INTELLIGENT ACTOR MOBILITY IN WIRELESS SENSOR AND ACTOR NETWORKS

A Dissertation
Presented to
The Academic Faculty

by

Sita Srinivasaraghavan Krishnakumar

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2008

INTELLIGENT ACTOR MOBILITY IN WIRELESS SENSOR AND ACTOR NETWORKS

Approved by:

Dr. Randal T. Abler, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. John A. Copeland
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Henry L. Owen, III
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Elliot Moore, II
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Kevin Haas
School of Civil and Environmental Engineering
Georgia Institute of Technology

Date Approved: May 14, 2008

To my parents

Mr. A.N. Srinivasa Raghavan & Mrs. S. Aarthi Raghavan

and my daughters

Sapna Krishnakumar & Vibha Krishnakumar

ACKNOWLEDGEMENTS

I would like to begin by thanking Dr. David Hertling, Associate Chair for Graduate Affairs Emeritus who believed in me and gave me an admission to the ECE Department, even though I had a Computer Science background. Else, I would have missed the honor of being associated with this great institution.

I am deeply indebted to my advisor Dr. Randal Abler, who took me under his wings and nurtured my ideas. He has been a true advisor in letting me identify and solve my problems, but setting boundaries when I got carried away. This has enabled me to finish my work in due time. He has also never doubted my work ethic when I had to tend to my children in times of need or otherwise. I would like to extend my sincere thanks to Dr. John Copeland, Dr. Henry Owen and Dr. Elliot Moore for graciously agreeing to serve on my committee. They have been active participants by helping to establish the expectations from my thesis. I would also like to thank Dr. Kevin Haas from the School of Civil and Environmental Engineering, who readily accepted to be on my defense committee. Many thanks are due to Dr. Benjamin Klein, who readily allowed the use of his node cluster. The simulations presented in this thesis would not have been possible otherwise.

My foremost advocates are my parents. I was raised in Neyveli - a small coal mining town in the southern state of Tamilnadu, India. My father is an honest, hard-working, self-made man to whom education came the hard way. My mom is the most benevolent person I know, who was denied education because she was a girl. Together, they educated their children - never once differentiating between me and my brother, instilled the importance of education, and, gave us the most memorable childhood. Their dream was to see me as a medical doctor, but I chose otherwise. A promise made to them as a teenager to earn a doctorate, grew with me to become my dream and has brought me to where I am today.

My dream of getting a PhD would not have become a reality if not for my husband, my rock, my Krishna - a true partner in every sense - a man many a woman wished they could

have by their side, but I am proud to say is by me! He has always been there to lean on to and I look forward to continuing my journey of life with him.

Also with me through this journey have been my adorable kids and my dear brother. My children have been a welcome distraction when I have been stumped and an inspiration with their determination as they mastered their many firsts! My brother is a constant presence in my life and was instrumental in my coming to the United States for higher education. He has been there whenever I needed him and his confidence in me is reassuring.

There are so many other people who have helped me, believed in me and constantly encouraged me, for which I am grateful. My lab mate Lane Thames has always been approachable, especially when I needed help with hardware. My baby-sitter, Jennifer Wojcian is such a gifted gal - I am lucky to have found someone like her. Other steadfast supporters are Krishna's uncle and aunt - Dr. & Mrs. Parthasarathy - who embraced me as their own niece, Mr. & Mrs. Raman who are surrogate parents to me and Gayathri Radhakrishnan, a dear friend.

Though this journey has been long, I have realized that my life is truly a blessing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Wireless sensor and actor networks are composed of sensor nodes that are low-cost, low-power devices with limited sensing, processing, and wireless communication capabilities and actor nodes that have better processing capabilities, higher transmission power, and longer battery life. They are deployed in situations where interaction is required between a network and the environment in which the network is deployed. This thesis studies the functioning of a single mobile actor deployed in a sparsely connected network.

The presence of a single actor eliminates the need for coordination and communication between actors and a sparsely connected network eliminates the need for location management. When deployed in a sparsely connected network, the actor has to do more than acting. It has to perform the additional duties of an event collector - collecting events from the naturally occurring groups of nodes - so that it can fulfill its primary obligation as an actor. The path taken by a mobile actor node is generated by a mobility model. The existing random mobility models are non-intelligent mobility models. While they may bring about a chance meeting between an actor and an event, there is no guarantee that these meetings will actually happen. This motivates the development of intelligent mobility models for the actor node, which will generate paths that are reflective of the network in which the actor is deployed.

In this thesis, intelligent mobility models for the actor node were developed using the inherent clustering information of a sparsely connected network. These models were applied to an actor node in networks of varying sparseness and the following conclusions were reached: (i) Existing random mobility models are unsuitable for an actor in a sparsely connected network. (ii) High probability of events can be sensed when a sparsely connected network is used. (iii) 100% event detection by the actor node is possible at higher speeds. (iv) When the single actor functioned as both an event collector and an actor, the number of events acted upon by the actor was very close to the number of events acted upon by an actor in a fully connected network. (v) The Correlation Theory developed in this research suggests using a combination of the intelligent mobility models to obtain the best performance results under all circumstances. (vi) Early detection of events can be supported where it is required.

In this thesis, the main goal was to develop intelligent mobility models for an actor node in a wireless sensor and actor network to maximize timely detection of events. This has been accomplished. Also, all of the above conclusions justify (from an economical and performance standpoint) the deployment of a single actor and a sparsely connected network, either individually or as a combination.

CHAPTER I

INTRODUCTION

A wireless sensor and actor network (WSAN), as the name suggests, comprises sensor nodes and actor nodes. The sensor nodes are low-cost, low-power devices with limited sensing, processing, and wireless communication capabilities. They can be deployed in harsh environmental conditions and are prone to node failures and communication failures. Hence they are deployed in large numbers. The actor nodes are resource rich with higher processing capabilities and longer battery life. There are fewer actor nodes than sensor nodes in an environment. Figure 1 represents a simple WSAN with a sink that is present to take care of overall communication and coordination. WSAN can be considered an extension to a wireless sensor network (WSN) which is composed entirely of sensor nodes and is deployed in situations where there is no need for intervention. A WSAN on the other hand, is used in situations where there is need for interaction between the network and the environment in which the network is deployed.

When an event occurs, the sensor nodes which sense the event can send the event detection information directly to the actor(s). Or the sensor nodes may send the information to a sink, which processes the information and then selects one or more actors to act. The former model is referred to as an automated architecture, while the latter model is referred to as semi-automated architecture [1]. These architectures are pictorially represented in Figure 2.

When there is a single stationary actor [1], it is assumed to be in the center of the network handling all the events that occur around it. The nodes in the immediate vicinity of the actor are known as forwarding nodes, as any information from the network reaches the actor through one of these nodes. These forwarding nodes are overburdened and deplete in energy

[1] Actor and actor node are used interchangeably to describe the physical entity which acts on the sensed information. Similarly, sensor and sensor nodes are also used interchangeably throughout this thesis.

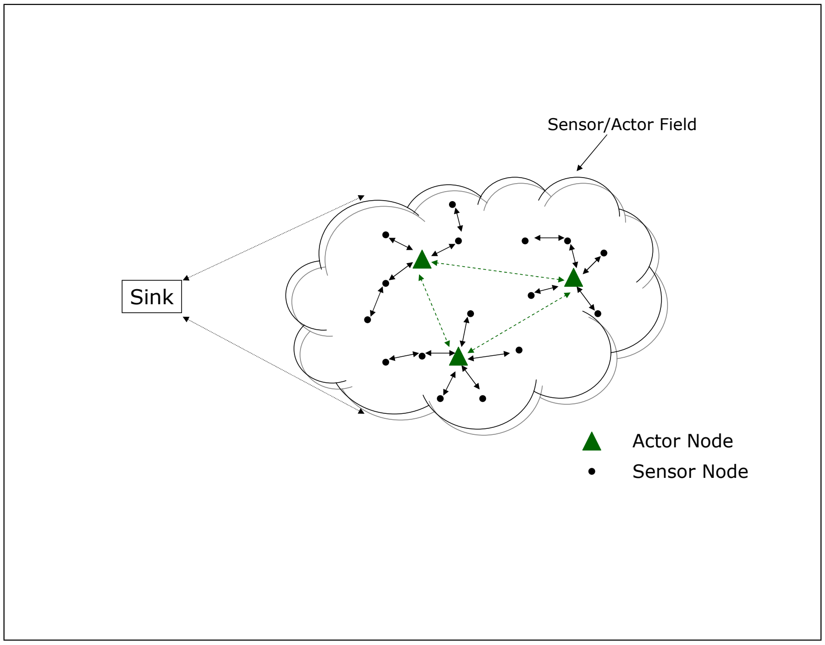

Figure 1: Example of a wireless sensor and actor network.

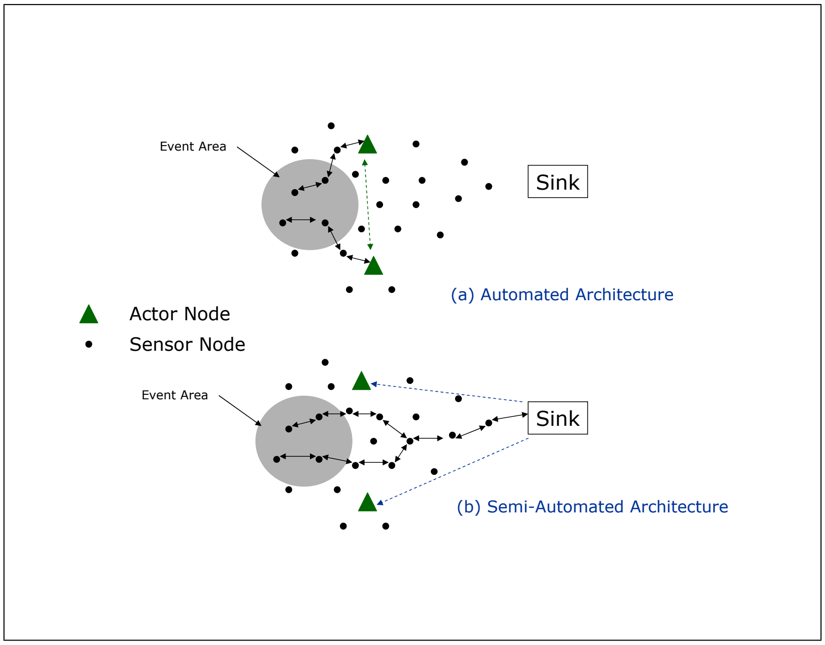

Figure 2: Architectures in wireless sensor and actor networks.

more quickly than other nodes in the network. Depending on the number of events and their location, these forwarding nodes may ultimately die. The actor is then disconnected from the rest of the network, as there is no path for the event detection information to reach the single stationary actor. Existing research deals with this problem in two ways:

- Using more than one stationary actor [19]: Events are generally assumed not to occur at the same place every time. Each of the events is then sensed by a different set of sensor nodes and different actors are chosen to act. This assures that the events in a field can be handled without any disruption. When there is more than one actor (stationary or mobile), there are a new set of issues that have to be dealt with. When a sensor node is near two actors, it has to decide which actor to send the event detection information to. Similarly, if more than one actor has received the same information from two different sets of sensor nodes, they have to communicate amongst themselves as to which one will take action. Else there will be duplication of action.

- Using a mobile actor [18], [20]: The primary advantage of a mobile actor is that it can get as close as possible to an event to take action. Also when the actor is mobile, any node in the network can be a forwarding node, thereby distributing the load throughout the network. If the mobile actor is deployed in a fully connected network, the path taken by a mobile actor is not critical since event detection information will always reach the actor. If there is more than one mobile actor deployed, then the issues with duplication of action remain. Having a mobile actor in a WSAN is similar to different mobile entities introduced in traditional WSNs to connect sparse networks and to mitigate network partitioning [25], [11], [23], [3], [21], [10], [8], [7]. But, the requirements for a WSAN are different when compared to a WSN [1].

A fully connected network is ideal regardless of whether the actor is stationary or mobile. All the sensor nodes then have one or more assured communication path(s) to the actor node for event detection information. A fully connected network can be formed using radios with longer transmission range or with dense sensor node deployment. The former may not result in a very energy efficient solution and the latter may not be the preferred deployment choice all the time. It is more challenging when a mobile actor is deployed in a sparsely connected network. A sparsely connected network may be formed in two ways:

- Limited transmission range on sensor nodes: The transmission range of a node is the diameter of a circle centered on the node location. A lower transmission range extends the longevity of the sensor node. Fewer numbers of nodes, deployed in a random manner generally produces a sparsely connected network. But by giving a high transmission range on the sensor nodes, a sparsely deployed network may be made into a fully connected network.

- Depletion of resources: Resources of the sensor nodes deplete with regular communication and may drain as a result of event monitoring and event detection. Over a period of time, a fully connected network may become a sparsely connected network.

In a sparsely connected network there are naturally occurring groups of nodes, known as clusters. Any event that occurs in a cluster is sensed by nodes in that cluster and data propagation is bounded by the cluster. Hence there is no need for complex routing protocols and a simple flooding algorithm may not be very expensive to use. Also, when a mobile actor is deployed in a fully connected network, the location of the actor has to be known so that the sensor nodes know the direction in which to forward the data. Sending information in the direction of the actor conserves resources on the sensor nodes when compared to flooding. This is known as location management and is not required in a sparsely connected network. Therefore, the advantages of a sparsely connected network are: (a) it can be randomly deployed (b) there is no need for location management and, (c) it is economical as fewer sensor nodes are deployed compared to a dense network. The disadvantages are that data is restricted to clusters and 100% detection cannot be guaranteed in the deployment area.

Similarly there are a number of advantages and disadvantages in deploying a single actor. The advantages are: (a) there is no need for communication between actors (b) there is no need for coordination amongst the actors when an event happens, and (c) actor nodes being expensive, keeping it to just one in a region is economical. The disadvantages are: (i) the actor can act on only one event at any given time (ii) if the network is sparsely connected, then the actor cannot be moving randomly and its path has to be modeled so that it can perform its duties efficiently.

Assuming that a single actor is sufficient to act upon an event, this research focuses on the case of a single actor in a sparsely connected network. The area assigned to an actor can also be considered a part of a larger network that has been singularly designated to this actor. The reason to study this combination is the expectation that the benefits outweigh the deficiencies:

- Demand of applications: There are many applications where a single actor in a sparsely connected network can be used. Example applications include, but are not limited to coastal monitoring, environmental monitoring and border patrol where sensor nodes are deployed to monitor specific characteristics or events. In the case of coastal monitoring, the events occur in specific regions like the coastline or shallow waters where sensor nodes mounted on buoys can be deployed. A probable actor is a manned boat which collects data from the sensor nodes and takes action if necessary.

- Resource conservation: Coordination and communication between actors consumes actor's resources and is eliminated with the presence of just one actor. Similarly, location management consumes energy on the sensor nodes and is unnecessary in a sparsely connected network. By eliminating these resource consuming processes, the actor's and sensor nodes' resources can be utilized towards performing their duties.

- Cost effectiveness: If it can be shown that an actor in a sparsely connected network can sufficiently handle events in an area, a sparse network with a single actor costs less to deploy than a fully connected network with one or more actors.

This combination of a single actor in a sparsely connected network can succeed only if there are competent mobility models. Therefore, the objective of this thesis is to generate intelligent mobility models for the actor in a sparsely connected network such that the number of events detected in a timely manner is maximized. There are many challenges in devising these mobility models:

1. The algorithms to generate the mobility models should support the evolving nature of wireless sensor networks.

2. Timely detection of events should be supported as required in WSANs. An event has to be occurring when the actor reaches the event location, so that the action taken by the actor is valid.

3. The algorithms have to model dynamic actor mobility. This means that the destinations for the actor to visit have to be calculated on the fly and should not be limited by the area of the network or the network structure.

4. The algorithms should support a wide variety of application scenarios with parameters that can be varied easily.

5. There has to be an easy way to test these mobility models. Extensive simulations have to be run to confirm if the intelligent mobility models are indeed effective in the chosen scenarios.

Some of the fundamental assumptions of this research are: (i) use of automated architecture, (ii) knowledge of sensor node locations, and (iii) stationary, fixed duration events. Some of the assumptions that are modifiable and can be relaxed in future work are: (i) single actor, (ii) constant actor speed, and (iii) fixed transmission power on sensor and actor nodes.

CHAPTER II

EXISTING RELATED MOBILITY MODELS AND MOBILE ENTITIES

The development of wireless sensor networks was originally motivated by military applications such as battlefield surveillance. However, they are now used in many civilian application areas such as habitat monitoring, healthcare applications, forest fires, and traffic control. The sensor nodes have a processing unit with limited computational power and memory, a sensing circuit, a communication device, and a power source, usually in the form of a battery. Because of limited resources and the harsh environmental conditions in which they are deployed, the sensor nodes are prone to node failures and communication failures. Hence they are deployed in large numbers. The sensor nodes sense the environment and forward the information to a sink, which collects the information and relays it to an end user.

A wireless sensor and actor network can be considered as a specialized WSN with the addition of resource-rich actor nodes that have better processing capabilities, higher transmission power, and longer battery life. Since actors have higher capabilities and can act on large areas, they are fewer in number compared to the number of sensor nodes in an environment. The actor is present in the network to take action based on the sensed information received from the sensor nodes. The data received by the actor should be valid at the time of reception for the action taken by the actor to be appropriate. For example, in the case of intruder detection, the intruder must still be in an area of reach when the actor gets the information to be able to take action. Other situations where a WSAN can be deployed require that event detection information reach the actor at the earliest possible time for a response to be taken. An example of this is a fire monitoring application. When there is a fire, the action of putting out the fire needs to be started by the actor as soon as possible to keep damage to a minimum. Hence, a WSAN has the additional requirement of event detection data reaching the actor in a timely fashion.

Another network of interest is a mobile ad-hoc network (MANET). MANET, as the name implies, is a network composed of wireless mobile nodes that are set up in an ad-hoc fashion. These mobile nodes cooperate to maintain network connectivity and perform routing functions. When the mobile nodes are sparsely distributed, network partitions can arise that last for a significant period of time. Sparse networks arise in natural situations like disaster scenarios. In such cases there is need for data collection from nodes that can maintain network connectivity only by using their long-range transmission radios.

2.1 Existing Random Mobility Models

The commonly used mobility models in WSAN and WSN research are the random walk mobility model, random waypoint mobility model, or a variation of the two. In the case of the random walk model, the actor moves towards a random destination at a certain speed. On reaching the destination, it chooses yet another random destination and continues towards that destination, and so on. In the case of the random waypoint mobility model, random destinations are picked uniformly in the region and the actor moves at a selected speed which is also chosen uniformly in an interval. Upon reaching the destination, the actor pauses for a pre-determined time period, and the process repeats itself afterwards. Random mobility models are non-intelligent mobility models. They are suitable when evaluating the performance of algorithms and protocols because, the results obtained are then not tailored to specific mobility models. When using random mobility models, evaluating against a fully connected network assures that there is a path available between the sensor nodes and the actor node or the sink for information exchange.

2.2 Mobile Entities in Research

Depending on the type of network, mobile entities have been introduced for different reasons:

- In a WSN, the sensor nodes around a sink are the nodes that forward information from the periphery of the network. The forwarding nodes are overburdened and drain resources faster than the other nodes and ultimately die. This can lead to network partitioning and the sink will be unable to receive any information. Mobile sink was introduced in WSNs to mitigate this problem. Mobile element and data MULE (Mobile Ubiquitous LAN Extensions) are mobile entities that were put forth to connect sparsely connected WSNs.

- In a WSAN with a single stationary actor, the same problem with overburdened forwarding nodes can occur depending on the number of events and their location. A natural progression is the introduction of a mobile actor, which can ease this problem.
- In a sparsely connected MANET, nodes can wait passively for their own mobility to allow them to re-connect with other mobile nodes. But such an encounter between nodes can be unpredictable and rare. Message ferries are mobile nodes that have been introduced in highly partitioned MANETs to aid with data delivery.

The presence of mobile entities has other added benefits like lifetime elongation [15], better security [2], and energy efficient data gathering [9].

2.2.1 Mobile Sinks

In a densely deployed WSN with a stationary sink, the sensor nodes around the sink forward information from the entire network to the sink. These nodes become hotspots and deplete resources faster than other nodes in the network. A mobile sink can mitigate this problem, as there are now different sets of forwarding nodes along the path of the mobile sink. This leads to uniform depletion of resources along the entire network.

For a dense WSN, multiple sensor nodes have to share a single communication channel for node-to-sink transmissions. Special multi-node transmission scheduling algorithms are proposed to ensure that there is a high rate of successfully received packets [22]. The chosen mobility model for the sink is a certain velocity and direction, which are obtained from a Global Positioning System.

Another strategy for a dense WSN requires the mobile sink to move in a circular path close to the periphery of a network. The network is also represented as a circular area [15]. The sink stays for a fixed duration of time at pre-determined points in the circular path that are close to sensor node locations. A joint routing and mobility scheme is proposed where the nodes within the circular path use a shortest-path routing scheme. Those nodes in the annulus (from the circular path to the periphery of the network) follow a two-step routing: round routing to a diameter of the circle passing through the current location of the mobile sink, followed by shortest-path routing to the mobile sink. This work concludes that the joint mobility and routing scheme balances the load of the network to alleviate hotspots and that it leads to the best performance results in terms of packets received and network lifetime.

With a mobile sink, its location has to be continuously propagated throughout the sensor field to keep all the sensor nodes updated with the direction in which to forward future data. With multiple sinks, frequent location updates can lead to excessive drain of sensors' limited battery power supply as well as increased collisions in wireless transmissions. A two-tier data dissemination approach is proposed that can provide scalable and efficient data delivery to multiple mobile sinks [14]. Each data source proactively constructs a grid structure that allows the sink to flood queries to a local cell only. The sensors located at the grid points are the only ones that acquire the forwarding information. They are called dissemination nodes. A query from the sink traverses two tiers to reach a source. The lower tier is the local cell where the sink is currently located and where the query is flooded. The second tier is made of the dissemination nodes at the grid points. The query is forwarded through the dissemination nodes to reach the data source. The mobile sink moves following the random waypoint mobility model.

2.2.2 Data MULEs

The idea behind using MULEs is to help connect sparse sensor networks. The MULE is a mobile entity that collects data from sensors as it passes by. The context of its applications includes habitat monitoring and other sparsely connected WSNs. The idea is to get a system in place that will provide the connectivity while keeping energy to a minimum at the cost of latency. A three-tier architecture of sensors, MULEs, and access points is proposed that would help connected sparse WSNs [21]. The sensors buffer their data until the MULE receives it and the MULE buffers the data until it can deliver it to an access point. The path taken by the MULE follows a random walk mobility model.

In a follow-up paper, mobility is exploited for energy-efficient non-real-time data collection in sparse sensor networks [9]. The results show that minimizing the communication responsibility of the resource-constrained sensors can extend the lifetime of the network. The mobility models used for the MULE in simulations were random walk model, random waypoint mobility model, deterministic arrival (fixed route and velocity), and Poisson arrival (inter-arrival time at the sensors is exponential). High data success ratio and low latency are achieved when the MULE arrival is deterministic. The random walk model performs the worst, as the MULE is always moving toward a random destination and there is no assurance that the MULE will come in the vicinity of a sensor node to collect data.

Another approach uses MULEs that move in straight lines [10]. Only nodes that are one-hop from the MULE can communicate with the MULE as it passes along. The one hop node is like a cluster head that has data aggregated from other nodes that are connected to it. For nodes that can reach more than one MULE, a centralized load-balancing algorithm is proposed that decides which MULE the cluster head will forward the data to.

2.2.3 Mobile Elements

Mobile elements are used for data collection, once again in sparsely connected networks. Data generation rates of sensors may vary, depending on the application environment. It may be necessary that some nodes be visited more frequently than others so that data is not lost. A partition-based scheme is proposed for scheduling so that no data is lost as a result of buffer overflow [8]. Nodes are partitioned into bins that are geographically close with similar buffer overflow times. The schedule for each bin is created by solving the traveling salesman problem. The individual paths for the bins are concatenated to generate the overall path. The delay is high using this method. To alleviate the delay, messages are classified into urgent and regular messages. Urgent messages are delivered ahead to meet the deadline [7]. The speed of the mobile element is also altered to help with delivery.

2.2.4 Mobile Relays

Mobile relays have been used in MANETs and WSNs. In MANETs, relays are used to increase the throughput capacity of the network. Direct communication between sources and destinations alone cannot increase the throughput, as they are apart most of the time. The source relays the data to every node in the network but itself and every node in the network is moving randomly. Since all the nodes are mobile, one of the relay nodes may be able to reach the destination sooner than the direct transmission from the source. This method is known as multi-path diversity and increases the throughput [6]. But it incurs heavy delay and cannot be used in applications requiring timely receipt of data. Rather than letting the nodes move randomly in a two-dimensional space, an extension work restricts the mobility model to a one-dimensional great circle [4]. The circles are random but remain fixed in time. The nodes move randomly in their own great circle. The results show that the throughput of the ad-hoc network is increased even with this restricted mobility of the nodes.

In WSNs, relays are used to prolong the lifetime of the network [24]. This paper considers a dense static WSN and gives arithmetical proof that it is sufficient if a mobile relay moves within a two-hop radius circle from the static sink, to maximize the lifetime of the network. The network is considered to be a circular area with the sink at the center. Starting from the center, the mobile relay traverses a path that forms a set of concentric circles around the sink with increasing radii, until it reaches the periphery of the two-hop radius circle. It stays on each point in this path for certain duration and relays traffic to the sink. This method alleviates the problem of overburdened nodes with a mobile relay that does not have to venture farther than two hops away from the sink.

2.2.5 Message Ferries

Message ferrying is a proactive routing scheme used in highly partitioned wireless ad-hoc networks [26]. Network connectivity in MANETs is generally maintained using radios with longer communication range. This can deplete the node battery rapidly. The other way to maintain network connectivity is to depend on existing node mobility. Existing node mobility results in low data delivery rates and large delays. Hence, a mobile node acts as a ferry to reach the disconnected nodes, collect the data and forward it along. Rather than use multi-hop communication, a mobile ferry reduces all communication to a single hop at the cost of delay. Given a highly partitioned network, this scheme finds the ferry route using a traveling salesman problem approximation algorithm followed by a local optimization technique to reduce the average delay. The ferry route generated is further extended to meet the bandwidth requirements between any two nodes.

The original scheme of message ferrying, which exploited non-randomness to help deliver data, is modified to improve data delivery performance and reduce energy consumption for mobile ad-hoc networks [25]. There are two ways in which proactive movement can be initiated by the ferries or the nodes:

1. The ferry moves in a predefined path and communicates with the nodes on the predefined path. A node that may want to send or receive data intermittently moves closer to the path of the ferry so that messages can be relayed.

2. The node proactively sends a message using long-range radio to the ferry saying it wants to communicate. The ferry changes its trajectory to meet the node that sent the request. Now that the ferry is nearby, the node communicates using short-range radio, which is less expensive. The ferry then continues on its predefined path.

A further improvement to the original scheme is the introduction of a power management framework [11]. The nodes are stationary or mobile and in different states like transmit, receive, idle, doze, and off. The ferry can be on a tight schedule or on a loose schedule. When the ferry is on a tight schedule, the node knows when the ferry is going to come its way, so it sleeps to conserve power. When the ferry is on a loose schedule, the node does not know when the ferry will come its way, but knows when it is definitely out of its transmission range. If the nodes are awake to receive a beacon from the ferry, they respond and pass the message along to the ferry. By keeping the nodes in less energy-consuming states for most of the time, this framework tries to conserve energy. The trade-off is that a ferry that is passing by may be missed, which can lead to delay in the delivery of messages.

The delay is managed by classifying messages into two types: urgent and regular [23], [3]. Urgent messages are delivered first to keep within the delay bounds as required by the application. Buffer restrictions can also be introduced on the nodes and the ferry. An elliptical forwarding scheme is designed that deals with buffer contention while aiding urgent messages to meet the deadline [3].

2.3 Mobile Actors in the WSAN area

Unlike WSNs and MANETs where mobile entities have been researched for the past few years, the concept of a mobile actor is relatively new in the field of WSANs. The requirements of a WSAN are different from those of a WSN [1]. It is essential that event sensing information be communicated to the actor at the earliest possible time. Since a centralized sink may not be present in a WSAN, coordination and communication between the actors and between the sensors and the actors have to be managed.

The coordination and communication problem has been studied and a hybrid location management scheme has been proposed to handle the mobility of actors along with a geographical routing algorithm for sensor-actor communication [18]. With a mobile actor, the location of the actor has to be known to the sensors, so the sensors can send information in the direction of the actor. The hybrid location management scheme has actors broadcasting updates, limiting their scope based on Voronoi diagrams. The sensors then predict the movement of the actors based on Kalman filtering of previously received updates. This scheme is shown to consistently reduce the energy consumption on sensors by avoiding over 75% of location updates. The sensor-actor communication uses forwarding rules based on geographical position in the presence of Rayleigh fading channels. An energy-efficient forwarding distance is derived in the presence of a fast fading channel. Then, the end-to-end delay is decreased by increasing the transmit power. Increasing the transmit power increases the forwarding range. The objective behind this is to trade off energy consumption for latency when the data has to be delivered within a given time bound. Coordination between actors is achieved by selecting a team of actors and their velocity to optimally divide the action workload. Additionally, the energy required to complete the action within the stipulated time is also minimized. A congestion-control algorithm is also proposed where multiple actors are forced to share the traffic generated in the event area. In simulations, the actor mobility is following the random waypoint mobility model to show the effectiveness of the hybrid location management scheme. Though this work assumes that the action and movement energy of the actor are orders of magnitude higher than the energy required for communication purposes, it does not analyze the impact of random actor movement on the movement energy. From the algorithms and simulations, the network under consideration seems to be fully connected.

Another research paper that deals with mobile actors proposes a real-time coordination and routing framework for sensor-actor coordination to achieve energy-efficient and reliable communication [20]. This framework configures sensors to form hierarchical clusters and elect a cluster head based on different application parameters. Only cluster heads are used to coordinate with actors to save energy. A backbone network is created by integrating the forward tracking and backtracking mechanisms, which provide all possible routes between the sensor nodes and actor nodes. The cluster head chooses the path to be taken by a packet so that it will reach the actor within a given delay bound. The actor is considered to be a location-aware mobile node, which knows the number of sensor nodes it is serving at any given point in time. The mobility model of the actor used in simulations is random walk. The metrics studied are the deadline-miss ratio and the average packet delay. In simulations with a mobile actor, even with sufficient delay, the deadline-miss ratio does not reach zero. This means either the actor did not receive the packet or there is no path for the packet to reach the actor. This work focuses on routing the packets within a given delay bound rather than controlling the mobility of the actor to aid in the reception of packets within a delay bound.

CHAPTER III

INTELLIGENT MOBILITY MODELS

As seen in Chapter II, existing research in the WSAN area deals mostly with a fully connected network. An example of a fully connected network of 500 nodes is shown in Figure 3. Due to dense deployment of sensor nodes, even with a limited transmission range on the sensor nodes this will result in a network where every node is reachable by every other node. When a network is fully connected, it guarantees one or more path(s) between the sensor nodes and the actor node for event detection information.

Figure 4 shows an example network of 100 uniformly distributed, randomly generated node locations in a 200 m x 200 m environment. Labels 0-99 represent the 100 sensor nodes, label 100 represents the actor node, and labels 101-110 represent the events. With a limited transmission range of 20 meters on the sensor nodes, Figure 5 shows the clusters formed by the sensor nodes. This is not a fully connected network, but a sparsely connected network. Unlike a fully connected network where all nodes form a single cluster, in sparsely connected networks there are groups of nodes which form clusters. An event sensed by one cluster cannot be sent beyond that cluster. The information can reach the actor only if the actor visits the cluster that has sensed the event. Sparsely connected networks can range from networks of larger size that form dense clusters (Figure 4) to networks of smaller size that form isolated clusters (Figure 6).

Figure 5 visualizes the need for an actor mobility model that would maximize the collection of valuable event detection information from the clusters, which have no other way to reach the actor. A mobility model produces the path taken by a mobile actor as it traverses the field and is defined by: (i) a set of points, S and, (ii) the order in which to visit the points in S to generate a path P. The mobility models currently used in WSAN research are the random walk mobility model or the random waypoint mobility model or variants thereof. Though it is obvious that random movement by an actor in a sparsely connected network

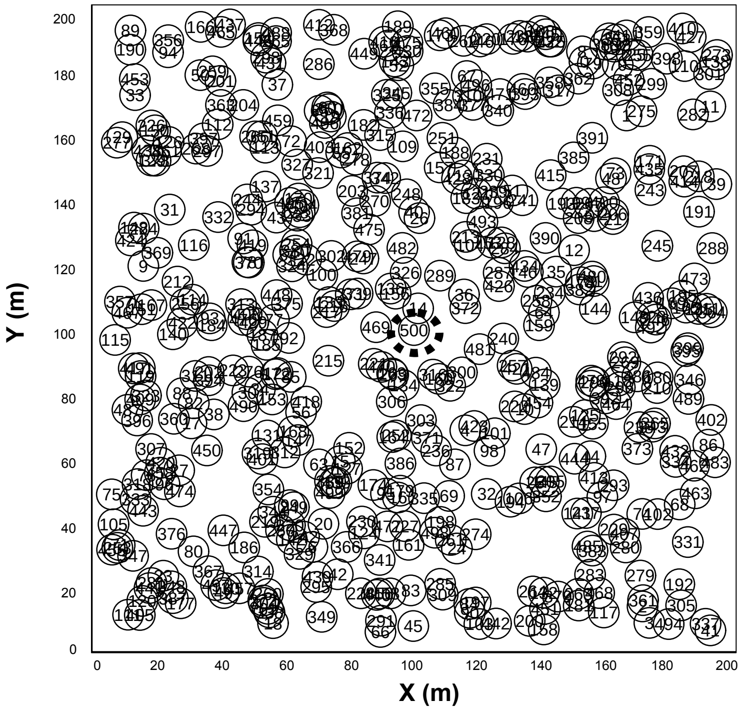

Figure 3: Densely populated network of 500 randomly placed nodes and an actor (dark dotted circle) in a 200 m x 200 m environment.

may not result in good performance, it is used in this thesis for a baseline comparison. The suitability of random mobility models in a fully connected network has not been studied and is beyond the scope of this research. The intelligent mobility models proposed in this thesis use the inherent clustering information of a sparsely connected network to devise mobility models for the actor with the goal of maximizing the number of events detected in a timely manner.

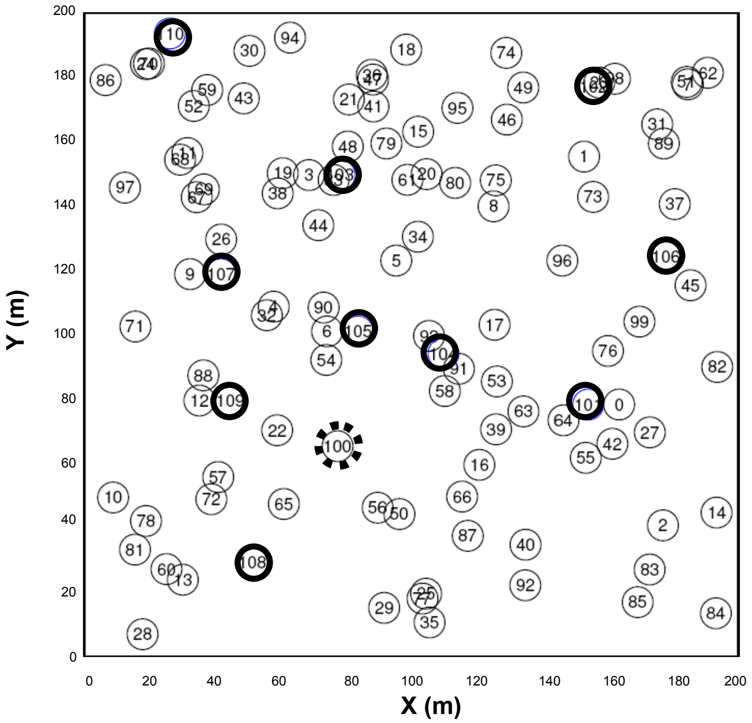

Figure 4: Network of 100 randomly placed nodes, 10 events (dark solid circles) and an actor (dark dotted circle) in a 200 m x 200 m environment.

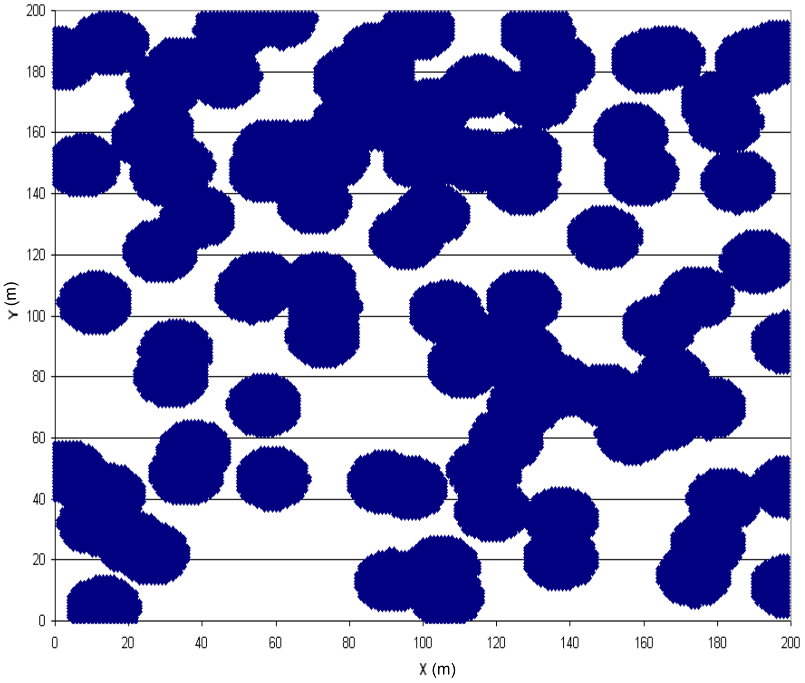

Figure 5: Coverage map for the network shown in Figure 4.

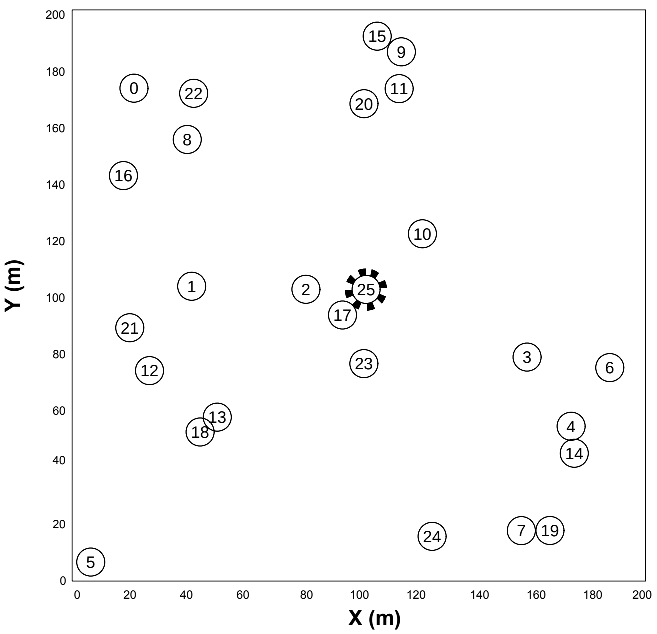

Figure 6: Sparsely populated network of 25 randomly placed nodes and an actor (dark dotted circle) in a 200 m x 200 m environment.

3.1 Static Mobility Models

Static mobility models are those in which the future visit to a destination by the actor is not dictated by its past visit to that destination. The paths generated by these static mobility models can be calculated in advance and can be used in situations where the actor's movement has to be pre-loaded. The static mobility models use only the clustering information of a sparsely connected network. The clusters' information can be generated using the sensor node locations and the sensor node transmission range. Sensor node locations can be obtained using localization techniques such as angle-of-arrival measurements, distance-related measurements, and received signal strength profiling techniques [16]. This research assumes that the location of the stationary sensor nodes is known. Given the node locations and the transmission range of the sensor nodes, the clusters in a network can be found using the algorithm shown in Figure 7.

The working of the clustering algorithm is explained for a simple network of 10 randomly generated node locations as seen in Figure 8. The circular region encompassing the node

```
/* find the neighbors of every node in the network */
/* tx_range - transmission range (diameter) of a sensor node */
for(every node $i$ in the network ) {
    for(every node $j$ in the network ) {
        if( node $i$ != node $j$ ) {
            if( distance between node $i$ and node $j$ < tx_range ) {
                node $i$ and node $j$ are neighbors;
            }
        }
    }
}

/* find paths emanating from every node */
for(every node $i$ in the network ) {
    find all possible paths in the network;
    cnt[i] = unique number of nodes reachable from node $i$;
}

/* cnt[i] denotes the size of the cluster to which node $i$ belongs */
for(every node $i$ that has not been assigned to a cluster ) {
    k = number of nodes that have the same cluster size as node $i$;
    if( k == cnt[i] ) {
         /*all k nodes are part of the same cluster*/
         assign a cluster_id and associate the nodes;
    }
    if( k > cnt[i] ) {
         go through the paths and find clusters of
                      size cnt[i] from the k nodes;
         assign each cluster a cluster_id and associate
                      the corresponding nodes with that cluster;
    }
    mark the k nodes as having been assigned to a cluster;
}
```

Figure 7: Pseudo code of the clustering algorithm.

number represents the transmission range of the node. When the algorithm shown in Figure 7 is run, the neighbors of each node are first found. The neighbors are: {1,3}, {2,4}, {3,1,9,10}, {4,2}, {5,8}, {6,7}, {7,6}, {8,5}, {9,3}, and {10,3}. Then the paths emanating from each node is found. The paths are: {1-3-9, 1-3-10}, {2-4}, {3-1, 3-9, 3-10}, {4-2}, {5-8}, {6-7}, {7-6}, {8-5}, {9-3-1, 9-3-10}, and {10-3-1, 10-3-9}. Next, the unique number of nodes reachable by every node is found. Every node includes itself in the reachable list. For the 10 nodes shown in Figure 8, the {node *i*, number of nodes reachable by node *i*} pairs are found to be as follows: {1,4}, {2,2}, {3,4}, {4,2}, {5,2}, {6,2}, {7,2}, {8,2}, {9,4}, and, {10,4}. The number of nodes reachable by node *i* also represents the size of the cluster to which node *i* belongs. In order to assign a node *i* to a cluster, the number of nodes which have the same cluster size as node *i* is found. For example, node 1 belongs to a cluster of size 4 (cnt[1]=4) and there are 3 (k=4) other nodes which also have a cluster size of 4. In this case, (k==cnt[i]) and all the four nodes (1, 3, 9, 10) belong to the same cluster. The four nodes are assigned with the same cluster_id without any further computation. On the other hand, node 2 belongs to a cluster of size 2 (cnt[2]=2) and there are 5 (k=6) other nodes which also have a cluster size of 2. This is a case of (k > cnt[i]) and the path information is used to identify the unique clusters: (2, 4), (5, 8), and (6, 7). In this manner, the clustering algorithm is repeated until every node has been assigned to a unique cluster_id.

When the speed of a mobile actor is known, a mobility model can generate a path for the actor for a given duration. From the clusters' information, a set of points to visit has to be found first. In this thesis, the centers of the clusters are used as points to visit. The center of a cluster is found by averaging the x and y coordinates, respectively, of all the sensor nodes belonging to that cluster. This work ignores the possibility that the geographic center of the cluster may not be part of the cluster. Next, for a given starting point, the set of points has to be ordered to form a path. Using the cluster centers, two static mobility models are defined.

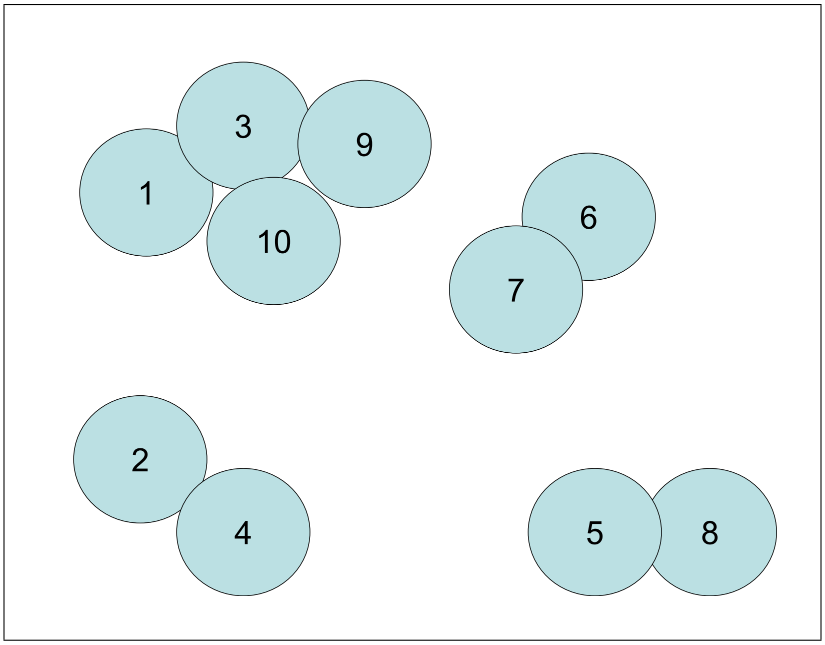

Figure 8: Example network of 10 randomly placed nodes shown with their sensing range.

3.1.1 Mobility Model 1 (MM1)

This mobility model uses a nearest neighbor greedy algorithm to find the destination to visit next. When a new destination to visit has to be found, the nearest unvisited neighbor is chosen as the next destination to visit for the actor node. Depending on the speed of the actor, the duration of a simulation may be longer than the time it takes to visit the points in S once. In such cases, the algorithm is repeated to generate the path P for the required duration. Figure 9 shows the steps involved in generating a path using this approach. Figure 11 (a) shows a path generated using mobility model MM1, covering all the points once.

3.1.2 Mobility Model 2 (MM2)

This mobility model uses a variation of MM1 where a neighbor with the nearest x-coordinate is found, to visit next. This generates a path which scans the entire field in a zigzag manner along the y-axis. Initially a list L is generated starting from the point with the smallest x-coordinate value. To find the next point in L, an unvisited neighbor with the nearest x-coordinate is found. All points in S are successively added to L using the nearest x-coordinate approach. If an actor follows this list, it would begin scanning from the left of

```
elapsed_time = 0;
if( elapsed_time < sim_time ) {
    num_visited = 0;
    add the starting point to $P$ and mark it as visited in $S$;
    $sp$ = starting point;
    num_visited++;

    /* find $m$ - the next destination of the actor */
    if( num_visited < total_num_clusters ) {
        find the closest unvisited point to visit next
                                        and assign it to $m$;
        add $m$ to $P$ and mark it as visited in $S$;
        dist = distance between $sp$ and $m$;
        elapsed_time + = $\frac{dist}{actor\_speed}$;
        /* make $m$ the new starting point */
        $sp = m$;
        num_visited++;
    }
    mark all points in $S$ as unvisited;

    /* go back to starting point */
    $m$ = starting point;
    dist = distance between $sp$ and $m$;
    elapsed_time + = $\frac{dist}{actor\_speed}$;
}
```

Figure 9: Steps to generate a path using MM1.

the field. But the current location of the actor could be anywhere in the field. Hence the closest point in L to the current location of the actor is found and new destinations to visit are found by scanning either backwards or forward in L. The algorithm to generate a path using this method is shown in Figure 10. With this approach some points may be visited more than once before all the points in S have been visited once, unlike MM1 where all points are visited once before they are re-visited. Similarly the nearest y-coordinate can also be used to find the next destination. By doing this, the field would be traversed in a zigzag manner along the x-axis. In this work, only the nearest x-coordinate approach is used in experiments to generate paths for the actor. Figure 11 (b) shows the ordered list and the subsequent path generated using mobility model MM2, covering all the points once.

The static mobility models defined above, along with preliminary simulation results were published in the first international conference dedicated to WSANs [12].

```
elapsed_time = 0;
/* first generate the ordered list L */
find the point in $S$ with the smallest x-coordinate value;
add this point to $L$ and mark it as visited in $S$;
num_visited = 1;
if( num_visited < total_num_clusters ) {
    find an unvisited point in $S$ with the closest x-coordinate to
                                        the current point;
    add that point to $L$ and mark it as visited in $S$;
    num_visited++;
}

/* now generate the path */
/* starting point is assumed to be a point on list closest to the actor's current */
/* location and the time for actor to reach this point is considered negligible */
scan list L to find the starting point;
add starting point to P;
$sp$ = starting point;
elapsed_time = 0;
/* find $m$ - the next destination of the actor */
backwards = FALSE;
if( elapsed_time < sim_time ) {
    if( start of list ) {
        forward = TRUE;
        backwards = FALSE;
    }
    else if (not end of list *and* backwards == FALSE )
        forward = TRUE;
    else {
        /* end of list or already scanning backwards */
        backwards = TRUE;
        forward = FALSE;
    }
    if( forward == TRUE )
        scan forward in L;
    else
        scan backwards in L;
    $m$ = next point in L;
    add $m$ to P;
    dist = distance between $sp$ and $m$;
    elapsed_time $+ = \frac{dist}{actor\_speed}$;
    /* make $m$ the new starting point */
    $sp = m$;
}
```

Figure 10: Steps to generate a path using MM2.

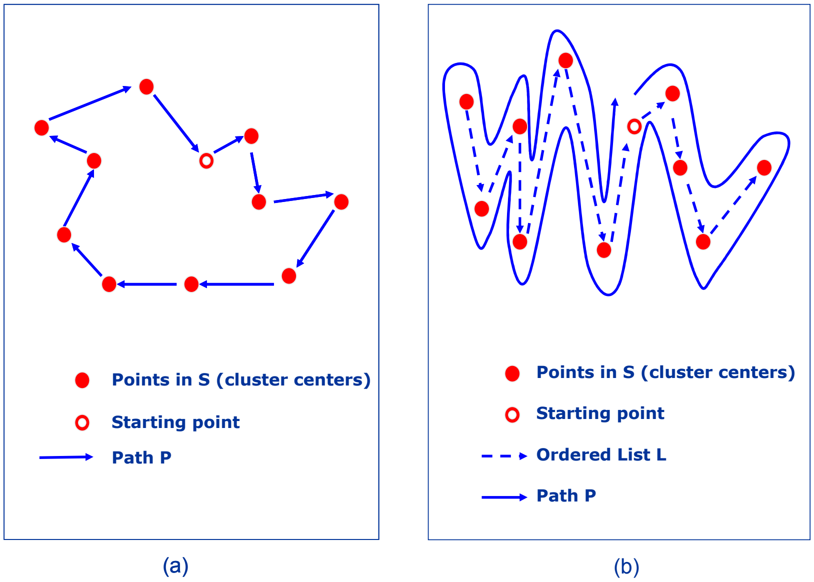

Figure 11: Paths generated for an example network with 11 clusters using static mobility model (a) MM1 and (b) MM2.

3.2 Dynamic Mobility Models

Dynamic mobility models are those in which the future visit to a destination by the actor is influenced by its past visit to that destination. These mobility models make use of not just the clustering information but also the area occupied by the clusters. The area of a cluster is of interest because, the larger the area of a cluster, the greater the chance of an event occurring in that cluster. Since the goal is to maximize the number of detected events, the area of the clusters is incorporated in deciding which cluster to visit next. The clusters' area can be found using the steps shown in Figure 12. To help define the dynamic mobility models, a statistical model for fixed duration events is proposed which incorporates the area of the clusters and the time of last visit to a cluster by the actor. The statistical model is defined in Section 3.2.1.

```
for( every square meter of the field ) {
    for( every node $n$ in every cluster $m$ ) {
        if( distance between node $n$ and center of this sq. m. < tx_range/2 ) {
            mark this sq. m. as covered by node $n$ of cluster $m$;
            increment cluster $m$'s area by 1 sq. m.;
        }
    }
}
```

Figure 12: Pseudo code to find the area of clusters in a network.

3.2.1 Statistical Model

Assuming that the following parameter values are known, (i) duration of an event, t_d (ii) rate of event occurrence, r (iii) time of last visit to cluster i by the actor, t_v^i (iv) area of cluster i, A_c^i (v) total area of the field, A_t, and, (vi) velocity of the actor, s; the following values are of interest:

1. Expected number of detectable events in cluster i, $E_d^i(t)$
2. Expected number of lost events in cluster i, $E_l^i(t)$

The statistical model for fixed duration events for every cluster i is shown in Figure 13. The events are independent and occur at a specific rate, r. The expected number of new events occurring in a cluster begins to grow since the time of last visit of the actor to that cluster, t_v^i. This is represented by a line denoted as $E_n^i(t)$ in Figure 13. With fixed duration events, any event that occurs, ends after time t_d. For example, an event that occurs at time t_1 seconds will end by time $(t_1 + t_d)$ seconds, and so on. The expected number of events which have occurred and are ending in cluster i is represented by the line parallel to $E_n^i(t)$ in Figure 13. This parallel line also represents $E_l^i(t)$, the expected number of lost events in that cluster. This is because if an event occurred in this cluster and the actor did not (re)visit the cluster within the duration of the event, then this event goes undetected, thereby accounting for a lost event. From these, $E_d^i(t)$ - the expected number of detectable events in cluster i - can be found. $E_d^i(t)$ begins to grow at time t_v^i, and becomes a constant after time $(t_v^i + t_d)$. The growth part of the $E_d^i(t)$ mirrors $E_n^i(t)$ until it reaches time $(t_v^i + t_d)$

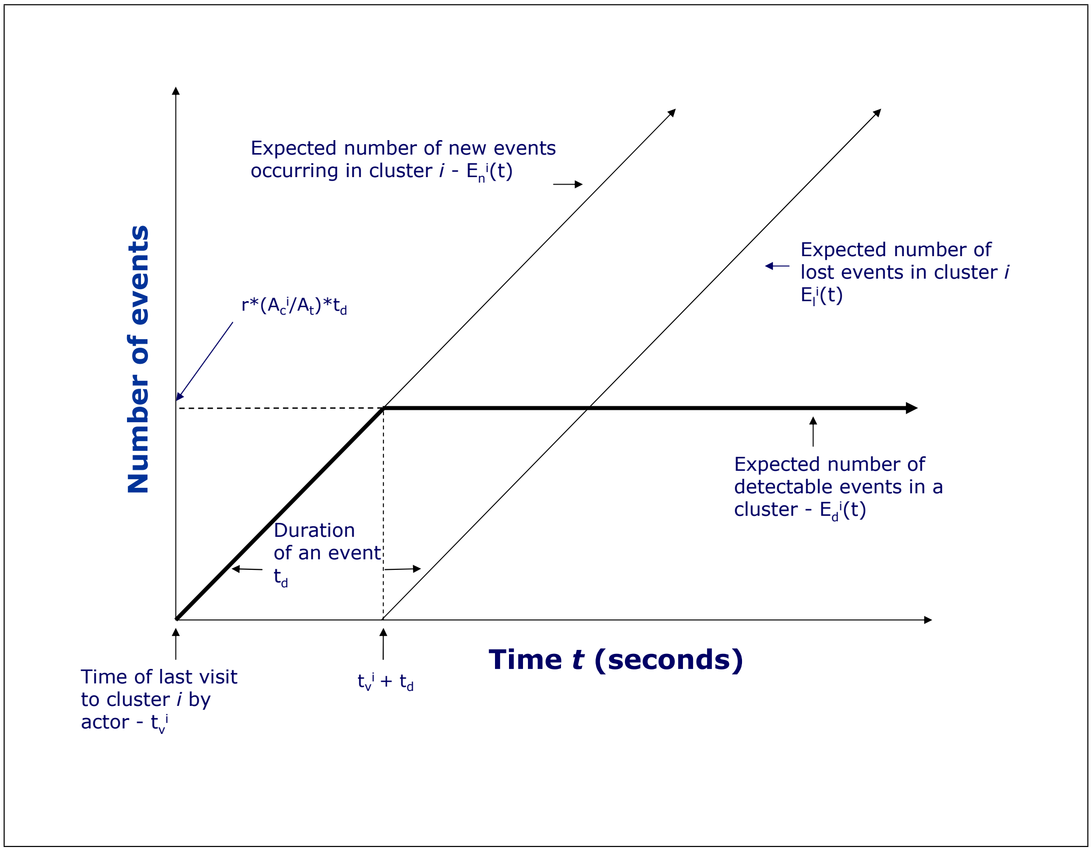
Number of events
Expected number of new events occurring in cluster i - $E_n^i(t)$
Expected number of lost events in cluster i $E_l^i(t)$
$r*(A_c^i/A_t)*t_d$
Expected number of detectable events in a cluster - $E_d^i(t)$
Duration of an event t_d
Time t (seconds)
Time of last visit to cluster i by actor - t_v^i
$t_v^i + t_d$

Figure 13: Statistical model for fixed duration events for cluster i.

and then levels off. $E_d^i(t)$ represents the expected number of events that will be detectable if the actor (re)visited cluster i at time t.

From Figure 13, simple line equations can be written to describe the statistical model as shown below. The slope of the parallel lines for cluster i is defined by m^i. The slope represents the rate of event detection for cluster i (events/second).

$$\boxed{m^i = r * \frac{A_c^i}{A_t}} \tag{1}$$

$E_n^i(t)$ is the equation of a line whose slope is m^i and passes through the point (t_v^i, 0).

$$\boxed{E_n^i(t) = m^i * (t - t_v^i), t \geq t_v^i} \tag{2}$$

$E_l^i(t)$ is the equation of a line whose slope is m^i and passes through the point ($t_v^i + t_d$, 0).

$$\begin{aligned} E_l^i(t) &= m^i * t - m^i * (t_v^i + t_d) \\ &= m^i * (t - t_v^i - t_d) \end{aligned}$$

$$\boxed{E_l^i(t) = m^i * (t - t_v^i - t_d)} \tag{3}$$

From equations (2) and (3), $E_d^i(t)$ can be defined as shown below:

$$\begin{aligned} E_d^i(t) &= m^i * (t - t_v^i)\ , t_v^i \leq t\ <\ t_v^i + t_d \\ &= m^i * t_d \qquad\quad , t \geq t_v^i + t_d \end{aligned} \tag{4}$$

It is clear that the expected values are dependent on m^i and, the slope m^i is influenced by the ratio of the area of cluster i to the total area of the field. The values shown in the above equations are maintained for every cluster in the network, at the actor node. Figure 14 shows a pictorial representation of the statistical model maintained for every cluster of a network at the actor node. Based on the time of last visit to a cluster by the actor, the expected values are different for each cluster, at the current time. This is represented by a

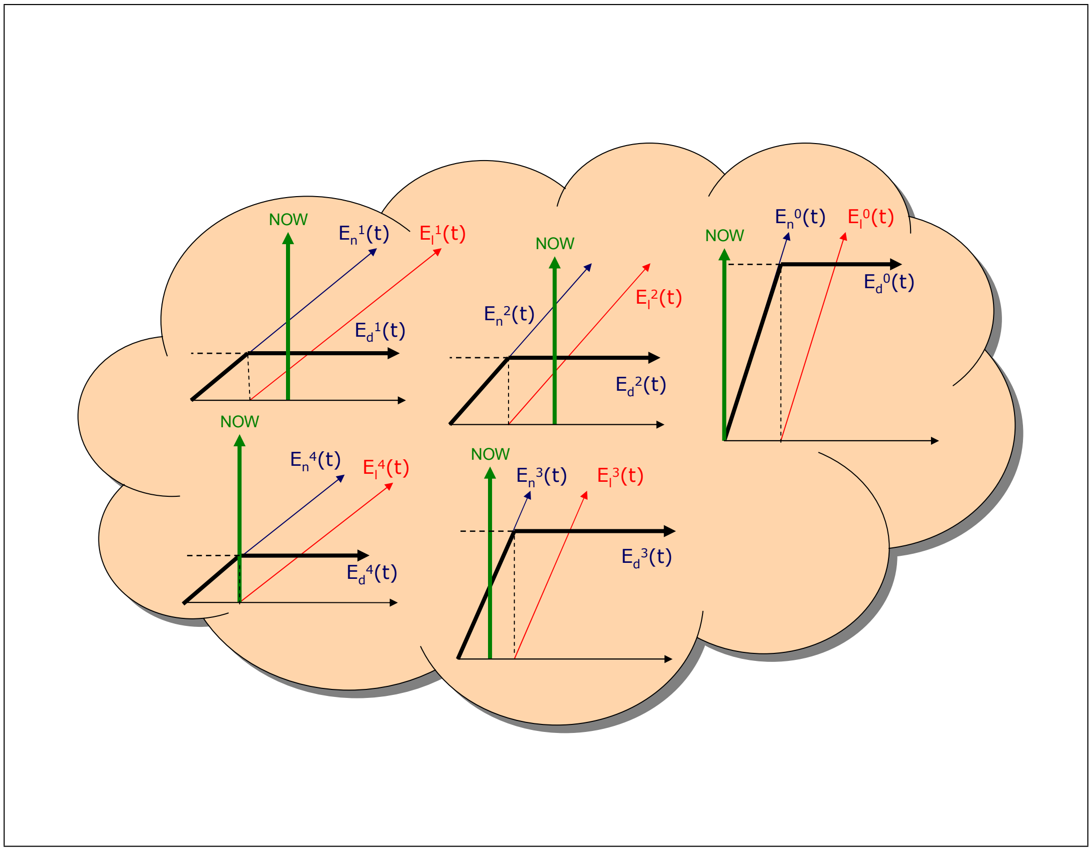

Figure 14: Actor's view of the network.

shifted y-axis marked with NOW in the figure. Depending on the mobility model chosen, using the values from the statistical model, the actor dynamically charts its course as to when and which cluster to visit.

This actor's view of the network can be easily translated into a tabular form for implementation purposes, as shown in Table 1. Given that the speed at which the actor is moving is known, t_c^i represents the time at which the actor can reach the center of cluster i, (x_c^i, y_c^i), from its current location. $E_d^i(t_c^i)$ represents the expected number of detectable events in cluster i at time t_c^i, when the actor node will reach the center of cluster i. Similarly, $E_l^i(t_c^i)$ represents the expected number of lost events in cluster i at time t_c^i, if the actor node does not (re)visit cluster i. The values in this table are updated periodically and destinations are chosen for the actor node depending on the application's choice of mobility model using these values.

Table 1: Tabular implementation of the actor's view of a network.

Cluster i	x_c^i	y_c^i	A_c^i	t_v^i	t_c^i	$E_d^i(t_c^i)$	$E_l^i(t_c^i)$
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
...							

The advantages of this model are that it is simple to implement, dynamic in decision making and consumes limited resources on the actor node. This statistical model helps the actor node to succinctly capture all the activity that happens in the network, in a transparent manner. Thus Table 1 will serve as a figure of merit to compare and assess different dynamic intelligent mobility models for the actor. Based on this statistical model, two dynamic mobility models are defined for the actor node.

3.2.2 Mobility Model 3 (MM3)

This mobility model decides the next destination for the actor node by choosing the cluster which has the largest value for $E_d^i(t)$. The clusters which have a large value for $E_d^i(t)$ are those that have a large cluster area. The basis for incorporating the area of clusters is that the number of events occurring in a cluster is proportional to the area occupied by the cluster. The actor node will try to visit these clusters as often as possible with the hope of maximizing the number of detected events. It is expected that with MM3, a good number of events will be detected with event detection information reaching the actor at the earliest. Since the actor visits clusters with larger areas more frequently, the number of clusters visited is limited, thereby accounting for a low latency whenever an event is detected.

3.2.3 Mobility Model 4 (MM4)

This mobility model decides the next destination for the actor node by choosing the cluster with the largest value for $E_l^i(t)$. By visiting a cluster that may have the largest number of lost events, the actor tries to detect these events, thereby minimizing the number of lost events. With this mobility model the expectation is that the most number of events will be detected by the actor node and that the packet latency will be nominal.

3.3 Mobility Model Expectations

The expectations from the different mobility models are discussed in this section. MM1 is a greedy algorithm and is expected to detect the least number of events. Since it goes from one nearest destination to the next in a pre-determined fashion, it is also expected to have a high latency of event detection packets. On the other hand, MM2 is expected to detect more events than MM1 as it scans the field by going across the y-axis in a zigzag manner. The packet latency with MM2 would be high just like MM1, as it is also going about the same path in a repetitive manner. The expectations for MM3 and MM4 have been discussed in Sections 3.2.2 and 3.2.3 respectively. The expectations from all the intelligent mobility models are summarized in the Table 2.

Table 2: Mobility model expectations.

	$MM1$	$MM2$	$MM3$	$MM4$
Number of events detected	Least	Low	High	Highest
Latency of event detection packets	High	High	Lowest	Low

CHAPTER IV

PERFORMANCE EVALUATION

The intelligent mobility models presented in Chapter III were evaluated with extensive simulations and the results are analyzed in this chapter.

4.1 Simulation Environment

The simulations were conducted using the network simulator ns-2 [17] in a Linux environment. The events and sensor nodes were represented in ns-2 using extensions from the Naval Research Laboratory [5]. A new actor node has been defined in ns-2 to handle the workings of an actor in a WSAN. Sensor node locations for a network were generated using a uniform random number generator. The sensor nodes were given a transmission range of 20 m and the actor node was given a transmission range of 30 m. The number of events that occurred in every simulation was fixed at 50. The events were detectable by any sensor node within a 30 m diameter of the event location. All these values are constant in any simulation. The stationary events have uniform random locations and the inter-arrival time between events follows an exponential distribution with a fixed rate parameter. An exponential distribution was chosen to model the inter-arrival time between events so that the independent events occur at a constant rate. The simulation duration is a product of the number of events and the inter-arrival time of events. All the simulation parameters are summarized in Table 3.

Simulating a WSAN requires event detection information to reach the actor node in a timely manner. This is essential because only then can the response taken by the actor be valid. With a mobile actor, the location of the actor is changing constantly. Generally, the actor node issues location updates periodically, so that the sensor nodes know the direction in which to forward the data. With a sparsely connected network, such location updates cannot reach the entire network because of the presence of the clusters. Therefore, event detection information is propagated through a cluster and is readily available at all the nodes in a cluster (after a propagation delay) to be picked up by the actor node. The event

Table 3: Simulation parameters.

Field dimension	200 m x 200 m
Number of sensor nodes	variable
Number of actor nodes	1
Transmission range: Sensor node	20 m
Transmission range: Actor node	30 m
Actor speed	variable
Acting duration	variable
Event diameter	30 m
Event duration	variable
Number of events	50
Rate of event occurrence	variable
Simulation duration	variable

detection information is propagated only when an event is in progress. A routing protocol is required to forward the event detection information within a cluster. This thesis' objective being to design intelligent mobility models for the actor node and not a sophisticated routing protocol, the simplest option of a broadcast routing protocol is chosen even though more effective schemes are desirable. Broadcast is an operation wherein every incoming new message at a node is distributed to all nodes in the network. If the set of forwarding nodes can be restricted while still ensuring that all the nodes receive the data, it would be more energy efficient. BCAST [13] is an optimized broadcast protocol that keeps track of one-hop and two-hop neighbors of a sensor node and sends information only to them. Only packets that would reach additional neighbors are re-broadcast. The disadvantage of this routing protocol is the cost of maintaining the neighborhood information.

4.2 Implementation in ns-2

Ns-2 is a discrete event simulator targeted at networking research. Ns-2 is written in C++ and an object oriented version of tool command language (Tcl) called OTcl. The mobility models and the clustering algorithm have been written in C++ and have been seamlessly integrated with the functionality of the actor node. The input to the simulator is a tcl file. The scenario files describing the network and the events are inputs to the tcl file. An example tcl file and snippets of an event scenario file have been shown in Appendix A. The different system parameter values can be set in the tcl file. These values are passed on to the simulator and are used by the mobility models to chart the path of the actor node.

When a simulation begins, the clusters of the network and the area of the clusters are found. Then, the centers of the clusters are found to use as points to visit. Depending on the mobility model chosen, the next destination for the actor is found and the actor moves towards that destination at the designated speed. If the actor encounters an event in its path, that information is logged. If the actor node has been configured to perform action (there is an acting duration provided in the tcl file), the actor pauses at the event location for the acting duration. In doing this, it mimics the time spent acting on the event. Then the mobility model calculates the actor's next destination, and the actor moves towards that destination. This process keeps repeating for the entire duration of the simulation.

With the dynamic mobility models, every time the actor visits a chosen cluster, the t_v^i of that cluster maintained at the actor node is updated with the current time. Other than this, the clusters visited by the actor en-route to a destination are also marked. Every so often, the actor checks to see if it is passing by a cluster. If yes, then it updates the t_v^i of that cluster with the current time. In this manner, the dynamic mobility models try to model the visits to clusters as best as possible.

4.3 Number of nodes in a cluster vs. Cluster area

In this section, the significance of the number of nodes in a cluster, the number of clusters in a network and area occupied by the clusters are discussed. For the example 100 node network shown in Figure 4, the clusters' information and their area were generated using the steps shown in Figures 7 and 12 respectively. Table 4 shows the details of the clusters. There are 23 clusters ranging in size from 1 to 24. The table also shows the nodes belonging to each cluster, the clusters' area and the percentage of the entire field that each cluster occupies. It is interesting to note that more than one-thirds of the clusters are of size 1. Also, the number of nodes in a cluster is not proportional to the area. For example, cluster 22 has 3 nodes and occupies an area of 470 sq. m. Cluster 0 has 19 nodes which is 6 times as many nodes as in cluster 22. But the area occupied by cluster 0 is 4619 sq. m., which is almost 10 times as that of cluster 22. Another example is that of cluster 8 with 24 nodes occupying 4746 sq. m. Though it has approximately 20% more nodes than cluster 0, it occupies only 2% more area. This is the reason behind the statistical model incorporating the area of the cluster and not the size of the cluster.

Since the intelligent mobility models are based on the idea of clustering, it would be valuable to know the limits beyond which these mobility models cannot be used. The number of clusters formed by networks of size 25 nodes to 200 nodes in a 200 m x 200 m area, in increments of 25 nodes was found and the results were plotted as shown in Figure 15. The results are the average of 10 configurations each of a particular network size. For example, 10 different networks of size 100 nodes each were generated and the clustering algorithm was run on each network. The number of clusters formed was averaged and plotted against 100 nodes in Figure 15. It is interesting to note that though the number of clusters formed initially increases, it decreases beyond 75 node networks. The most clusters are formed by networks of size 50 nodes each. Networks with 200 nodes seem to be the upper limit, after which nodes in a network may form one giant cluster in the 200 m x 200 m field.

Along with the number of clusters formed, the area occupied by the clusters was also noted. The results were averaged and plotted as shown in Figure 16. As the number of nodes in a network increases, the area occupied by the clusters also increases. It is interesting to note that as the number of nodes in the network is doubled (25, 50, 100, 200), the area occupied by the clusters falls in each of the 10,000 sq. m. categories (0-10,000 sq. m., 10,000-20,000 sq. m., 20,000-30,000 sq. m., 30,000-40,000 sq. m.).

Table 4: Cluster information for the network shown in Figure 4.

Cluster #	**Size**	**Nodes in cluster**	**Area A_c (sq. m.)**	A_c/A_t **(%)**
0	19	0 16 17 27 39 40 42 53 55 58 63 64 66 76 87 91 92 93 99	4619	11.5475
1	2	1 73	563	1.4075
2	2	12 88	474	1.1850
3	2	23 98	420	1.0500
4	2	50 56	458	1.1450
5	2	57 72	466	1.1650
6	4	14 2 83 85	940	2.3500
7	4	25 29 35 77	731	1.8275
8	24	3 5 8 15 18 19 20 21 33 34 36 38 41 44 46 47 48 49 61 74 75 79 80 95	4746	11.8650
9	5	4 6 54 90 32	999	2.4975
10	5	10 78 60 13 81	1019	2.5475
11	6	7 31 37 51 62 89	1102	2.7550
12	11	9 11 26 30 43 52 59 67 68 69 94	2205	5.5125
13	1	22	313	0.7825
14	1	28	253	0.6325
15	1	45	307	0.7675
16	1	65	311	0.7775
17	1	71	311	0.7775
18	1	82	158	0.3950
19	1	84	164	0.4100
20	1	96	316	0.7900
21	1	97	300	0.7500
22	3	24 70 86	470	1.1750

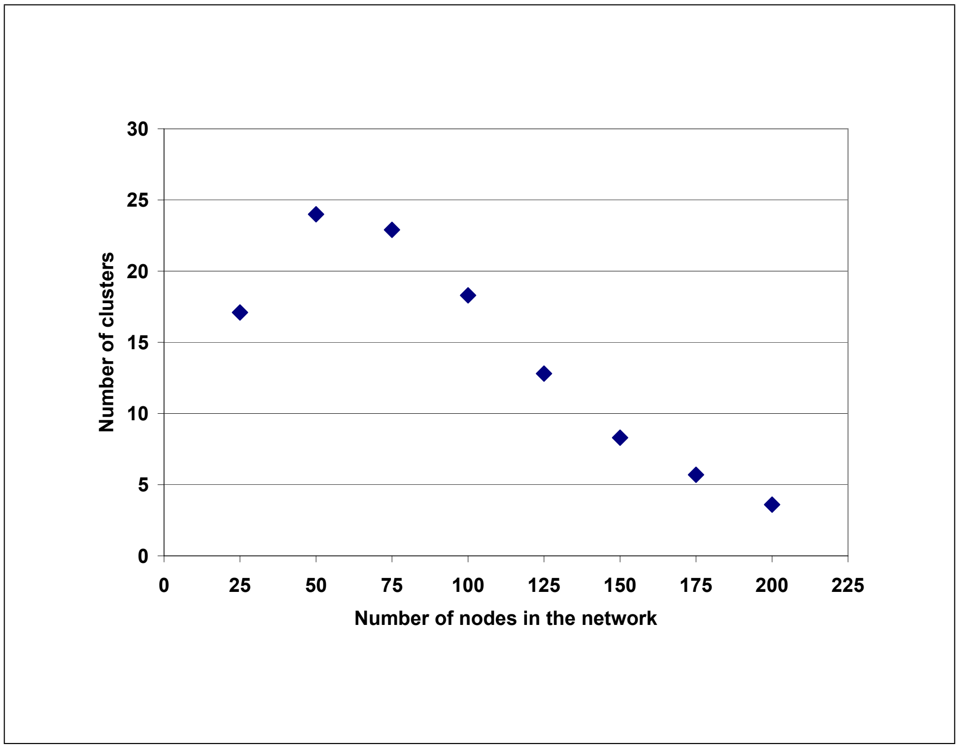

Figure 15: Average number of clusters formed by sparse networks in a 200 m x 200 m field.

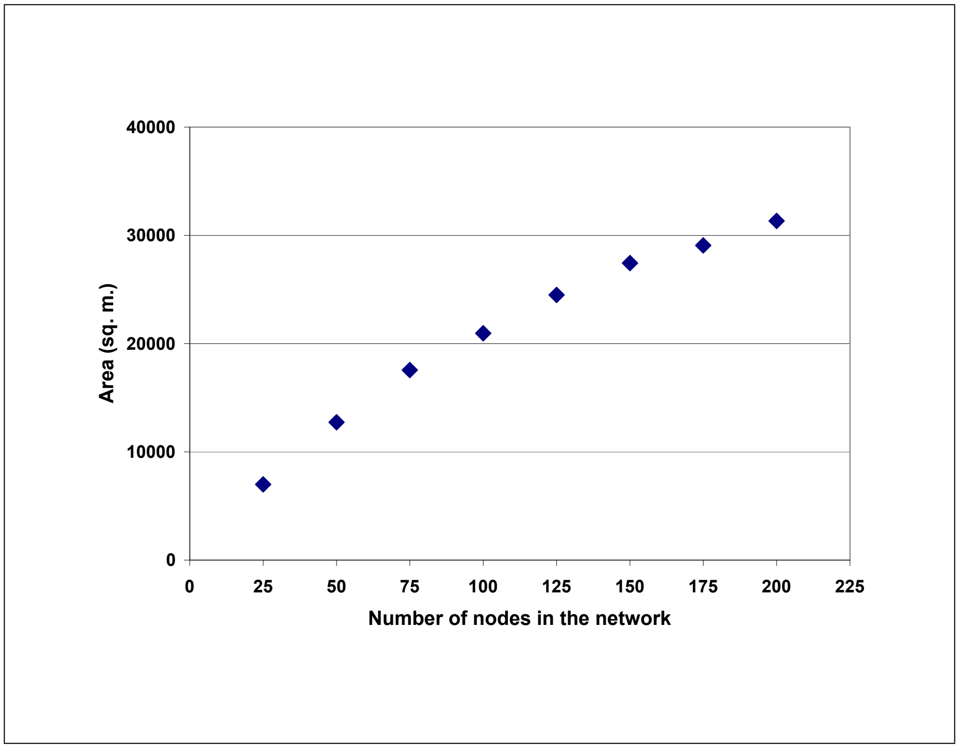

Figure 16: Average area covered by clusters of networks in a 200 m x 200 m field.

4.4 Events sensed in sparsely connected networks

With a fully connected network - assuming there is a dense deployment of sensor nodes in the entire field - there is guarantee of 100% detection of events by the sensor nodes. With sparsely connected networks, there are pockets where there is no sensor node coverage. Hence every event that occurs may not get sensed. So it would be useful if there was some metric on the number of events that can be sensed using sparsely connected networks. To study this, 50 events with a range of 30 meters were introduced in 10 network configurations each of 25, 50 and 100 nodes. The same set of events was deployed in each case. The average inter-arrival time between events was 432 seconds and the events lasted for 300 seconds each. All other parameters were as shown in Table 3. The number of events sensed by the sensor nodes was averaged and plotted as shown in Figure 17.

It is obvious that the number of events sensed will increase as the event diameter is increased. So the interesting observation from this figure is not the fact that an average of 46 out of 50 events were sensed in 100 node networks, but the fact that when the number of nodes in a network was reduced to a fourth, the number of events sensed was not quartered, but halved. These simple results show that a high probability of events can be sensed even with a sparsely connected network composed of a few nodes.

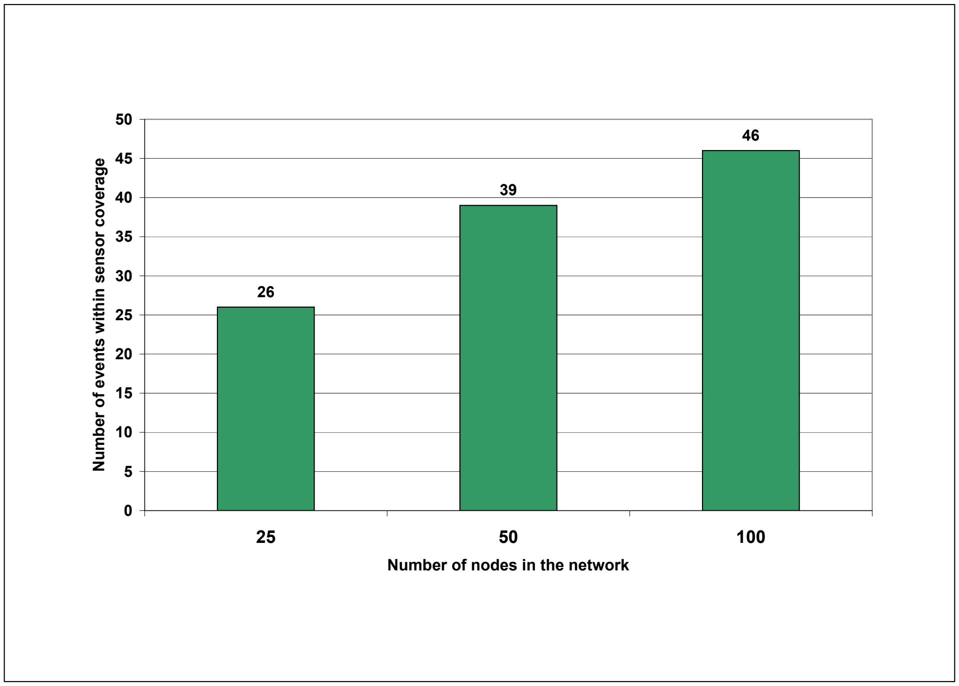

Figure 17: Average number of events sensed in sparsely connected networks, event diameter = 30 m, t_d = 300 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

4.5 Simulation Results - 1

The first set of simulations presented compares a non-intelligent mobility model against an intelligent mobility model. The random walk mobility model was chosen for the non-intelligent case and MM1 was chosen for the intelligent case. The random walk mobility model has been described in Section 2.1. MM0 represents the random walk model in Figures 18 and 19. The chosen configuration was 50 node networks. For the results shown in Figure 18, the event duration was 300 seconds and the inter-arrival time between events was 432 seconds. For the results shown in Figure 19, the event duration was 75 seconds and the inter-arrival time between events was 216 seconds. There were 50 events in both scenarios and approximately 37-39 events were sensed by the sensor nodes as marked with dotted lines in the figures.

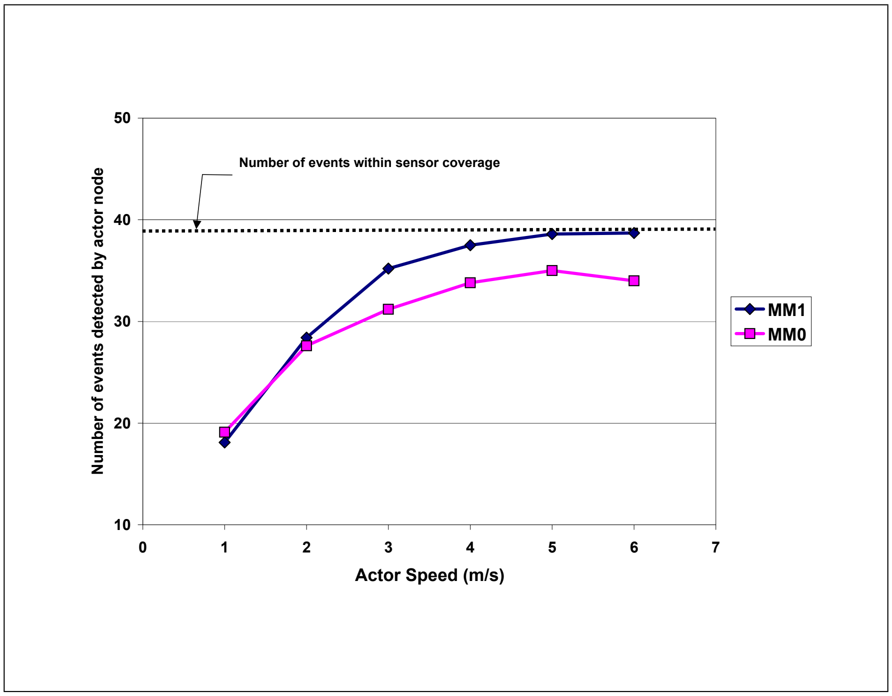

Figure 18: Comparison between an intelligent and a non-intelligent mobility model - 1, 50 node networks, t_d = 300 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

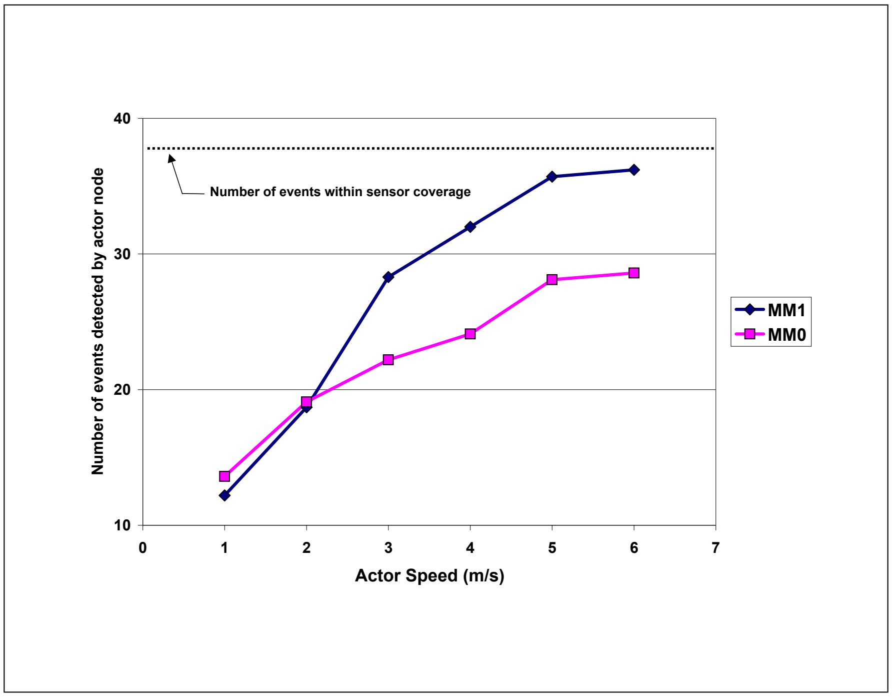

Figure 19: Comparison between an intelligent and a non-intelligent mobility model - 2, 50 node networks, t_d = 75 seconds, 1/r = 216 seconds, 200 m x 200 m area, 50 events.

4.6 Simulation Analysis -1

There are a few conclusions that can be made from the simulation results shown in Section 4.5:

1. Random walk may detect more events by chance at lower speeds, but that does not carry forward at higher speeds.

2. At higher speeds, as the intelligent mobility model moves towards 100% detection of events by the actor node, random walk is unable to match it in performance.

3. When the event duration is reduced along with the inter-arrival time of events, the gap in performance widens as seen in Figure 19.

4. Though there may be specific scenarios in which the random walk model may come close to performance of an intelligent model, it is not guaranteed and cannot be used with confidence. With an intelligent model and ample actor speed, it is possible to achieve 100% detection of events by the actor node with respect to the number of events sensed by sensor nodes.

4.7 Simulation Results - 2

The first set of simulations has shown that the intelligent mobility models are superior to non-intelligent mobility models in a sparsely connected network. In these set of simulations, the different intelligent mobility models are compared against one another to see how they perform in sparsely connected networks. The actor node acts just as an event collector and there is no acting duration incorporated. The chosen configurations include 25, 50 and 100 node networks. The same set of 50 events is applied across all the networks. The simulation duration was set to 21600 seconds (6 hours). The rate parameter of the exponential distribution was set to 0.0023 (= 1/432 = 50/21600) and events were generated accordingly. The event duration was set to 300 seconds. The actor speed was varied from 1-6 m/s. The number of events detected by the actor as it goes through the field was noted. Also noted was the latency of the event detection packets reaching the actor node. Latency of an event detection packet is the time between the start of an event and the first event

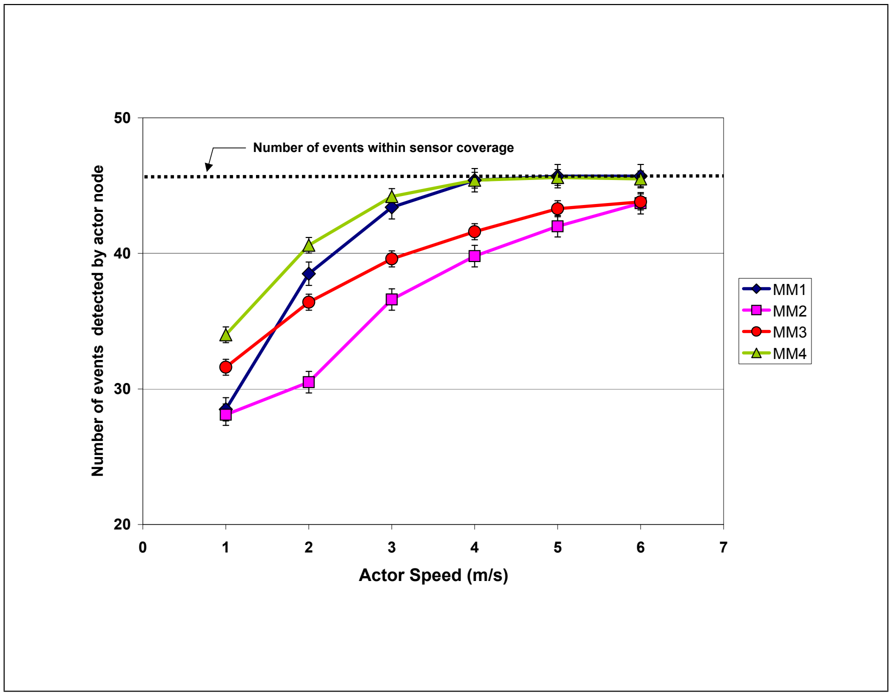

Figure 20: Average number of events detected by actor node, 100 node networks, t_d = 300 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

detection packet reaching the actor node, from any of the sensor nodes which sensed the event. The results across the 10 networks in each case were averaged and plotted with 95% confidence interval as shown in Figures 20 through 25.

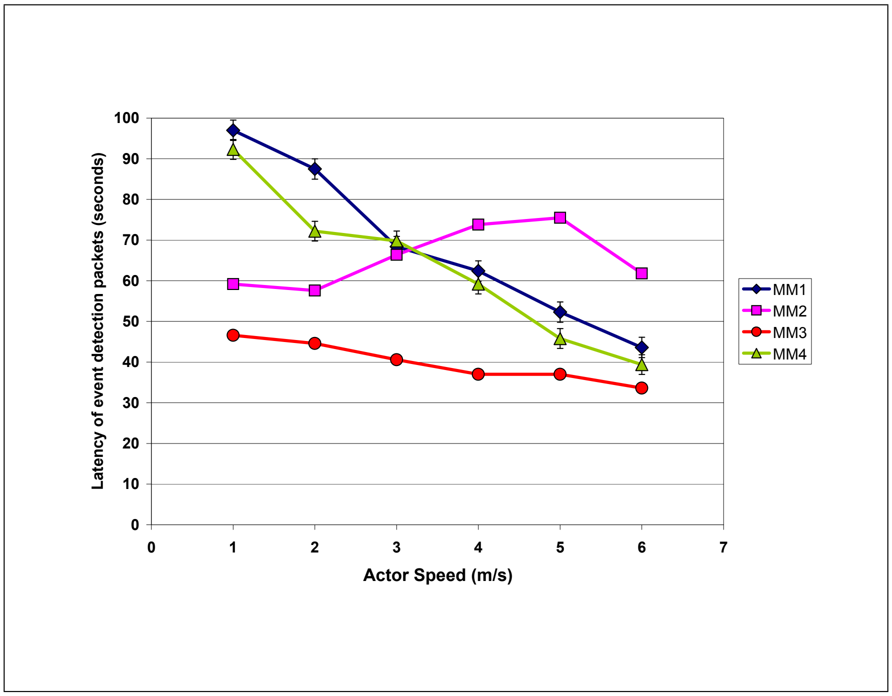

Figure 21: Average latency of event detection packets, 100 node networks, t_d = 300 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

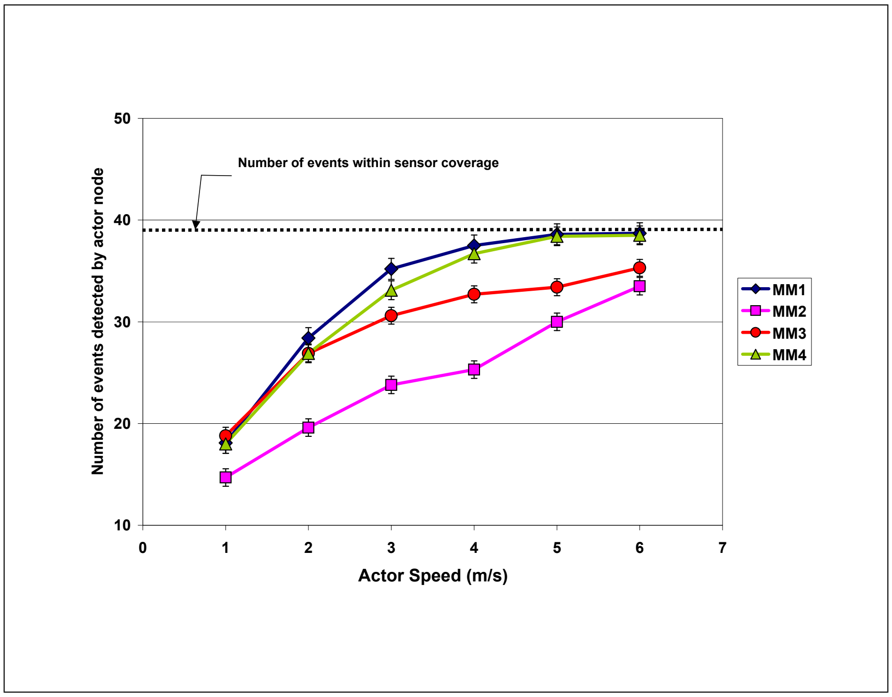

Figure 22: Average number of events detected by actor node, 50 node networks, t_d = 300 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

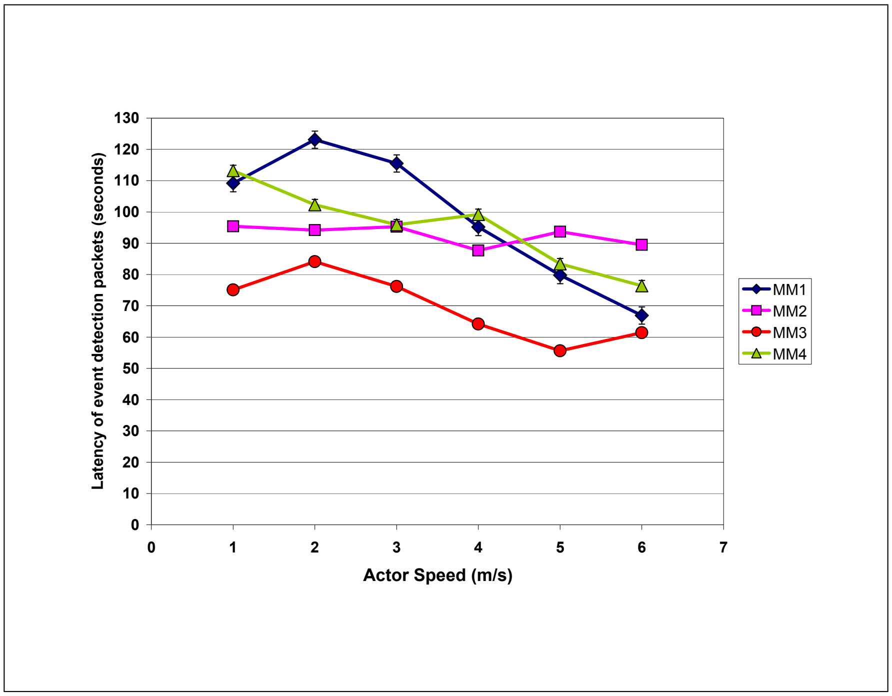

Figure 23: Average latency of event detection packets, 50 node networks, t_d = 300 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

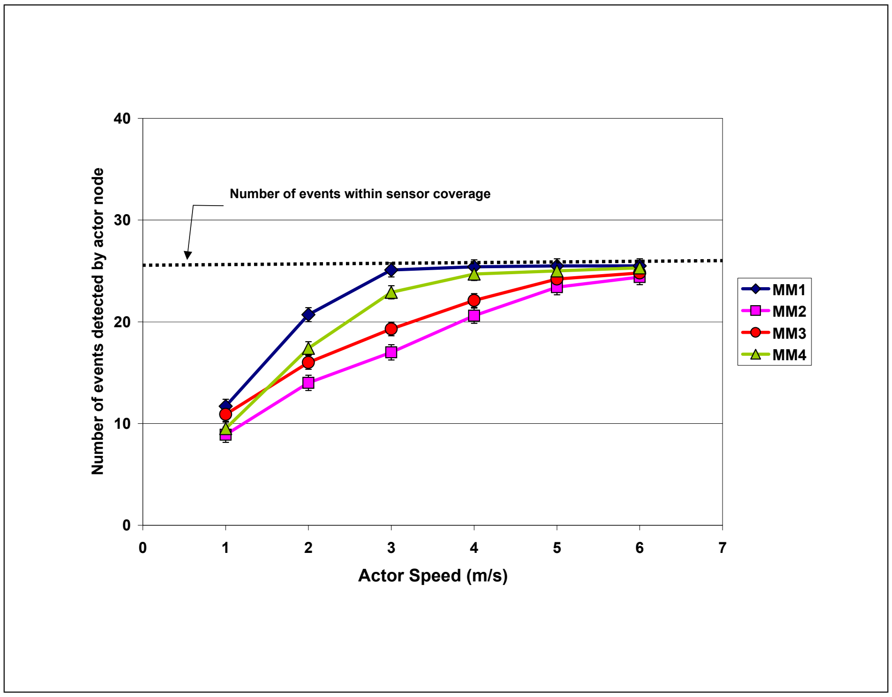

Figure 24: Average number of events detected by actor node, 25 node networks, t_d = 300 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

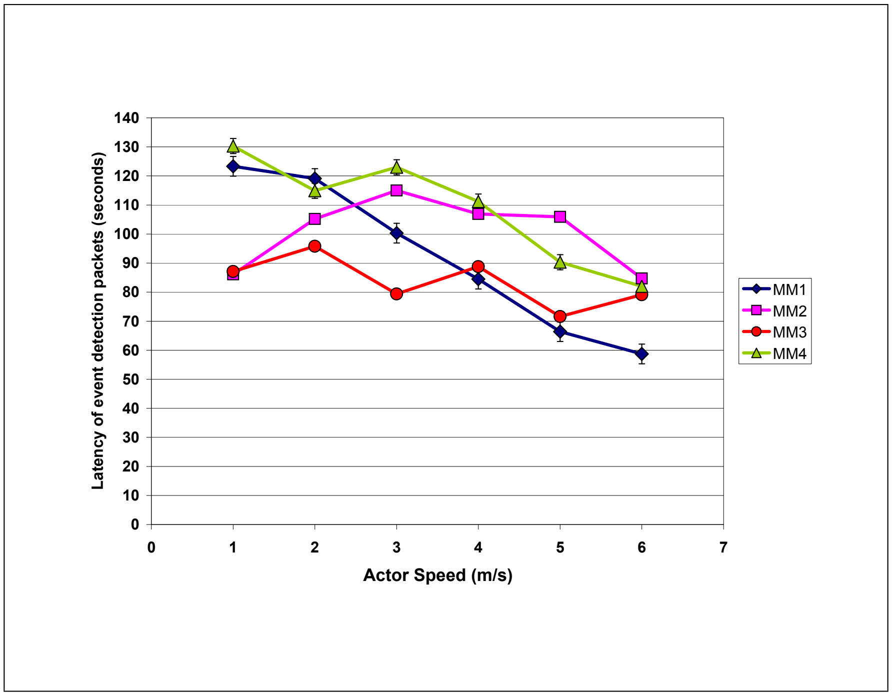

Figure 25: Average latency of event detection packets, 25 node networks, t_d = 300 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

4.8 Simulation Analysis - 2

There are a series of observations that can be made from the second set of simulation results:

1. At higher speeds, the number of events detected by the actor node using all the mobility models tries to converge.

2. At lower speeds MM4 detects more number of events than any other mobility model for networks composed of 100 nodes each. This can be deduced from Figure 20.

3. As the networks get sparser, MM1 performs better than MM4 at lower speeds itself. At higher speeds, MM4 catches up with MM1. Figures 22 and 24 present these results.

4. The average number of events sensed by sensor nodes is shown in Figures 20, 22 and 24 with a dotted line. At higher speeds using MM1 and MM4, the actor is able to detect 100% of the events that were sensed by the sensor nodes. This proves that some of the intelligent mobility models can give 100% detection by the actor node where ever there is sensor coverage.

5. The unexpected result is that MM1, a greedy nearest neighbor algorithm implementation is able to do as well as or better than MM4 at higher speeds in a few cases.

6. With respect to packet latency, MM3 stands out from the rest of the mobility models by providing the lowest latency across networks of size 100 and 50 nodes each. This is because it visits clusters of larger area more frequently with the goal of maximizing number of detected events. Clusters of larger area are fewer in a network and since it sticks to those clusters, it is able to provide a low latency. This fails to hold in the case of 25 node networks as seen in Figure 25. As the network gets sparser, there are more isolated clusters that occupy lesser area. As a result, the actor is forced to visit more clusters which occupy smaller area and is unable to provide low latency on event detection packets.

7. From Figures 21, 23 and 25, it is obvious that the latency of event detection packets is higher as the network gets sparser. Even the very efficient MM3 has higher latency for 50 node networks when compared to 100 node networks across all speeds.

8. Event detection packet latency was worse for MM2 than MM1, even though they are both static mobility models. This is yet another change in value when compared to the values in Table 2.

From the observations above, the table presented in Chapter III with the mobility model expectations (Table 2) can now be modified as shown in Table 5. The major miscalculation was that MM3 would perform better than MM1, but that did not happen. When dealing with sparsely connected networks, it is becoming clear that visiting every cluster is better than visiting a few selected clusters.

Table 5: Mobility model performance.

	$MM1$	$MM2$	$MM3$	$MM4$
Number of events detected	High	Low	Low	Highest
Latency of event detection packets	Low	High	Lowest	Low

4.9 Simulation Results - 3

In the previous set of simulations, the actor acted just as an event collector and there was no acting duration incorporated. In this set of simulations, by introducing an acting duration, the goal was to study if there would be any difference in the way the mobility models would operate. Going forward, only MM1 and MM4 are used in simulations to understand why MM1 matches MM4 in performance after certain speeds. The simulation parameters for this set of simulations are shown in Table 6.

To decide on the acting duration for the simulations, a simple limitation analysis was conducted. Assuming that the actor had knowledge of every single event that occurred in the field (like it would be in a fully connected network), a small program was written to find out the number of events the actor would be able to act upon, if it sequentially serviced the events one-by-one. The acting duration was varied as a multiple of the event duration (acting duration = {1/2, 1, 2, 4} x event duration) and the results are plotted as shown in Figure 26.

As a reference, the average number of events within sensor coverage in the sparsely connected 100 node networks is shown with a dotted line in Figure 26. As expected, the actor is capable of acting on fewer events as the acting duration is increased. With an acting duration of 75 and 150 seconds, at higher speeds all 50 events can be serviced. With an acting duration of 300 and 600 seconds, the number of events that can be acted upon falls below the number of events within sensor coverage. Hence the simulations were run against acting durations of 300 and 600 seconds and the results are plotted as shown in Figures 27 and 28 respectively.

Table 6: Simulation parameters to study the effect of acting duration.

Field dimension	200 m x 200 m
Number of sensor nodes	100
Number of actor nodes	1
Transmission range: Sensor node	20 m
Transmission range: Actor node	30 m
Actor speed	1-6 m/s
Acting duration	variable
Event diameter	30 m
Event duration	150 seconds
Number of events	50
Rate of event occurrence	every 432 seconds
Simulation duration	21600 seconds

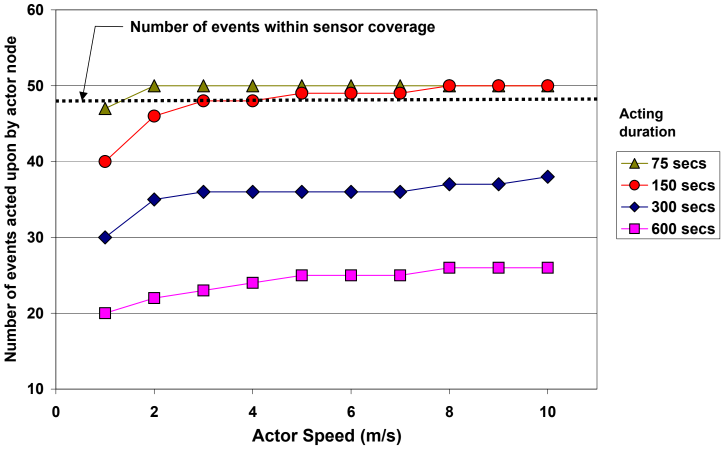

Figure 26: Limitation analysis for a fully connected network, t_d = 150 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

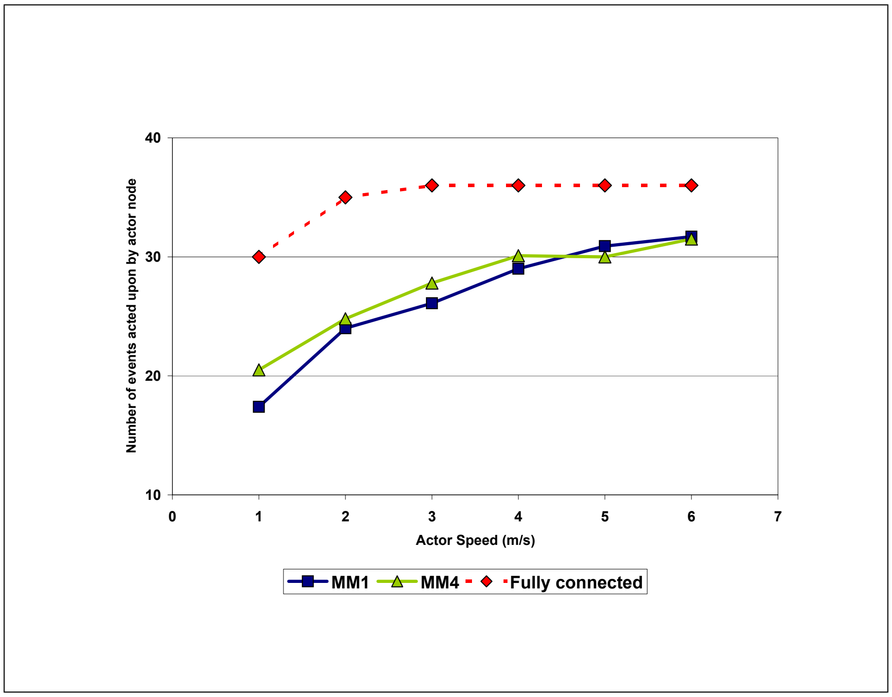

Figure 27: Average number of events acted upon by the actor node, 100 node networks, acting duration = 300 seconds, t_d = 150 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

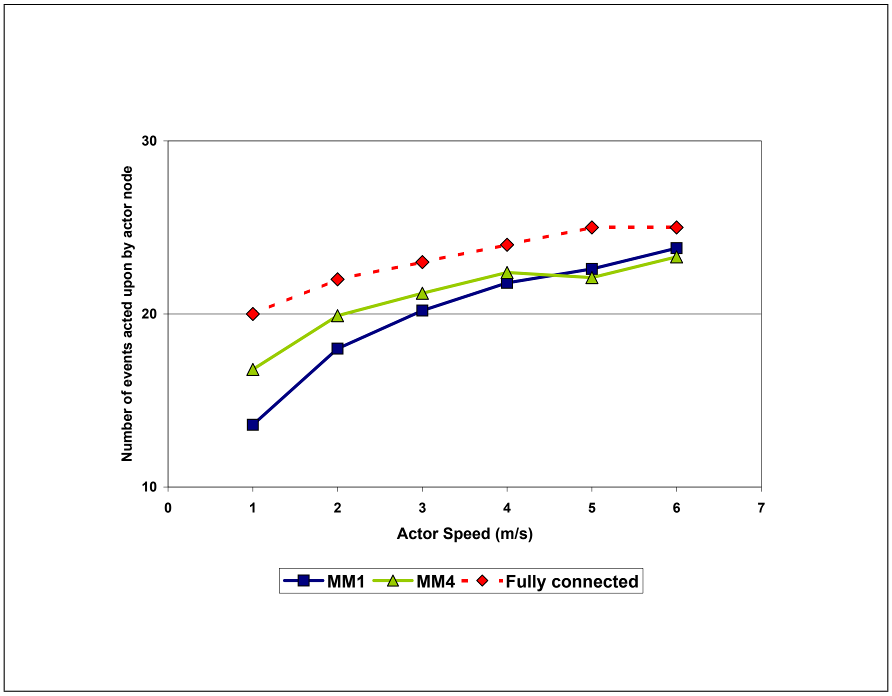

Figure 28: Average number of events acted upon by the actor node, 100 node networks, acting duration = 600 seconds, t_d = 150 seconds, 1/r = 432 seconds, 200 m x 200 m area, 50 events.

4.10 Simulation Analysis - 3

The third set of simulations has led to the following observations:

1. When the acting duration is >= 4 times the event duration, MM4 comes very close to the case of an actor in a fully connected network. This is an important finding, because it strengthens the case for deployment of a single actor in a sparsely connected network in the place of a single actor in a densely deployed, fully connected network.

2. The issue still remains as to how MM1 is able to catch up with MM4 or even perform better than MM4 at higher speeds.

3. These results re-emphasize the fact that given a sparsely connected network and ample actor resources, visiting all the clusters in a network may be beneficial.

4.11 Correlation Theory

The unanswered question that remains is this: Why is MM1 able to detect as many or more events than MM4 at higher speeds? Also, can a prediction be made about the speed beyond which MM4 is unlikely to perform better than MM1? There were many concepts tried to understand why MM1 was able to perform as good as or better than MM4 with respect to the average number of detected events, as shown below:

1. The networks were made sparser. The hope was that since MM4 incorporated cluster area, it may be better suited in judging where events may happen, unlike MM1 which would visit every cluster. The sparser the network, the lower the speed is at which MM1 can exceed MM4 in performance as seen in Figures 20, 22 and 24.

2. Initially only clusters of size > 1 were used in the static mobility models. This automatically helped eliminate a whole bunch of clusters. Clusters of size 1 would have been visited rarely with the dynamic mobility models as they tend to occupy less area, but nonetheless would have been visited by MM4. When all clusters were incorporated in the simulations, it took longer for MM1 to catch up with MM4, but it ultimately did catch up.

3. Decreasing the event duration from 300 seconds to 100 seconds. As the event duration was reduced, the number of events detected by the actor node with every mobility model decreased evenly and at high speeds, MM1 performed just as good as MM4.

4. Increasing the event rate and simultaneously decreasing the event duration also did not help. At intermediate speeds, MM1 was sometimes better than MM4 and finally at higher speeds, they were even.

Therefore, the following are the known facts: (i) MM4 is better than MM1 at low speeds for networks of larger sizes and with acting incorporated. Depending on the sparseness and other system parameters, the difference in the number of detected events can vary from few to many. (ii) The performance can match up only after a certain speed. At least in the 100 node networks studied, MM1 did not start out to be better than MM4 at 1 m/s. And, at higher speeds, MM1 can sometimes give better performance than MM4.

MM1 performing better than MM4 at higher speeds re-affirms that visiting every cluster is better than visiting clusters selected on criteria, especially when resources are available. The explanation and reasoning for this performance match up is described with the help of Figure 29. Consider 2 events E1 and E2 that are ideally separated in occurrence by the inter-arrival time of events, $1/r$ seconds. Since this thesis works with fixed duration events, the events occur for a duration of t_d seconds each. An event is detectable by an actor at any time within the t_d seconds. Assuming that the actor has detected the first event E1, soon after it has happened; it has approximately $(t_d + 1/r)$ seconds before when it has to reach the location of the next event, to be able to detect E2. If within this $(t_d + 1/r)$ duration, an actor can visit every cluster in the network once, then the chance of the actor missing an event is minimized. Therefore given a set of points, the nearest neighbor algorithm can be run to find the time it would take the actor to visit all the clusters once with an actor speed of 1 m/s. From this, the time to visit all the clusters once as the actor speed is increased can be easily deduced. Plotting the actor speed versus the time, it is now possible to tell the speed at which all clusters can be visited once by the actor for any combination of $(t_d + 1/r)$ seconds. This is the speed beyond which MM4 may not be able to detect more number of

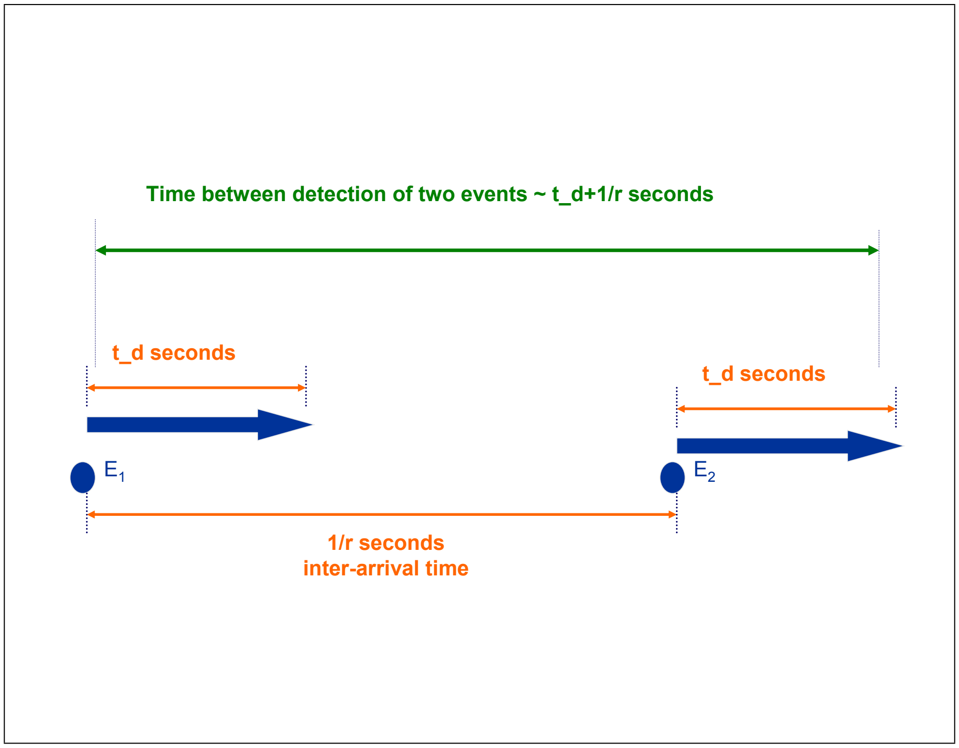

Figure 29: Explanation of correlation theory.

events than MM1 and is defined as the "crossover speed". Therefore crossover speed is the speed before which MM4 may perform better and after which MM1 may perform better. This reasoning correlates the actor speed and event detection time and is aptly referred to as the "Correlation Theory". In the following sections, this theory is applied to networks of different sizes.

4.11.1 100 node networks

Figure 30 shows the case of a 100 node network and the time it takes the actor to visit all the clusters once using the nearest neighbor algorithm, which is the basis for MM1. Also marked are the many $t_d + 1/r$ duration combinations that have been used in the simulations so far. Choosing a time combination for $(t_d + 1/r)$ as 366 seconds, the correlation theory leads to the prediction that beyond a crossover speed of 3 m/s, it is unlikely that performance with MM4 will be better than with MM1. There were 3 sets of simulations run to confirm if this prediction was correct:

1. $t_d = 216$ *seconds*, $1/r = 150$ *seconds*, case of event duration > inter-arrival time of

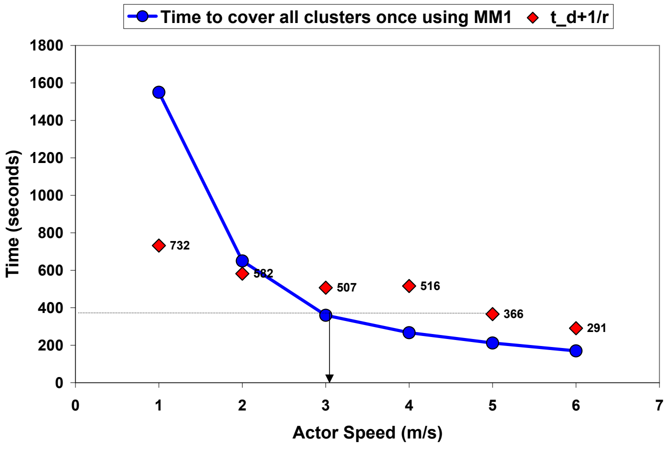

Figure 30: Correlation theory applied to 100 node networks, 200 m x 200 m area.

events

2. $t_d = 150$ *seconds*, $1/r = 216$ *seconds*, case of event duration < inter-arrival time of events

3. $t_d = 183$ *seconds*, $1/r = 183$ *seconds*, case of event duration = inter-arrival time of events

The results of these simulations are plotted as shown in Figures 31 - 33. The simulations were run against 10 networks of size 100 nodes each for different actor speeds and the results were averaged. For the case of $t_d >= 1/r$ seconds, the performance of MM1 is as good as MM4 or better, after a speed of 3 m/s as seen in Figures 31 and 33. In the case of when the event duration is less than the inter-arrival time ($t_d < 1/r$), it takes a little more than 3 m/s (almost 5 m/s) before when MM1 can catch up with MM4. This is because when the event duration is reduced, predicting which cluster to visit does better than following a static pattern. It takes a little more speed on the part of the actor to be as good as MM4 in performance as seen from Figure 32. Also evident from all the simulations is the fact that when the actor has an ample speed, it does better than MM4 by visiting each

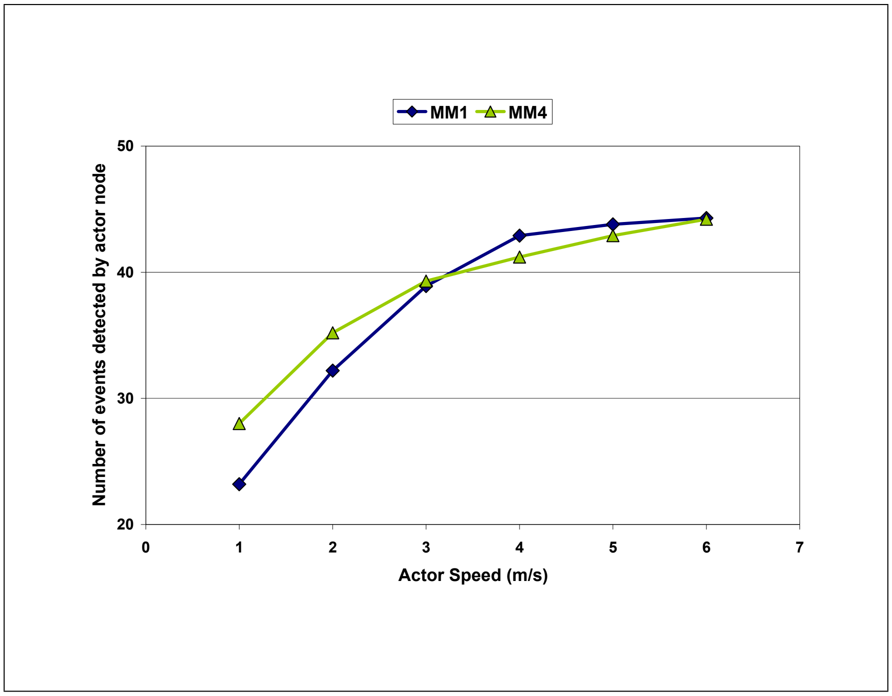

Figure 31: Average number of events detected by actor (event duration > inter-arrival time), 100 node networks, t_d = 216 seconds, 1/r = 150 seconds, 200 m x 200 m area, 50 events.

and every cluster. So these simulations have concurred with the reasoning behind why and when MM1 catches up with MM4 in performance.

In these set of simulations, there was no acting on the part of actor. It acted as a pure event collector. When there is acting duration incorporated, MM4 should obviously be better in predicting which cluster to visit next so that the number of lost events is minimized. This means, it should take higher speed on the actor's part for MM1 to catch up with MM4 when acting duration is incorporated. The simulations shown in Section 4.9 had a t_d of 150 seconds and $1/r$ = 432 seconds. From Figure 30, the crossover speed is close to 2 m/s. With an acting duration of 300 seconds, crossover speed is approximately 4.5 m/s. The same goes for an acting duration of 600 seconds also. These are deduced from Figures 27 and 28 respectively. Therefore when acting duration is incorporated, the correlation theory can provide the lower limit on the actor speed before which MM1 cannot outperform MM4.

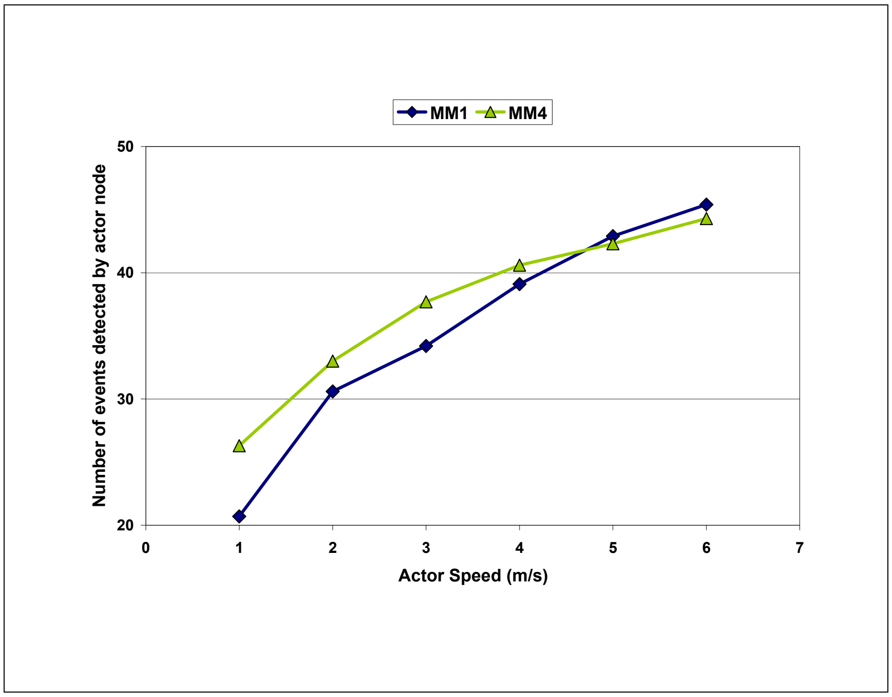

Figure 32: Average number of events detected by actor (event duration < inter-arrival time), 100 node networks, t_d = 150 seconds, 1/r = 216 seconds, 200 m x 200 m area, 50 events.

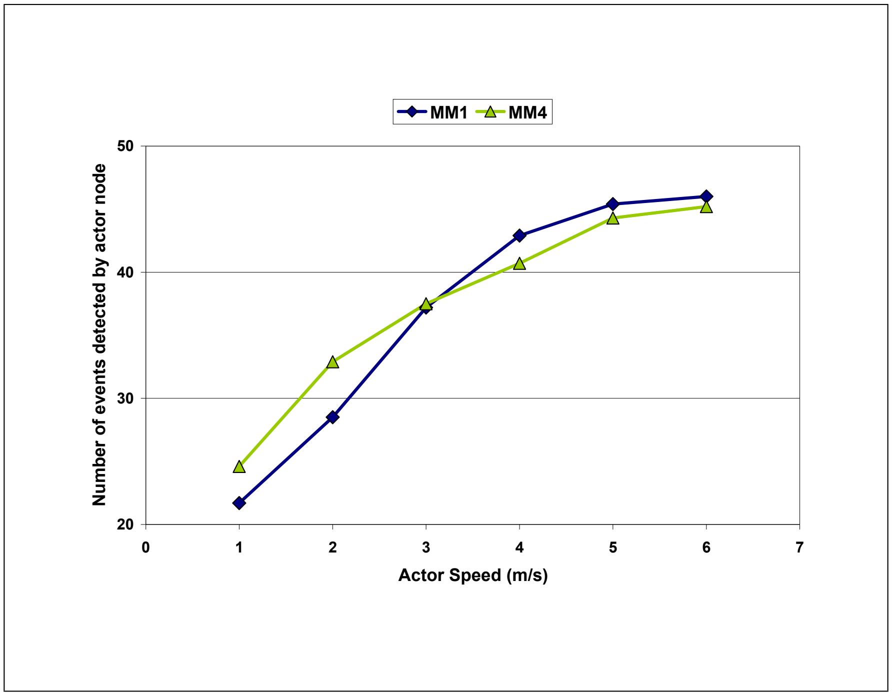

Figure 33: Average number of events detected by actor (event duration = inter-arrival time), 100 node networks, t_d = 183 seconds, 1/r = 183 seconds, 200 m x 200 m area, 50 events.

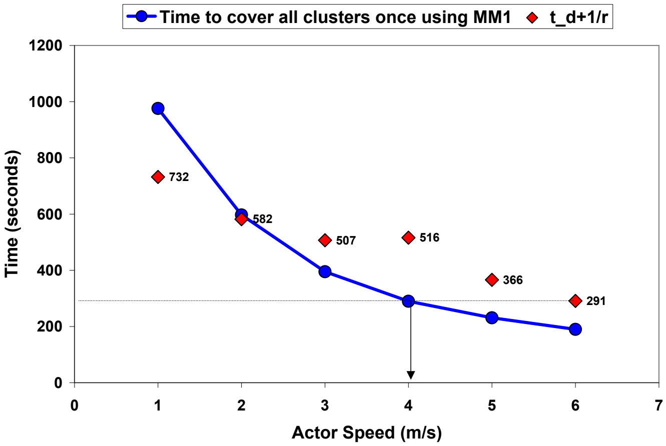

Figure 34: Correlation theory applied to 50 node networks, 200 m x 200 m area.

4.11.2 50 node networks

To make sure that the correlation theory presented in Section 4.11 holds for networks of any size, a study was done for 50 node networks as well. Figure 34 shows the time it takes to cover all the clusters once using MM1 and some combinations of $(t_d + 1/r)$. For a time combination of 291 seconds, Figure 34 predicts that the crossover apeed is approximately 4 m/s. Simulations were run with $t_d = 75$ seconds and $1/r = 216$ seconds in 10 networks of 50 node configurations each. The result for this set of simulations is plotted in Figure 35. The lines for MM1 and MM4 cross closer to 3 m/s and >3 m/s, MM1 takes over. Since this is a sparser network when compared to 100 node networks, MM1 is better even before the crossover speed and definitely superior after the crossover speed. Higher actor speeds may be required before a convergence in performance can be noticed.

From these results, it can be said with certainty that for networks of any size, given the time it would take the actor node to visit all the clusters once using MM1, the crossover speed can be predicted using the correlation theory. To get the best of both worlds (MM1 and MM4), an adaptive algorithm which combines both the mobility models can be devised.

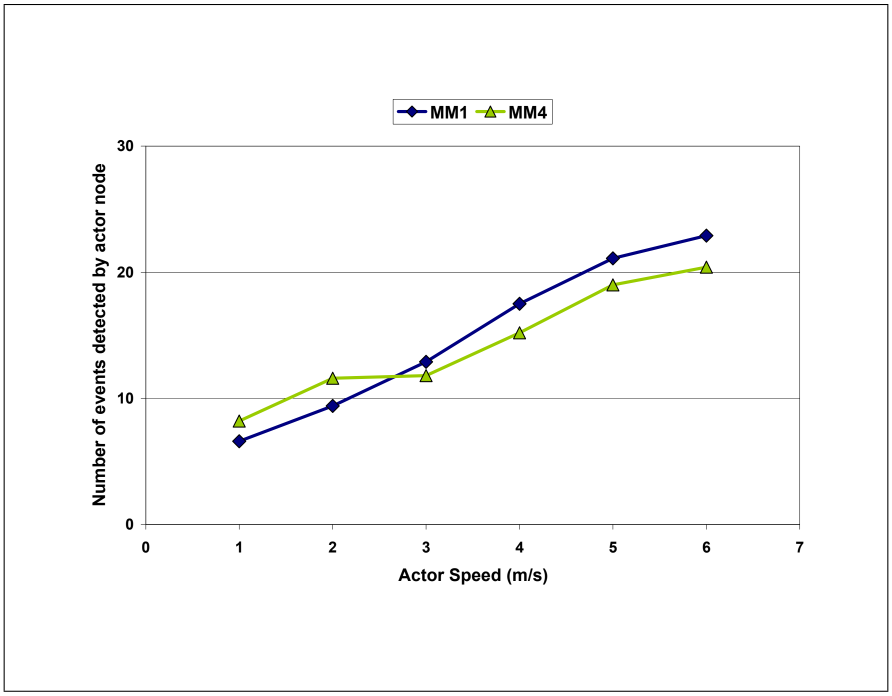

Figure 35: Average number of events detected by actor in 50 node networks (event duration < inter-arrival time), t_d = 75 seconds, 1/r = 216 seconds, 200 m x 200 m area, 50 events.

The correlation theory is very adaptable and can be extended easily. Consider that it takes twice as long to visit all the clusters in a network once using MM1 at 1 m/s when compared to the values shown in Figure 34. Doubling all the values from Figure 34, a new graph can be plotted as shown in Figure 36. Now for the same $t_d + 1/r = 291$ seconds, it can be predicted that it will take an actor speed of approximately 7 m/s before when the performance of MM1 can be better than MM4. If an application has an actor deployed that cannot move at this speed, it becomes obvious as to which mobility model has to be used for the maximum number of events to be detected.

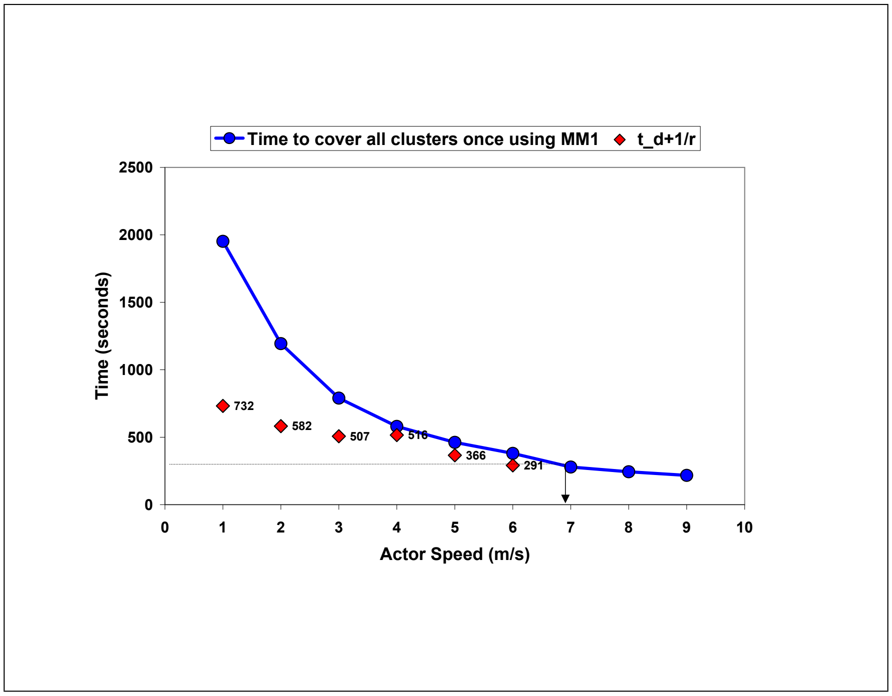

Figure 36: Extendability of correlation theory.

CHAPTER V

CONCLUSIONS, CONTRIBUTIONS AND FUTURE WORK

5.1 Conclusions

The presence of a single mobile actor in a sparsely connected network has been studied in this thesis. A sparsely connected network demands the presence of a mobility model for the actor node to enable it to perform its duties effectively. Intelligent mobility models for the actor node were generated and their performance was evaluated. The following are the conclusions:

- **Random mobility models are unsuitable for sparsely connected networks:** Random mobility models are non-intelligent mobility models and are not reflective of the network in which they are used. They may perform as good as an intelligent mobility model by chance in cases where (i) the area occupied by clusters constitutes to a larger area of the total field, or (ii) events occur for a sustained duration, or (iii) the inter-arrival time between events is large. But in other situations, the chance meetings cannot be translated into good performance when compared to an intelligent mobility model. Simulation results have been presented in Section 4.5 that support this conclusion.

- **A sparsely connected network can sense a high probability of events in an area of deployment:** A network of 25 nodes occupies approximately 25% of a 200 m x 200 m area and a network of 100 nodes occupies approximately 55% of the same area. Given the same set of events occurring in the field, the 100 node networks can sense 90% of the events and the 25 node networks can sense 50% of the events. Results from Sections 4.3 and 4.4 directly support this claim. This leads to the conclusion that a sparsely connected network can sense a high probability of events, and that the average number of sensed events is not proportional to the number of nodes in a network but to the total area occupied by the clusters of the network.

- **100% event detection by a single actor is possible:** In a WSAN, events have to be detected in a timely manner so that the action taken is valid. In a sparsely connected network, an event can be sensed by a sensor node only if it happens within the transmission range of the sensor node. Also, an actor in a sparsely connected network can get that event detection information only if it visits the cluster to which the sensor node that sensed the event belongs to, when the event is occurring. When the single actor acts as an event collector, it is possible to achieve 100% event detection with respect to the number of events sensed by the sensor nodes in the network. This has been shown with extended simulations in Chapter IV for a variety of scenarios. This is an important finding because it validates the intelligent mobility models that have been defined. To support 100% detection, a second actor can be employed to take care of the action.

- **Single actor in a sparsely connected network can be as good as an actor in a fully connected network when it also acts:** The primary function of an actor in a WSAN is to take action on an event. When it is deployed in a sparsely connected network, it takes on the additional responsibility of an event collector. When acting duration was incorporated, simulation results presented in Section 4.9 have shown that the number of events serviced can be very close to the results obtained with a single actor deployed in a fully connected network. This supports the economical aspect of deploying a sparsely connected network against a fully connected network when there is going to be only one actor in the field to take care of both detection and action.

- **Specific situations warrant specific mobility models:** This thesis has presented a suite of intelligent mobility models for an actor node in a sparsely connected network. At lower speeds in networks that have more area occupied by clusters, MM4 (the mobility model which tries to minimize the number of lost events) was the most efficient with respect to the number of detected events. At higher speeds in the same networks, the greedy nearest neighbor algorithm which is the basis for MM1 also did as well as MM4. The correlation theory presented in Section 4.11 not only explains why MM1 can perform as well as MM4, but also predicts the crossover speed beyond which this may happen. This prediction is of great use because it can help potential actors identify the mobility models they can use to achieve the best possible results. It also allows for the creation of a combination algorithm where in MM4 can be used until the actor reaches the crossover speed and MM1 can be used after the crossover speed to get the best possible results under all circumstances. Also, as the event duration decreases, the dynamic mobility models perform better in networks with dense clusters.

- **Early detection of events can be supported:** For applications that require detection of events at the earliest possible time, an actor may be allowed to follow MM3. The clusters that do not get visited as often using MM3 can be handed over to a second actor for monitoring.

5.2 Contributions

A mobile actor in a sparsely connected network is an area that has not been studied in the WSAN area. The contributions made from this thesis can be summarized as follows:

- **Novel concept:** Using the inherent clusters of a sparsely connected network to generate intelligent mobility models for the actor is a unique solution that results in an actor which moves following a path that is reflective of the network in which it is deployed. Simulations have proven that this idea is successful in maximizing the number of events detected in a timely manner. A suite of mobility models has been presented which allows for a wide variety of application scenarios where the mobility models can be used, as desired.

- **Adaptive clustering algorithm:** There are situations where the set of nodes active at any given time may vary. This leads to different clusters at different times. Example scenarios include (i) networks that use solar powered sensor nodes, or (ii) networks where the sensor nodes are following a sleeping pattern, or (iii) networks where resources have drained due to event occurrence. Since the clustering algorithm is adaptive, it can be re-applied to update the current state of the network. The intelligent mobility models can self-adjust to the current clustering structure and there is no need for external intervention. It remains to be studied as to how often the clustering algorithm has to be run to refresh the clustering information. This would require modeling of resource depletion on event detection and subsequent action.

- **Correlation theory:** The correlation theory has explained why MM1 is able to perform as well as or better than MM4 after the crossover speed. Given an actor node and its capabilities, it is now possible to decide the mobility model suitable to an application.

- **Implementation in ns-2:** The mobility model algorithms have been seamlessly integrated into ns-2. Example tcl code which is used to run a simulation is shown in Appendix A. The parameters that can be set from the tcl file are the mobility model, actor transmission range, acting duration, event duration, rate of event occurrence, actor speed, starting location of the actor node, and, network topology file. The network topology file has the area of the field, the sensor node transmission range and locations of the sensor nodes, which can all be modified. Changing any of the system parameters or loading a new network does not require the simulator to be rebuilt. This gives great flexibility to run numerous simulations to study the effect of varying the parameters, before the actual deployment of a network.

5.3 Future Work

There are many other extensions possible to the current work that can be studied further. Limiting the scope to a single actor, the following are some opportunities for future work:

- **Non-uniform rate of event occurrence:** Currently, there is a fixed rate of event occurrence across the entire field. It would be interesting to study the effect of the mobility models if different clusters have different rate of event occurrences. For example, a rectangular field can be divided into four quadrants with a different rate of event occurrence in each quadrant. It would be interesting to see if the dynamic mobility models are always superior when compared to the static mobility models and if the correlation theory continues to hold.

- **Apply the intelligent mobility models to fully connected networks:** There are many clustering algorithms available which can help form clusters based on criteria. Applying a limited broadcast of event detection information and letting the actor use the intelligent mobility models to chart its course, it would be valuable to study if location management can be eliminated. Location management is used primarily to broadcast the location of the actor so that the actor receives the event detection information at the earliest possible time, no matter where in the field the actor is. Visiting clusters periodically will be enforced by the intelligent mobility models, but simulations have to be conducted to see if performance can be maintained without location management.

- **Account for every event that occurs:** Currently, the sensor nodes do not maintain outdated information. Also, when a second event happens in the same location as a currently occurring event, information about the first undetected event is overwritten. By introducing memory on the sensor nodes, the next time an actor visits this cluster it can be informed about the lost event and this can be used to adjust the path generated by the mobility model. The current mobility models have to be enhanced to accommodate a feature like this. This also requires changes to be made to the sensor nodes in order to maintain history information.

- **Alter transmission range of sensor nodes:** If a new event happens in a cluster and the sensor nodes which sense the event are aware of the presence of the actor in the nearby vicinity, then they can use their long range transmission radios to alert the actor of this event. This is a temporary increase in transmission range of the sensor nodes and the expenditure in resources to do this has to be justified.

- **Optimize MM1:** Since the correlation theory has proven that beyond specific speeds MM1 can perform as well or better than MM4, MM1 can be further optimized to see if it can always perform better than MM4 in networks of all sizes and across all speeds of the actor.

- **Dynamic clustering:** Currently, the clustering information is static. Once found, it is retained for the duration of the simulation. The results obtained using the intelligent mobility models are dependant on the accuracy of the clustering information. If the resource depreciation due to an event can be modeled, simulations can be run to understand when partitioning of clusters occur. This can help to decide the frequency at which the clustering algorithm has to be re-applied so that dynamic clustering can be introduced.

- **Relax the constraint of visiting cluster centers:** The intelligent mobility models use the centers of clusters as points to visit. This constraint can be relaxed if it can be shown that it is sufficient to visit the edge of a cluster and still detect 100% of the sensed events in a timely manner.

APPENDIX A

PROGRAM LISTINGS

This appendix shows example code from files that are used in simulations. Simulations were run using the ns-2 simulator in a Linux environment. The input to the simulator is a tcl file. The tcl file has the definition of the network nodes and location information of nodes and events to run a simulation. The location information can be generated independently using a random number generator and can be stored separately as a scenario file. The tcl file reads the scenario file and enables the node and event information at times described in the scenario file. The ns-2 simulator is invoked with a simple command as shown below:

```
$ns wsan.tcl
```

A.1 Example scenario file

In this section, code fragments are shown to depict how five events are created (there are 100 nodes in the network labeled 0-99, actor node has label 100 and the events are labeled 101-105), initialized, enabled at their start times and disabled after the event's duration (216 seconds, in this case).

Listing A.1: Creating events

```
$node_(101) set X_ 185.447471
$node_(101) set Y_ 172.687770
$ns_ at 0.0000000  ‘‘$node_(101) setdest 185.447471 172.687770 50.0’’
$ns_ at 151.097716 ‘‘$node_(101) setdest 185.447471 172.687770 50.0’’
$node_(102) set X_ 43.140655
$node_(102) set Y_ 78.518660
$ns_ at 0.0000000  ‘‘$node_(102) setdest 43.140655 78.518660 50.0’’
$ns_ at 179.771102 ‘‘$node_(102) setdest 43.140655 78.518660 50.0’’
$node_(103) set X_ 112.227650
```

```
$node_(103) set Y_ 34.407180
$ns_ at 0.0000000  ‘‘$node_(103) setdest 112.227650 34.407180 50.0’’
$ns_ at 350.438970 ‘‘$node_(103) setdest 112.227650 34.407180 50.0’’
$node_(104) set X_ 97.415232
$node_(104) set Y_ 162.556358
$ns_ at 0.0000000  ‘‘$node_(104) setdest 97.415232 162.556358 50.0’’
$ns_ at 637.380905 ‘‘$node_(104) setdest 97.415232 162.556358 50.0’’
$node_(105) set X_ 58.864230
$node_(105) set Y_ 89.009639
$ns_ at 0.0000000  ‘‘$node_(105) setdest 58.864230 89.009639 50.0’’
$ns_ at 740.187918 ‘‘$node_(105) setdest 58.864230 89.009639 50.0’’
```

Listing A.2: Initializing events

```
$ns_ at 0.0 {[$node_(101) set netif_(0)] set Pt_ 0.000}
$ns_ at 0.0 {[$node_(102) set netif_(0)] set Pt_ 0.000}
$ns_ at 0.0 {[$node_(103) set netif_(0)] set Pt_ 0.000}
$ns_ at 0.0 {[$node_(104) set netif_(0)] set Pt_ 0.000}
$ns_ at 0.0 {[$node_(105) set netif_(0)] set Pt_ 0.000}
```

Listing A.3: Enabling events

```
$ns_ at 151.097716 {[$node_(101) set netif_(0)] set Pt_ 0.281838}
$ns_ at 179.771102 {[$node_(102) set netif_(0)] set Pt_ 0.281838}
$ns_ at 350.438970 {[$node_(103) set netif_(0)] set Pt_ 0.281838}
$ns_ at 637.380905 {[$node_(104) set netif_(0)] set Pt_ 0.281838}
$ns_ at 740.187918 {[$node_(105) set netif_(0)] set Pt_ 0.281838}
```

Listing A.4: Disabling events

```
$ns_ at 367.097716 {[$node_(101) set netif_(0)] set Pt_ 0.000}
$ns_ at 395.771102 {[$node_(102) set netif_(0)] set Pt_ 0.000}
```

```
$ns_ at 566.438970 {[$node_(103) set netif_(0)] set Pt_ 0.000}
$ns_ at 853.380905 {[$node_(104) set netif_(0)] set Pt_ 0.000}
$ns_ at 956.187918 {[$node_(105) set netif_(0)] set Pt_ 0.000}
```

A.2 Example tcl file

The following is a complete example of a tcl file which configures 100 sensor nodes, an actor node and 50 event nodes. The various system parameters that can be modified are also listed. After loading the scenario files, the simulation is run for 6 hours.

```
# Filename: wsan.tcl


# ======================================================================
# Define options
# ======================================================================
set val(prop)          Propagation/TwoRayGround ;# radio-propagation model
set val(netif)         Phy/WirelessPhy          ;# network interface type
set val(mac)           Mac/802_11               ;# MAC type
set val(PHENOMmac)     Mac                      ;# MAC type for phenomena
set val(ifq)           Queue/DropTail/PriQueue  ;# interface queue type
set val(ll)            LL                       ;# link layer type
set val(ant)           Antenna/OmniAntenna      ;# antenna model
set val(ifqlen)        50                       ;# max packet in ifq
set val(rp)            BCAST                    ;# routing protocol
set val(x)             200                      ;# grid width
set val(y)             200                      ;# grid hieght
set val(scenfile)      ./wsan.scn               ;#scenario file name
set val(eventfile)     ./events.scn             ;# event file name
set val(num_events)     50                      ;# number of events
set val(num_nodes)      100                     ;# number of sensor nodes
```

```
Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1

set val(energymodel)    EnergyModel     ;# simulate energy consumption
set val(energyBunny)    5000.0          ;#enough energy for nodes
set val(txPower)        0.36            ;# transmission power
set val(rxPower)        0.395           ;# reception power
set val(idlePower)      0.035           ;# idle consumption
set val(sleepPower)     0.001           ;# sleep power consumption
set val(transPower)     0.2             ;# transition power
set val(transTime)      0.005           ;# transition time
set val(simTime)        21600           ;#simulation duration - 6 hours
set val(total_nodes) [ expr {$val(num_nodes)+$val(num_events)+1} ];

# ======================================================================
# Main Program
# ======================================================================
set ns_      [new Simulator]
set tracefd  [open /dev/null w]
$ns_ use-newtrace
$ns_ trace-all $tracefd

set topo        [new Topography]
$topo load_flatgrid $val(x) $val(y)
set god_        [create-god $val(total_nodes)]
$god_ off
$god_ allow_to_stop
$god_ num_data_types 1
$ns_ set WirelessNewTrace_ ON
```

```
#configure phenomenon channel and data channel
set chan_1_             [new Channel/WirelessChannel]
set chan_2_             [new Channel/WirelessChannel]

# Transmit power for sensor nodes is 4.80696e-7 (20m)
Phy/WirelessPhy set RXThresh_ 4.80696e-7
Phy/WirelessPhy set CSThresh_ 4.80696e-7
# configure sensor nodes
$ns_ node-config \
        -adhocRouting $val(rp) \
        -llType $val(ll) \
        -channel $chan_2_ \
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
        -phyType $val(netif) \
        -macType $val(mac) \
        -PHENOMmacType $val(PHENOMmac) \
        -PHENOMchannel $chan_1_ \
        -topoInstance $topo \
        -rxPower $val(rxPower) \
        -txPower $val(txPower) \
        -idlePower $val(idlePower) \
        -sleepPower $val(sleepPower) \
        -transitionPower $val(transPower) \
        -transitionTime $val(transTime) \
        -energyModel $val(energymodel) \
        -initialEnergy 5000 \
      -agentTrace ON \
      -routerTrace ON \
      -macTrace OFF \
      -movementTrace OFF

for {set i 0} {$i < $val(num_nodes) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0
        $god_ new_node $node_($i)
        }

# Transmit power for actor nodes is 2.13643e-07 Watts (30m)
Phy/WirelessPhy set RXThresh_ 2.13643e-07
Phy/WirelessPhy set CSThresh_ 2.13643e-07
# configure actor node
$ns_ node-config \
        -adhocRouting $val(rp) \
        -phyType $val(netif) \
        -channel $chan_2_ \
        -macType $val(mac) \
        -agentTrace ON \
        -PHENOMchannel "off"

#actor node number
set a $val(num_nodes)
set node_($a) [$ns_ node]
$node_($a) random-motion 0
$god_ new_node $node_($a)
```

```
# Transmit power for phenom nodes is 2.13643e-07 Watts (30m)
Phy/WirelessPhy set RXThresh_ 2.13643e-07
Phy/WirelessPhy set CSThresh_ 2.13643e-07
# configure phenomenon node with the PHENOM routing protocol
$ns_ node-config \
        -adhocRouting PHENOM \
        -phyType $val(netif) \
        -macType $val(PHENOMmac) \
        -channel $chan_1_ \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace OFF \
        -movementTrace OFF

set m [expr {$val(num_nodes)+1} ]
puts $m
for {set i $m} {$i < $val(total_nodes) } {incr i} {
   set node_($i) [$ns_ node]
   $node_($i) random-motion 0
   $god_ new_node $node_($i)
   $ns_ initial_node_pos $node_($i) 10
   [$node_($i) set ragent_] pulserate 1
   [$node_($i) set ragent_] phenomenon CO
}

################################################################################
# Attach the sensor agent to the sensor node, and build a conduit thru
# which recieved PHENOM packets will reach the sensor agent's recv
# routine.
# attach a Sensor Agent (i.e. sensor agent) to sensor node
for {set i 0} {$i < $val(num_nodes) } {incr i} {
  set sensor_($i) [new Agent/SensorAgent]
  $ns_ attach-agent $node_($i) $sensor_($i)
  # specify the sensor agent as the up-target for the sensor node's
  # link layer configured on the PHENOM interface, so that the
  # sensor agent handles the received PHENOM packets instead of
  # any other agent attached to the node.
  #
  [$node_($i) set ll_(1)] up-target $sensor_($i)
}

################################################################################
# setup UDP connections to data collection point, and attach sensor apps
$node_($a) set X_ 100
$node_($a) set Y_ 100
$node_($a) set Z_ 0.000000000000
$ns_ at 0.000 ‘‘$node_($a) setdest 100 100 2.0’’
set sink [new Agent/UDP]
$ns_ attach-agent $node_($a) $sink
set actorapp [new Application/ActorApp]
$actorapp attach-agent $sink
[$node_($a) set ll_(0)] up-target $sink

# topology file
$ns_ at 0.0 ‘‘$actorapp topofile nodes.txt’’
$ns_ at 0.0 ‘‘$actorapp outfile results.txt’’
# mobility model
$ns_ at 0.0 ‘‘$actorapp mob_mod 3’’
```

```
# actor tx_range
$ns_ at 0.0 ‘‘$actorapp actor_tx 30.0’’
# actor speed (in m/s)
$ns_ at 0.0 ‘‘$actorapp speed 1’’
# actor location
$ns_ at 0.0 ‘‘$actorapp start_x 100’’
$ns_ at 0.0 ‘‘$actorapp start_y 100’’
# rate of event occurrence eg: 1 every 432 seconds = 1/216=0.0046
$ns_ at 0.0 ‘‘$actorapp event_rate 0.0046’’
#event duration in seconds
$ns_ at 0.0 ‘‘$actorapp event_dur 150’’
$ns_ at 0.0 ‘‘$actorapp nodeid $node_($a)’’
$ns_ at 0.0 ‘‘$actorapp acting_dur 0’’
$ns_ at 0.5 ‘‘$actorapp start $sink’’

for {set i 0} {$i < $val(num_nodes)} {incr i} {
  set src_($i) [new Agent/UDP]
  $ns_ attach-agent $node_($i) $src_($i)

  #attach and start sensor app to sensor node
  set app_($i) [new Application/SensorApp]
  $app_($i) attach-agent $src_($i)
  $ns_ at 0.5 ‘‘$app_($i) start $sensor_($i)’’
}

# Load in the scenario file
source $val(scenfile)
source $val(eventfile)
```

```
#Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(num_nodes) } {incr i} {
  $ns_ at $val(simTime) ‘‘$node_($i) reset’’;
}
$ns_ at $val(simTime).1 ”stop”
$ns_ at $val(simTime).1 ‘‘$actorapp stop’’
$ns_ at $val(simTime).1 ‘‘puts \”NS_EXITING...\” ; $ns_ halt’’
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
}

#Begin command line parsing
puts ‘‘Starting Simulation...’’
$ns_ run
```

REFERENCES

[1] Akyildiz, I. and Kasimoglu, I., "Wireless sensor and actor networks: Research challenges," *Ad Hoc Networks Journal (Elsevier)*, pp. 351–367, 2004.

[2] Capkun, S., Hubaux, J.-P., and Butty, L., "Mobility helps security in ad hoc networks," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, (New York, NY, USA), pp. 46–56, ACM Press, 2003.

[3] Chuah, M. C. and Yang, P., "A message ferrying scheme with differentiated services," *MILCOM*, vol. 3, pp. 1521–1527, 2005.

[4] Diggavi, S., Grossglauser, M., and Tse, D., "Even one-dimensional mobility increases ad-hoc wireless capacity," *IEEE International Symposium on Information Theory (ISIT)*, 2002.

[5] Downard, I., "Simulating sensor networks in ns-2," *NRL/FR/5522–04-10073, Naval Research Laboratory, Washington, D.C., May 2004.*

[6] Grossglauser, M. and Tse, D. N. C., "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 477–486, 2002.

[7] Gu, Y., Bozdag, D., and Ekici, E., "Mobile element based differentiated message delivery in wireless sensor networks," *International Symposium on a World of Wireless, Mobile and Multimedia Networks*, vol. 0, pp. 83–92, 2006.

[8] Gu, Y., Bozdag, D., Ekici, E., Ozguner, F., and Lee, C.-G., "Partitioning based mobile element scheduling in wireless sensor networks," *secon*, pp. 386–395, 2005.

[9] Jain, S., Shah, R. C., Brunette, W., Borriello, G., and Roy, S., "Exploiting mobility for energy efficient data collection in wireless sensor networks," *Mob. Netw. Appl.*, vol. 11, no. 3, pp. 327–339, 2006.

[10] Jea, D., Somasundara, A., and Srivastava, M., "Multiple controlled mobile elements (data mules) for data collection in sensor networks," *International Conference on Distributed Computing in Sensor Systems*, 2005.

[11] Jun, H., Zhao, W., Ammar, M. H., Zegura, E. W., and Lee, C., "Trading latency for energy in wireless ad hoc networks using message ferrying," *percomw*, vol. 00, pp. 220–225, 2005.

[12] Krishnakumar, S. S. and Abler, R. T., "Intelligent actor mobility in wireless sensor and actor networks," in *IFIP International Federation for Information Processing, Volume 248, Wireless Sensor and Actor Networks*, pp. 13–22, 2007.

[13] Kunz, T., "Multicasting in mobile ad-hoc networks: achieving high packet delivery ratios," in *CASCON '03: Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*, pp. 156–170, IBM Press, 2003.

[14] Luo, H., Ye, F., Cheng, J., Lu, S., and Zhang, L., "Ttdd: two-tier data dissemination in large-scale wireless sensor networks," *Wirel. Netw.*, vol. 11, no. 1-2, pp. 161–175, 2005.

[15] Luo, J. and J.-P.Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," *INFOCOM*, vol. 3, pp. 1735–1746, 2005.

[16] Mao, G., Fidan, B., and Anderson, B. D., "Wireless sensor network localization techniques," *Computer Networks*, 2007.

[17] McCanne, S. and Floyd, S., "ns–network simulator," *Available from http://www-mash.cs.berkeley.edu/ns/.*

[18] Melodia, T., Pompili, D., and Akyildiz, I. F., "A communication architecture for mobile wireless sensor and actor networks," in *Proceedings of IEEE Secon'06*, (Reston, VA, USA), 2006.

[19] Melodia, T., Pompili, D., Gungor, V. C., and Akyildiz, I. F., "A distributed coordination framework for wireless sensor and actor networks," in *Proceedings of ACM MobiHoc'05*, (Urbana-Champaign, IL), 2005.

[20] Shah, G. A., Bozyigit, M., Akan, O. B., and Baykal, B., "Real-time coordination and routing in wireless sensor and actor networks," in *6th International Conference, NEW2AN 2006*, (St. Petersburg, Russia), 2006.

[21] Shah, R., Roy, S., Jain, S., and Brunette, W., "Data mules: Modeling a three-tier architecture for sparse sensor networks," *IEEE SNPA Workshop*, 2003.

[22] Song, L. and Hatzinakos, D., "Dense wireless sensor networks with mobile sinks," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05).*, pp. iii/677– iii/680, 2005.

[23] Viswanathan, R., Li, J. T., and Chuah, M. C., "Message ferrying for constrained scenarios," *wowmom*, vol. 01, pp. 487–489, 2005.

[24] Wang, W., Srinivasan, V., and Chua, K.-C., "Using mobile relays to prolong the lifetime of wireless sensor networks," in *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, (New York, NY, USA), pp. 270–283, ACM Press, 2005.

[25] Zhao, W., Ammar, M., and Zegura, E., "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, (New York, NY, USA), pp. 187–198, ACM Press, 2004.

[26] Zhao, W. and Ammar, M. H., "Message ferrying: Proactive routing in highly-partitioned wireless ad hoc networks," in *FTDCS '03: Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03)*, (Washington, DC, USA), p. 308, IEEE Computer Society, 2003.

VITA

Sita hails from India and has been in the USA for the past 15 years. She has enjoyed living in the Southern part of the country, especially in beautiful Savannah for the last 10 years. She and her husband plan to relocate to their native land on completion of her Ph.D. so that their children can be close to their respective families. Sita is mother to two beautiful daughters.