ON-SITE TRACKING IN WIRELESS SENSOR NETWORKS

by

Baljeet Singh Malhotra

B.Tech

National Institute of Technology, Jalandhar

India, 1999

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in
MATHEMATICAL, COMPUTER AND PHYSICAL SCIENCES
(COMPUTER SCIENCE)

THE UNIVERSITY OF NORTHERN BRITISH COLUMBIA

April 2005

Your file Votre référence
ISBN: 0-494-04636-8
Our file Notre référence
ISBN: 0-494-04636-8

DEDICATED TO MY FATHER

"DADY JI".

Abstract

Tracking is one of the oldest practices that has evolved along with the advancement of technology. A similar practice, called hunting has been in existence since time immemorial. The primary objective in tracking is to find the whereabouts of a moving target. This target could be a human, or an animal, or a vehicle, etc. The task of tracking a moving target may have different objectives. Based on how these objectives are achieved, we classify the tracking problem into two broad categories: *On-site tracking* and *Off-site tracking*. The basic difference between these two approaches is the need for the physical presence of a mobile sink in the region of interest to track the target.

This thesis mainly deals with the On-site tracking problem in the context of wireless sensor networks. First, we characterize the On-site tracking problem for a single target case, and propose an ant-based approach to solve the problem. Then, we generalize the problem for the multiple targets case, and extend our ant-based approach to solve the generalized problem.

Next, we present the design of OSTSim, a simulation software. We developed this simulator for the performance study of the algorithms that could solve the On-site tracking problem in sensor networks. In addition to the basic ant-based algorithms, we proposed two efficient algorithms to solve the On-site tracking problem in sensor networks. Theoretical bounds for the tracking time and the number of messages generated by the sensor nodes have been derived for our algorithms. An extensive simulation study has been conducted, and the results show that our algorithms are efficient.

Contents

List of Figures

List of Tables

Publications from this thesis

1. Baljeet S. Malhotra and Alex A. Aravind. Energy Efficient On-Site Tracking of Mobile Target in Wireless Sensor Networks. Proceedings of International. Conference on Sensors, Sensor Networks and Information Processing (ISSNIP'04), Melbourne, Australia, pp. 43-48, 14-17 Dec.2004.

2. Baljeet S. Malhotra and Alex A. Aravind. A Master-Sink Based Model for On-Site Tracking of Multiple Mobile Targets in Wireless Sensor Networks. Proceedings of International Conference on Intelligent Sensors and Information Processing (ICISIP'05), Chennai, India. 4-7 Jan.2005.

3. Baljeet S. Malhotra and Alex A. Aravind. Path-adaptive On-Site Tracking in Wireless Sensor Networks. Submitted to the IEICE Transactions (Special Section on Parallel and Distributed Computing and Networking), 2005.

4. Baljeet S. Malhotra and Alex A. Aravind. OSTSim: Simulation Software for the On-site Tracking in Wireless Sensor Networks. Proceedings of the Western Canadian Conference on Computing Education (WCCCE'05), Prince George, BC, Canada, 5-6 May, 2005.

Acknowledgments

Thanks are due to my mentor, Dr. Alex Aravind without whom I could not have finished this thesis. I am very much indebted to him for his erudite supervision throughout my research pursuits at UNBC that produced this thesis. I am also very thankful to Dr. Waqar Haque and Dr. Patrick Montgomery for serving on my graduate committee and providing me with their valuable suggestions.

I extend a special thanks to Dr. Jernej Polajnar for his assistance and interest in my research work. I am very thankful to Dr. Staffen Lindgren for his interesting talk on the ants that triggered some of the initial thoughts outlined in this thesis.

The work carried out in this thesis is financially supported by the Advance System Institute of BC, Canada, through the ASI graduate scholarship. Here, I would like to take this opportunity to thank Dr. Bob Tait, Dean of Graduate Studies at UNBC for encouraging me to successfully compete for the ASI scholarship, and for the Graduate Travel award and a continuing TA. One more time many thanks to Dr. Waqar Haque, Chair Computer Science program, for generously supporting my research via RA and travel grants. I am also thankful to SPEATBC for their support via a scholarship. Thanks to Rob Lucas and Paul Stokes for providing me with computer related resources. My thanks are also due to the external examiner for his/her time to read this thesis.

I want to thank Dr. Rezaei, Dr. Deo, Dr. Brown, and Dr. Zahir for their general guidance and encouragement. I must thank my friends at UNBC, Sharmin, Sean, Tyler, Jey, Travis, and Jaco for engaging me in useful research discussions. A warm thanks to Dr. Mahi for her dinner invitations, and Abhinav, Pruthvi and Srinivas for giving me nice company.

I will be forever thankful to my most loving brother Navjot and most loving friend Manav, who have influenced my life in so many ways. Lastly but most importantly I can not thank enough my parents and my wife without whom my existence is meaningless. Their love and support will always be a driving force behind me.

Chapter 1

Introduction

1.1 Background

In the past few decades, advances in computer science and engineering, miniaturization of hardware devices, and technological improvements in network and communication infrastructure have revolutionized the computing and communication environment around us. The eighties and nineties of the past century have seen the rapid growth of the world wide web that has a significant impact on our day-to-day lives. In the last one-and-a-half decades ubiquitous computing has led the way to embed tiny devices in various physical objects and places. Wide spread usage of mobile devices like lap-tops, palm-tops, PDAs, mobile phones, etc. is redefining the ways in which information is exchanged between such devices in different parts of a geographical region. Integration of local and global information provided by thousands or even millions of such devices has started to make an impact on the information processing systems. Moreover, the opportunistic usage or on-demand availability of such useful information will change the way the world wide web works today.

Mobile computing is a technology which enables people to connect their mobile computing devices to network whenever and wherever they go[1]. Today most of the wireless networks, such as cellular telephones, personal communications systems, and wireless local area networks, are supported by static infrastructure (also called

backbone). The infrastructure consists of fixed base stations or access points, which are connected either through wires or by long range wireless transmissions to act as gateways and bridges in the network.

To cope with the demands of mobility and portability in using computers, mobile computing technologies are being enabled by rapidly emerging wireless communication systems based on radio and infrared transmission mechanisms. The history of wireless networks started in the early sixties[2] and the interest has been growing ever since. Ubiquitous access to information, anytime and anywhere, will characterize whole new kinds of information systems in the 21st century. Some of the challenges faced by these networks are the issues of mobility, energy efficiency, and security.

1.2 Wireless Sensor Networks

Wireless sensor networks (WSN) present a promising opportunity for realizing many practical applications that will become part of our daily lives[3, 4, 5]. Small, inexpensive, intelligent devices equipped with processor, memory, and radio components would work together in a coordinated fashion to report the phenomena of interest happening around them. These miniature sized devices (generally referred to as sensor nodes) are characterized by their limited power source and ad-hoc deployment in abundance due to cost effectiveness. Since recharging or replacing batteries for these sensor nodes is normally difficult because of various practical reasons, energy is considered as the most crucial resource in sensor networks.

1.2.1 Architecture

Due to specific objectives of various applications, WSNs do not have a fixed "fit-for-all" architecture. As surveyed in [6], the architecture of such WSNs would drastically differ at a node level as well as at the network level. At the node level chip size, storage capacity, and computational and communication power, are some of the important design considerations. At the network level nodes organization, and their communi-

cation strategies on a collective basis would influence the architecture. However, a common desirable characteristic across all the WSNs is minimal power consumption.

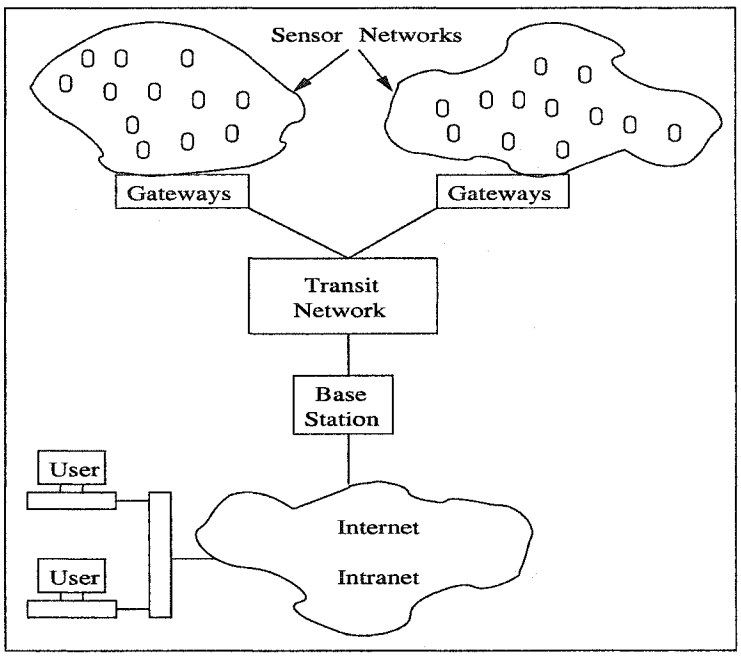

Figure 1.1: An Architecture of Wireless Sensor Networks

In most cases WSNs are required to be integrated with the existing wired or wireless networks. Figure 1.1 represents a typical design of the WSN architecture. In such networks observers sitting at various locations would be able to access the sensor nodes remotely.

1.2.2 Characteristics

WSNs have been characterized according to several parameters like node deployment, node capabilities, applications, energy and communication constraints etc. [3, 6, 7]. Some of these general characteristics include:

- *Ad-Hoc Deployment*: Nodes are generally designed to be deployed in a random fashion. An example of such deployment is where the nodes are dropped from an airplane onto a geographical region of interest, hence creating ad hoc networks.

- *Dynamic Topology*: The network topology may change randomly and rapidly at unpredictable times.

- *Scalability*: Due to their size and cost-effectiveness, nodes can be deployed in abundance. Nodes in hundreds or even thousands would cover a large geographical region. Their abundance would provide more accurate and up-to-date information about their physical environment.

- *Application Specific*: It is very likely that sensor nodes are designed for specific applications. That means functionality of nodes would be highly dependent upon the type of applications for which they are designed.

- *Energy Constraints*: In most cases, sensor nodes in a network rely on a limited supply of energy from the batteries or other exhaustible means. Furthermore, energy consumption of individual sensor nodes account for the overall lifetime of the deployed network.

- *Bandwidth Constrained*: Sensor nodes will primarily be dependent on their wireless radio components for communication. These components normally would have a limited bandwidth of communication channels.

- *Robustness*: In many applications, sensor nodes will be deployed to perform in extreme conditions that are not suitable for human interactions. In such cases, physical damages and individual failure of nodes will be common. In spite of this, WSNs are expected to perform well.

- *Self-Reconfiguration*: Due to energy constraints, ad hoc deployment, and robustness, WSNs are expected to have self-reconfiguration capabilities. Since in most of the cases sensor nodes would be stationary, the tasks of networking and self-reconfiguration would mostly depend on nodes' knowledge about their relative positions.

1.2.3 Applications

A future has been envisioned in which these sensor nodes would play crucial roles around us[8]. Surveillance, tracking, and smart spaces are some of the important applications of these networks. We list some of the most popular applications next.

- *Military Applications*: Military applications are one of the promising areas in which WSNs are being explored on a large scale. Such applications include tracking of moving objects, and monitoring of hostile environments. Deployment of sensor nodes in a hostile environment will reduce human injuries and other monetary costs. Nodes can be dropped from a plane over a vast geographical region to detect harmful and dangerous materials. Tracking of tanks and other vehicles in a war zone may provide the observer with better strategic decisions.

- *Environmental Studies*: Nodes capable of measuring variations in temperature, humidity, pressure, etc. can be very useful in environmental health monitoring systems. For example sensor nodes can be deployed for an early warning system to check the spread of forest fires. Habitat monitoring is one such application in which sensor nodes deployment have been experimented with successfully[9]. Environmental studies that involve visits to regions of harsh weather can be benefited greatly from the help of sensor nodes. Sensor nodes in such regions may be deployed to collect data over a period of time without involving human experts. This might reduce the operational costs. Also, such networks would avoid human interference with the natural habitat, which otherwise, in most cases require continuous study and hence frequent visits to the region of interest. This way WSNs tend to reduce the negative side effects of such studies, and at the same time, are capable of providing useful information about the habitants in the deployed region.

- *Civilian Applications*: In civilian applications, sensor nodes can be deployed to solve many urban problems like traffic congestion, vehicular parking, and

security. Sensor nodes can be deployed along the busy highways for route information and traffic diversions in case of accidents. Sensor nodes can also be deployed within the vehicles to collect and exchange useful information as they cross each other while they are moving along the roads/highways. Parking management is another application where sensor nodes can be used for effective parking services in busy urban places.

- *Industrial Applications* : Tracking inventory through sensor nodes in a warehouse is another application which has generated interest in retail and other related industries. Based on the information available with these sensor nodes, deployed in a large warehouse, inventory items can be managed efficiently.

Despite the feasibility and practicality of WSNs, there are some issues and many challenges to overcome to realize these applications in the near future. We list some of these issues and challenges next.

1.2.4 Issues and Challenges

- The foremost challenge posed by WSNs is the development of energy efficient sensor nodes. A major energy consumer in a sensor node is the radio component[10]. In a comparison study it is revealed that 3000 instructions can be executed for the same cost as the transmission of one bit over 100m[10]. In most cases nodes are battery operated and expected to be in deployed in a large number. In such condition changing batteries is not feasible when power is exhausted. Therefore conservation of energy posses a great challenge in these networks.

- Sensor nodes are expected to be deployed in a large number in most of the applications. The unpredictable nature of deployment conditions introduces significant scalability and reliability concerns[11]. Exposure of the hardware components to their deployed environment in extreme conditions, poses a great challenge for making these sensor nodes durable and robust.

- On the software front providing a high degree of efficiency to application level software, while keeping the precise control of various components at the lower level of sensor nodes, is still challenging[12] due to their deployment in a large number. However, the software development is extended to provide libraries for routing, tracking, synchronization and querying to support various applications. As the technology evolves, sensor networks must provide a unified architectural platform for existing as well as future applications.

- Nodes are expected to perform multiple tasks of sensing, processing and communication. Hence all these components are required to be on a single chip due to their size restrictions and energy constraints, which is one of the design issues.

- In most of the WSN related simulation studies a wide variety of simulation softwares is in common use, and that includes ns2[13], GloMoSim[14], Qualnet[15], Opnet[16]. Some of the challenges in using these simulation tools are discussed in Chapter 4.

This thesis deals with a particular type of tracking application that we have termed On-site tracking. An outline of the organization of the thesis is presented in the following section.

1.3 Contribution

This thesis contains the following contribution.

- We classify the tracking problem into two broad categories of *On-site tracking* and *Off-Site tracking* problem.

- Based on our classification, we present a taxonomy of the tracking problems.

- We characterize the On-site tracking problem for a single target case in the wireless sensor networks context, and propose an ant-based approach to solve the problem[17].
- We generalize the On-site tracking problem in sensor networks for the multiple target case and extend our ant-based approach to solve the problem[18].
- We propose two efficient algorithms to solve the On-site tracking problem in sensor networks[19].
- We present the design of a simulation software, which we call OSTSim[20] that we built for the performance study of the proposed algorithms.

1.4 Thesis Organization

The fundamentals of tracking and a taxonomy of the tracking problems is presented in Chapter 2. Next, in Chapter 3, we discuss On-site tracking in wireless sensor networks. In Chapter 4, the design of the OSTSim, a simulation software that we built and implemented for conducting a performance study of the methods that could solve the On-site tracking problem. The next two Chapters 5 and 6 can be read independently of Chapter 4. In Chapter 5, we present our ant-based approach to solve the On-site tracking problem. Next in Chapter 6, a path adaptive approach for On-site tracking has been discussed, and two efficient protocols have been proposed to solve the problem. Finally, in Chapter 7, we conclude the thesis while outlining some future directions to extend the work carried out in this thesis.

Chapter 2

Tracking

2.1 Introduction

Tracking is one the oldest practices that has evolved along with the advancement of technology. A similar practice, called hunting has been in existence since time immemorial. The very survival of a large majority of animals and other species is dependent on their tracking and hunting skills. For example, ants are considered a highly sophisticated and organized species, because of their ability to track down their food sources by an effective communication mechanism with the help of pheromones[21]. The adaptability to learn and invent new techniques for tracking with the help of sensing capabilities, power, and speed has proved to be very useful for many species, including humans. In ancient times humans used various techniques, like identification of foot prints, to track down animals for hunting purposes. In the modern world, tracking techniques are used to locate the objects of interest. Sniffing capabilities of dogs are utilized to locate harmful and dangerous materials. Sophisticated scientific equipment can be seen in use at airports, public places, buildings etc., for tracking people, goods, vehicles, and other objects.

Today the term "tracking" is used in various contexts, e.g. tangible and intangible entities such as tracking a parcel, economic growth, messages on the Internet, animals in the forest, and so on. In this thesis, we mainly deal with the tracking of tangible

entities in which the primary objective is to track the whereabouts of moving objects in WSN context. Next, we present a taxonomy of the tracking problem.

2.2 A Taxonomy of the Tracking Problems

The tracking problem has received considerable attention from the research community[22, 23, 24, 25, 26, 27]. There are two main entities that are involved in a typical tracking problem. They are:

- *Target* and
- *Tracker (Sink).*

A target is an entity of interest that is required to be tracked. This entity could be a living or non-living object such as human, animal, vehicle, etc. A tracker is an entity that is interested in tracking the moving target. Having said that, a tracker could be a stationary or non-stationary, living or non-living object. Examples of a tracker could be a human sitting in a fixed base station where the information about the target is being collected, or a moving vehicle that is receiving the information about the target. For generality, we often use the term **sink** to refer to the tracker.

The task of tracking a moving target may have different objectives. One such objective could be reporting the latest position of the moving target to a sink. Upon receiving the information, appropriate actions are anticipated. Imagine the situation of a battlefield, and consider that there are unfriendly vehicles that are roaming in the region of interest that are required to be tracked. Providing information about the unfriendly vehicles to the friendly forces would help them to make better strategic decisions. In this case the objective is to just report the position or any other relevant information to the sink. The sink may utilize this information to take appropriate actions that may include alarming the friendly forces. In such cases the sink need not actually be present in the region of interest.

Now consider another scenario of habitat monitoring in which a team of life scientists are riding in a vehicle (mobile sink) to track an animal to provide it with medical treatment. Here the vehicle has to actually move around in the region of interest to track the animal. Several approaches can be applied to provide the scientists with the latest information on the moving animal. In this case, a sink has to be physically present in the region of interest where the target is located.

We have seen two examples of tracking in which the moving target is being tracked; but each has a different objective. The basic difference in these two approaches is the need for the actual presence of a sink in the region of interest. Based on this observation, we classify the tracking problem into two broad categories.

- *On-Site Tracking*: In which a sink is eventually required to be present in the vicinity of the target, and
- *Off-Site Tracking*: In which a sink is not required to be present in the vicinity of the target.

Based on the awareness of the target and the sink, i.e. their knowledge about each other, we can further classify the tracking problem. There are four possibilities that can be considered here.

C1: The sink is aware of the target, and the target is unaware of the sink.

C2: The sink is unaware of the target, and the target is aware of the sink.

C3: Both are aware of each other.

C4: Both are unaware of each other.

A taxonomy based on the above classifications is presented in Figure. 2.1. Some of the combinations presented in the taxonomy have interesting applications and some of them do not. For example:

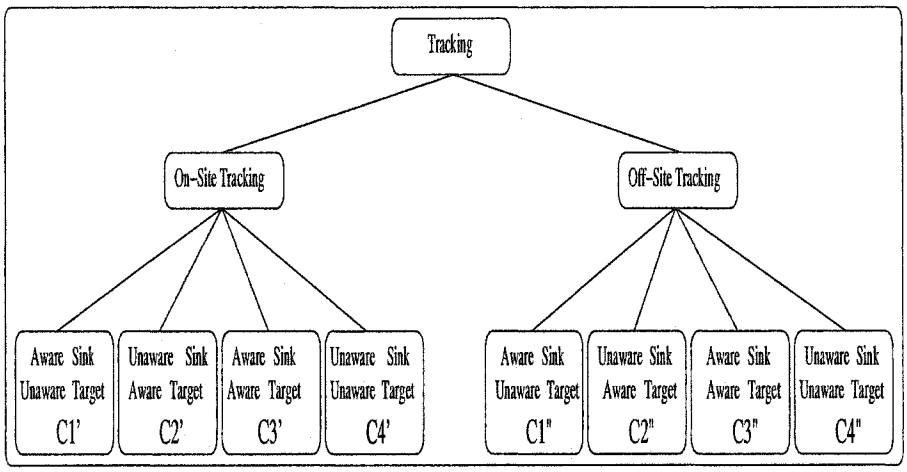

Figure 2.1: A taxonomy of the tracking problems

- Class C1' has several interesting applications such as tracking of vehicles, humans, etc. Ant-based On-site tracking presented in this thesis is an example of this class.

- Class C3' is a more challenging problem than C1'. In this class, a target is aware that it is being tracked, and hence it may devise an escape strategy, which makes the tracking harder. The pursuer-evader problem[27] is an example of this class.

- Class C1" has several interesting applications such as the tracking of vehicles, humans, etc. in which a moving target is unaware of being tracked. In such applications information can be collected at a base station and appropriate actions can be initiated subsequently.

- Again, class C3" is more challenging than C1" as in this case a target is aware of being tracked. For example, intruding activities at a border of a country where intruder might be aware that he or she is being monitored.

- To the best of our knowledge, we are not aware of any interesting applications for other classes C2', C4', C2", and C4".

Based on specific applications a sink can assume various roles such as pursuer, tracker, observer, etc. In the next section we discuss some of the popular tracking

approaches.

2.3 Tracking Approaches

2.3.1 GPS Based Tracking

The majority of today's tracking applications are based on Global Positioning Systems (GPS). With the advancement of technology it has become more viable and affordable, but GPS has its limitations. Some of the major constraints are the high costs and bulkiness of GPS receivers, and its non-usability in most indoor environments. The other key factor involved is the accuracy. The effects of the position accuracy in a GPS based tracking system as mentioned in [28] are:

- *Clock Errors*: Both the satellite and the receiver require very precise clocks to function properly. In that, the receiver's clock is typically a weak link due to cost considerations.

- *Atmospheric Errors*: Satellite signals travel over 20,000km, including a trip through the Earth's ionosphere and troposphere, and in both these regions charged particles distort the signal. In particular, for northern users such as Canadians, this error becomes greater due to the longer signal path through these latitudes.

- *Multipath Errors*: These result when the satellite signal is reflected off a nearby object, like a person, a building, a roof, trees, dense foliage, a mountain, etc. Unless the GPS device has a clear sky view, i.e. unobstructed in all directions and a minimum of 4 satellites in view, multipath errors are very likely.

- *Receiver Noise*: This depends on the quality of the electronics employed in the GPS unit and translates into the cost of the unit. Consumer GPS units are lower cost and higher noise devices.

- *Relativistic Corrections*: Both of Einstein's theories of general relativity and special relativity must be incorporated into the software/firmware built into the receivers. Expert physicists have questioned the correctness of the software/firmware. Errors in the subtle relativistic corrections lead to errors of tens of meters or larger in positional accuracy.

2.3.2 Sensor Networks Based Tracking

Recently, sensor networks have emerged as an alternative to the GPS based tracking systems due to their accuracy and versatility in sensing a variety of physical phenomena. Also, sensor network based tracking provides a viable option for other types of tracking where GPS based tracking may not be applicable such as indoor tracking.

Sensor networks are typically designed to monitor phenomena of interest happening around them. This task can be achieved as sensor nodes sense these phenomena, collect the data, and report to the observer at desired times. One of the fundamental utilizations of this collected data is that it can be used to track the moving targets. Since sensor networks are also equipped with radio components, they can spread the information about the target as soon as they detect its presence. We discuss in detail WSN based tracking in Chapter 5.

2.4 Motivation

As emphasized before, conserving energy is one of the main objectives of any sensor network application. Applying traditional routing based solutions to solve the tracking problem is generally costly. Moreover, the mobility of the sink introduces additional communication and computational overheads to keep track of its location in the network[29]. Recently, mobility has been explored in sensor networks for energy efficient data collection[30, 31, 32]. Physical mobile entities such as humans, robots, vehicles, and animals equipped with specialized sensor nodes may move around in the sensor field to collect the data by direct interactions with the sensor nodes. This

effectively reduces the communication costs.

Consider the On-site tracking of a moving target (C1' in Figure 2.1). In this particular class of application the mobility of the sink may be dependent on the mobility of the target. Recall our previous example of habitat monitoring in which a team of life scientists are riding in a vehicle to track an animal. Here the vehicle has to follow the movement of the animal to track it. The vehicle can be equipped with a powerful sensor node (making it a mobile sink) to collect the data from the sensor nodes along the track of the animal, instead of these nodes continuously routing the information to the moving sink. We believe that, this mobility dependency between the sink and the target can be effectively exploited to reduce the communication overheads.

2.5 Related Work

The tracking problem in general has been addressed in many previous attempts and various solutions have been proposed[22, 23, 24, 25]. In [22], a clustering based approach is proposed in which sensor nodes perform the task of sensing, predicting and communicating, and then they repeat these tasks as required by the cluster heads. In another approach[23], sensors detect the presence of a target for a threshold value. Nodes broadcast an alert message when this threshold value is reached and three similar messages have been received from their neighbor nodes. A trajectory of moving target is estimated as nodes alert their neighbor nodes while broadcasting these messages. In [26], a data centric approach called *directed diffusion* is presented in which named data is used to diffuse the interest in the network. Data returns on multiple paths of gradient setup. Enforcing the return data to use the most optimum path is one of the key features of this approach. In [29], the authors present a grid based approach to address the problem of continuous delivery of data from the source to the mobile sinks.

Recently mobility for data collection has been explored in sensor networks[30, 31,

32] for energy efficient data gathering. In [30], a University bus shuttle is being used as a mobile observer for data collection from the nodes deployed in the region of interest. An approach, which exploits the mobile nodes present in the sensor region as data forwarding agent is presented in [31]. Software based mobile agents have been proposed to solve the tracking problem in [24].

Our approach is different from [22, 23, 24, 25] in the way the sink communicates with the sensor nodes. In all of these approaches, the communication is based on multi-hop routing in contrast to the single-hop approach in which the mobile sink directly communicates with the sensor nodes. The primary objective of [30, 31, 32] is to exploit the mobility for data collection in sensor networks; but our focus is to utilize the mobility dependency for tracking the moving target. In this thesis we propose a set of algorithms that can be used to solve the On-site tracking problem. These algorithms include ant based-tracking and adaptive On-site tracking methods that are presented in chapters 5 and 6 respectively. Before we present these algorithms, we formally characterize the On-site tracking problem in the next chapter.

Chapter 3

On-site Tracking in Wireless Sensor Networks

In this chapter, we formally characterize the On-site tracking problem in the wireless sensor networks context. In order to do this, next we present the system model and the problem statement.

3.1 System Model and Problem Statement

We consider the sensor network system as a **quadruple** $S = < NR, SN, MUT, MT >$, where

- NR represents the network region,
- SN is a set of stationary sensor nodes deployed in NR,
- MUT is the mobile unit for tracking the moving target, and
- MT is the mobile target.

We make the following assumptions.

Assumption 3.1 *The network region, NR is connected. That is, a sensor node in NR can communicate to any other sensor node in NR.*

Assumption 3.2 *Sensor nodes are suitably deployed to cover NR. That is, every point in NR is in the sensing range of at least one sensor node.*

Assumption 3.3 *Each sensor node is aware of its position and has limited energy, memory, and computing power.*

Assumption 3.4 *The MUT is not constrained by energy, memory, and computing power.*

Assumption 3.5 *The speed of the MUT is greater than the speed of the MT.*

A particular target may be tracked in many time periods, each time for a particular mission. For example, a wounded animal might require continuous monitoring and treatment until its wound heals. The same animal might be tracked in a later time period for a different purpose that might require less frequent observations. Based on this observation we introduce the concepts of tracking mission and mission period.

Definition 3.1 *A* **tracking mission** *is the task of tracking a particular mobile target, and the duration in which the target is to be tracked is called its* **mission period**.

Next we characterize the On-site tracking problem in sensor networks. To formalize the problem, we introduce the concepts of *desired distance* and *frequency of closeness* as follows.

Definition 3.2 *The* **desired distance** δ *is defined as a threshold value of distance between a moving target (say MT) and a tracking sink (say MUT), required to initiate some actions at a particular time during the mission period, if necessary.*

Definition 3.3 *The* **frequency of closeness** f *between the target MT and its the tracking sink MUT is defined as the number of times MT and MUT are within the desired distance δ, during the mission period.*

The value of desired distance, δ, is application specific and normally much smaller than the diameter of the tracking region. Similarly, the frequency of closeness f is also application specific. Some applications might require one time closeness to the target, and others might require closeness for a finite number of times or may remain closer to the target during the entire mission. Different tracking missions of a same mobile target may have different values of δ and f.

Definition 3.4 *If the tracking of a moving target achieves the frequency of closeness, f, then we say that it is* **On-site tracking**.

The problem is to devise a method to achieve the On-site tracking for a single target. We present the algorithms to solve the On-site tracking problem of class C1' (shown in Figure 2.1) in Chapter 5 and 6.

3.2 A Generalization

On-site tracking of multiple targets by multiple sinks require coordination among the sinks, mainly to determine which sink tracks which target. Next we present the modified system model for the multiple targets case.

We consider the sensor network system as a **quadruple** $\mathcal{S} = < \mathcal{NR}, \mathcal{SN}, \mathcal{MUT}, \mathcal{MT} >$, where:

- $\mathcal{NR}$ represents the network region,
- $\mathcal{SN}$ is a set of n stationary sensor nodes deployed in the target region,
- $\mathcal{MUT}$ is a set of l mobile units (sinks) for tracking the moving targets. The tracking units are labeled as $MUT_1, MUT_2, ..., MUT_l$, and
- $\mathcal{MT}$ is a set of m moving targets that is to be tracked. The moving targets are labeled as $MT_1, MT_2, ..., MT_m$.

All of the assumptions except Assumption 3.5 considered in the single target case are assumed to hold for the multiple targets case. We substitute Assumption 3.5 with the following assumption.

Assumption 3.6 *The speed of any mobile target (MT_i) is less than the speed of any mobile tracking unit (MUT_j).*

Next we characterize the On-site tracking problem for multiple targets case by restating the definitions of the desired distance and the frequency of closeness.

Definition 3.5 *The* **desired distance** δ_{ij} *is defined as a threshold value of distance between a mobile target (say MT_i) and its corresponding tracking sink (say MUT_j), required to initiate some actions at a particular time during the mission period, if necessary.*

Definition 3.6 *The* **frequency of closeness** f_{ij} *between any target MT_i and its corresponding tracking sink MUT_j is defined as the number of times MT_i and MUT_j are within the desired distance δ_{ij}, during the mission period.*

Definition 3.7 *If the tracking of a MT_i by a MUT_j, $\forall i, j$, achieves the frequency of closeness f_{ij} then we say that it is* **On-site tracking**.

The problem is to devise a method to achieve the On-site tracking of multiple mobile targets. We present the algorithms to solve the C1' class (shown in Figure 2.1) of this problem in Chapter 5.

For simplicity, in this thesis we assume that the network region is a two dimensional plane with no obstacles in it, and the number of sinks is greater than or equal to the number of targets (i.e, $l \geq m$).

3.3 Generic Solutions for the On-site Tracking

On-site tracking can be solved by many existing general tracking methods. However, due to their generality these methods can be costly. In this section, we sketch some of these methods that will be used later as references for a comparative analysis.

The way the nodes and the sink communicate about the target, and how the sink makes a move to track the target based on the obtained information are the two factors that significantly affect the performance of the On-site tracking methods. The simplest approach to solve the On-site tracking problem would be continuously flooding the network with the target information. Another approach is to use an optimized routing method in which the messages travel to many intermediate nodes for every update. Such an approach is presented in [29].

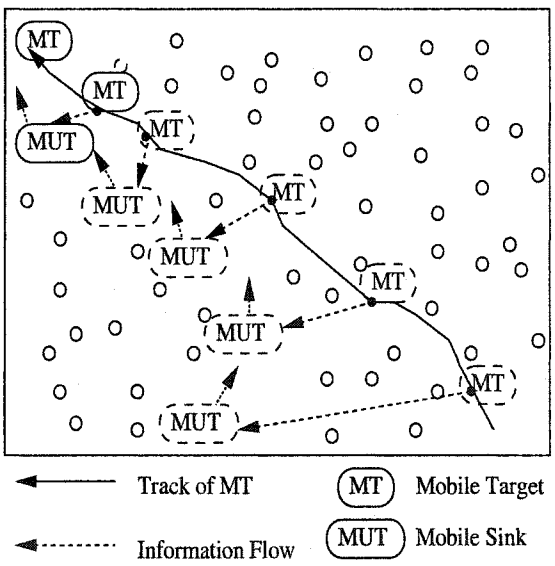

Figure 3.1: Routing based On-site tracking

The nodes surrounding the *MT* initiate the routing and then by multiple hopping the information about the *MT* reaches the mobile sink. Continuous reporting about the target is required by the nodes as the sink continuously changes position. A situation is depicted in Fig. 3.1. The arrows from the trace line toward *MUT* refer to a continuous communication between the mobile sink, *MUT*, and the sensor nodes. Depending on the information obtained, the *MUT* changes its direction to reach the mobile target, *MT*.

Maintenance of some logical structure, like a grid[29] is normally required to effectively keep track of the location of the sink as well as the target. This complicates the solution and it is costly because mobile sink requires a continuous update from the

sensor nodes to effectively keep track of the target in the network region. Previously discussed approaches (flooding and grid based) for the On-site tracking normally require multiple hopping of messages, which might waste a considerable amount of energy, and subsequently that would reduce the life time of the network.

The objective in this thesis is to explore the solutions for the On-site tracking problem that can effectively exploit the mobility dependency inherent in the problem. Before we discuss about these solutions in Chapters 5 and 6, in Chapter 4 we present OSTSim, a simulation software that we built for the performance study of the methods discussed in this thesis. However, Chapters 5, 6 and 7 can be read independently of Chapter 4.

Chapter 4

OSTSim: On-site Tracking Simulator

4.1 Introduction

This chapter presents the architecture of a simulation software, which we call OSTSim. We developed OSTSim to evaluate the performance of the algorithms that can solve the On-site tracking problem.

As mentioned in Chapter 1, there are many public domain softwares available to conduct the simulation studies in wireless networks. Since most of these simulators were initially developed for specific studies and then modified thereafter by the contributions of other researchers in an ad-hoc fashion, they lack structure and proper documentation. Their minimal documentation, increased size, and generality normally:

- make the learning curve steep,
- incur huge execution time,
- allow less control for certain modifications and extensions, and
- lack features required for the specific studies.

Therefore, during our search for a simulator, we eventually chose to develop our own simulation test-bed. The major advantage we have with this decision is a better understanding of the underlying design of the software that has in fact provided us a better control over the simulator.

OSTSim is limited to conducting the performance study of the On-site tracking methods in wireless sensor networks. For the On-site tracking methods, we are mainly interested in the energy spent by the sensor nodes and the tracking time of the sink. We compute the energy expenditures based on the messages sent and received by the sensor nodes. We assume that the communication network is reliable. To develop OSTSim, we systematically followed a methodology that we present next.

4.2 Problem Statement

A sensor network with specified number of targets and sinks are assumed. For a given tracking approach for a specified set of parameters such as sink speed, desired distance, and area size, the simulator should compute the tracking time of the sink and the energy consumption of the sensor nodes. We develop the simulation system through analysis and design. The analysis phase involves making the use-case diagrams, gathering the system specifications, and constructing the ER diagrams. The design phase involves constructing the class diagrams. We start with the use-case diagram[33].

4.3 Use-Case Diagram

Constructing an use-case diagram involves: (i) identification of the actors, (ii) identification of the use-cases (the ways of using the system), and (iii) refining the use-cases and setting the relationships. The researcher who is interested in evaluating the performance of the algorithms, is the only actor in the system. This actor can use the system by setting the parameters of the simulator and getting the results on defined

metrics. Figure 4.1 represents the use-case diagram of OSTSim.

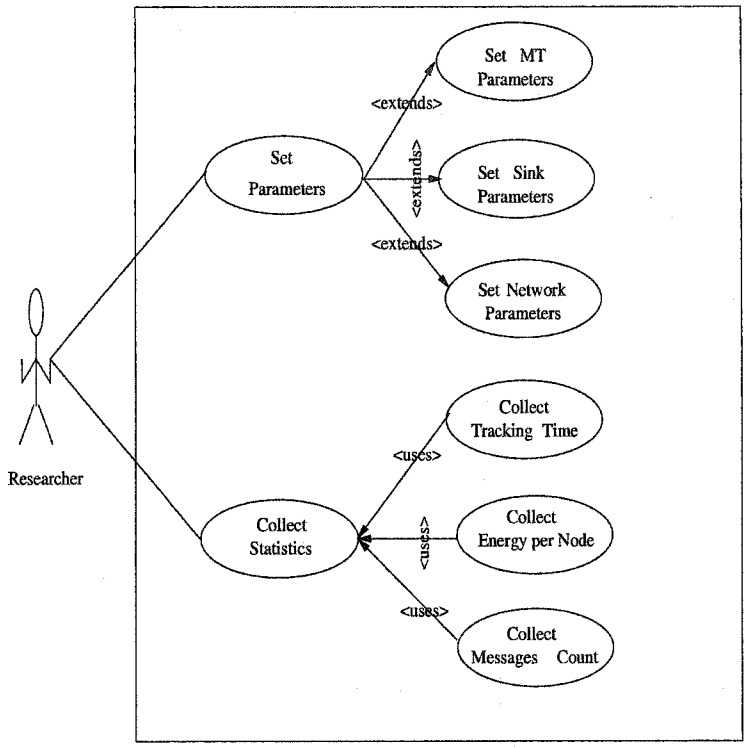

Figure 4.1: OSTSim use-case diagram

4.4 System Specification

The sensor network consists of the following.

- A geographical region of area size a.
- A set of n sensor nodes with communication range r.
- A set of l mobile sinks each with a maximum speed v_{mut}.
- A set of m mobile targets each with a maximum speed v_{mt} and a random mobility pattern.

We make the following system assumptions.

- The geographical region is a two-dimensional rectangular plane without obstacles.
- The sensor nodes are suitably deployed to cover the network region.
- The targets never cross the network boundary.
- We assume the same radio model as referred in [34]. In this model $E_r = 50nJ/bit$ and $E_s = 50 + .1 \times R^2\ nJ/bit$, where E_r is the energy required to receive one bit and E_s is the energy required to send one bit at R distance.

Figure 4.2 represents the setup of three main constituents of OSTSim.

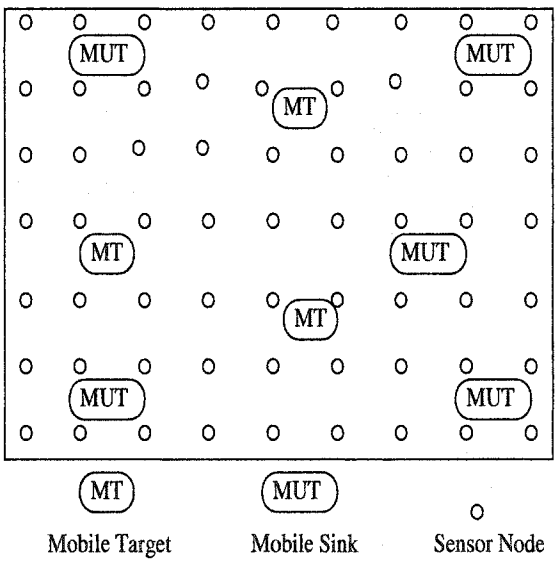

Figure 4.2: OSTSim system setup

Following are the main activities in the system.

- Sensor nodes collect and store the information of the targets.
- Sensor nodes supply the information about the targets when sinks request them.
- The system (simulator) collects the statistics.
- Direct Inputs:

- Transmission range of a sensor node, r.
- Size of the network region, a.
- Sink and Target speeds, v_{mut} and v_{mt} respectively.
- Desired Distance, δ.
- Mission Period, p_m.
- Number of simulation runs.

- Derived Inputs:
  - The number of sensors required is computed based on the network area and the transmission range of a sensor node.
  - The sensor nodes are placed in such a way that they cover the region.
  - The initial positions of the sinks and targets are determined randomly.

Based on the system specifications, next, we construct the entity relationship (ER) diagram for the system.

4.5 Entity Relationship (ER) Diagram

The design of an ER diagram involves: (i) identification of the entities in the system, (ii) identification of the characteristics of these entities, and (iii) identification of the relationships between these entities. Figure 4.3 represents the ER Diagram of system. Next, based on the use-case diagram, the system specifications and the ER diagram, we construct the class diagram.

4.6 Class Diagram

A class diagram depicts the structural aspect of the system. A class essentially has three logical components: data attributes, operations that involves services from other

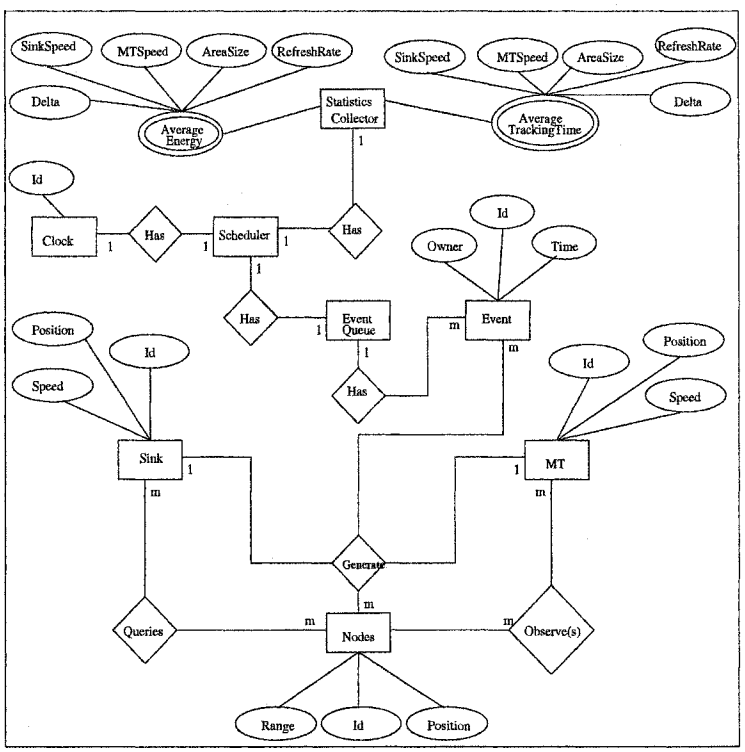

Figure 4.3: OSTSim ER Diagram

classes, and operations to access the member attributes of the class. Its development involves mainly the following steps.

1. Identification of the objects and their data attributes. This can be obtained by analyzing the problem specification, the use-case diagram and the ER diagram.

2. Abstraction of the objects into the classes.

3. Identification of the relations among various objects (referred to as *links*), and abstracting them into the relations between the corresponding classes (referred to as *associations*). This involves finding the relation, labeling it properly, and determining its cardinality.

4. Refine the class diagram to identify all the main operations using suitable interaction diagrams.

5. Further refining the class diagram to get a final class diagram.

4.6.1 Identification of Objects

Mainly there are three classes of objects involved in the system.

1. Objects in the simulation system.
    - Simulation-Interface,
    - Scheduler,
    - Statistics-Collector,
    - Simulation-Clock, and
    - Event-Queue.
2. Objects in the sensor network.
    - Sensor Node.
3. Objects in the On-site tracking methods.
    - Mobile Target, MT and
    - Mobile Sink, MUT.

4.6.2 Abstraction of Classes

In this system, each object has its corresponding class. Description of all the classes is given in Table 4.1.

1. Simulation-Interface,
2. Scheduler,
3. Statistics-Collector,
4. Simulation-Clock,
5. Event-Queue,

Class	Data Members (attributes)
Simulation-Interface	**int:** NofNodes, NofSimulations, NofMT, NofSink **float:** Length, Breadth, SensingRange, CellSize, DesiredDistance
Scheduler	**object:** EventQueue, Clock, StatisticalData
Statistics-Collector	**object:** StatisticalData
Sensor-Node	**int:** NodeID, NodexPos, NodeyPos **object:** TargetData
MT	**int:** MTID **float:** MTxPos, MTyPos, MTNewxPos, MTNewyPos, MTSpeed **bool:** MTCaptured **object:** StatisticalData
MUT	**int:** MUTID **float:** MUTxPos, MUTyPos, MUTNewxPos, MUTNewyPos, MUTSpeed **object:** StatisticalData

Table 4.1: Description of OSTSim classes

6. Sensor-Node,

7. MT, and

8. MUT.

Next, we identify various associations existing among these classes.

	Simulation-Interface	**Scheduler**	**Statistics-Collector**	**Sensor-Node**	**MT**	**MUT**
Simulation-Interface	–	Initiates (1-1)	Initiates (1-1)	Initiates (1-n)	Initiates (1-1)	Initiates (1-1)
Scheduler	Initiates (1-1)	–	Updates (1-1)	Updates (1-m)	Updates (1-m)	Updates (1-m)
Statistics-Collector	Supplies (1-1)	– (1-1)	–	–	–	
Sensor-Node	– (m-1)	–	–	–	–	Reports (m-m)
MT	–	–	–	–	–	–
MUT	–	–	–	Requests (m–m)	Tracks (1-1)	–

Table 4.2: Association among various OSTSim classes

4.6.3 Identification of Associations

Table 4.2 contains the association among the various classes. This table allows us to draw the first level class diagram representing the association among various classes, as shown in Figure 4.4. Classes in this diagram contain the main data members. The next step is to identify the operations of these classes, which invoke services from other classes. This can be achieved through building and analyzing interaction diagrams that we do in the next section.

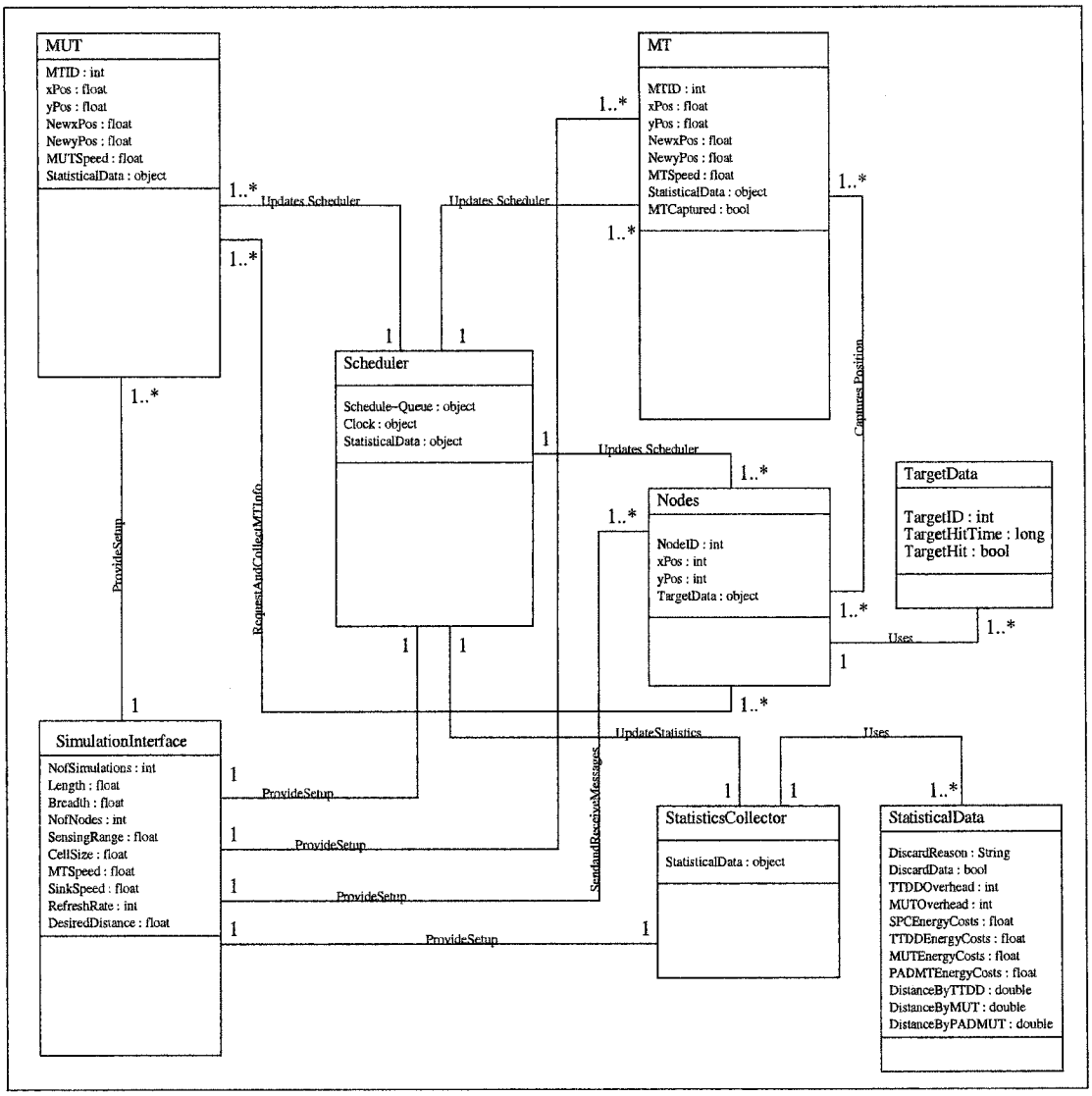

Figure 4.4: Class Digram - After first refinement

4.6.4 Interaction Diagram

Interaction diagrams illustrate the dynamic behavior of a system. That is, showing interaction among the objects. In this context, only the objects which have non-trivial interaction are considered. Objects considered in the interaction diagram are: (1) Nodes (2) Mobile Target, MT, (3) Mobile Sink MUT, and (4) System Interface. The interactions of these objects are given in Figure 4.5.

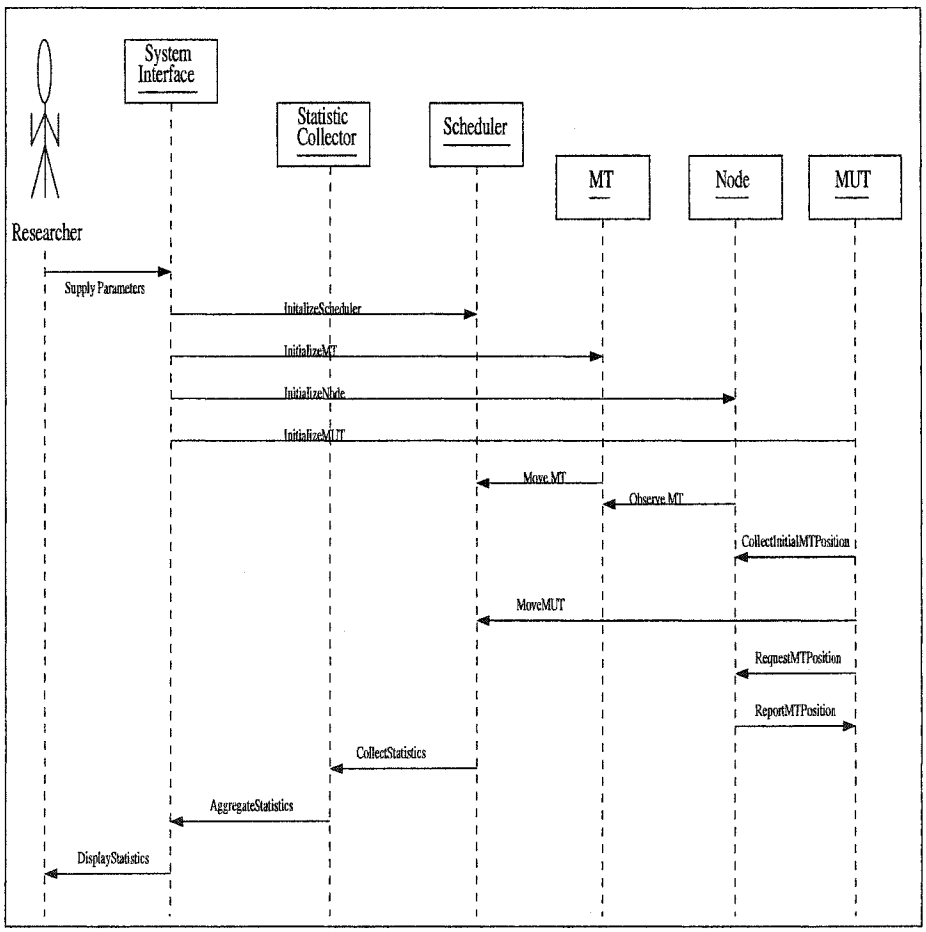

Figure 4.5: Interaction diagram of OSTSim

4.6.5 Refinement of Classes

Based on the interactions between various objects as shown in the interaction diagram in Figure 4.5, we refine the classes to include the operations. After the first

refinement and then including the operations, the class diagram is presented in Figure 4.4. Further refinement involves carefully walking through the operations given in the class diagram as shown in the Figure 4.4 to check their completeness. This refinement is shown in Figure 4.6. Complete class diagram listing all the required operations and the member variables required to perform the system operations is given in Figure 4.7.

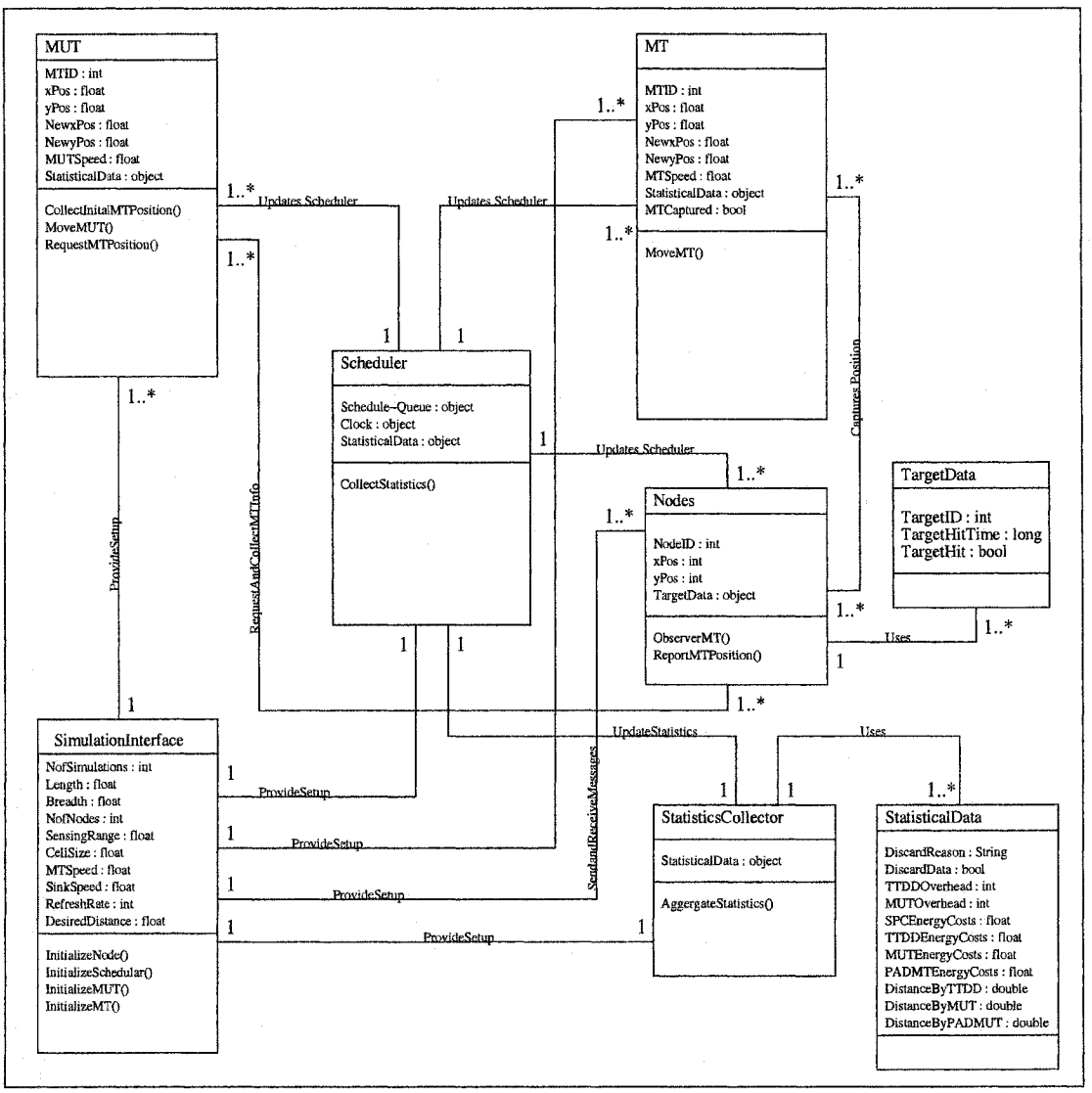

Figure 4.6: Class Digram - After second refinement

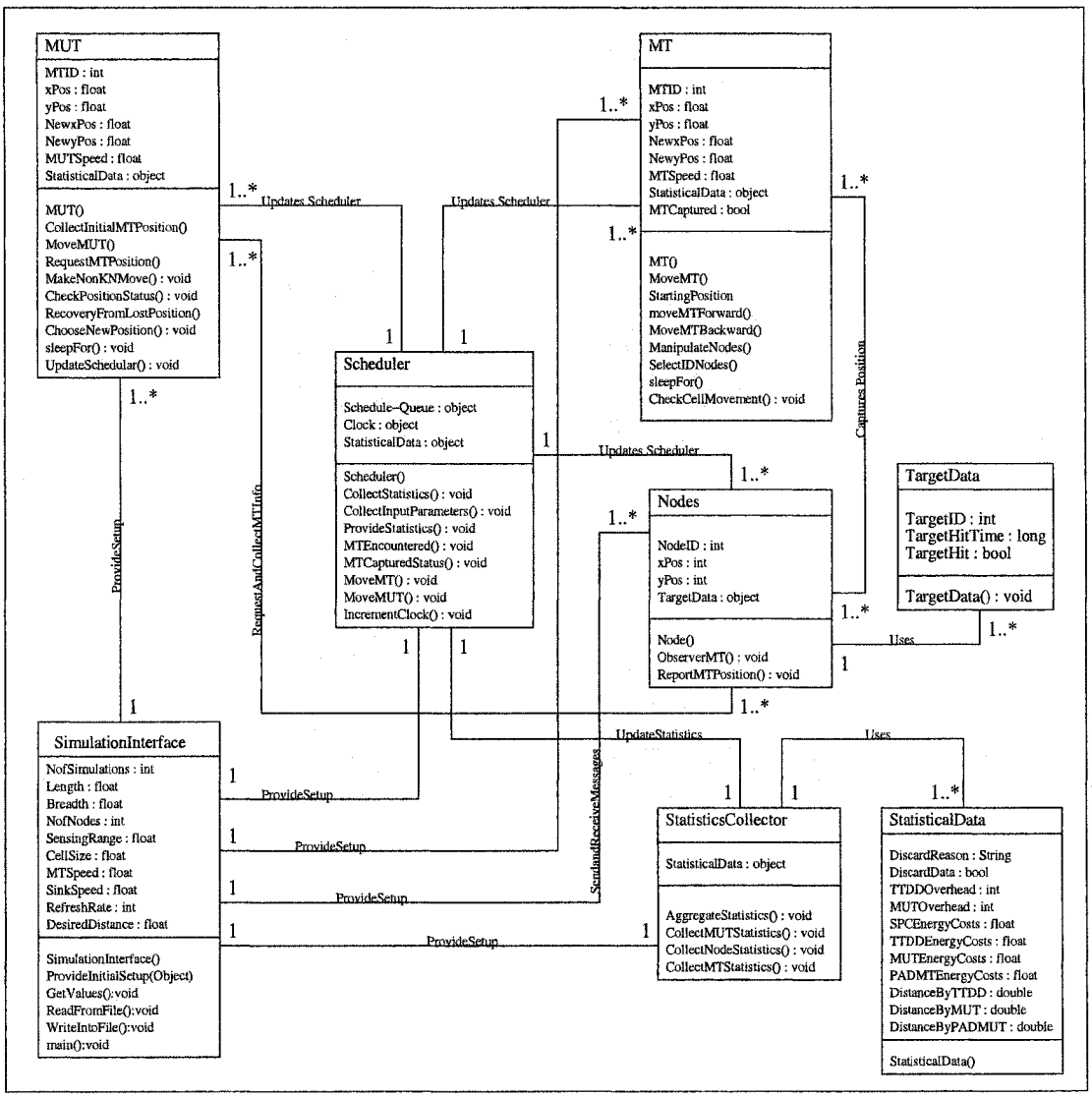

Figure 4.7: Class Digram - Final

4.7 Activity Diagrams

Finally, we identify and expand the key functions of the system in terms of the activity diagrams. Next, we draw the activity diagrams for the following:

- Scheduler (Figure 4.8), and
- Mobility of MT (Figure 4.9).

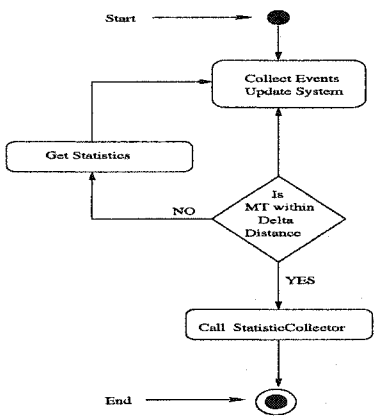

Figure 4.8: Activity diagram of the Scheduler

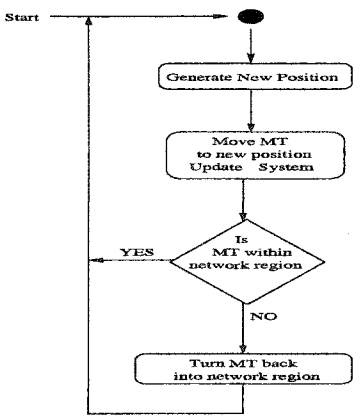

Figure 4.9: Activity diagram of the MT

4.8 Implementation

We used Java 2 (SDK, SE v1.4.2.03) to develop OSTSim. We effectively utilized the Java thread programming to build this model. It was a fruitful learning experience on concurrent programming.

4.9 Conclusion

In this Chapter we elaborated the steps that we followed to develop our simulation test bed OSTSim. It is used to evaluate the performance of the algorithms discussed in the next two Chapters 5 and 6.

Chapter 5

Ant-Based On-site Tracking

5.1 Introduction

This chapter presents an ant-based method that exploits the mobility dependency between the *MT* and the *MUT* to solve the On-site tracking problem. Ant-based approaches have been adapted to solve the routing problems in computer networks and mobile ad hoc networks[35, 36].

5.2 Basic Idea

To solve the On-site tracking problem, the *MUT* derives its tracking strategy based on the behavior of ants. While locating food, an ant leaves a trail of a substance called *pheromones* for other ants to follow and find the food source[21]. The direction of the path to the food source is guided by the intensity of pheromones. In our system a *MUT* inherits this ant behavior to track the moving target *MT*. Each node stores information about the target with a time stamp. *MUT* collects and uses these time stamps from the stationary sensor nodes, along the track of the *MT*, to compute the direction of its velocity vector.

5.3 Detailed Description

Our On-site tracking method consists of three logical steps: (1) *Reporting the initial position of the MT*, (2) *Initiation of tracking*, and (3) *Tracking*. We will explain these three steps in detail next.

5.3.1 Reporting the Initial Position

A sensor node, say s, which observes the MT first, reports this information to the MUT to initiate the tracking, and also inhibits other sensor nodes from redundant reporting. To achieve this, s sends a message about the MT to the entire network, and hence, to the MUT. The nodes that encounter the MT and have already received this message will only record the information about the MT, and not report to the network. It is possible that more than one node could observe the MT and report simultaneously; but eventually the nodes will stop the redundant reporting.

For our later references, we call the nodes which encounter the MT as the *knowledgeable nodes.* For example s becomes the first *knowledgeable node* in the network.

5.3.2 Initiation of Tracking

Once the MUT receives the information about the MT, it has to visit a *knowledgeable node* from where it can start tracking the MT. To achieve this MUT moves towards the first *knowledgeable node.* On the way if it encounters another *knowledgeable node* then it starts tracking the MT from that position.

5.3.3 Tracking

The main objective of the ant-based tracking is to avoid the huge communication costs incurred by the sensor nodes due to frequently updating the MUT with the location of the MT. This requires intelligent decision-making on the part of the MUT. An ant uses the pheromones to locate the food source whereas MUT uses the

information collected from the *knowledgeable nodes* to reach the MT. Data available at the *knowledgeable nodes* is the useful information about the MT with an associated timestamp. The value of this timestamp is the latest time at which that node has encountered the MT.

MUT computes its direction of velocity vector based on the timestamps collected from the *knowledgeable nodes*. The order between the timestamps of two *knowledgeable nodes* sets the direction for the MUT. If the collected timestamps are equal, then the MUT randomly chooses one of the *knowledgeable nodes* and move towards that node. There it collects data from the neighbors of the chosen *knowledgeable node*. If it finds data with a larger timestamp, then it moves in that direction. Otherwise, it retreats back to one of the unchosen *knowledgeable nodes* and repeats the process until it finds a *knowledgeable node* with a larger timestamp. By Assumption 3.2, the MUT will eventually find such a *knowledgeable node*, and then proceed in its direction.

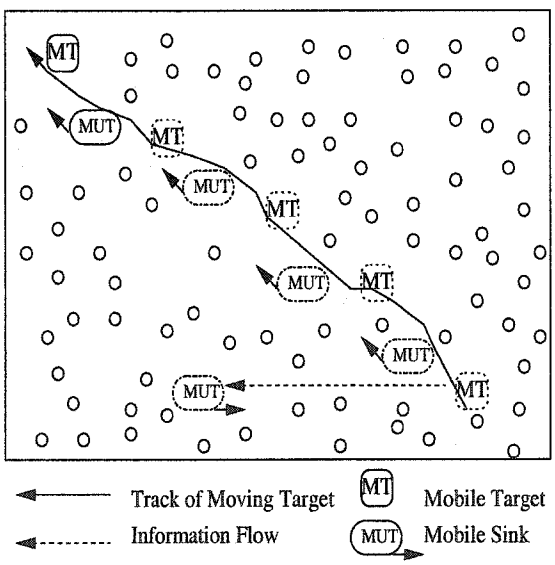

Figure 5.1: Ant-based On-site tracking

A typical scenario of ant-based On-site tracking is depicted in Figure 5.1. The dotted smooth rectangles around the MUT and the MT refer to their past positions, and the solid smooth rectangles indicate their current positions. The solid line across the diagonal region indicates the track of the MT. First, the initial position of the

MT is reported to the MUT. Then, MUT moves towards the MT's reported position and follows the track of the MT as shown in the Figure 5.1.

5.4 Theoretical Analysis

5.4.1 Terminology

We introduce the following parameters for our analysis.

- r - transmission range of the sensor nodes and the MUT.
- d_0 - diameter of the network region NR. That is, the distance between the farthest points in the network.
- a - area of NR.
- $n = |SN|$ - the number of stationary nodes in NR.
- d - distance traveled by the MT, from its initial position, before it is tracked.
- v_{mt} - velocity of the MT.
- v_{mut} - velocity of the MUT.
- t - tracking time.
- m_t - total number of messages generated by the sensor nodes during the tracking time, t.

5.4.2 An Upper Bound on the Tracking Time

Based on the parameters outlined in subsection (5.4.1), we derive an upper bound on the tracking time, t.

Definition 5.1 *The nodes that are within the transmission range of a node, say n_0, are called the* **neighbors of** n_0.

Theorem 5.1 *Ant-based On-site tracking approach assures that the tracking time,* $t \leq \frac{ad_0 - 2\delta rpn - ap}{av_{mut} - 2v_{mt}rpn}$, *where* $p = \frac{(r+\sqrt{2}r)}{2}$

Proof: To cover the maximum distance, assume that the MT travels in a straight line and does not repeat its path as shown in Figure 5.2. It is easy to infer from Figure 5.2 that the maximum number of *knowledgeable nodes* beyond δ distance from the MT is:

$$\frac{2r(d-\delta)}{a}n \tag{5.1}$$

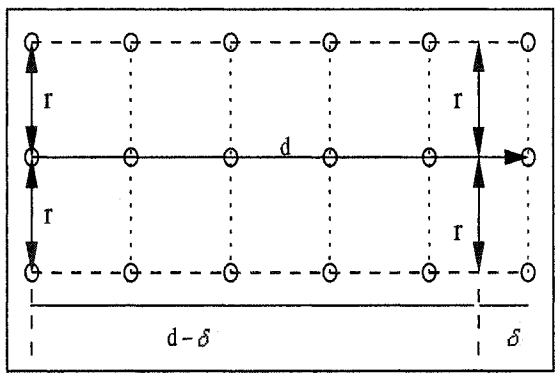

Figure 5.2: Knowledgeable area swept by MT's path

By Assumption 3.2, during tracking, the MUT will be either within the desired distance δ to the MT or the MUT will have a message from at least one of the neighbor *knowledgeable nodes* with a higher timestamp. In the later case, if the MUT has only one neighbor with a higher timestamp, then the MUT can move to that particular neighbor and proceed thereon. Note that only the *knowledgeable nodes* with higher timestamp will reply. If more than one *knowledgeable node* have equal timestamps, then the MUT may have to visit all these *knowledgeable nodes.*

Consider Figure 5.3, in which the MUT's current position is represented by the black circle in the center. The MUT might have visited this position from any of the four neighbor nodes, L, U, D, and R. Without loss of generality, we assume that the MUT has moved to its current position from L. Now to move to the next position the

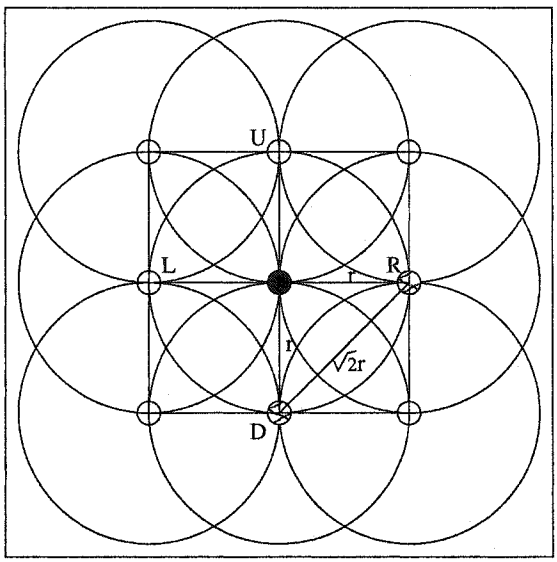

Figure 5.3: Movement of MUT between two knowledgeable nodes

MUT may get a maximum of two replies with equal timestamps from the remaining three *knowledgeable nodes*. That is, either from the pair U and R, or from the pair D and R. In either case, the maximum distance that the *MUT* should travel to visit the pair is:

$$r + \sqrt{2}r \tag{5.2}$$

This pattern of traveling $r + \sqrt{2}r$ distance to visit a pair of *knowledgeable nodes*, as explained through Figure 5.3, could repeat until the tracking is complete. Since we have $\frac{2r(d-\delta)}{a}n - 1$ *knowledgeable nodes*, the maximum distance that the *MUT* could travel to visit all the *knowledgeable nodes* is:

$$\left(\frac{\frac{2r(d-\delta)}{a}n - 1}{2}\right)(r + \sqrt{2}r) \tag{5.3}$$

For brevity we use $p = \frac{r+\sqrt{2}r}{2}$. By adding d_0, the maximum distance traveled by the *MUT* to reach the first *knowledgeable node*, into the equation (5.3), we get the total distance traveled by the *MUT* as follows.

$$d_0 + \left(\frac{2(d-\delta)r}{a}n - 1\right)p \tag{5.4}$$

Since the distance traveled by the *MUT* in time t is, tv_{mut}, we get the following inequality.

$$t \leq \frac{d_0 + \left(\frac{2(d-\delta)rn}{a} - 1\right)p}{v_{mut}} \tag{5.5}$$

By substituting $d = tv_{mt}$ in the equation (5.5), we obtain:

$$t \leq \frac{ad_0 + 2(tv_{mt} - \delta)rpn - ap}{av_{mut}} \tag{5.6}$$

Finally, by solving the equation (5.6) for t, we get:

$$t \leq \frac{ad_0 - 2\delta rpn - ap}{av_{mut} - 2v_{mt}rpn} \tag{5.7}$$

Hence the proof.

Corollary 5.1 *Ant-based On-site tracking approach assures that the target can be tracked in a finite time, if* $v_{mut} > \left(\frac{r^2(1+\sqrt{2})n}{a}\right) v_{mt}$

5.4.3 An Upper Bound on the Number of Messages Generated by the Sensor Nodes

In this section we derive an upper bound on m_t.

Theorem 5.2 *Ant-based On-site tracking approach assures that the number of messages generated by the sensor nodes during* t, $m_t \leq \frac{2rnv_{mt}(ad_0 - 2rnp\delta - ap - anp) + a^2nv_{mut}}{a(av_{mut} - 2rnpv_{mt})}$, *where* $p = \frac{(r+\sqrt{2}r)}{2}$

Ant-based tracking involves two logical stages of message generation by the sensor nodes: (i) initial flooding and (ii) direct communication with the *MUT* during tracking. Let m_x and m_y, respectively, be the messages generated in these two stages. We

compute the upper bounds for m_x and m_y separately and then add them to get the upper bound for m_t.

Assume that initially there are more than one sensor node that capture the position of the MT (first *knowledgeable nodes*). Even if all of the first *knowledgeable nodes* send messages simultaneously, rest of the sensor nodes would choose only one of the first *knowledgeable nodes* to forward their message. Therefore, each node in the network will forward at most one message during the flooding. Hence, we get the following first equation.

$$m_x = n \tag{5.8}$$

During tracking, the MUT can get response from at most all the *knowledgeable nodes*. The number of knowledgeable nodes is $\frac{2dr}{a}n$, where $d = tv_{mt}$. Substituting for d, we get:

$$m_y \leq \frac{2trv_{mt}}{a}n \tag{5.9}$$

By adding equations (5.8) and (5.9), we obtain the following inequality.

$$m_t \leq \frac{an + 2trnv_{mt}}{a} \tag{5.10}$$

By substituting the value for t from Theorem 5.1 in the equation (5.10), we get:

$$m_t \leq \frac{2rnv_{mt}(ad_0 - 2rnp\delta - ap - anp) + a^2nv_{mut}}{a(av_{mut} - 2rnpv_{mt})}. \tag{5.11}$$

This completes the proof.

From Theorem 5.2, an upper bound on the total energy spent by the sensor nodes during tracking can be easily calculated. These upper bounds reflect the costs for worst case scenarios. Average case analysis would help to understand the normal behavior of the system. Computing the average case values for t and m_t is quite complex. The complexity arises due to the variabilities of the parameters such as the

initial positions of the MUT and the MT (they can be at any position in the entire network), mobility pattern of the MT, the speed of the MT (may vary from 0 to v_{mt}), etc. However, to see the closeness to the simulation results we derive the upper bounds for a simplified average case.

Theorem 5.3 *Assume that*

(i) The total number of messages generated initially n,

(ii) MUT travels $\frac{d_0}{2}$ *to reach the first knowledgeable node,*

(iii) MT travels with an average velocity of $\frac{v_{mt}}{2}$*, and*

(iv) MUT visits only half of the total number of knowledgeable nodes.

Then,

(a) $t \leq \frac{ad_0 - 2\delta rnp - 2ap}{2av_{mut} - v_{mt}rnp}$

(b) $m_t \leq \frac{rnv_{mt}[ad_0 - 2ap(1+2an) - 2\delta rnp] + 8a^3nv_{mut}}{2a(2av_{mut} - rnpv_{mt})}$

The proof is similar to the proofs of Theorems 5.1 and 5.2. In the next section we present our simulation study.

5.5 Simulation Study

In the simulation, we are mainly interested in studying the performance of our ant-based On-site tracking method. In addition to that, we are also interested in comparing it with other existing methods that can be used to solve the On-site tracking problem. Though the On-site tracking problem has been introduced in this thesis, the methods like TTDD[29] and shortest path communication (SPC) can be used to solve the problem. TTDD uses a logical grid structure for the communication. SPC is a theoretical method that uses a shortest path between two nodes for the communication with zero path-maintenance cost. In this section, we compare our ant-based approach with TTDD and SPC.

5.5.1 Experimental Setup

The parameters for the simulation are set as follows.

1. Speed of the moving target is fixed as $1m/s$.
2. Speed of the moving sinks varies from $5m/s$ to $15m/s$.
3. a, square area, varies from $100m \times 100m$ to $2000m \times 2000m$.
4. Transmission range r is computed as $(1/10)^{th}$ of the side of the square region.
5. Desired distance, δ, varies from $5m$ to $30m$, and frequency of closeness, f, is fixed as 1.

5.5.2 Results Analysis

Simulation results are mainly collected for three performance metrics: (i) average number of messages generated by the nodes, (ii) average energy consumption per node, and (iii) average time taken for tracking the MT. We investigate these three metrics by, (1) varying the size of the area, (2) varying the speed of the Sink or MUT, and (3) varying the desired distance, δ. Results obtained for each value of

these varying parameters are an average of 100 simulation runs. For the comparison purposes in the graphs, we refer our ant-based approach as *MUT*.

Experiment 1 (*Area vs. Energy used per sensor node*): In this experiment, we are interested in computing the average amount of energy spent by the sensor nodes for message communication. Sensor nodes spend energy for both sending and receiving the messages as mentioned previously. Therefore, we first calculate the number of messages sent and received by the sensor nodes in the entire network, and then compute the average energy spent per sensor node. The results are summarized in graphs as shown in Figure 5.4.

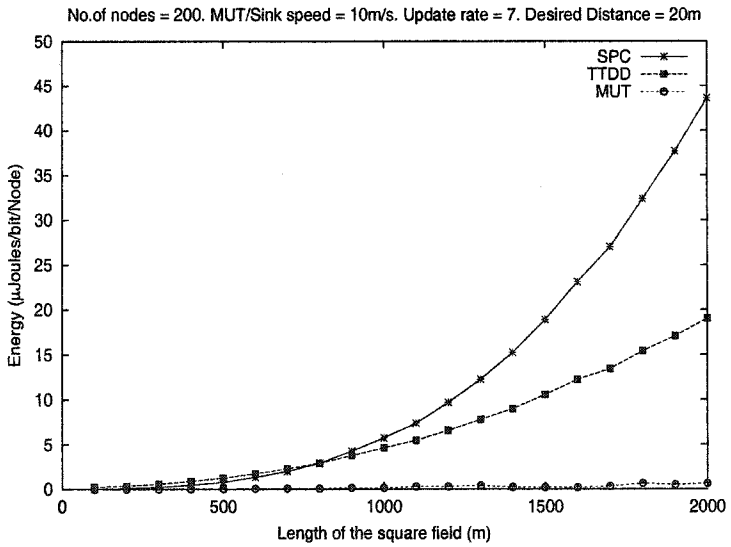

Figure 5.4: Area vs. Energy/Node

Observation 1: The average energy consumption per node is quite small in the case of *MUT*, and it increases slightly as the area increases. On the other hand, the energy consumption in *SPC* and *TTDD* increases rapidly as we increase the area size. We note that for an area size of $2000 \times 2000\ m^2$, the average energy consumption per node for SPC, TTDD and *MUT* are 43.618 μJoules/bit, 19.045 μJoules/bit and 0.63 μJoules/bit respectively. This shows that *MUT* is highly energy efficient.

Experiment 2 (*Area vs. Tracking Time*): In this experiment, we are interested in comparing the average time taken for tracking the MT in our method with that of TTDD. The results are summarized in graphs as shown in Figure 5.5.

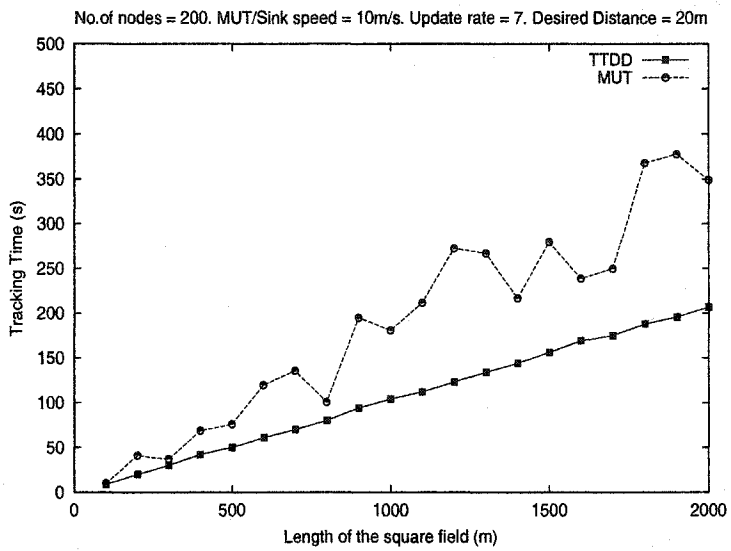

Figure 5.5: Area vs. Time

Observation 2: The average tracking time in the case of TTDD increases steeply with the increase in area, and so is the case with MUT but with a random behavior. This is because MUT normally travels a larger distance than TTDD while trying to achieve the desired distance.

Note: We conducted the study by fixing the value of r and increasing the number of nodes when area increases. We observed a similar behavior.

In the next two experiments, we consider the varying speed of the sink. In these experiments we consider the network area to be a square of size 2000×2000 m^2 with 200 nodes in it. The maximum speed of the moving target is fixed at 3 m/s, though, we allowed the MT to have a random movement.

Experiment 3 (*Sink speed vs. Energy used per sensor node*): In this experiment, we are interested in finding how sink's speed affects the energy consumption by the nodes. The results are summarized in graphs as shown in Figure 5.6.

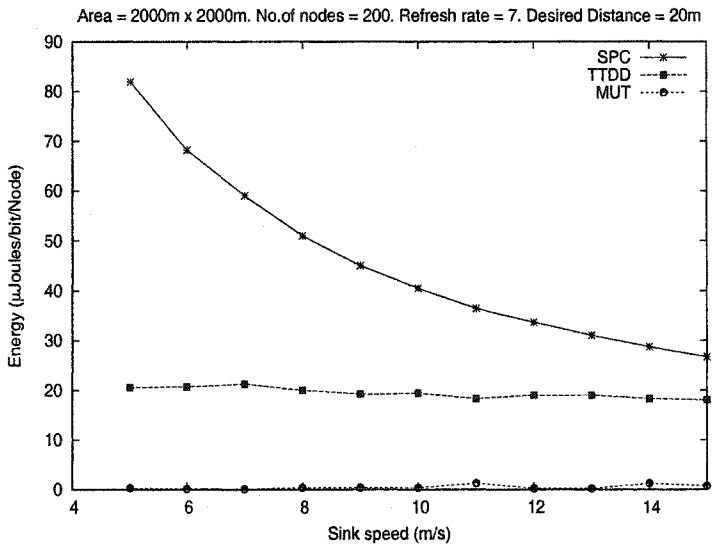

Figure 5.6: Speed vs. Energy/Node

Observation 3: As we vary the speed of the sinks from 5 m/s to 15 m/s, we observe that message generation reduced in the case of SPC. As the sink's speed increases it is expected to track the target faster. There is a slight decrease in the case of TTDD, but MUT's average number of messages remains almost constant. In terms of energy consumption, there is a decrease in energy consumption per node in SPC and TTDD, but it remains almost constant in the case of *MUT*. In 100 simulation runs with the sink speed of 10 m/s, the average energy consumption per node in SPC, TTDD and *MUT*, respectively, are 40.489 μJoules/bit, 19.406 μJoules/bit, and 0.345 μJoules/bit.

Experiment 4 (*Sink speed vs. Tracking Time*): In this experiment, we vary the sink's speed to observe its affect on the tracking time. The results are summarized in graphs as shown in Figure 5.7.

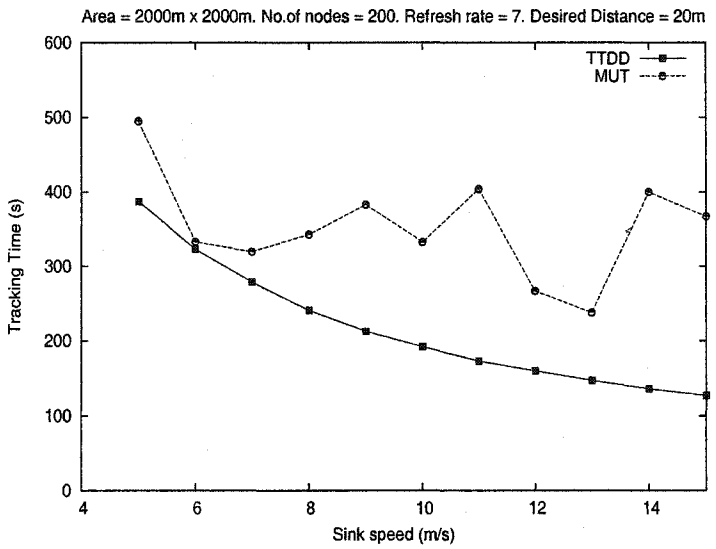

Figure 5.7: Speed vs. Time

Observation 4 The average tracking time in the case of TTDD decreases sharply with the increase in speed as it is obvious that the sink with more speed will be able to track the target faster. In the case of *MUT*, though, the average tracking time is quite random and is comparatively higher than that of TTDD as expected, but it also decreases as *MUT*'s speed increases.

Next, we evaluate the performance by taking third parameter which is desired distance, δ. In this experiment we vary δ while keeping the speed of sinks and the network area fixed. In the network area of 2000 $\times$ 2000 m^2 with 200 nodes in it, we vary δ from 10m to 50m while keeping sink speed fixed at 10 m/s.

Experiment 5 (*Desired Distance vs. Energy used per sensor node*): In this experiment, we evaluate *MUT*'s performance in terms of energy consumption. As we vary δ, we observe its impact on the average energy consumption by nodes. The results are summarized in graphs as shown in Figure 5.8.

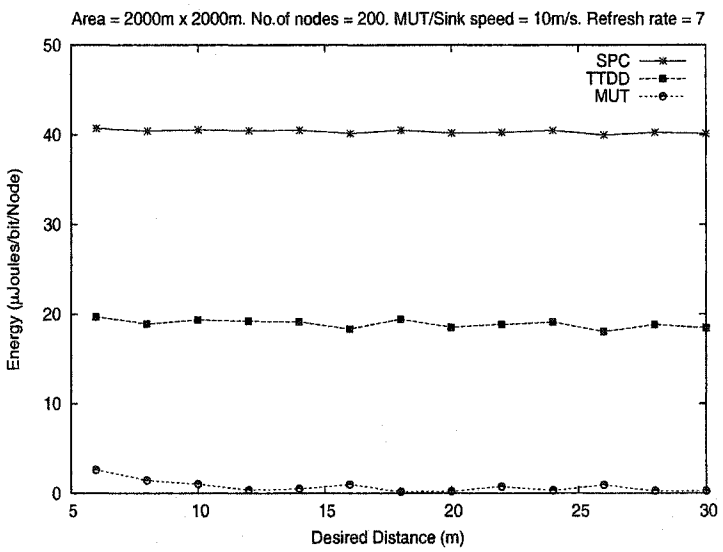

Figure 5.8: Desired Distance vs. Energy/Node

Observation 5: There are small variations in energy consumption in the case of *MUT*. However, it is quite efficient than SPC and TTDD that can be deduced from Figure 5.8. Like other experiments we ran 100 simulations for the varying parameter, which is δ in this experiment. We observe that the average energy consumption per node in SPC, TTDD and MUT is 40.196 μJoules/bit, 18.549 μJoules/bit and 0.219 μJoules/bit respectively for a δ value of 20m.

Experiment 6 (*Desired Distance vs. Tracking Time*): In this experiment, we are interested in observing the average tracking time of MUT against the varying value of δ. The results are summarized in graphs as shown in Figure 5.9.

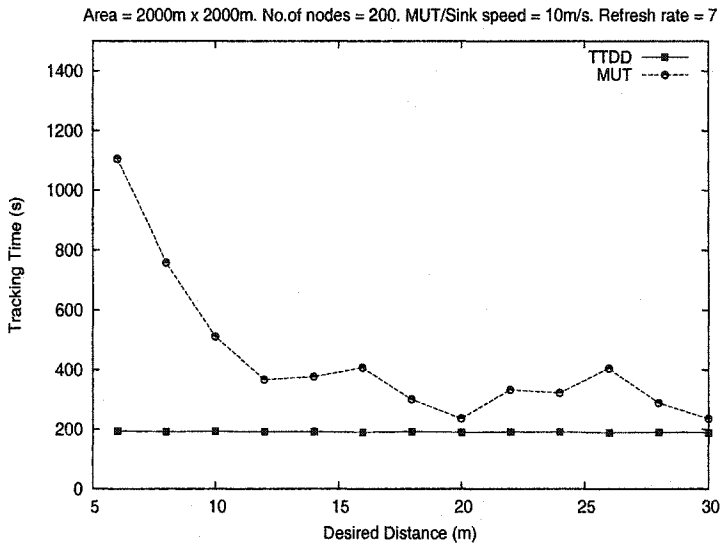

Figure 5.9: Desired Distance vs. Time

Observation 6: Like previous experiments for the average tracking time of MUT, in this experiment also, MUT's average tracking time is random and higher than that of TTDD. However, it gradually decreases as the value of δ increases. This is expected, because MUT has to capture the target from a large distance instead of following it closely.

5.5.3 Conclusion

From the simulation results, it is easy to see that energy consumption in our ant-based approach is quite less than that of TTDD based approach. On the other hand tracking time in our approach is higher than that of TTDD. In the next section we generalize the ant-based approach to solve the On-site tracking problem for the multiple targets case.

5.6 Generalization of the Ant-based Approach

On-site tracking of multiple targets by multiple sinks requires coordination among all the sinks, mainly to determine which sink tracks which target. For such coordination, we introduce a master sink called MUT_M. As described previously in the single target case, our On-site tracking method consists of three logical steps. Here, we modify those three steps to generalize the ant-based approach for the multiple targets case. These three modified steps are: (1) *Reporting the initial position of each target MT_i, to the master MUT_M*, (2) *Initiation of trackings by the MUT_M*, and (3) *Tracking the individual targets by the MUTs.* Next we explain these three steps in detail.

5.6.1 Reporting the Initial Positions of the MTs

The initial reporting is based on the demand of the master MUT_M. A sensor node, say s, which observes the MT_i first (at a time equal or greater than the time specified by the MUT_M) reports this information to the master MUT_M to initiate tracking and also inhibits other sensor nodes from redundant reporting. To achieve this, s sends a message about the MT_i to the entire network and hence to the MUT_M. The nodes which encounter the MT_i and have already received the reporting message will only record the information about the MT_i and do not report again to the network. It is possible that more than one node could observe the MT_i and report simultaneously; but eventually the nodes will stop the redundant reporting.

In this case for all i (i.e. 1 to m), each MT_i will have a set of *knowledgeable nodes.* It is quite possible that one particular node is a *knowledgeable node* for more than one MT_i.

5.6.2 Initiation of Trackings by the MUT_M

Once the MUT_M receives the information about all the MT_i from the sensor nodes, it allocates each MUT_j with the mission to track a particular MT_i. The MUT_M makes this decision based on the reported positions of the first *knowledgeable nodes*

corresponding to all the MTs and the starting positions of all the MUTs.

5.6.3 Tracking the Individual Targets by the MUTs

Once the MUT_j has been assigned to track the MT_i , it has to reach a *knowledgeable node* corresponding to the MT_i, and from there it can start tracking the MT_i. To achieve this, the MUT_j move towards the first *knowledgeable node* corresponding to the MT_i. On the way if it encounters another *knowledgeable node* of the MT_i, then it starts tracking the MT_i from that position.

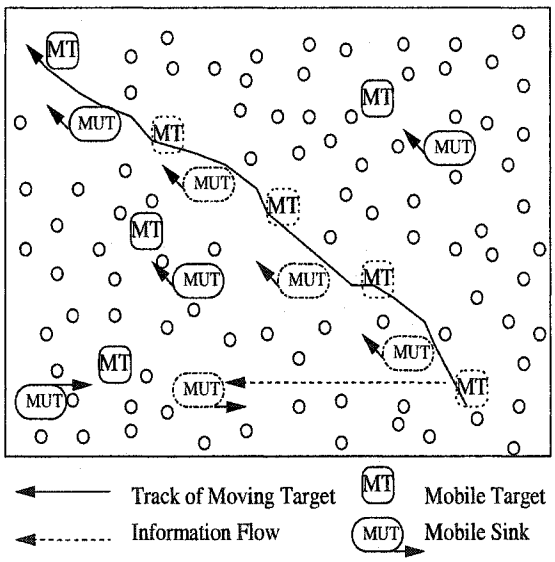

Figure 5.10: On-site tracking of multiple targets

Tracking a target from its first *knowledgeable node* is similar to the single target case as discussed in the Section 5.3.2. That is, the MUT_j computes its direction of velocity vector based on the timestamps collected from the *knowledgeable nodes.* The order between the timestamps of two *knowledgeable nodes* sets the direction for the MUT_j. If the collected timestamps are equal then the MUT_j randomly chooses one of the *knowledgeable nodes* and move towards that node. There it collects data from the neighbors of the chosen *knowledgeable node.* If it finds data with a larger timestamp, then it moves in that direction. Otherwise it retreats back to one of the unchosen *knowledgeable nodes* and repeats the process until it finds a *knowledgeable*

node with a larger timestamp. By assumption 3.2, MUT_j will eventually find such a *knowledgeable node* and then proceed in that direction. By assumption 3.6, the MUT_j will reach the MT_i in a finite time and thereon it can be within the desired distance, if necessary.

A typical scenario of ant-based On-site tracking of multiple targets is depicted in Figure 5.10. The dotted smooth rectangles around the MUT and the MT refer to their past positions, and the solid smooth rectangles indicate their current positions. The solid line across the diagonal region indicates the track of the MT.

5.7 Simulation Study

In the simulation, we are mainly interested in studying the performance of ant-based On-site tracking method for the multiple targets case. In addition to that, we also compared it with the TTDD[29], and SPC as discussed previously. The experimental setup is same as given in the Section 5.5.1. In addition to that setup, in the following experiments we consider the simplistic case of five MUTs and five MTs.

5.7.1 Simulation Experiments and Results Analysis

Again simulation results are mainly collected for three performance metrics: (i) average number of messages generated by the nodes, (ii) average energy consumption per node, and (iii) average time taken for tracking the MT. We investigate these three metrics by, (1) varying the size of the area, (2) varying the speed of the Sink or MUT and (3) varying the desired distance, δ. Results obtained for each value of these varying parameters are an average of 100 simulation runs. Next, we discuss our experiments.

Experiment 7 (*Area vs. Number of messages generated by nodes*): In this experiment, we are interested in calculating the average number of messages generated by the sensor nodes for various sizes of the network area. The results are summarized in graphs as shown in Figure 5.11.

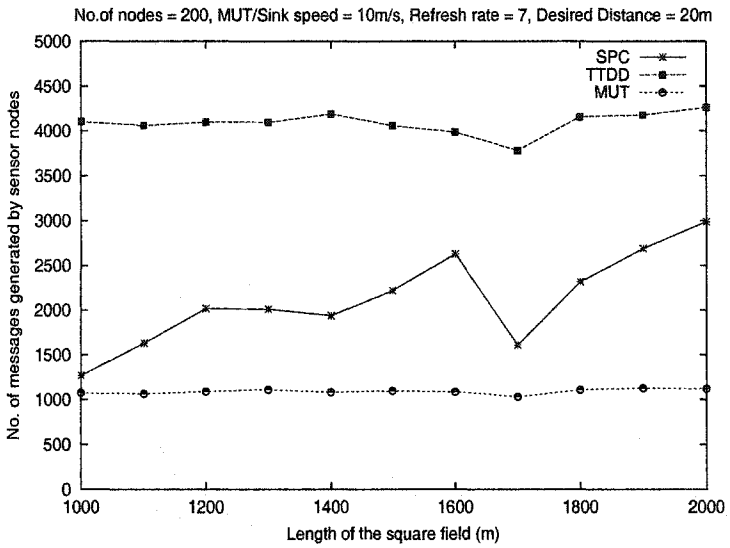

Figure 5.11: Area vs. Number of Messages

Observation 7: The message generation in SPC increases rapidly as the area increases. There is a slight increase in TTDD, but *MUT* has the lowest communication overhead and remains almost constant as the size of the area increases.

Experiment 8 (*Area vs. Energy used per sensor node*): In this experiment, we compute the amount of energy spent by the nodes for message communication. The results are summarized in the graphs as shown in the Figure 5.12.

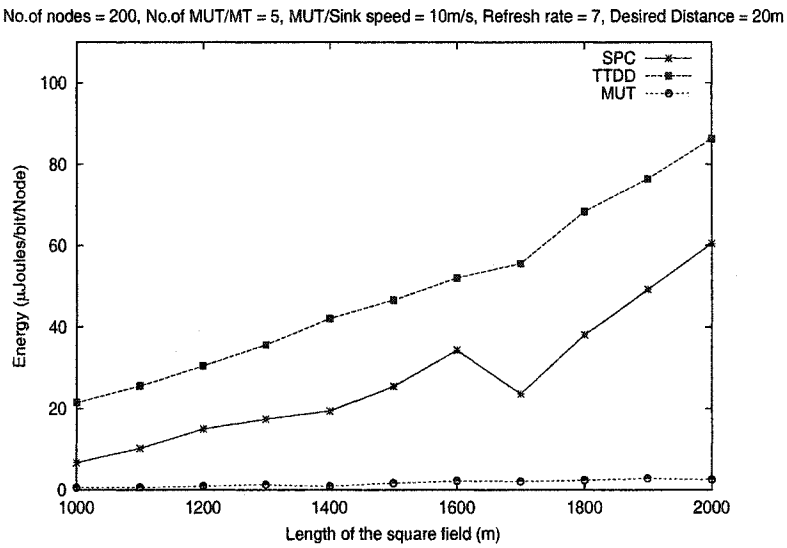

Figure 5.12: Area vs. Energy/Node

Observation 8: The energy consumptions for both SPC and $TTDD$ increases sharply as the area increases. In contrast to that, the energy consumption is quite less in the case of MUT, and it increases slightly as the area increases. We note that for an area size of 2000 $\times$ 2000 m^2, average energy consumption per node for SPC, TTDD and MUT are 60.489 μJoules/bit, 86.311 μJoules/bit and 2.548 μJoules/bit respectively, which shows that MUT is highly energy efficient.

Experiment 9 (*Area vs. Tracking Time*): In this experiment we compare the average tracking time in our method with the average tracking time in TTDD. The results are summarized in graphs as shown in Figure 5.13.

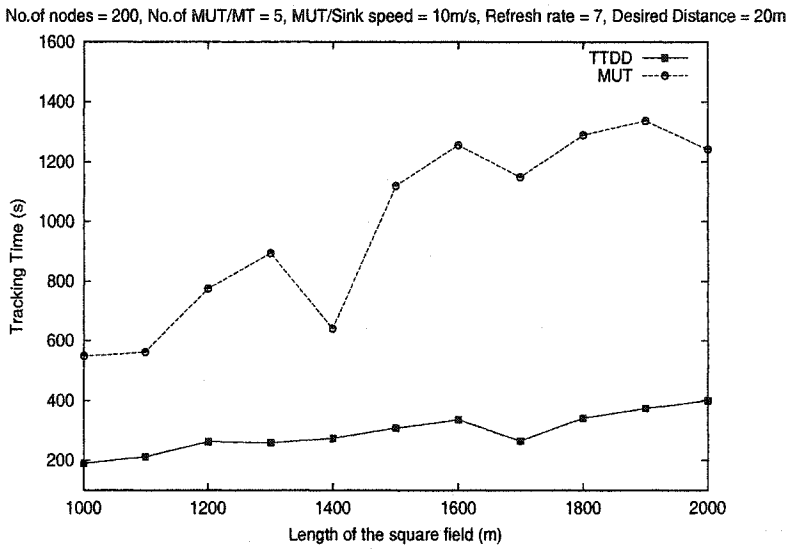

Figure 5.13: Area vs. Tracking Time

Observation 9: The average tracking time in the case of TTDD increases with the increase in area and so is the case with *MUT* but with a random behavior. This is because *MUT* normally travels a larger distance than TTDD while trying to achieve the desired distance.

In the next three experiments we consider the varying speed of the sinks.

Experiment 10 (*Sink speed vs. Number of messages generated by nodes*): In this experiment we compute the number of messages transmitted by nodes for a varying value of sink's speed as specified previously. The results are summarized in graphs as shown in Figure 5.14.

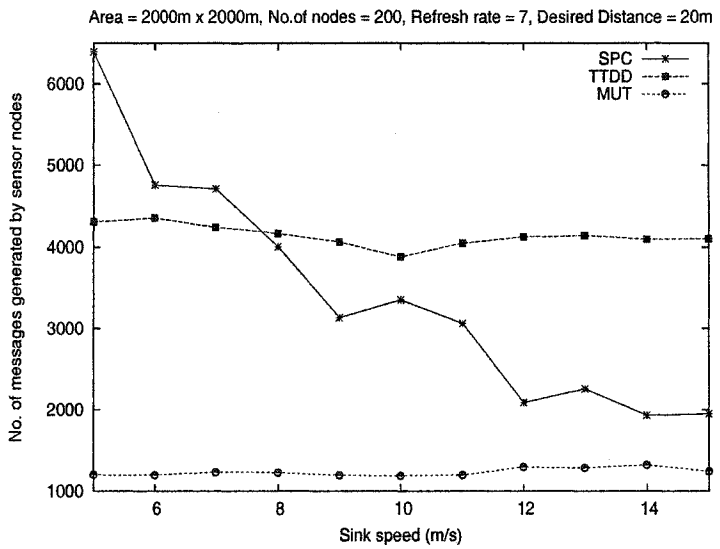

Figure 5.14: Sink speed vs. Number of Messages

Observation 10: As shown in Figure 5.14, the sink's speed does not greatly affect the message generation in the case of *MUT*. There is slight decrease in the case of TTDD and a sharp decrease in the case of SPC. Overall *MUT* has the least communication overhead.

Experiment 11 (*Sink speed vs. Energy used per sensor node*): In this experiment, we are interested in finding how sink's speed affects the energy consumption in nodes. The results are summarized in graphs as shown in Figure 5.15.

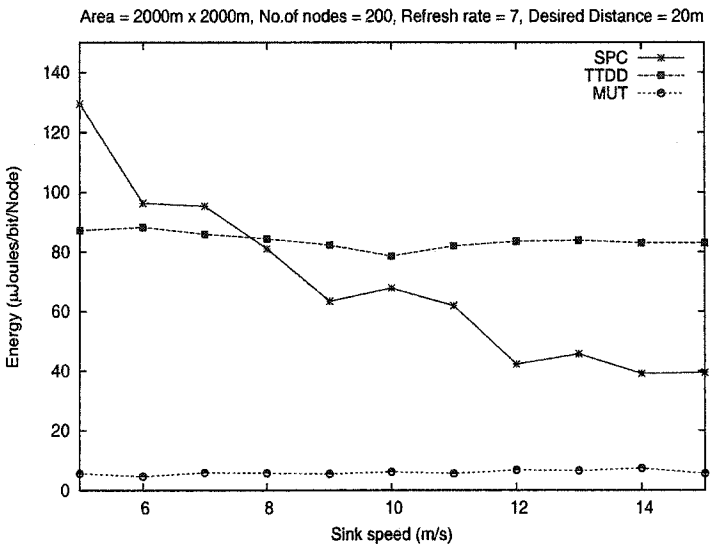

Figure 5.15: Sink speed vs. Energy/Node

Observation 11: As we vary the speed of the sinks from 5 m/s to 15 m/s, we observe that average energy consumption decreases quite rapidly in the case of SPC. On the other hand in the case of TTDD and *MUT*, it remains almost constant. For 100 simulation runs, we observe that the average energy consumption in SPC, TTDD and *MUT* is 39.453 μJoules/bit, 83.03 μJoules/bit, and 5.575 μJoules/bit respectively for the sink speed of 10 m/s.

Experiment 12 (*Sink speed vs. Tracking Time*): In this experiment we vary the sink's speed to observe its affect on the tracking time. The results are summarized in graphs as shown in Figure 5.16.

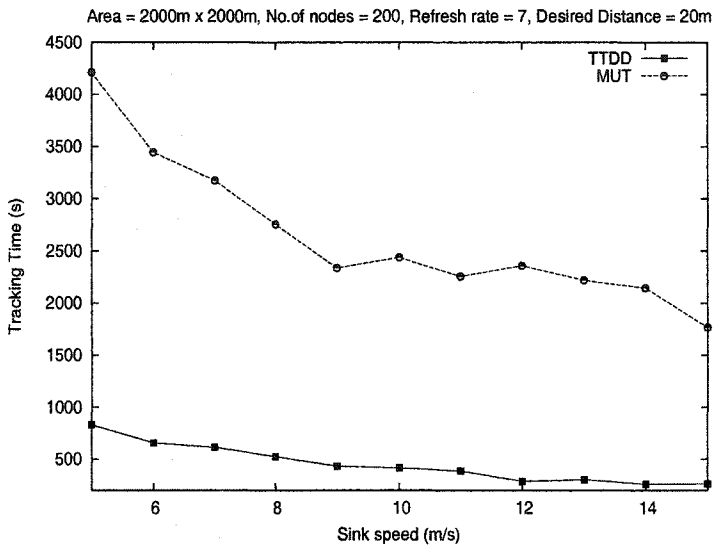

Figure 5.16: Sink speed vs. Time

Observation 12: The average tracking time in the case of TTDD decreases as the sink's speed increases. In the case of *MUT* the tracking time decreases as well, but it is comparatively higher than that of TTDD as expected. It is interesting to note that an increased sink's speed improves the tracking time much faster in both the cases of TTDD and *MUT*. On the other hand increased sink's speed has lesser impact on the energy consumptions in both the cases of TTDD and *MUT*.

In next three experiments we evaluate the performance by varying the value of δ.

Experiment 13 (*Desired Distance vs. Number of messages generated by nodes*): In this experiment we vary δ and observe its impact on the message generation by the nodes. The results are summarized in graphs as shown in Figure 5.17.

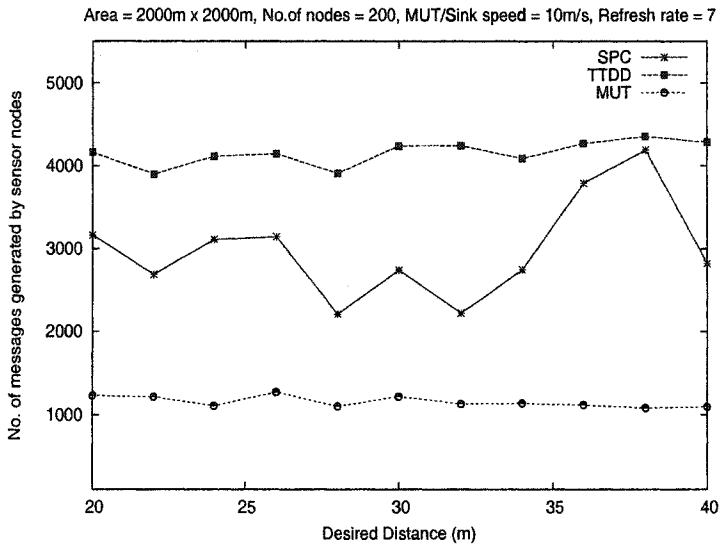

Figure 5.17: Desired Distance vs. Number of Messages

Observation 13: There are variations in the number of messages generated in all the three cases of SPC, TTDD and *MUT*. But overall message generation by the nodes in the case of *MUT* is much less as compared to SPC and TTDD.

Experiment 14 (*Desired Distance vs. Energy used per sensor node*): In this experiment we vary δ to observe its impact on the average energy consumption in nodes. The results are summarized in graphs as shown in Figure 5.18.

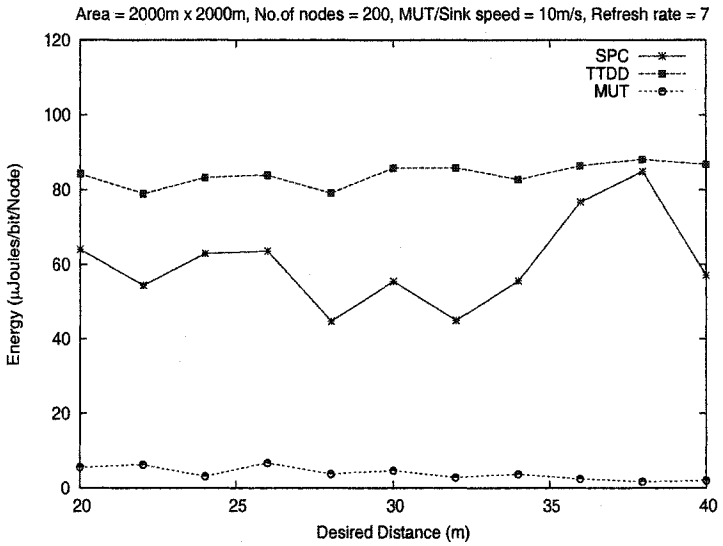

Figure 5.18: Desired Distance vs. Energy/Node

Observation 14: There are small variations in energy consumption in the case of *MUT*, but it is quite efficient than SPC and TTDD as shown in Figure 5.18. We observe that the average energy consumption per node in SPC, TTDD and *MUT* is 57.089 μJoules/bit, 86.845 μJoules/bit and 2.036 μJoules/bit respectively for a δ value of 20m.

Experiment 15 (*Desired Distance vs. Tracking Time*): In this experiment observe the tracking time of MUT against the varying value of δ. Results are summarized in graphs as shown in Figure 5.19.

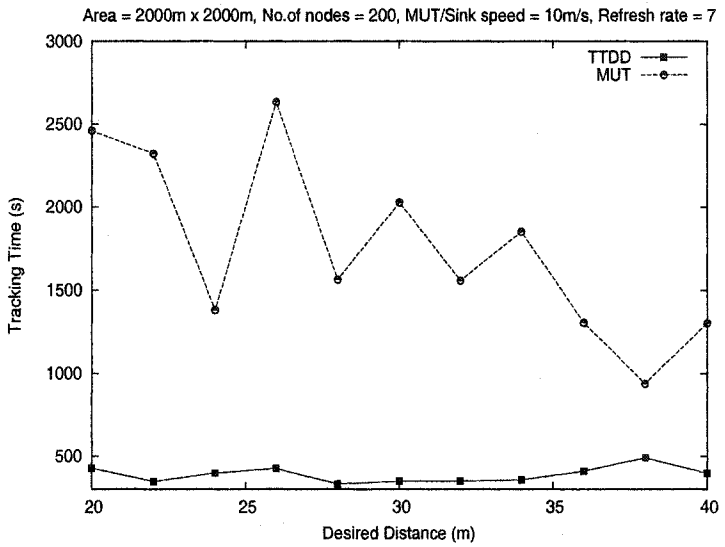

Figure 5.19: Desired Distance vs. Time

Observation 15: Like previous experiments for the tracking time of the MUT, in this experiment also MUT's average tracking time is random and higher than that of TTDD, but it gradually decreases as the value of δ increases. This is expected as MUTs have to capture their target from a large distance instead of following it closely.

It is interesting to note that in the experiments for the tracking time, MUT's tracking time is quite random as compared to the TTDD based approach. The reason is that MUT follows the track of the MT. Since the MT is allowed to have a random movement, and therefore MT's traced path highly influence the tracking time of the MUT.

5.8 Conclusion

In this chapter, we proposed an ant-based approach to solve the On-site tracking problem for a single target case, and then we generalized the ant-based approach to solve the problem for the multiple targets case. In our simulation results, it is shown that energy expenditures of the sensor nodes in our approach is quite less than that of the TTDD and SPC based approaches. However, in our approach the *MUT* takes a longer time to track the target as compared to the TTDD based approach. The next step in our research was focused on reducing the tracking time of the *MUT*, which we discuss in the next chapter.

Chapter 6

Adaptive On-site Tracking

6.1 Introduction

The basic solution proposed in Chapter 5 to solve the On-site tracking problem is shown to be energy efficient. In that approach, sensor nodes simply store the information about the target with a timestamp. The *MUT* visits these sensor nodes and collects these timestamps to compute a direction towards the moving target. Since the velocity of the *MUT* is assumed to be greater than that of the target; the *MUT* would eventually capture the target. This approach reduces the communication overhead tremendously, but has the limitation of increased tracking time.

The reason for the increased tracking time is that *MUT* follows the track of the target by visiting a large number of *knowledgeable nodes* along the track of the target. Consider the case in which the initial position of the target has been reported to the sink. By the time the sink reaches the reported position, the target may have moved to another position in the network region. Even though the sink would move faster than the target, the sink would have to visit a maximum number of sensor nodes to collect the timestamps, and compute the direction of its velocity vector. That would increase the tracking time significantly.

We believe that the tracking time can be reduced, if the information possessed by the sensor nodes is the latest. Consider the case in which the initial position of

the target has been reported to the *MUT*. Now, by the time the *MUT* moves to the reported position, the target may have moved to a new position; but the sensor node at the initial position would report the latest position of the target, and hence the *MUT* would be able to track the target faster by moving directly to the reported position. Updating the *MUT* with the latest information can be achieved by routing the latest information of the target along the track.

6.1.1 Basic Idea

The main objective of the adaptive On-site tracking is to reduce the tracking time in the basic approach proposed in Chapter 5, while conserving the energy of the sensor nodes. This can be achieved by a strategy of supplying the latest information about the *MT* when the *MUT* visits a *knowledgeable node* in the network. Using the latest information, the MUT can move directly towards the latest position as reported by the sensor nodes bypassing the intermediate *knowledgeable nodes*. This way the *MUT*'s tracking time is reduced as it does not visit all the *knowledgeable nodes* along the track of the *MT*. The energy efficiency of the sensor nodes depends on the information maintenance strategy.

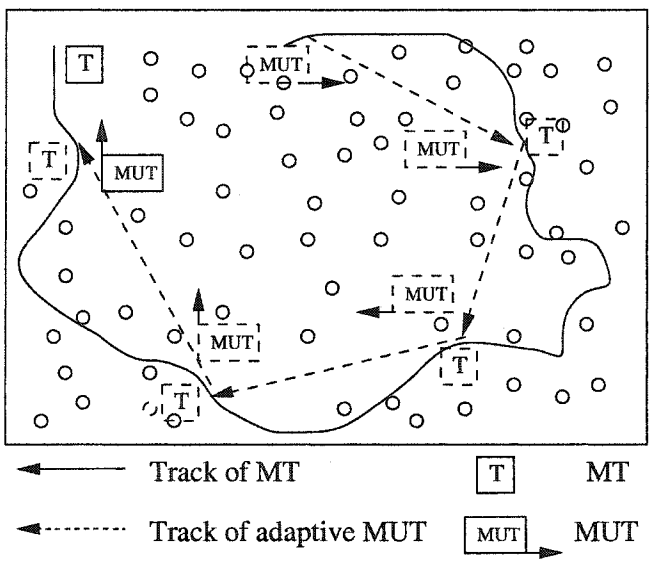

Figure 6.1: Adaptive On-site tracking by MUT

A typical scenario of adaptive On-site tracking is depicted in Figure 6.1. The dotted smooth rectangles around the *MUT* and the *MT* refer to their past positions, and the solid smooth rectangles indicate their current positions. The solid line across the diagonal region indicates the trace of the *MT*. First, the initial position of the *MT* is reported to the *MUT*. Then, the *MUT* moves towards the *MT*'s initial position and adaptively follows the track of the *MT* as shown in the Figure 6.1.

6.2 Detailed Description

The algorithms presented here for the adaptive On-site tracking consists of three logical steps:

1. *Reporting the initial position of the MT*,
2. *Maintaining the latest information about the MT in the knowledgeable nodes along the MT's track*, and
3. *Deciding the MUT's tracking strategy using the latest information collected during the previous step.*

Initial reporting remains same as described for the ant based algorithm given in Chapter 5. The main contribution in this approach is in steps 2 and 3.

6.2.1 Terminology

We introduce some terminology that will be used to describe the adaptive On-site tracking approach.

Definition 6.1 A latest knowledgeable node *is a sensor node that has most recently observed the MT.*

It is quite possible that there could be more than one *latest knowledgeable node* in the network at any instance of time during the tracking.

Definition 6.2 *Let the MUT ask a knowledgeable node, say A, for the latest information about the MT. If A has that information then it will supply the information immediately. Otherwise, it will initiate a request message for the latest information in the network, obtain the information, and then supply this to the MUT. Here, we call A the* **guide**, *and the knowledgeable node that supplied the latest information about the MT the* **reporter**.

A *latest knowledgeable node* can assume the role of *reporter* that will be explained in the next section. With this terminology, first we explain *MUT*'s tracking strategy.

6.2.2 Tracking Strategy

The *reporters* might change as the *guides* change over the time. In our adaptive On-site tracking, the *first knowledgeable node* that *MUT* visits is the first *guide*. Then, it performs the following tasks repeatedly until it encounters the *MT* within the distance, δ.

1. It asks the *guide* for the latest information about the *MT* supplied by the *reporter*.

2. Then the *MUT* travels straight to the *reporter*.

3. The *reporter* becomes the *guide*.

This way the *MUT* visits only the *first knowledgeable node* and the *reporters*. This brings us to the question of maintaining the latest information about the MT in the network.

6.2.3 Maintaining the Latest Information about the *MT*

Updating the *knowledgeable nodes* along the track of the *MT* can be achieved in number of ways. We list two of them next.

1. *Self Updated Path:* The sensor nodes continuously update the path.

2. *On-demand Updated Path:* The sensor nodes update the path only when the *MUT* demands.

In the first approach, the *MUT* is lazy in its nature, and it assumes that *knowledgeable nodes* along the path of the *MT* would always be updated in advance before the *MUT* visits them to get the information of the *latest knowledgeable node.* Hence, we named it as (L)azy (AD)aptive (M)obilt (U)nit for (T)racking (*LADMUT*).

In the second approach, the *MUT* pro-actively asks the sensor nodes to build a path for the messages to travel and get the latest information of the *MT*. (PA)th is updated on (D)emand by the (M)obile (U)nit for (T)racking, and hence named as *PADMUT*. This approach is simple modification of the first approach; but it is more energy efficient.

Self Updated Path Approach (*LADMUT*)

In this approach, the *MUT* achieves the task of tracking as each node in the network does the following.

- Whenever a node encounters the *MT* it stores the information about the *MT* (along with the timestamp) and becomes the *latest knowledgeable node.* Then it sends its observation to its neighbors.

- If a node is a *knowledgeable node* and it receives the information about the *MT* from its neighbor, then it does the following. If the received timestamp is larger than its own timestamp, then update the information about the *latest knowledgeable node* and forward this new information to its neighbors. Otherwise ignore the message.

In summary, the value of the timestamp possessed by a particular *knowledgeable nodes* at any given point of time is the latest time at which that particular *knowledgeable node* or the *latest knowledgeable node* has encountered the *MT*. Any other *knowledgeable node* receiving information from two different *knowledgeable nodes* may

choose either of them to store and forward the information of the selected *knowledgeable node*. This way all the *knowledgeable nodes* in the network would have the latest information about the MT.

An Optimization of $LADMUT$

In $LADMUT$, the communication flow between the *latest knowledgeable node* and the rest of the *knowledgeable nodes* is continued until the MUT reaches the MT within the δ distance. This would unnecessarily cost more energy. Instead, if the flow of information between two *reporters* on the track of the MT is stopped once MUT has reached the later *reporter*; it would save the energy of the *knowledgeable nodes*. To achieve this, the *reporters* that have already been visited by the MUT could simply stop forwarding the information to their neighbors. That is, the MUT initiates the stoppage of message flow incrementally as it visits the *reporters* along the path of the MT.

On-Demand Updated Path Approach ($PADMUT$)

In this approach, the MUT demands the latest information when it visits the *knowledgeable nodes*. This way, the information about the latest position is routed only when needed and that would save considerable amount of energy. The update involves subtle details. For example, when the MUT visits a *knowledgeable node* along the track, we do not want the request message to travel along the downstream track.

Processing of the request for the latest information by the MUT involves three steps: (1) Sending an upstream request message along the track of the MT, (2) Determining the latest *knowledgeable node*, and (3) Forwarding a downstream reply message along the track of the MT. Next, we explain these three steps in detail.

1. **Sending an upstream request message along the track of the MT**

   A *guide* initiates a request message to get the latest information of the MT after receiving a query message from the MUT. Initiating a request message

involves simply forwarding the timestamp to its neighbor *knowledgeable nodes.* The nodes that receive the request compare the received timestamp with their own timestamp and perform the following task.

- If the received timestamp is less than its own timestamp, then forward the request message with its own timestamp.

This step is repeated until the request message reaches the *latest knowledgeable node*, the *reporter*. The timestamp of the *guide* assures that the request message does not travel (i.e. downstream) towards the nodes that have older timestamp than the *guide.* It is quite possible that there may be more than one *guide* during any demand; but *knowledgeable nodes* would respond to just one *guide.*

2. **Determining the latest knowledgeable node (reporter)**

   A *knowledgeable node* would choose to become a *reporter* after it has received a request message, and if it has not received a timestamp greater than its own timestamp within a thresh hold value of time. Obviously, the *latest knowledgeable node* would not receive such a message. However, the converse need not be true. That is, a node which does not receive such message need not be a *latest knowledgeable node.* For example, a *knowledgeable node*, say A, which is neither the latest nor has any neighbor with a higher timestamp would also not receive such a message. This problem can be solved if one of A's neighbors could notice the problem and alert its neighbors. The problem can be noticed by a *knowledgeable node* if it receives an equal timestamp and a larger timestamp from its neighbors. When a *knowledgeable node* notices this problem, it could immediately send an alert message to its neighbors, to inform of the existence of correct reporters.

3. **Forwarding a downstream reply message along the track of the MT.**

After receiving the request message, the *reporter* sends the downstream reply message. The downstream reply message travels as follows.

- If the received timestamp is greater than its own timestamp and has participated in the upstream request, then forward the reply message with its own timestamp.

This process avoids forwarding the redundant messages and also assures that the reply message does not travel beyond the *guide*.

6.3 Theoretical Analysis

6.3.1 Terminology

We introduce the following parameters for our analysis.

- r - transmission range of the sensor nodes and the MUT.
- d_0 - diameter of the network region NR. That is, the distance between the farthest points in the network.
- a - area of NR.
- $n = |SN|$ - the number of stationary nodes in NR.
- d - distance traveled by MT, from its initial position, before it is tracked.
- v_{mt} - velocity of the MT.
- v_{mut} - velocity of the MUT.
- t - tracking time.
- m_t - total number of messages generated by the sensor nodes during the tracking time, t.

- k - the maximum number of reporters that the MUT would visit (number of reportings) to achieve the desired distance.

6.3.2 An Upper Bound on the Tracking Time

Based on the above parameters, we derive an upper bound on the tracking time, t, for On-demand updated path approach.

First we derive an upper bound for k.

Lemma 6.1 *On-demand updated path approach for the On-site tracking assures that the maximum number of reportings in which MUT can reach within the desired distance of the MT,* $k \leq \lceil \log_{v_r}(\delta/d_0) \rceil$, *where* $v_r = \frac{v_{mt}}{v_{mut}}$

Proof:

We consider the worst case scenario that the MT always travels in a straight line between two consecutive reporters. Let d_i represent the distance between the i^{th} and $(i+1)^{th}$ reporters. The distance traveled by the MT between the first and second reporter can be computed as follows.

$$d_1 = d_0 \left(\frac{v_{mt}}{v_{mut}} \right) \tag{6.1}$$

Similarly the distance between the second and the third reporter is:

$$d_2 = d_1 \left(\frac{v_{mt}}{v_{mut}} \right) = d_0 \left(\frac{v_{mt}}{v_{mut}} \right)^2 \tag{6.2}$$

In general, we have:

$$d_i = d_0 \left(\frac{v_{mt}}{v_{mut}} \right)^i \tag{6.3}$$

After reaching the k^{th} *reporter*, the MUT must be within the δ distance from the MT. Therefore we have:

$$d_k \leq \delta \tag{6.4}$$

Now substituting the value of $i = k$ in the equation (6.3), and solving equation (6.4) for the value of k, we get:

$$k \leq \log_{v_r}(\delta/d_0) \tag{6.5}$$

Since k must be an integer value, $k \leq \lceil \log_{v_r}(\delta/d_0) \rceil$.

Theorem 6.1 *On-demand updated path approach for the On-site tracking assures that the tracking time,* $t \leq \frac{d_0\left(1-v_r{}^{\lceil \log_{v_r}(\delta/d_0)\rceil}\right)v_r+(d_0-\delta)(1-v_r)}{v_{mut}(1-v_r)}$, *where* $v_r = \frac{v_{mt}}{v_{mut}}$

Proof:

From the equation (6.4) of the Lemma (6.1), the total distance traveled by the MT, d is:

$$d \leq \sum_{i=1}^{k} d_i = d_0 \left[\left(\frac{v_{mt}}{v_{mut}}\right) + \left(\frac{v_{mt}}{v_{mut}}\right)^2 + ... \left(\frac{v_{mt}}{v_{mut}}\right)^k \right] \tag{6.6}$$

By simplifying the equation (6.6) we get:

$$d \leq d_0 \left(\frac{1 - {v_r}^k}{1 - v_r}\right) v_r \tag{6.7}$$

To consider the worst case scenario of the tracking time, we assume that the MUT does not find any other *knowledgeable node* before reaching the first reporter. Therefore the total tracking time, t, of the MUT is:

$$t \leq \frac{d_0 + d - \delta}{v_{mut}} \tag{6.8}$$

Now by substituting the value of d from the equation (6.7) into the equation (6.8), and then solving it we have:

$$t \leq \frac{d_0(1-{v_r}^k)v_r + (d_0-\delta)(1-v_r)}{v_{mut}(1-v_r)} \tag{6.9}$$

Now substitute the value of k in the equation 6.9, we have:

$$t \leq \frac{d_0\left(1-{v_r}^{\lceil \log_{v_r}(\delta/d_0)\rceil}\right)v_r + (d_0-\delta)(1-v_r)}{v_{mut}(1-v_r)} \tag{6.10}$$

Hence the proof.

Corollary 6.1 *On-demand updated path approach for the On-site tracking assures that the target can be tracked in a finite time, if* $v_{mut} > v_{mt}$.

6.3.3 An Upper Bound on the Number of Messages Generated by the Sensor Nodes

In this section we derive an upper bound on m_t for the On-demand updated path approach for On-site tracking.

Theorem 6.2 *On-demand updated path approach for the On-site tracking assures that the number of messages generated by the nodes during* t, $m_t \leq \frac{an(1-v_r)+4rnd_0(1-{v_r}^k)v_r}{a(1-v_r)}$, *where* $v_r = \frac{v_{mt}}{v_{mut}}$

On-demand updated path approach for tracking involves three logical stages of message generation by sensor nodes: (i) initial flooding, (ii) sending the upstream query message by the reporting *knowledgeable nodes* on the demand of the MUT, and (iii) sending the downstream reply message initiated by the latest *knowledgeable nodes*. Let m_x, m_y, and m_z respectively, be the messages generated in these three stages. We compute the upper bounds for m_x, m_y, and m_z separately and then add them to get the upper bound for m_t. The value of m_x is given in the equation (5.8).

Total number of upstream query messages generated by the *knowledgeable nodes* would be the total number of *knowledgeable nodes* between the *guide* and the *reporter* during any request by the MUT. Therefore the total number of upstream query

messages generated during any reporting (for $i = 1, 2...k$) gives us the following inequality.

$$m_y \leq \left(\frac{2rd_i}{a}\right) n \tag{6.11}$$

Similarly, m_z is:

$$m_z \leq \left(\frac{2rd_i}{a}\right) n \tag{6.12}$$

An upper bound on the total number of messages generated by the senor nodes during the tracking time, t, can be computed by adding all the upstream query messages, downstream reply messages during all the reportings along with the total number of messages generated during the flooding phase. From equations (6.10), (6.11) and (6.12) we have:

$$m_t \leq n + 2\sum_{i=1}^{k} \left(\frac{2rd_i}{a}\right) n \tag{6.13}$$

By substituting the value of $\sum_{i=1}^{k} d_i$ from equation (6.7) into equation (6.13), and then by simplifying it, we have:

$$m_t \leq \frac{an(1 - v_r) + 4rnd_0(1 - {v_r}^k)v_r}{a(1 - v_r)} \tag{6.14}$$

This completes the proof.

From Theorem 6.2, an upper bound on the total energy spent by the sensor nodes during tracking can be easily calculated. These upper bounds reflect the costs for worst case scenarios. Average case analysis would help to understand the normal behavior of the system. As discussed previously, computing the average case values for tracking is quite complex. The complexity arises due to the variabilities of the parameters such as the initial positions of the MUT and the MT (they can be any at position in the entire network), mobility pattern of MT, speed of the MT (may vary from 0 to v_{mt}), etc. However, to see the closeness to the simulation results, we

compute a simplified average case values for t and m_t next.

Theorem 6.3 *Assume that*

(i) MUT travels $\frac{d_0}{2}$ *to reach the first knowledgeable node,*

(ii) MT travels with an average velocity of $\frac{v_{mt}}{2}$, *and*

Then,

(a) $t \leq \frac{d_0(1-{v_r}^k)v_r+(d_0-\delta)(1-v_r)}{2v_{mut}(1-v_r)}$

(b) $m_t \leq \frac{2an(1-v_r)+4rnd_0(1-{v_r}^k)v_r}{2a(1-v_r)}$

where $k \leq \lceil \log_{v_r}(2\delta/d_0) \rceil$ *and* $v_r = \frac{v_{mt}}{2v_{mut}}$.

The proof is similar to the proofs of Theorems 6.1 and 6.2.

6.4 Simulation Study

In the simulation, we are mainly interested in studying the performance of the On-demand updated path approach for the adaptive On-site tracking. In addition to that we are also interested in comparing it with the ant based method presented in Chapter 5 and other existing methods like TTDD[29].

6.4.1 Experimental Setup

We carried out experiments for the various on-site tracking scenarios using our software simulation model as discussed previously. To simulate all the approaches in the identical conditions, we use sinks that start from the same position. The initial position of the *MT* is generated randomly, and is allowed to have a random movement. The mobility pattern of the *MT* is restricted in such a way that it remains in the network region. As described previously, the sensor nodes are deployed suitably to cover the network area (Figure 4.2). The number of sensor nodes required to cover

the network area is computed based on the network area and the transmission range of a sensor node.

The parameters for the simulation are set as follows.

1. Speed of the moving target is fixed as $5m/s$.
2. Speed of the moving sinks varies from $8m/s$ to $16m/s$.
3. Area size, a, varies from $15000m \times 15000m$ to $22000m \times 22000m$.
4. The transmission range r is computed as $(1/40)^{th}$ of the side of the square region.
5. Desired distance, δ, varies from $20m$ to $50m$.
6. Frequency of closeness, f, is fixed as 1.

6.4.2 Results Analysis

Simulation results are mainly collected for three performance metrics: (i) average no. of messages generated by the nodes, (ii) average energy consumption per node, and (iii) average time taken for tracking the MT. We compute the energy costs based on the messages transmitted and received by the sensor nodes We investigate these three metrics by, (1) varying the size of the area, (2) varying the speed of the Sink or MUT and (3) varying the desired distance, δ. Results obtained for each value of these varying parameters are an average of 100 simulation runs.

In the following experiments we depict the sink with the ant-based approach for On-site tracking as MUT, and the sink with On-demand updated path approach for on the adaptive On-site tracking as $PADMUT$.

Experiment 1 (*Area vs. Energy used per sensor node*): In this experiment, we are interested in computing the average amount of energy spent by the sensor nodes on message communication. Sensor nodes spend energy for both sending and receiving the messages as mentioned previously. Therefore, we first calculate the number of messages sent and received by the sensor nodes in the entire network, and then compute the average energy spent per sensor node. The results are summarized in graphs as shown in Figure 6.2.

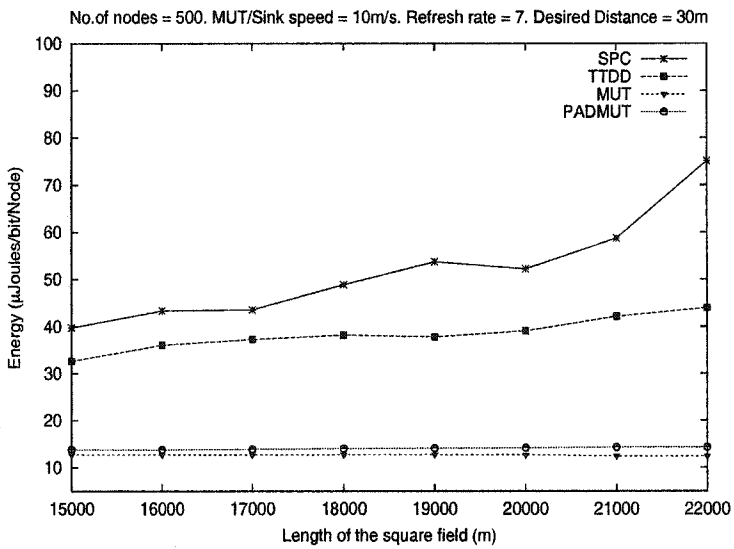

Figure 6.2: Area vs. Energy/Node

Observation 1: The average energy consumed per node is quite less in the case of the *MUT* and *PADMUT* as compared to TTDD and SPC. *PADMUT* has slightly higher values than *MUT*. It increases slightly for both *MUT* and *PADMUT* as the area increases, but it increases more rapidly for *SPC* and *TTDD*. We note that, for an area size of 20000 x 20000 m^2, the average energy consumption per node for SPC, TTDD, *MUT* and *PADMUT* are 52.204 μJoules/bit, 39.04 μJoules/bit, 12.69 μJoules/bit and 14.16 μJoules/bit respectively. This shows that *PADMUT* is slightly expensive than *MUT* but it is highly energy efficient as compared to the TTDD and SPC methods.

Experiment 2 (*Area vs. Tracking Time*): In this experiment, we are interested in comparing the average time taken for tracking the MT in $PADMUT$ method with that of TTDD. The results are summarized in graphs as shown in Figure 6.3.

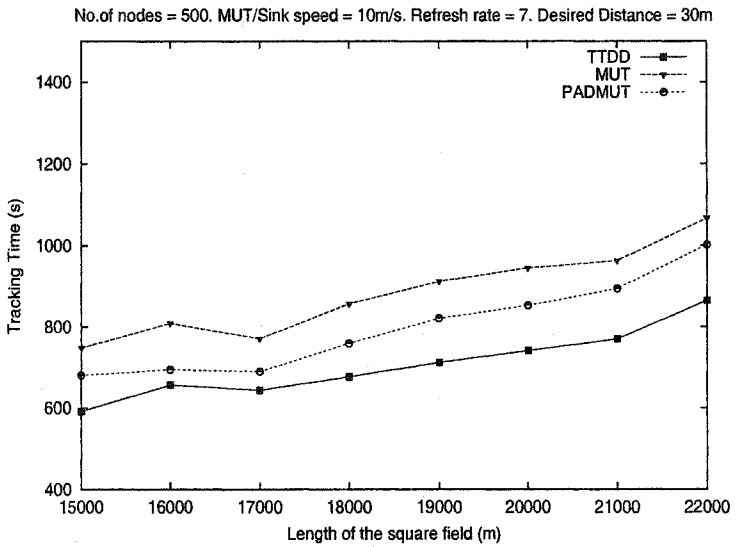

Figure 6.3: Area vs. Time

Observation 2: The average tracking time in the case of TTDD increases with the increase in the area, and so is the case with MUT and $PADMUT$. MUT's tracking time has been improved by applying $PADMUT$ approach of tracking. Large tracking time for the MUT is because; it normally travels a larger distance as compared to TTDD while trying to achieve the desired distance.

Note: We conducted the study by fixing the value of r and increasing the number of nodes when increasing the area. We observed a similar behavior.

To verify the simulation results for the tracking time of $PADMUT$, we compared the simulation results with the theoretical upper bound on tracking time as derived in Section 6.3. We observe that the experimental values are well below the upper bound calculated. The results are summarized in graphs as shown in Figure 6.4.

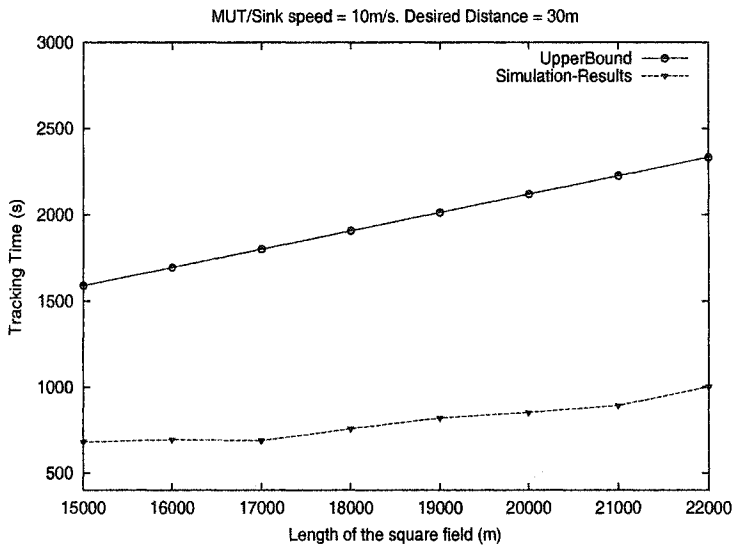

Figure 6.4: Area vs. Time

In the next two experiments, we consider the varying speed of the sink. We vary the speed of the sinks while keeping the network area and number of nodes fixed. In these experiments we consider the network area of size 20000 x 20000 m^2 with 4000 nodes in it. The maximum speed of the moving target was fixed at 5 m/s.

Experiment 3 (*Sink speed vs. Energy used per sensor node*): In this experiment, we are interested in finding how the sink's speed affects the energy consumption by the nodes. The results are summarized in graphs as shown in Figure 6.5.

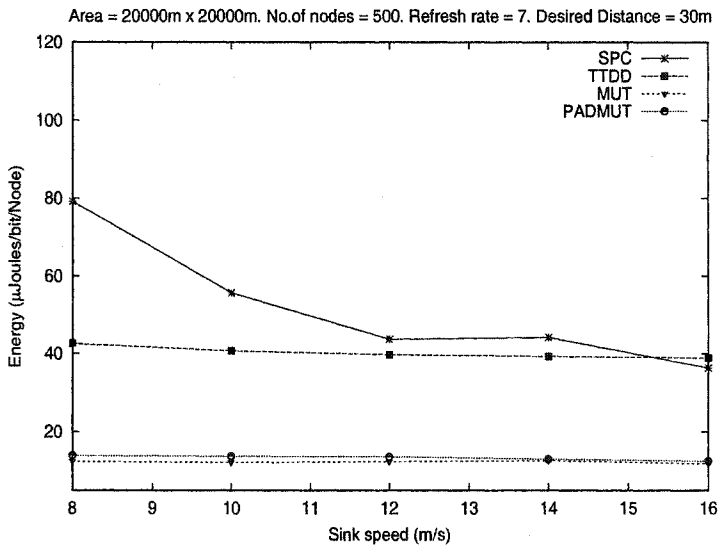

Figure 6.5: Speed vs. Energy/Node

Observation 3: As we vary the speed of the sinks from 8 m/s to 16 m/s, we observe that the message generation reduced in the case of SPC. As the sink's speed increases it is expected to track the target faster. There is a slight decrease in the case of TTDD, but MUT's average number of messages remains almost constant and also in the case of *PADMUT*. In terms of energy consumption, there is a decrease in the energy consumption per node in SPC and TTDD, but it remains almost constant in the case of *MUT* and *PADMUT*. In 100 simulation runs with the sink speed of 12 m/s, the average energy consumption per node in SPC, TTDD, *MUT* and *PADMUT*, respectively, are 43.72 μJoules/bit, 39.771 μJoules/bit, 12.412 μJoules/bit and 13.694 μJoules/bit.

Experiment 4 (*Sink speed vs. Tracking Time*): In this experiment, we vary the sink's speed to observe its affect on the tracking time. The results are summarized in graphs as shown in Figure 6.6.

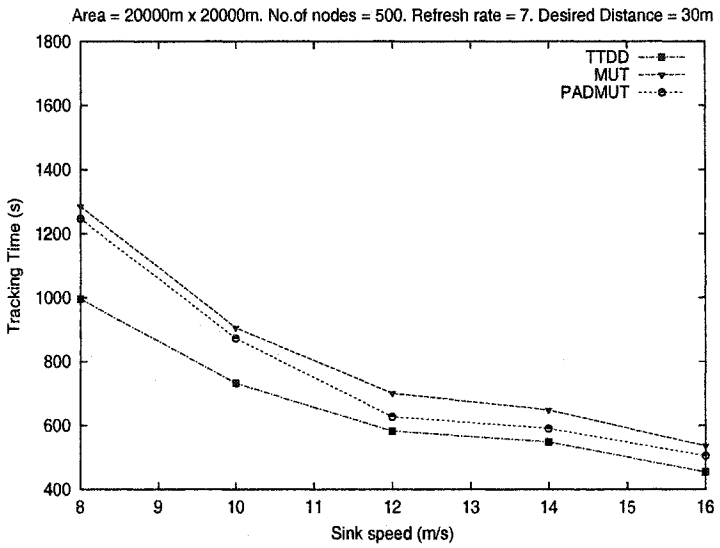

Figure 6.6: Speed vs. Time

Observation 4: The average tracking time in the case of TTDD decreases with the increase in speed as it is obvious that the sink with more speed will be able to track the target faster. The average tracking time in *MUT* and *PADMUT* is comparatively higher than that of TTDD, but the *PADMUT* performs better than the *MUT*.

To verify the simulation results for the tracking time of $PADMUT$, we compare it with the upper bound on tracking time for the varying speed of the MUT. We observe that the experimental values are well below the upper bound calculated in the analytical section. The results are summarized in graphs as shown in Figure 6.7.

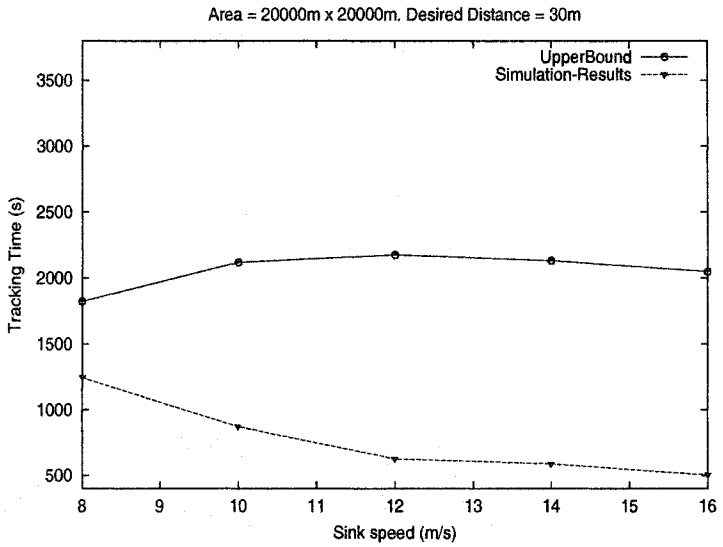

Figure 6.7: Speed vs. Time

Next, we evaluate the performance by taking third parameter that is desired distance, δ. In this experiment, we vary δ, while keeping the speed of sinks and network area fixed. In the network area of 20000 x 20000 m^2 with 4000 nodes in it, we vary δ from 20m to 50m while keeping the sink speed fixed at 10 m/s.

Experiment 5 (*Desired Distance vs. Energy used per sensor node*): In this experiment, we evaluate *PADMUT*'s performance in terms of energy consumption. As we vary δ, we observe its impact on the average energy consumption by nodes. The results are summarized in graphs as shown in Figure 6.8.

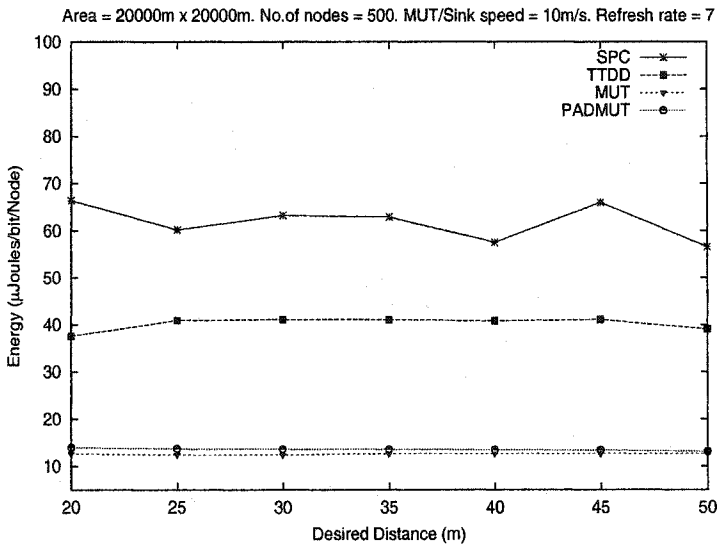

Figure 6.8: Desired Distance vs. Energy/Node

Observation 5: There are small variations in energy consumption in the case of *PADMUT*. It is more efficient than SPC and TTDD as shown in Figure 5.8, but consumes little higher energy than the *MUT*. We ran 100 simulations for each sample value of δ. We observe that the average energy consumption per node in SPC, TTDD, MUT and PADMUT is 62.876 μJoules/bit, 41.09 μJoules/bit, 12.685 μJoules/bit, and 13.588 μJoules/bit respectively for a δ value of 35m.

Experiment 6 (*Desired Distance vs. Tracking Time*) In this experiment, we are interested in observing the average tracking time of *MUT* against the varying value of δ. The results are summarized in graphs as shown in Figure 6.9.

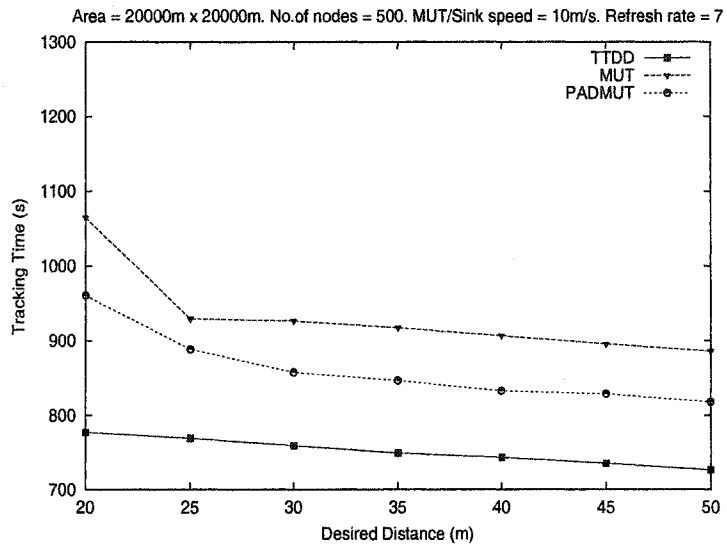

Figure 6.9: Desired Distance vs. Time

Observation 6: Like the previous experiments for the average tracking time of the *PADMUT*, in this experiment also it performs better than *MUT*. It's tracking time gradually decreases as the value of δ increases. This is expected because *PADMUT* has to capture the target from a large distance instead of following it closely.

We also verified that the experimental values obtained for the tracking time (by varying the δ value) are well below the upper bound calculated in the analytical section. The results are summarized in graphs as shown in the Figure 6.10.

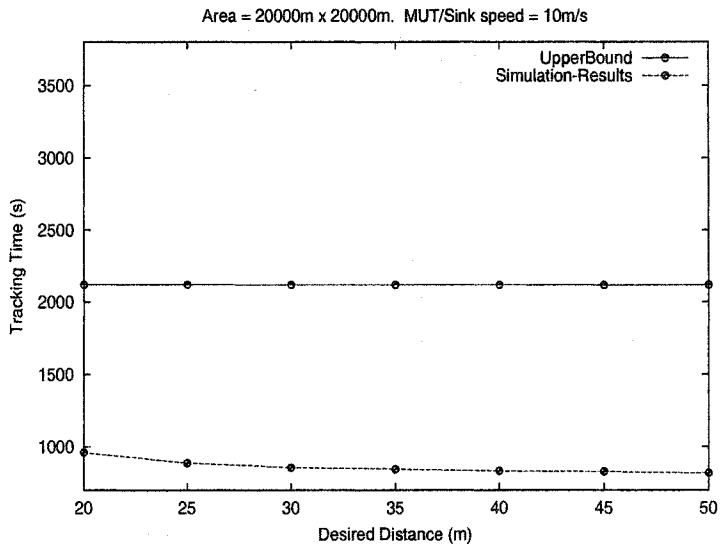

Figure 6.10: Desired Distance vs. Time

6.5 Conclusion

In this chapter, we proposed two new routing algorithms to solve the On-site tracking problem. It can be deduced from the simulation results that the adaptive nature of the On-site tracking reduces the tracking time of the *PADMUT* as compared to the basic approach presented in Chapter 5. *PADMUT* achieves this without compromising much on the energy expenditures. In the next chapter we conclude this thesis while outlining some of the future directions of our research.

Chapter 7

Conclusion and Future Directions

7.1 Conclusion

Recently, wireless sensor networks have received increasing attention from the research community. Many applications based on these networks will be realized in the near future. Tracking a moving object using sensor nodes is one such important application. A considerable attention has been paid, and various approaches have been proposed in order to solve the problem.

Our contribution via this thesis are many fold. We classified the tracking problem into two broad categories of *On-site tracking* and *Off-site tracking* problem. Based on this classification we are able to present a taxonomy of the tracking problems. To the best of our knowledge, no such classification on this problem has been available in the literature.

Our focus in this thesis is On-site tracking in the context of wireless sensor networks. After formally characterizing this problem for single target case, we generalized the problem for multiple targets case. We proposed a class of algorithms to solve the problem for both the cases. The first set of algorithms are based on an ant-based approach. Through our extensive simulation study we showed that our algorithms are energy efficient. We also derived theoretical bounds for the tracking time and the number of messages generated by the senor nodes. In these ant based algorithms,

we observed that, the tracking time is comparatively higher than that of the existing approaches that lead to our investigation for more efficient algorithms to solve the problem under consideration. One of the other known limitations of the ant-based algorithms is that they are not generic in nature, and therefore, they may not be directly applicable to other type of tracking problems.

In our second set of algorithms, we devised a path adaptive approach for the On-site tracking. These algorithms performed well on the tracking time without compromising much on the energy expenditures. Performance of our algorithms have been revealed in the simulation study. We also computed the theoretical bounds on the tracking time and the number of messages generated for our algorithms.

Our other contribution in this thesis is the design of a simulation software called OSTSim that we built for the performance study of the algorithms that could solve the On-site tracking problem. Developing this simulator was a worth while experience.

7.2 Future Directions

There are many directions in which the work presented in this thesis can be expanded. We outline some of them next.

- The ant-based algorithms presented in Chapter 5 can be explored to look for more optimal solutions for the tracking related problems.
- There are many variations of our proposed algorithms that can be incorporated in the generalized On-site tracking problem.
- An other interesting area to explore is the mobility pattern of the targets, and to see that, which tracking strategy suits most for a particular type of target. Our intuitive ideas are that the nature of the target and the path it traces will highly influence the type of solutions we design for a particular type of target.

Bibliography

[1] P. Sun. Connectionless Routing Protocols for Mobile Ad-Hoc Networks. MSc Thesis, University of Northern British Columbia, 2004.

[2] Bell Labs, Lucent Bell Labs's Top Innovations, http://www.bell-labs.com/about/history/timeline.html

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. Computer Networks (Elsevier), Vol. 38, pp. 393-422, Mar. 2002.

[4] L. Clare, G. Pottie, and J. Agre. Self Organizing Distributed Sensor Networks. In SPIE Conf. on Unattended Ground Sensor Technologies and Applications, Orlando, FL, USA, pp. 229-237, Apr.1999.

[5] M. Dong, K. Yung, and W. Kaiser. Low Power Signal Processing Architectures for Network Microsensors. In Proc. of Int. Symposium on Low Power Electronics and Design, Monterey, CA, USA, pp. 173-177, Aug. 1997.

[6] H. Karl and A. Willig. A short survey of wireless sensor networks. TKN Technical Reoprt TKN-03-018. Berlin, Germany, Oct. 2003.

[7] K. Akkaya and M. Younis, "A Survey of Routing Protocols in Wireless Sensor Networks, " in the Elsevier Ad Hoc Network Journal (to appear)

[8] L. Kleinrock. An internet vision: the invisible global infrastructure. Ad Hoc Networks, published by Elsevier B.V. 2003.

[9] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In the 2002 ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02), Atlanta GA, September 28, 2002. (also Intel Research, IRB-TR-02-006)

[10] G.J.Pottie and W.J.Kaiser. Wireless integrated network sensors. Communications of the ACM, 43(5):51-58, May 2000.

[11] D. Ganesan, A. Cerpa, W. Ye, Y. Yu, J. Zhao, and D. Estrin Networking Issues in Wireless Sensor Networks. Journal of Parallel and Distributed Computing (JPDC), Special issue on Frontiers in Distributed Sensor Networks, Elsevier Publishers, To appear.

[12] J. Suh and M. Horton. Current Hardware and Software Technology for Sensor Networks. First International Workshop on Networked Sensing Systems (INSS2004), University of Tokyo, Japan. June 22-23 2004.

[13] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/

[14] GloMoSim, Global Mobile Information Systems Simulator Library, http://pcl.cs.ucla.edu/projects/glomosim/

[15] QualNet Network Simulator by Scalable Network Technologies, http://www.scalable-networks.com/

[16] Opnet, http://www.opnet.com/

[17] B. S. Malhotra and A. A. Aravind. Energy-Efficient On-Site Tracking of Mobile Target in Wireless Sensor Networks. Proc. of the Intl. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP '04), Melbourne, Australia, pp. 43-48, 14-17 Dec. 2004.

[18] B. S. Malhotra and A. A. Aravind, A Master Sink Based Model for On-Site Tracking of Multiple Mobile Targets in Wireless Sensor Networks. Proceedings of the Second International Conference on Intelligent Sensing and Information Processing (ICISIPA05), Chennai, India, 4-7 Jan. 2005.

[19] B. S. Malhotra and A. A. Aravind. Path-adaptive On-Site Tracking in Wireless Sensor Networks. Submitted to the IEICE Transactions (Special Section on Parallel and Distributed Computing and Networking), 2005.

[20] B. S. Malhotra and A. A. Aravind. OSTSim: Simulation Software for the On-site Tracking in Wireless Sensor Networks. Proceedings of the Western Canadian Conference on Computing Education (WCCCE'05), Prince George, BC, Canada, 5-6 May, 2005.

[21] A. Dussutour, V. Fourcassie, D. Helbing, and J. L. Deneubourg. Optimal Traffic Organization in Ants Under Crowded Conditions. Nature 428, pp. 70-73, Mar. 2004

[22] H. Yang and B. Sikdar. A Protocol for Tracking Mobile Targets using Sensor Networks. Proc. of IEEE workshop on sensor Network Protocols and Applications, Anchogare, AK, pp. 71-81, May 2003.

[23] R. Gupta and S. R. Das. Tracking Moving Targets in a Smart Sensor Networks. Proc. of VTC Fall 2003 Symposium, Oct. 2003.

[24] Y. C. Tseng, S. P. Kuo, H. W. Lee, and C. F. Huang. Location Tracking in a Wireless Sensor Networks by Mobile Agents and Its Data Fusion Strategies. The Computer Journal, Volume 47, Issue 4, pp. 448-460, July 2004.

[25] T. Hegazy, G. Vachtsevanos. Dynamic Agents Self-Deployment for Tracking Moving Targets Based on Target Motion Model. 12th Mediterranean Conf. on Control and Automation (MED04), Aydin, Turkey, WSEAS Transactions on Circuit and Systems, Issue 3, Volume 3, pp. 514-520, May 2004.

[26] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed Diffusion for Wireless Sensor Networking. IEEE/ACM Transactions on Networking, Vol. 11, No. 1, pp. 2-16, Feb. 2003.

[27] M. Debirbas, A. Arora, and M. Gouda. Pursuer-Evader Tracking in Sensor Networks. To appear in Sensor Network Operations, IEEE Press, Sept. 2004.

[28] C. Arpin. Director of Research and Development Boomerang Tracking Inc., Global Positioning System (GPS) Errors and Limitations for Vehicle Tracking, A White Paper. http://www.boomerangtracking.com/en/pdf/whitepaper.pdf

[29] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang. TTDD: Two-Tier Data Dissemination in Large-scale Sensor Networks. Proc. of Eigth Annual Int. Conf. on Mobile Computing and Networking (MOBICOM'02), Atlanta, Georgia, USA, ACM Press, pp. 148-159, Sept. 2002.

[30] A. Chakrabarti, A. Sabharwal, and B. Aazhang. Using Predictable Observer Mobility for Power Efficient Design of Sensor Networks. 2nd Int. Workshop on Information Processing in Sensor Networks (IPSN'03), Berkeley, California, USA, April 2003.

[31] S. Jain, R. Shah, W. Brunette, G. Borriello, and Sumit Roy. Exploiting Mobility for Energy Efficient Data Collection in Sensor Networks. 2nd Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks. WiOpt 2004, University of Cambridge, UK, 2004.

[32] P. Venkitasubramaniam, Q. Zhao, and L. Tong. Sensor Networks with Multiple Mobile Access Points. Proc.of the 38th Annual Conf. on Information Sciences and Systems (CISS'04). Princeton, NJ, USA, Mar. 2004.

[33] H. E. Erikson and M. Penker. UML Toolkit. Wiley Computer Publications. John Wiley and Sons, Inc. 1998.

[34] W. H. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. Proc. of the Hawaii Intl. Conf. on System Sciences, Maui, Hawaii, USA, pp. 1-10, Jan. 2000.

[35] G. DiCaro and M. Dorigo. Two ant colony algorithms for best-effort routing in datagram networks. In Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98), Las Vegas, USA, pp. 541-546, IASTED/ACTA Press, 1998.

[36] M. Gunes, U. Sorges, and I. Bouazizi, ARA - The ant-Colony Based Routing Algorithm for MANETs. Proc. of Int. Conf. on Parallel Processing Workshops (ICPPW'02), Vancouver, BC, Canada, p.79, Aug. 2002.

[37] J. Ibriq and I. Mahgoub. Cluster-Based Routing in Wireless Sensor Networks: Issues and Challenges. Proceedings of the 2004 Symposium on Performance Evaluation of Computer Telecommunication Systems (SPECTS'04), ISBN: 1-56555-284-9, 2004.