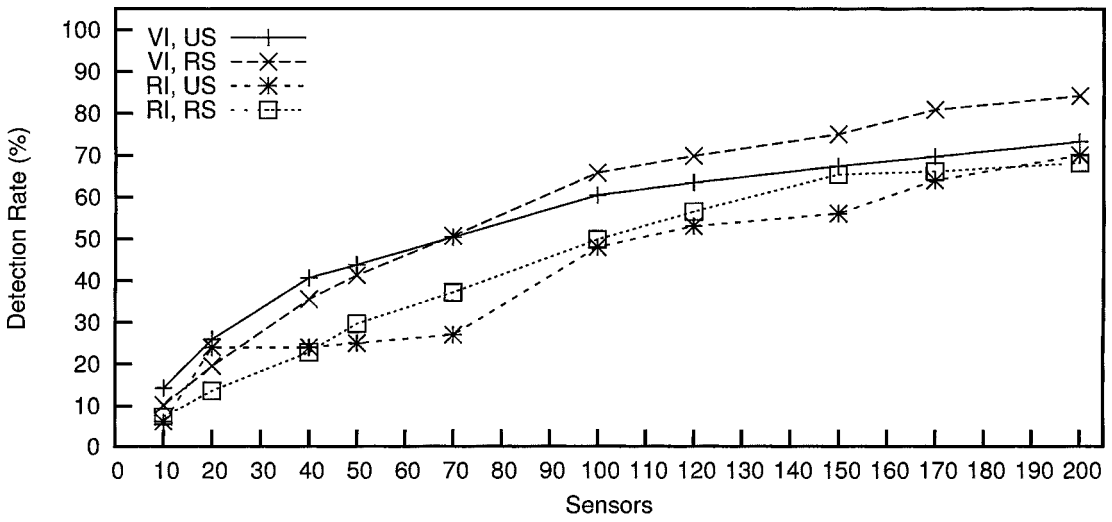

Figure 9.2: Random Moving Sensor Performance

Random Straight Line Moving Sensors

Similarly, random straight line sensors also perform well when infiltrated by a vertically moving intruder. Uniform and random deployment of these sensors perform equally well. A randomly deployed intruder has a higher success rate when evading random straight

line sensors. Random straight line sensors share the same shortcomings that affects randomly moving sensors - intruders with random entry and exit points are noticeably more successful at infiltration.

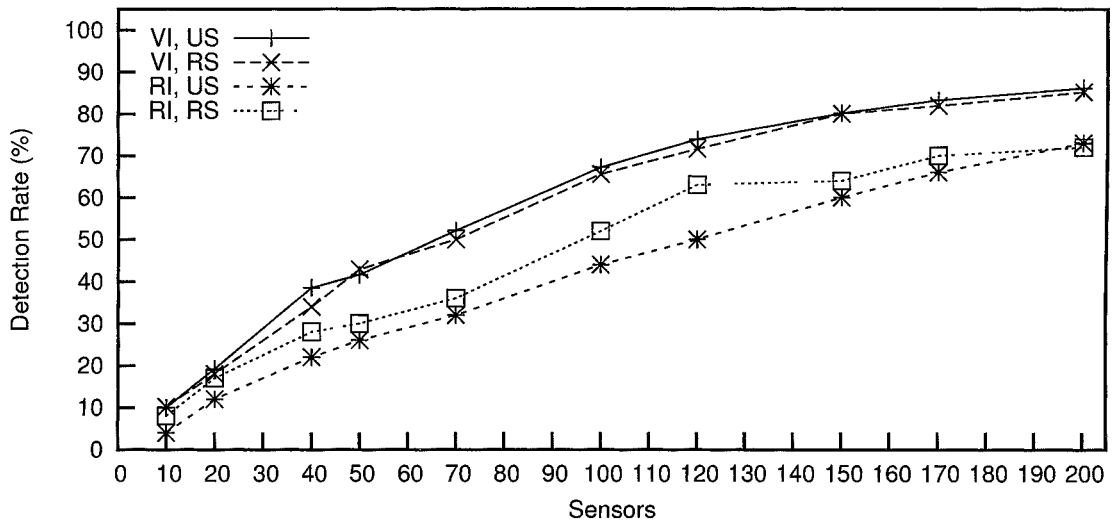

Figure 9.3: Random Straight Line Sensor Performance

Sine Wave Moving Sensors

Sensors which move in a sine wave pattern with constant acceleration perform somewhat well against both classes of intruder, however random deployments against an intruder with a random entry and exit point are not quite as effective. Despite this sensors guided by this method of mobility are somewhat of a general purpose solution when the cost of higher energy expenditure is not a point of concern.

Sensors which move in a sine wave pattern with a constant velocity along the X axis perform well against both classes of intruder however the variable amount of acceleration to maintain the X axis velocity raises large concerns regarding the amount of energy required to fuel these movement patterns. Nonetheless, random distribution of these against a vertically moving intruder succeeds in detection more than any other type of intruder and deployment method. If energy resources permit the usage of this mobility model then it would be most beneficial in border situations where the movement pattern is known to be vertical.

What these results have shown is that static sensors can provide a competitive rate of detection coupled with comparitively low energy consumption. However, one weakness lies

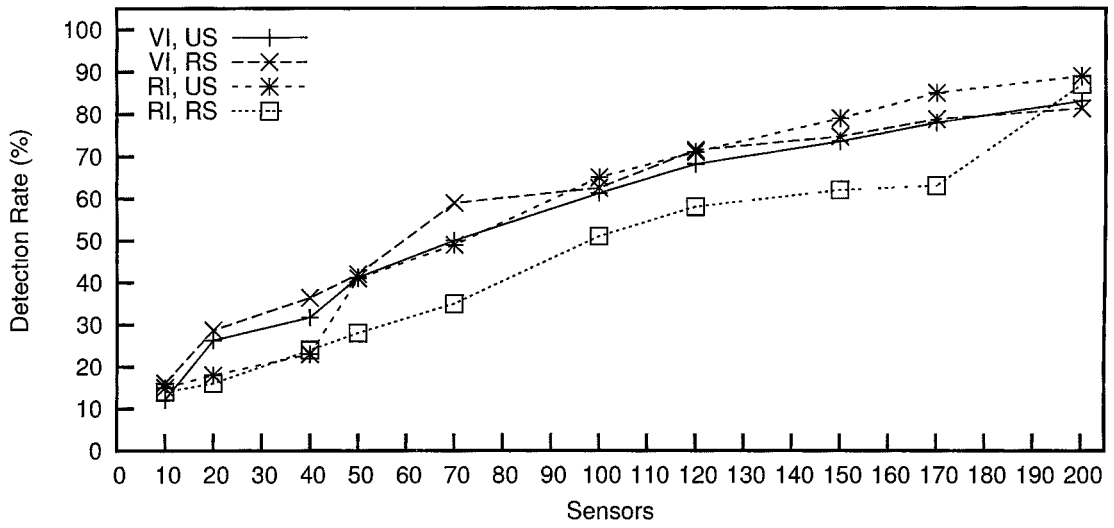

Figure 9.4: Sine Wave (Arc Length) Sensor Performance

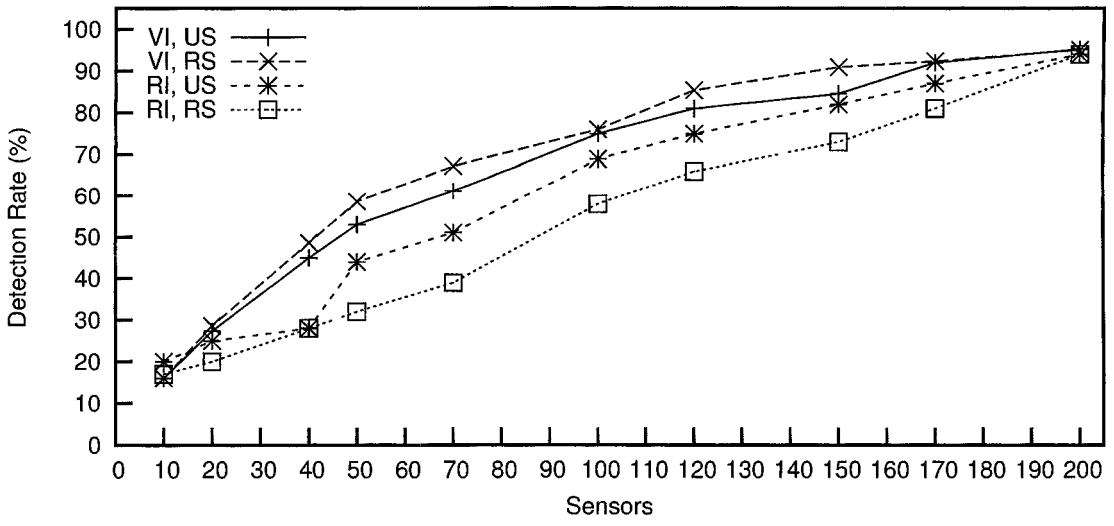

Figure 9.5: Sine Wave (X axis) Sensor Performance

in the detection rate yielded by low sensor densities. Thus, when the quantity of deployed sensors is low static sensors do not provide a viable means of detecting infiltrations. Only at a quantity of 150 sensors do static sensors provide an equal or greater amount of detectibility than random moving sensors, and only at 200 sensors do static sensors produce a higher rate of detection than sensors moving in a straight line toward randomly chosen boundaries. Conversely, static sensors yield much higher detection rates when the intruder enters the area at a randomly chosen boundary. Random moving and random boundary sensors are unable to match the performance of static sensors against this class of intruder, regardless of the area's sensor density.

The significance of these discoveries is that a sensor does not necessarily have to be mobile to be effective - against certain classes of intruder and certain types of infiltration static sensors do provide adequate means of protection. Areas of interest having no designated entry and exit points benefit from static sensors the most. Yet, consider a scenario in which a border between two independant areas is under surveilance. Many borders are divided by naturally occuring geographic bodies such as rivers, channels, streams, forests, etc. An intruder enters the boundary seperating both ends of the boarder at one side then move to the boundary at the opposite side. Static sensors would not perform optimally in this environment. Random boundary sensors would provide the second greatest probability of detection, just under sine wave sensors with constant X axis velocity. Overall randomly moving sensors do not provide scenario-based benefits that random boundary and static sensors provide. At low sensor quantities this mobility model performs approximately equal to or better than random boundary sensors, however at 70 sensors or greater its performance begins to normalize and does not increase at the rate of mobile boundary sensors. The constraints of maintaining sensor quantities at a narrow range between the minimum and maximum values can become burdensome.

Table 1 shows the percentage by which the rate of detectability increases as the quantity of sensors increases. The increase in percentage from one quantity to the next shows that as the number of sensors increase the additional percentage of additional detectability increases. The results show that as deployments surpass 120 sensors the percentage by which the detection rate increases is significantly lower than the rates afforded by lower sensor

quantities. This is significant because the threshold between large and small improvements is known for a particular area and scenario developers may calculate the range of desired sensor quantities. In the context of a vertically moving intruder against uniformly deployed sensors, one can initially deploy more than 120 sensors, however if the quantity drops below that threshold then the benefit of restoring a quantity of more than 120 sensors produces diminishing returns. The random moving, random boundary and static mobility models differ in these respective percentages. Static sensors generally experience the greatest amount of growth from one sensor quantity to the next, while sensors employing the random moving model see the lowest. Moderate to high sensor quantities see similar increases when guided by the sine wave and random boundary (straight line) models.

Static Sensors				
	VI		RI	
k	US	RS	US	RS
10	-	-	-	-
20	910.0	200.0	200.0	53.8
40	125.7	53.3	3.3	110.0
50	22.4	52.2	54.8	4.8
70	43.4	28.6	31.3	38.6
100	36.5	31.1	17.5	26.2
120	17.6	5.1	13.5	-5.2
150	11.7	6.5	1.2	16.4
170	13.5	18.2	4.7	4.7
200	9.9	3.8	2.2	4.5

Random Moving Sensors				
	VI		RI	
k	US	RS	US	RS
10	-	-	-	-
20	82.4	91.2	300.0	83.8
40	57.1	82.6	0.0	67.6
50	7.6	16.0	4.2	30.3
70	15.1	22.8	8.0	25.3
100	19.8	29.8	77.8	33.9
120	5.0	6.1	10.4	13.5
150	6.3	7.4	5.7	15.8
170	3.4	7.9	14.3	1.1
200	5.0	4.0	9.4	3.1

Random Straight Line Sensors				
	VI		RI	
k	US	RS	US	RS
10	-	-	-	-
20	92.0	76.5	200.0	112.5
40	100.5	88.9	83.3	64.7
50	8.1	26.2	18.2	7.1
70	25.2	16.6	23.1	20.0
100	29.0	31.2	37.5	44.4
120	10.0	9.1	13.6	21.2
150	8.4	11.7	20.0	1.6
170	4.0	2.4	10.0	9.4
200	3.4	4.1	10.6	2.9

Sine Wave (Arc Length) Sensors				
	VI		RI	
k	US	RS	US	RS
10	-	-	-	-
20	116.6	78.3	20.0	14.3
40	20.9	26.8	27.8	50.0
50	29.8	15.4	78.3	16.7
70	21.5	40.2	19.5	25.0
100	22.4	6.1	32.7	45.7
120	11.3	14.4	9.2	13.7
150	8.0	4.5	11.3	6.9
170	6.0	5.5	7.6	1.6
200	6.5	3.2	4.7	38.1

Sine Wave (X Axis) Sensors				
	VI		RI	
k	US	RS	US	RS
10	-	-	-	-
20	71.1	78.8	25.0	17.6
40	64.3	69.9	12.0	40.0
50	18.1	20.6	57.1	14.3
70	15.4	14.8	15.9	21.9
100	22.6	12.9	35.3	48.7
120	8.0	12.4	8.7	13.8
150	4.3	6.4	9.3	10.6
170	8.9	1.5	6.1	11.0
200	3.5	3.0	8.0	16.0

Table 9.1: Increase In Detection Rate For Mobility Models (%)

CHAPTER 10

BROADCASTING INTRUDER DATA

Broadcasting is a necessity in many applications of mobile ad hoc networks (MANETs). Within the context of incomplete sensor networks broadcasting is critical in providing sensors the capability to communicate and perform as well as networks for which there are a sufficient number of nodes for the geographic volume. In addition to message transmission, broadcasting in sensor networks also assists in routing protocols by way of route discovery [5]. Intrusion detection is greatly improved by effective broadcasting. At the point at which an intruder is first detected its coordinates and direction of travel are known. If these bits of information are broadcast to other detecting sensors then more information may be calculated: movement patterns, rate of acceleration, velocity, communication range, and exit point. In large collections this data can provide invaluable information about the strategies used by intruders and could be used to create heuristics which guide detecting sensors according to the information surrounding future initial detections of intruders.

Objective

The purpose of this experiment is to provide insight into the requirements for convergence with respect to node density, communication range, and the time required to reach convergence in an incomplete sensor network. While complete sensor networks are all but guaranteed to converge faster than an incomplete sensor network, we search for ways that an incomplete sensor network can produce competitive results. Simple Flooding is used as the broadcasting mechanism; the algorithm begins with a sensor broadcasting a packet to all neighbors after which all of the source's neighbors re-broadcast the packet [12]. This sequence of events occurs until all sensors have received the broadcasted packet. While Simple Flooding is quite primitive it is regarded as a protocol for broadcasting in

ad hoc networks with low sensor densities and / or high mobility [12]. Since these two characteristics are common to an incomplete sensor network the results below are presented as a worst-case scenario in the potential achievement of a sensor network's broadcasting capabilities; undoubtedly there exist more sophisticated and effective broadcast protocols. Additionally, properties that gauge the effectiveness of a broadcast model are usability and realism. Usability means that the model is abstract enough that the algorithm designers disregard the low-level details of hardware and general enough to allow algorithms to be designed [2]. Conversely, realism is a measure of how well the model represents real systems. The following simulation experiment aims to achieve a good representation of broadcasting in an incomplete sensor network.

While there has been research in broadcasting within MANETs the usability of designed protocols within incomplete sensor networks has not been addressed in large detail. These experiments are presented with the anticipation that others will follow suit and continue to examine the effect of newly designed protocols on networks with deficiencies in node densities.

These simulations seek to provide insight into the following problem: given a number of nodes, what combinations of communication range and elapsed time will yield a dissemination rate of 100%? If complete distribution of a message cannot be achieved by way of a particular range and time, what is the percentage of sensors who have received a broadcast? Within a 150×50 area Simple Flooding is performed among $k = 10, 20, 40, 50, 70, 100, 120, 150, 170, 200$ sensors. Four communication ranges are examined: 1m, 5m, 10m, and 20m. For all simulations the sensors move randomly within the area in which they are deployed.

Related Literature

Extensive research has been based in broadcasting among ad hoc wireless sensor networks. These protocols include many sophisticated techniques - techniques designed to not only increase the success rate of broadcasting information to all sensors but to satisfy energy, communication, and computation constraints of the sensor network [6]. Probability-based protocols like GOSSIP and Fireworks always broadcast messages originated from the

source yet re-broadcast received messages at probability p [17]. Obviously when $p = 100\%$ these protocols are analogous to flooding protocols [32]. Networks employing area-based methods determine the coverage area as redundant data packets are received. Additionally there is much effort in developing neighbor-based protocols. Neighbor relationships are formed between sensors via "Hello" packets creating link or vector based tables by which sensors can choose to re-broadcast received messages. While the amount of research for broadcasting / multicasting is undisputed the concerns of how proposed protocols perform in an incomplete network have been largely unaddressed.

[6] also studies energy consumption when broadcasting is the mechanism for situation awareness in MANETs. However, the study assumes that sensors in the network are aware of the coordinates of all other sensors in the area. This study examines only static sensors and thus this research extends the amount of knowledge in similar scenarios but with mobile nodes. Additionally, this study gives sensors the power control to reach sensors in any region of the area of interest. Therefore while sensors are of homogenious hardware architecture their transmit and receive thresholds are of heterogenious distribution. This simulation deploys all nodes with the same transmission radius.

Localized techniques for broadcasting in large ad hoc networks are proposed in [17]. These methods anticipated a very high probability of disseminating a message from a source to all other mobile sensors. One method aims for the network topology to become increasingly sparse while another utilizes probabilistic flooding. Peng and Lu propose an alternative to mere flooding in mobile ad-hoc networks in an attempt to improve conservation of network bandwidth and energy [20]. The model is adaptive to the size of the network and utilizes reverse learning in determining which sensors have already received a particular broadcast message. Both [15] and [10] proposes algorithms. that reduces message redundancy and excess energy consumption by strategically placing relay sensors exclusively designated for broadcast transmissions [5] describes a protocol that utilizes directional antennas to execute self-pruning. This directional self-pruning (DSP) protocol equips nodes with only 2-hop neighborhood information achieved via "hello" packets.

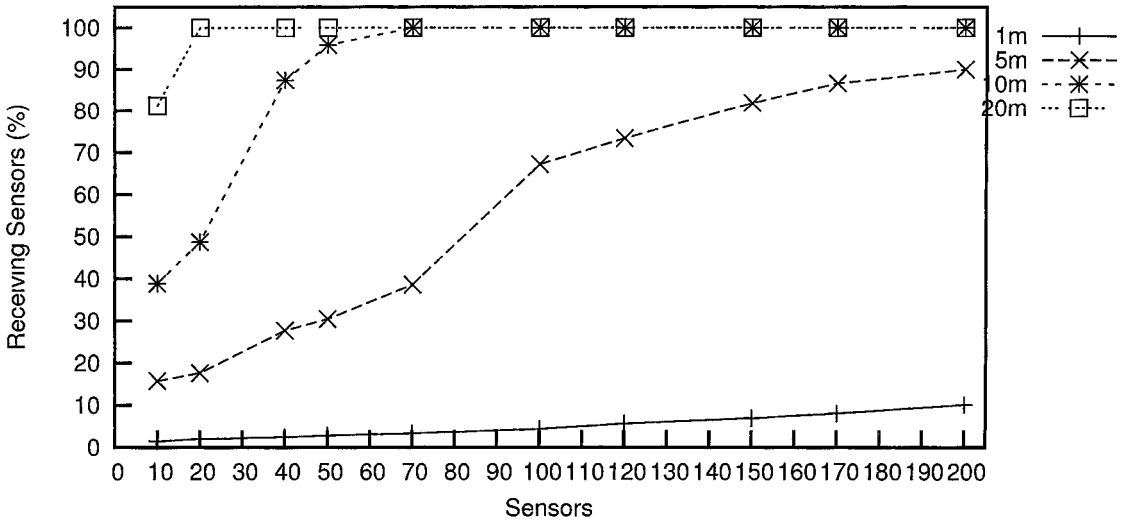

Figure 10.1: Broadcast Rates for Elapsed Time 50s

Rate of Convergence by Quantity and Range

Figure 15 indicates that a communication range of 1m is not sufficient in achieving reliable broadcasting for any number of nodes. The implications of these figures suggest that sensors deployed for intrusion detection must have a larger threshold for transmitting and receiving data. Otherwise, sensors who are first to detect an infiltration are incapable of providing to other sensors such information as intruder coordinates and direction. With communication range of 10m a minimum of 40 sensors are needed to transmit broadcast messages to 80% of the entire sensor population.

At twice the elapsed time of the initial simulations do sensors become equipped to broadcast with a communication range of 5m. However, one hundred sensors or greater are needed to transmit broadcast messages to 80% of the entire sensor network. Thus, the number of deployed sensors will greatly affect the strategy behind their deployment. At one hundred nodes it is possible to feasibly track more advanced intruder information. When the number of sensor is less than one hundred the deployment strategy may best benefit from the same approach as a 1m communication range and elapsed time of 50 seconds.

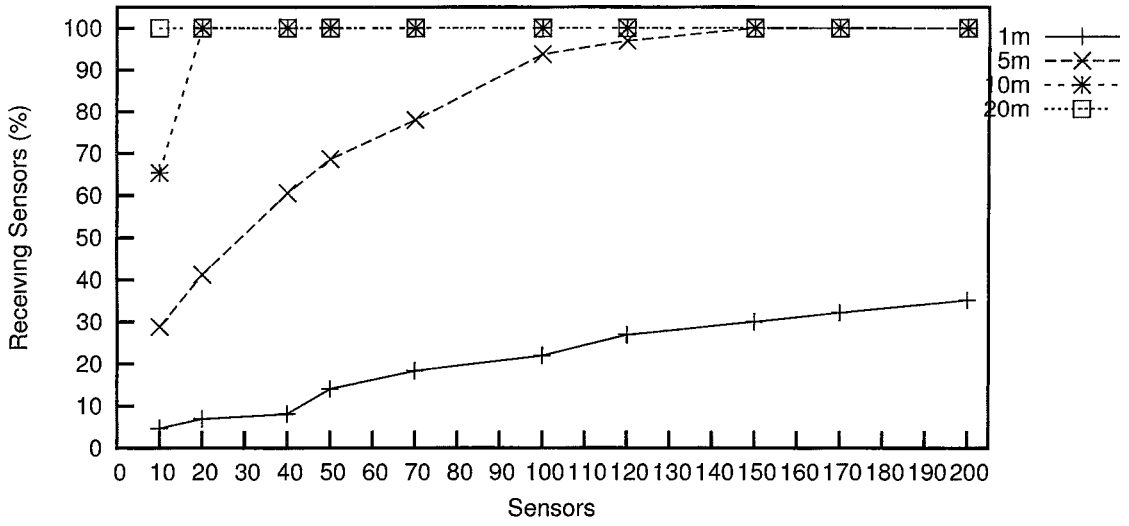

Figure 10.2: Broadcast Rates for Elapsed Time 100s

Average Convergence Time by Quantity and Range

Figure 15 illustrates that a communication range of 10m and a sensor density of 70 nodes is the lowest combination for which a sensor can broadcast intruder data to all other defending sensors. We now wish to know the average elapsed time required for each combination of communication range and sensor density achieving full convergence. Figure 17 provides the average elapsed time for sensor densities that reached full convergence with a communication range of 10m. Seventy sensors equipped with a communication range of 10m converges at an average elapsed time of just above 19 seconds. Deploying an additional 30 nodes places 100 sensors in the surveillance area and reduces the convergence time by a factor of 2/3. As the number of deployed sensors increases from 70 to 200 the average convergence time decreases however the improvement is less pronounced as more nodes are deployed.

The average time of convergence for sensors with a communication range of 20m shows a significantly higher result for deployments of 10 sensors than deployments with higher sensor quantities. This is explained by the fact that randomly generated coordinates may deploy sensors far away from one another. Our defending sensors move randomly at a velocity of 1m/s and as such may have to travel a significant distance to arrive within another node's sensing radius. Doubling the number of sensors in the area reduces the convergence

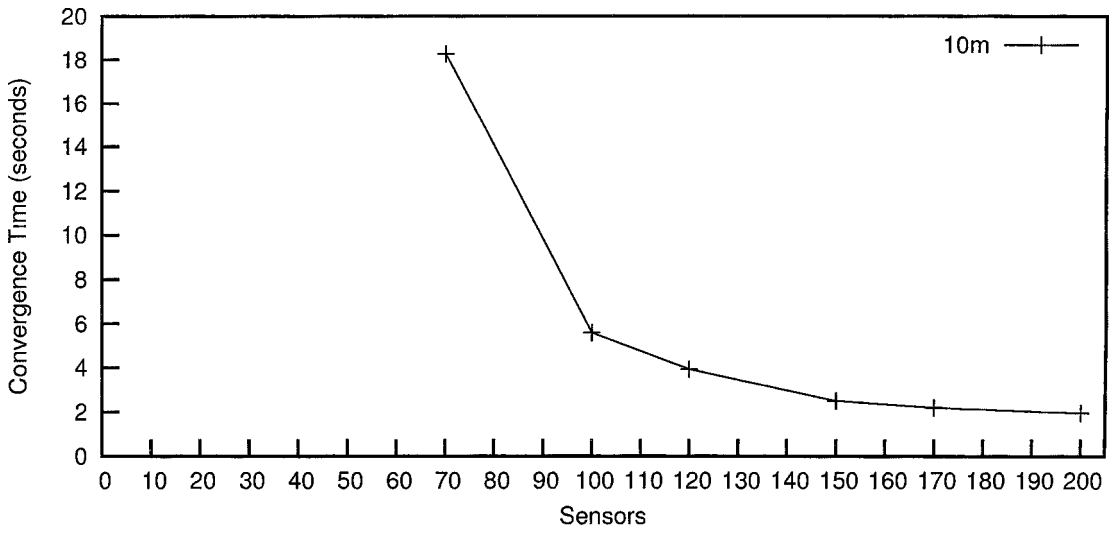

Figure 10.3: Convergence Time by Node Density, Signal Strength = 10m

time by a factor of approximately 1/3. When 40 sensors are present the convergence time reduces quite significantly. Figures 18 and 19 show the relationship between communication range and convergence time for deployments of the various sensor quantities. Figure 19 shows this relationship for deployments of 40 sensors or greater.

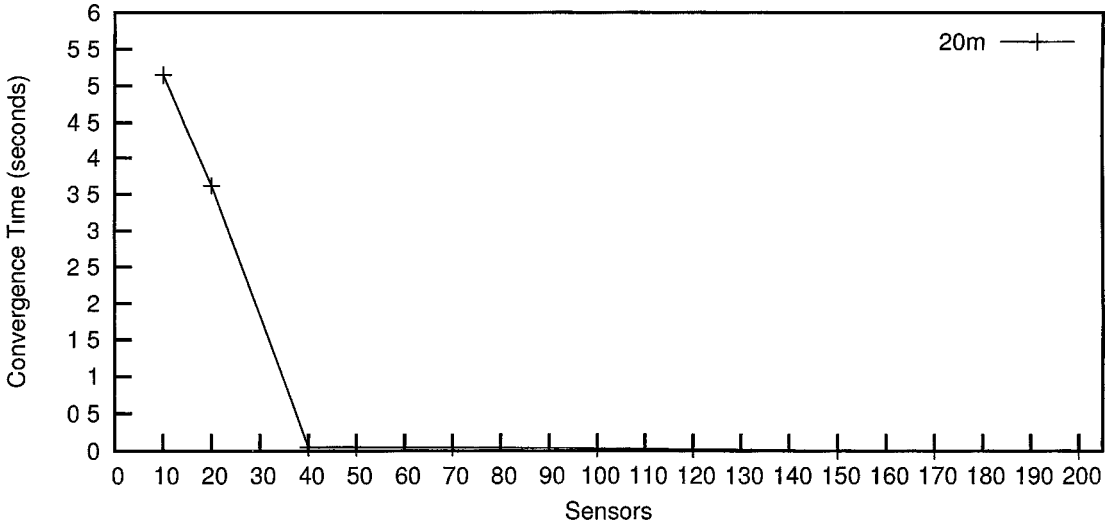

Figure 10.4: Convergence Time by Node Density, Signal Strength = 20m

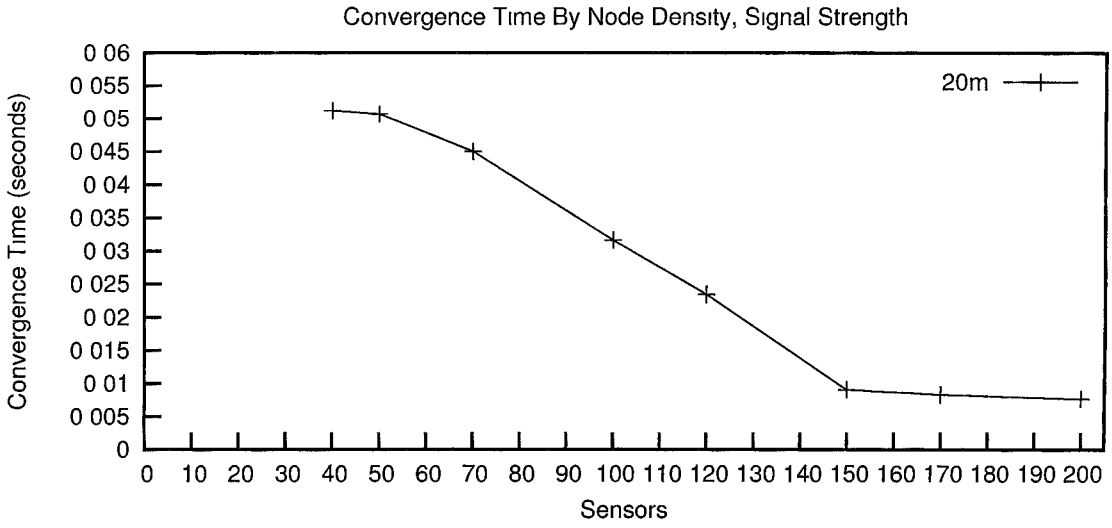

Figure 10.5: Convergence Time by Node Density, Signal Strength = 20m (Scaled)

Observations

When sensors are alotted a range of 10 meters or more and an elapsed time of 100 seconds the broadcast rate under Simple Flooding fast approaches 100%. A range of 20 meters yields a rate of 100% for any number of nodes yet the energy required to maintain this range is significantly higher than a node maintaining a 1 meter range. If energy and technological constraints do not allow for a larger communication range then networks with extremely low node density are best deployed behind a preemptive strategy designed to point all resources at a particular area. The game of Football's "blitz" defense is one which stems from the same strategy and motivations. In fact, in order for the sensors deployed in earlier simulations to achieve a decent broadcast rate each sensor must have a transmit and receive threshold of 10m.

These results indicate that when the number of sensors are low and energy resources are abundant a wider range is appropriate. When the node density is high and the energy resources are scarce a feasible compromise is found in deploying sensors with a 5 meter range. Lastly areas with both low node density and low energy resources do not provide many options in terms of broadcasting or detectability. Deploying static nodes, trading off mobility for increased transmission range is one such option. Sensors may be deployed densely in a particular area, leaving other areas vulnerable yet also providing sensors adequate means

of communication by placing them within the range of other sensors.

Recall that the communication range of the detectability expiriment was 1 meter. Obviously such a short range in relation to the size of the surveilance area means that sensors Will not arrive at a useable level of convergence. Thus, the range must be increased. Increasing the range by a factor of five does not yield a 1oo% convergence rate however it does provide a node the means to communicate information to other nodes residing nearby. Even a small amount of information may be compiled into data that is beneficial to future deployment strategies.

These results are significant in that the insight into the average convergence time for various sensor quantities and communication ranges supply a wealth of information about what is achieved in monitoring an area for intrusions by way of incomplete sensor networks. Given this information one can gauge the expectations of what the deployment strategy, sensor quantity and communication range can achieve in terms of detectability and dissemination of data. Nodes in high sensor quantities and densities do not require the large coverage radius than those in a network composed of low sensor densities.

CHAPTER 11

CONCLUSIONS

Several mobility models across both randomly deployed and uniformly deployed networks have shown to have varying degress of success in detecting infiltrations. The empirical data supports the intuition that none of the proposed mobility models are ideal for every scenario. For instance, static sensor deployments are most effective in detecting breaches of the monitored area when their deployment is random and not uniform. Real world examples of where static sensors perform well are the following: protected airspace, wildlife habitats, and areas considered to be adversarial / contended by the military.

Likewise, mobile sensors whose movements are based on random number generation are most effective when the intruder moves in a straight line from the bottom of the area (Y = 0) to the top of the area (Y = maxY). If a sensor network is deployed to detect intrusions across a border such as the one between Mexico and the United States then mobile sensors will provide the highest rate of detection.

Sine wave movements with a constant velocity along the wave arc perform similiarly well against both classes of intruder when positioned randomly and uniformly. When these nodes are guaranteed a constant velocity along the X axis the rate of detection is quite high, however the energy consumption resulting from the positive and negative acceleration is a major concern and in some scenarios unrealistic. With these observations in mind the complexity imposed upon scientists and engineers in choosing an appropriate deployment method and mobility model is reduced.

The above deployment strategies and mobility models are supplemented by effective means of broadcast communication. We have learned that the 1 meter communication range utilized by defending sensors is severely inadequate in achieveing full dissemination of intruder data. Ideally, given a sufficient amount of time any number of nodes could achieve

a 100% broadcast rate however an increase in communication range greatly accelerates the rate at which this is achieved. In our simulations we have converged seventy sensors inside a 150×50 area in at most 50 seconds with a 10m receive and transmission threshold. Given twice this amount of time a network of twenty sensors with a 10m communication range can broadcast intruder statistics to all nodes. On average, 10 sensors with a communication range of 20 meters converge in as little as 5 seconds and 150 sensors converges at approximately one-thousandth of a second. The curves in Figure 18 and Figure 19 show that the communication range employed by sensors is much more important than their quantities. The additional speed of convergence as more sensors are added is not linear and has diminishing returns. The relationship between the size of the area and the communication range dictates the threshold at which the costs of increasing the sensor quantity is equal to or greater than the benefit from deploying additional sensors.

Future Research

As wireless sensor networks continue to raise interest from academic, commercial and governmental organizations the need for additional research will become more abundant. Incomplete sensor networks as a subset of this research can greatly benefit from:

Pseudo-Randomness

Psuedo-randomness with respect to deployment and movement are two ways to manipulate the perceived sensor density in networks where the density is low. Additionally pseudo-randomness can provide sensors in a network with an algorithm to determine the location of other sensors; anyone without access to the algorithm does not have the insight to reliably predict the location of sensors. Sensors deployed and moved according to pseudo-random algorithms may also positively affect the rate of sensors who receive broadcast messages - as well as the time it takes to receive them.

Hybrid Networks

While the experiments presented herein explore networks composed of a homogenious mobility model, there exists the potential for sensor networks to be populated with

sensors adhering to many patterns of movement. Among the multitude of possible combinations are networks composed of both static and mobile sensors, sensors with varying axes on which they base the majority of their movements, and sensors with different mobility models. Some mobility models may be complimentary to other models, providing more significant means of both intrusion detection and broadcasting.

Heuristics

As intruders are detected defending sensors can either broadcast or multicast information to other sensors. This information can include (but is not limited to) the coordinates of the detection, the angle at which the intruder entered the sensor's sensing radius and the elapsed time in which the sensor stayed within the sensing radius. When collectively shared this information can be manipulated to form heuristics about the intruder's favored points of entry, velocity, direction of travel, frequency at which detections occur in a particular area, etc. Sensors can be initially deployed randomly or uniformly, yet as time passes the deployment coordinates are determined by a heuristic function based on previously gathered data. Similarly sensor movements can be initialized to some random or uniform means, however as data is gathered they move in patterns dictated by a heuristic function driven by such variables as elapsed time since simulation start, frequency of detections in the area in which they're travelling, and / or the angle at which the most sensors detect an intruder.

Mobile Sensor Alternatives

The deployment methods and mobility models discussed herein are applicable not only to sensors that have mobile capabilities. Assuming that mobile sensors have a much higher cost than their static counterparts, many static sensors can be used in place of mobile nodes, evenly distributed throughout the area and made to emulate movement patterns by cooperatively managing the idle states of all sensors in the area. For example, where a mobile sensor would travel in a straight line static sensors would be turned on one sensor at a time. When some elapsed time has passed the sensor returns to its idle state and a neighboring sensor repeats the process. Strategically managing idle states in this manner also creates possibilities for enhanced broadcasting capabilities.

Energy Expenditure

While energy expenditure has been a focus of sensor network technology the importance of conservation in an incomplete sensor network cannot be overstated. There is a need for more sophisticated algorithms that govern active and idle states utilized by sensors. Furthermore transmission range, speed, rate / frequency of movement and overall amount of mobility are subject to the problem of energy expenditure.

APPENDIX

Provided below is the source code created to achieve our various deployment scenarios and mobility models. Line numbers are provided as well as brief descriptions embedded in the source code.

Uniform Deployed Random Straight Line Moving Sensors

```
00001 #!/usr/bin/perl
00002 # ------------------------------------------------------
00003 # Eric Reeves
00004 # Department of Computer Science
00005 # Texas State University - San Marcos
00006 # er40634@txstate.edu
00007 #
00008 # Filename:  sensors.pl
00009 #
00010 # Description:  Creates the scenario file for defending
00011 # sensors using the random straight line mobility model.
00012 # This script ensures that each sensor is moving for the
00013 # entire simulation.
00014 # ------------------------------------------------------
00015
00016 # ---------
00017 # Variables
00018 # ---------
00019 $nodes = $ARGV[0];    # Number of Nodes
00020 $time  = $ARGV[1];    # Simulation Time
00021 $maxX  = $ARGV[2];    # Max X Coordinate
00022 $maxY  = $ARGV[3];    # Max Y Coordinate
00023 $speed = 1.000;       # Speed of Node (For now, static)
00024 $i     = 0;           # Loop Index
00025 $j     = 0;       # Loop Index
00026 $n     = 0;           # Loop Index
00027 $xVal  = 0;           # X Value
00028 $yVal  = 0;           # Y Value
00029 $xDiv  = 0;           # Interval of uniform deployment along X axis
00030 $yDiv  = 0;       # Interval of uniform deployment along Y axis
00031 $perY  = 10;          # Nunber of nodes deployed on Y axis.
00032 $dist1 = 0;           # Distance
```

```
00033 $dist2 = 0;           # Distance
00034 $rndDest = 0;         # Random Destinations:
00035 $xDest1 = 0;
00036 $yDest1 = 0;
00037 $xDest2 = 0;
00038 $yDest2 = 0;
00039 @x     = ();        # Stores X coordinates
00040 @y     = ();          # Stores Y coordinates
00041
00042
00043 # ----------------------------------------------------
00044 # Only execute the script with the required number of
00045 # command line parameters.  Display a message and abort
00046 # otherwise.
00047 # ----------------------------------------------------
00048 if ($#ARGV != 3) {
00049   print "usage:  ./sensors.pl numNodes simTime maxX maxY\n";
00050   exit;
00051 }
00052
00053 $yDiv = $maxY / (($nodes / 10) + 1);
00054 $xDiv = $maxX / ($perY + 1);
00055
00056 $i = $yDiv;
00057
00058 while ($i < $maxY && $n < $nodes) {
00059   $j = $xDiv;
00060   while ($j < $maxX) {
00061     print "\$node_($n) set X_$j\n";
00062     print "\$node_($n) set Y_$i\n";
00063     print "\$node_($n) set Z_0.000000000000\n";
00064     $x[$n] = $j;
00065     $y[$n] = $i;
00066     $n += 1;
00067     $j += $xDiv;
00068   }
00069   $i += $yDiv;
00070 }
00071
00072 # ----------------------------------------------------
00073 # 1.  Randomly generate the boundary that each sensor will travel to.
00074 # 2.  Determine the length of the line from point of deployment to boundary.
00075 # 3.  At the time equal to that length, generate a new and different
00076 boundary.
00077 # 4.  Do this two times to make sure that the node is always moving for
the
00078 #    entire simulation.
00079 # ----------------------------------------------------
00080
```

```
00081 for ($n = 0; $n < $nodes; $n++) {
00082
00083   $rndDest1 = rand(4) % 4;
00084
00085   if ($rndDest1 == 0) {
00086     $yEnd = (rand($maxY));
00087     $xEnd = $maxX - 0.01;
00088   }
00089   elsif ($rndDest1 == 1) {
00090     $yEnd = (rand($maxY));
00091     $xEnd = 0.01;
00092   }
00093   elsif ($rndDest1 == 2) {
00094     $yEnd = $maxY - 0.01;
00095     $xEnd = (rand($maxX));
00096   }
00097   else {
00098     $yEnd = 0.01;
00099     $xEnd = (rand($maxX));
00100   }
00101
00102   print "\$ns_at 0.0 \"\$node_($n) setdest $xEnd $yEnd $speed\"\n";
00103
00104   $dist1 = sqrt( ($x[$n] - $xEnd)**2 + ($y[$n] - $yEnd)**2 );
00105
00106   if ($dist1 < $time) {
00107
00108     while ( ($rndDest2 = rand(4) % 4) == $rndDest1) {}
00109
00110     if ($rndDest2 == 0) {
00111       $yEnd = (rand($maxY));
00112       $xEnd = $maxX - 0.01;
00113     }
00114     elsif ($rndDest2 == 1) {
00115       $yEnd = (rand($maxY));
00116       $xEnd = 0.01;
00117     }
00118     elsif ($rndDest2 == 2) {
00119       $yEnd = $maxY - 0.01;
00120       $xEnd = (rand($maxX));
00121     }
00122     else {
00123       $yEnd = 0.01;
00124       $xEnd = (rand($maxX));
00125     }
00126
00127     print "\$ns_at $dist1 \"\$node_($n) setdest $xEnd $yEnd $speed\"\n";
00128
00129     $dist2 = $dist1 + sqrt( ($x[$n] - $xEnd)**2 + ($y[$n] - $yEnd)**2 );
```

```
00130
00131     if ($dist2 < $time) {
00132       while ( ($rndDest2 = rand(4) % 4) == $rndDest1 ||$rndDest3 ==
00133               $rndDest2) {}
00134
00135       if ($rndDest3 == 0) {
00136         $yEnd = (rand($maxY));
00137         $xEnd = $maxX - 0.01;
00138       }
00139       elsif ($rndDest3 == 1) {
00140         $yEnd = (rand($maxY));
00141         $xEnd = 0.01;
00142       }
00143       elsif ($rndDest3 == 2) {
00144         $yEnd = $maxY - 0.01;
00145         $xEnd = (rand($maxX));
00146       }
00147       else {
00148         $yEnd = 0.01;
00149         $xEnd = (rand($maxX));
00150       }
00151
00152       print "\$ns_at $dist2 \"\$node_($n) setdest $xEnd $yEnd $speed\"\n";
00153     }
00154   }
00155 }
00156
00157 #  print "\$ns_at $x \"\$node_($i) setdest $xArr[$i] $yArr[$i] $speed\"\n";
00158
00159 for ($i = 0; $i < $nodes; $i++) {
00160   for ($j = $i + 1; $j < $nodes; $j++) {
00161     #print "\$god_set-dist $i $j 167777215\n";
00162   }
00163 }
```

Random Intruder Deployment Script

```
00001 #!/usr/bin/perl
00002 # ----------------------------------------------------
00003 # Eric Reeves
00004 # Department of Computer Science
00005 # Texas State University - San Marcos
00006 # er40634@txstate.edu
00007 #
00008 # Filename:  intruder.pl
00009 #
00010 # Description:  Creates the coordinates for an intruder.
00011 # The X coordinate is randomly generated and bound by
00012 # $maxX. It is always deployed at the bottom of the area
```

```
00013 # at Y = 0.  The intruder moves from Y = 0 to Y = maxY in
00014 # a straight line.
00015 # ------------------------------------------------------
00016
00017 # ---------
00018 # Variables
00019 # ---------
00020 $nodes = $ARGV[0];          # Number of Nodes
00021 $time  = $ARGV[1];          # Simulation Time
00022 $incr  = $ARGV[2];          # Time Increment
00023 $maxX  = $ARGV[3];          # Max X Coordinate
00024 $maxY  = $ARGV[4];          # Max Y Coordinate
00025 $destY = $maxY - 0.1;       # Destination Y Coordinate
00026 $speed = 1.000;             # Speed of Node (For now, static)
00027 #$speed = 0.033;             # Speed of Node (For now, static)
00028 $rndX  = 0;                 # Random X Coordinate
00029 $rndY  = 0;
00030 $xStart = 0;
00031 $yStart = 0;
00032 $rndDir = 0;
00033 $rndDest = 0;
00034 $xDest = 0;
00035 $yDest = 0;
00036
00037 # ------------------------------------------------------
00038 # Only execute the script with the required number of
00039 # command line parameters.  Display a message and abort
00040 # otherwise.
00041 # ------------------------------------------------------
00042 if ($#ARGV != 4) {
00043   print "Usage:  ./intruder.pl nodes maxTime timeIncr maxX maxY\n";
00044   exit;
00045 }
00046
00047 # ------------------------------------------------------
00048 # Generate the intruder's position at a random position
00049 # along the X axis.  The Y coordinate is initialized to
00050 # 0.
00051 # ------------------------------------------------------
00052 $rndDir = rand(4) % 4;
00053 if ($rndDir == 0) {
00054   $yStart = (rand($maxY));
00055   $xStart = $maxX;
00056 }
00057 elsif ($rndDir == 1) {
00058   $yStart = (rand($maxY));
00059   $xStart = 0.0;
00060 }
00061 elsif ($rndDir == 2) {
```

```
00062   $yStart = $maxY;
00063   $xStart = (rand($maxX));
00064 }
00065 else {
00066   $yStart = 0.0;
00067   $xStart = (rand($maxX));
00068 }
00069
00070 while ( ($rndDest = rand(4) % 4) == $rndDir) {}
00071
00072 if ($rndDest == 0) {
00073   $yEnd = (rand($maxY));
00074   $xEnd = $maxX - 0.01;
00075 }
00076 elsif ($rndDest == 1) {
00077   $yEnd = (rand($maxY));
00078   $xEnd = 0.01;
00079 }
00080 elsif ($rndDest == 2) {
00081   $yEnd = $maxY - 0.01;
00082   $xEnd = (rand($maxX));
00083 }
00084 else {
00085   $yEnd = 0.01;
00086   $xEnd = (rand($maxX));
00087 }
00088
00089 print "\$node_($nodes) set X_$xStart\n";
00090 print "\$node_($nodes) set Y_$yStart\n";
00091 print "\$node_($nodes) set Z_0.000000000000\n";
00092 print "\$ns_at 0.000000000000 \"\$node_($nodes) setdest $xEnd $yEnd
00093                                          $speed\"\n";
00094
```

Main TCL Control Script for NS-2 Simulator

```
00001 # Copyright (c) 1997 Regents of the University of California.
00002 # All rights reserved.
00003 #
00004 # ----------------------------------------------------
00005 # Eric Reeves
00006 # Department of Computer Science
00007 # Texas State University - San Marcos
00008 # er40634@txstate.edu
00009 #
00010 # Filename:  sensors.tcl
00011 #
00012 # Description:  Main control script for NS-2 Simulator.
00013 # This particular instance of the script enables a 20m
```

```
00014 # communication range and runs for 100 seconds.
00015 # ---------------------------------------------------
00016 #
00017 # simple-wireless.tcl
00018 # A simple example for wireless simulation
00019
00020 # ========================================================================
00021 # Define options
00022 # ========================================================================
00023 set val(chan)           Channel/WirelessChannel    ;# channel type
00024 set val(prop)           Propagation/TwoRayGround   ;# radio-propagation
00025         model
00026 set val(netif)          Phy/WirelessPhy            ;# network interface
00027         type
00028 set val(mac)            Mac/802_11                 ;# MAC type
00029 set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
00030 set val(ll)             LL                         ;# link layer type
00031 set val(ant)            Antenna/OmniAntenna        ;# antenna model
00032 set val(ifqlen)         100                         ;# max packet in ifq
00033 set val(nn)             21                         ;# number of mobilenodes
00034 set val(rp)             DSDV                       ;# routing protocol
00035 set val(sc)             "./scen-nodes"
00036 set val(sc_intruder)    "./scen-single-straight-intruder"
00037 set val(header)   "./header"
00038 set opt(overlap)  0
00039
00040 source $val(header)
00041
00042 set coverage [HalfCellCoverage 0 0 3600 3600 $opt(overlap)]
00043 set power [SetPt $coverage]
00044 puts "power calculated:  $power"
00045
00046 Phy/WirelessPhy set Pt_$power
00047
00048 # ========================================================================
00049 # Main Program
00050 # ========================================================================
00051
00052
00053 #
00054 # Initialize Global Variables
00055 #
00056
00057 set ns_   [new Simulator]
00058 $ns_use-newtrace
00059
00060 set tracefd     [open sensor.tr w]
00061 $ns_trace-all $tracefd
00062
```

```
00063 set nf [open out.nam w]
00064 $ns_namtrace-all $nf
00065
00066 # set up topography object
00067 set topo       [new Topography]
00068
00069 $topo load_flatgrid 150 50
00070
00071 #
00072 # Create God
00073 #
00074 #create-god $val(nn)
00075
00076 set  god_[create-god $val(nn)]
00077
00078 #
00079 #  Create the specified number of mobilenodes [$val(nn)] and "attach" them
00080 #  to the channel.
00081 #  Here two nodes are created :  node(0) and node(1)
00082
00083 # configure node
00084
00085         $ns_node-config -adhocRouting $val(rp) \
00086        -llType $val(ll) \
00087        -macType $val(mac) \
00088        -ifqType $val(ifq) \
00089        -ifqLen $val(ifqlen) \
00090        -antType $val(ant) \
00091        -propType $val(prop) \
00092        -phyType $val(netif) \
00093        -channelType $val(chan) \
00094        -topoInstance $topo \
00095        -agentTrace ON \
00096        -routerTrace ON \
00097        -macTrace OFF \
00098        -movementTrace OFF
00099   $ns_node-config -rxPower $power -txPower $power
00100
00101   for {set i 0} {$i < $val(nn) } {incr i} {
00102     set node_($i) [$ns_node]
00103     $node_($i) random-motion 0    ;# disable random motion
00104   }
00105
00106 #
00107 # Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes
00108 #
00109 #
00110
00111 puts "Loading sensor scenario file..."
```

```
00112 source $val(sc)
00113 puts "Loading intruder scenario file..."
00114 source $val(sc_intruder)
00115
00116 set j [expr $val(nn)-1]
00117
00118 set sink_($j) [new Agent/TCPSink]
00119 $ns_attach-agent $node_($j) $sink_($j)
00120
00121 for {set i 0} {$i < $j } {incr i} {
00122
00123   #set tcp_($i) [new Agent/TCP]
00124   #$tcp_($i) set class_2
00125   #$ns_attach-agent $node_($i) $tcp_($i)
00126
00127   #$ns_connect $tcp_($i) $sink_($j)
00128
00129   set ping_($i) [new Agent/Ping]
00130
00131   #$ns_attach-agent $node_($i) $ping_($i)
00132
00133         $node_($i) attach $ping_($i) 9999
00134         set ll_($i) [$node_($i) set ll_(0)]
00135         $ns_at 0.1 "$ping_($i) set-ll $ll_($i)"
00136
00137   #$ns_at 1.0 "$ping_($i) bcast"
00138 }
00139
00140 #
00141 # Tell nodes when the simulation ends
00142 #
00143 for {set i 0} {$i < $val(nn) } {incr i} {
00144     $ns_at 100.0 "$node_($i) reset";
00145 }
00146 $ns_at 100.0 "stop"
00147 $ns_at 100.01 "puts \"NS EXITING...\" ; $ns_halt"
00148 proc stop {} {
00149     global ns_tracefd nf
00150     $ns_flush-trace
00151     close $tracefd
00152     close $nf
00153     exit 0
00154 }
00155
00156 puts "Starting Simulation..."
00157 $ns_run
00158
00159
```

Broadcast / Convergence Script

```
00001 #!/usr/bin/perl
00002 # ------------------------------------------------------
00003 # Eric Reeves
00004 # Department of Computer Science
00005 # Texas State University - San Marcos
00006 # er40634@txstate.edu
00007 #
00008 # Filename:  bcast.pl
00009 #
00010 # Description:  This script parses the sensor.tr trace
00011 # file left by the NS-2 simulator.  It specifically looks
00012 # for broadcast events maintaining a list of sensors,
00013 # if they've received the original message (the intruder
00014 # information) and the ID of the latest packet received.
00015 # The network is considered to be converged on the intruder
00016 # data once all of the sensors have received the original
00017 # message.
00018 # ------------------------------------------------------
00019
00020 $nodes  = $ARGV[0];
00021 $start  = int(rand($ARGV[0] - 1));
00022 $bCount = 0;
00023 @trArr  = ();
00024 @sArray = ();
00025 @pArray = ();
00026
00027 # Initialize the sensor and packet ID arrays.
00028 for ($i = 0; $i <= $nodes; $i++) {
00029   $sArray[$i] = 0;
00030   $pArray[$i] = 0;
00031 }
00032
00033 open FILE, "<sensor.tr";
00034 while (<FILE>) {
00035   if ($bCount < $nodes) {
00036   $bCount = 0;
00037   @trArr = split();
00038
00039   # Legend:
00040   # -------
00041   # $trArr[0]:  s/r (Send / Receive)
00042   # $trArr[2]:  (Time of Event)
00043   # $trArr[30]:  -Is (Sending Node)
00044   # $trArr[32]:  -Id (Destination Node)
00045   # $trArr[40]:  -Ii (Packet ID)
00046
```

```
00047   # If the intruder is detected, flag the detector and add the packet id.
00048   if ($trArr[0] eq "r"&& ($trArr[30] eq "$start.255"||$trArr[32] eq
00049       "$start.255")) {
00050     ($sender, $sndPort) = split(/\./, $trArr[30]);
00051     ($receiver, $recvPort) = split(/\./, $trArr[32]);
00052
00053     if ($sender eq "$start") {
00054       #print "sender:  $trArr[32], packet:  $trArr[40]\n";
00055       $sArray[$trArr[32]] = 1;
00056       $pArray[$trArr[32]] = $trArr[40];
00057     } else {
00058       # $receiver eq "$start"
00059       #print "receiver:  $trArr[30], packet:  $trArr[40]\n";
00060       $sArray[$trArr[30]] = 1;
00061       $pArray[$trArr[30]] = $trArr[40];
00062     }
00063   }
00064   elsif ($trArr[0] eq "r"&& $trArr[30] ne "$start.255"&& $trArr[32] ne
00065         "$start.255") {
00066     ($sender, $sndPort) = split(/\./, $trArr[30]);
00067     ($receiver, $recvPort) = split(/\./, $trArr[32]);
00068
00069     if ($sArray[$sender] == 1 && $trArr[40] >= $pArray[$sender]) {
00070       $sArray[$trArr[32]] = 1;
00071       $pArray[$trArr[32]] = $trArr[40];
00072     }
00073   }
00074   elsif ($trArr[0] eq "s") {
00075     ($sender, $sndPort) = split(/\./, $trArr[30]);
00076     ($receiver, $recvPort) = split(/\./, $trArr[32]);
00077
00078     if ($receiver eq "$start") {
00079       #print "receiver:  $trArr[32], packet:  $trArr[40]\n";
00080       $sArray[$trArr[32]] = 1;
00081       $pArray[$trArr[32]] = $trArr[40];
00082     }
00083   }
00084
00085   for ($i = 0; $i <= $nodes; $i++) {
00086     if ($sArray[$i] == 1) {
00087       $bCount++;
00088     }
00089   }
00090   if ($bCount == $nodes) {
00091     $toc = $trArr[2];
00092   }
00093   }
00094
00095 }
```

```
00096 close FILE;
00097
00098 $bCount = 0;
00099 for ($i = 0; $i <= $nodes; $i++) {
00100   if ($sArray[$i] == 1) {
00101     $bCount++;
00102   }
00103 }
00104
00105 printf("%.2f, %.2f\n", ($bCount/$nodes) * 100, $toc);
```

Main Perl Simulation Control Script

```
00001 #!/usr/bin/perl
00002 # ------------------------------------------------------
00003 # Eric Reeves
00004 # Department of Computer Science
00005 # Texas State University - San Marcos
00006 # er40634@txstate.edu
00007 #
00008 # Filename:  loop.pl
00009 #
00010 # Description:  Runs the number of simulations provided
00011 # at the command line.  This script is tailored to
00012 # determine the time of convergence when the broadcast
00013 # rate is 100%.  At the end of all simulations the final
00014 # statistics are displayed.
00015 # ------------------------------------------------------
00016
00017 $limit     = $ARGV[0];
00018 $node_num  = $ARGV[1];
00019 $x_size    = $ARGV[2];
00020 $y_size    = $ARGV[3];
00021 $time_incr = 1.00;
00022 $sim_time  = 12502.0;
00023 $i = 0;
00024 $hits = 0;
00025 $bcast_total = 0;
00026 $bcast_toc = 0;
00027 $bcast_floor = 0;
00028
00029 if ($#ARGV != 3) {
00030   print "Usage:  ./loop.pl limit numNodes maxX maxY\n";
00031   exit;
00032 }
00033
00034 while ( $i < $limit ) {
00035   $setdest_cmd = "./sensors.pl  $node_num $sim_time $x_size $y_size > scen-
00036                  nodes";
```

```
00037   $intruder_cmd = "./intruder.pl $node_num $sim_time $time_incr $x_size
00038                    $y_size > scen-single-straight-intruder";
00039
00040   $setdest_cmd_status = `$setdest_cmd`;
00041   $intruder_cmd_status = `$intruder_cmd`;
00042
00043   $tcl_file = "sensor.tcl";
00044
00045   $ns_cmd = "ns ".  $tcl_file;
00046   $ns_cmd_status = `$ns_cmd`;
00047   print $ns_cmd_status .  "\n";
00048   $trace_file = "sensor.tr";
00049
00050   #$bcast_status = `./bcast.pl $node_num`;
00051   @bcast_status = split(/,/, `./bcast.pl $node_num`);
00052
00053   $wc_str = "cat ".  $trace_file .  " |grep \"^r\" |grep \"Ni ".  $node_num
00054             .  "\" |grep -v \"Nx ".  ($x_size - 0.01) .  "\" |grep -v \"Ny
00055             ".  ($y_size - 0.01) .  "\" |wc -l";
00056   $wc_l = `$wc_str`;
00057   if ( $wc_l > 0) {
00058     $hits++;
00059   }
00060
00061   $i++;
00062   print "Iteration $i:  hits/limit=$hits/$limit\nBroadcast:  $bcast_status[0]
00063                                ";
00064   print "Convergence Time:  $bcast_status[1]\n";
00065   $bcast_total += $bcast_status[0];
00066   $bcast_toc += $bcast_status[1];
00067
00068   if ("$bcast_status[0]"eq "100.00") {
00069     $bcast_floor++;
00070   }
00071 }
00072
00073 printf("\nFinal status:  %.2f\n", $bcast_total/$limit);
00074 printf(" TOC: %.2f\n", $bcast_toc/$bcast_floor);
```

REFERENCES

[1] J Aslam, Z. Butler, F. Constantine, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. *Proceedings of ACM International Conference on Embedded Sensor Systems(SenSys)*, pages 150–161, 2003.

[2] Gregory Chockler, Murat Demirbas, Seth Gilbert, Nancy Lynch, Calvin Newport, and Tina Nolte. Reconciling the theory and practice of (un)reliable wireless broadcast. *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops*, 2005.

[3] Thomas Clouqueur, Veradej Phipatanasuphorn, Ramanathan Parameswaran, and Kewal K. Saluja. Sensor deployment strategy for target detection. *WSNA '02*, pages 42–48, September 2002.

[4] Thomas Clouqueur, Veradej Phipatanasuphorn, Parameswaran Ramanathan, and Kewal K. Saluja. Sensor deployment strategy for detection of targets traversing a region. *Mobile Networks and Applications*, 8:453–461, 2003.

[5] Fei Dai and Jie Wu. Efficient broadcasting in ad hoc wireless networks using directional antennas. *IEEE Transactions on Parallel and Distributed Systems*, 17(4), April 2006.

[6] Arjan Durresi, Vamsi Paruchuri, S. Sitharama, and Rajgopal Kannan. Optimized broadcast protocol for sensor networks. *IEEE Transactions on Computers*, pages 1013–1025, August 2005.

[7] Kevin Fall and Kannan Varadhan. The ns manual, May 2005.

[8] Yong Gao, Kui Wu, and Fulu Li. Analysis on the redundancy of wireless sensor networks. *WSNA'03*, September 2003.

[9] Guanghui He and Jennifer Hou. Tracking targets with quality in wireless sensor networks. *Proceedings of the 13th IEEE International Conference on Network Protocols*, 2005.

[10] Chih-Shun Hsu, Jang-Ping Sheu, and Yen-Jung Chang. Efficient broadcasting protocols for regular wireless sensor networks. *Proceedings of the 2003 International Conference on Parallel Processing*, 2003.

[11] Chi-Fu Huang and Yh-Chee Tseng. The coverage problem in a wireless sensor network. *WSNA'03*, September 2003.

[12] Jorjeta Jetcheva, Yih-Chun Hu, and David Maltz. A simple protocol for multicast and broadcast in mobile ad hoc networks. *Internet Draft: draft-ietf-manetsimple -mbcast-01.txt*, July 2001.

[13] Qun Li, Michael De Rosa, and Daniela Rus. Distributed algorithms for guiding navigation across a sensor network. *MobiCom '03*, pages 313–325, September 2003.

[14] X. Li, P. Wan, and P. Frieder. Coverage in wireless sensor networks. *IEEE Trans. on Computers*, 52(6):753–763, June 2003.

[15] Stephanie Lindsey and Cauligi S. Raghavendra. Energy efficient broadcasting for situation awareness in ad hoc networks. *Proceedings of the 2001 International Conference on Parallel Computing*, 2001.

[16] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. *WSNA '02*, pages 88–97, September 2002.

[17] L. Orecchia, A. Panconesi, C. Petrioli, and A. Vitaletti. Localized techniques for broadcasting in wireless sensor networks. *DIALM-POMC '04*, pages 41–50, October 2004.

[18] J. Pan, Y.T. Hou, L. Cai, Y. Shi, and S.X. Shen. Topology control for wireless sensor networks. *Proceedings of the ACM MobiCom*, pages 286–299, 2003.

[19] Sung Park, Andreas Savvides, and Mani B. Srivastaa. Simulating networks of wireless sensors. *Proceedings of the 2001 Winter Simulation Conference*, 2001.

[20] Wei Peng and Xi-Cheng Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 129–130, 2000.

[21] Wuxu Peng. Proposal on detectability in incomplete sensor networks. 2005.

[22] Adrian Perrig, John A. Stankovic, and David Wagner. Security in wireless sensor networks. *Communications of the ACM*, pages 53–57, June 2004.

[23] Venkatesh Rajendran, Katia Obraczka, and J.J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. *SenSys '03*, pages 181–192, November 2003.

[24] Meguerdichian S., F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage problem in wireless ad hoc sensor networks. *Proceedings of the IEEE INFOCOM*, pages 1380–1387, 2001.

[25] Paolo Santi. Topology control in wireless ad hoc and sensor networks. *ACM Computing Surveys*, 37:164–194, June 2005.

[26] Andreas Savvides, Sung Park, and Mani Srivastava. On modeling networks of wireless microsensors. *Technical Report TM-UCLA-NESL-2000-11-001*, November 2000.

[27] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. *Proceedings of 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 32–41, September 2002.

[28] Robert Szewczyk, Eric Osterweil, Joseph Polastre, Michael Hamilton, Alan Mainwaring, and Deborah Estrin. Habitat monitoring with sensor networks. *Communications of the ACM*, 47:34–40, June 2004.

[29] Giacomino Veltri, Qingfeng Huang, Gang Qu, and Miodrag Potkonjak. Minimal and maximal exposure path algorithms for wireless embedded sensor networks. *SenSys '03*, pages 40–50, November 2003.

[30] Atul Verma, Hemjit Sawant, and Jindong Tan. Selection and navigation of mobile sensor nodes using a sensor network. *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications*, 2005.

[31] Wikipedia. Sine waves.

[32] Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. *MOBIHOC '02*, pages 194–204, June 2002.

[33] Ting Yan, Tian He, and John A. Stankovic. Differentiated surveillance for sensor networks. *SenSys '03*, pages 51–62, November 2003.

[34] W Zhang and G Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. *Proceedings of IEEE INFOCOM 2004*, 2004.

VITA

Eric Daniel Reeves was born in Kerrville, Texas on December 17, 1978, the son of Daniel Wayne and Linda Reeves. Eric attended high school at Huntsville High School in Huntsville, Texas where during his senior year he completed his first Computer Science courses. After graduating from high school in May of 1997 he attended Southwest Texas State University the following Fall semester. After his first two semesters at Southwest Texas State Eric transfered to the University of Texas at Austin where he completed the requirements for the degree of Bachelor of Arts in Computer Science. Between August 2002 and January 2004 Eric pursued the Cisco Certified Network Associate certification. Between January 2004 and May 2006 he was entered in the Graduate College of Texas State University-San Marcos. Currently he works as Senior Network Engineer at one of the largest employers of technically-based students in San Marcos.

Permanent Address: 1805 N IH 35, Apt 18G

San Marcos, Texas 78666

This thesis was typed by Eric Daniel Reeves.