Secure Localization and Node Placement Strategies for Wireless Networks

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

Santosh Pandey

Certificate of Approval:

Farooq Anjum
Research Scientist
Telcordia Technologies, Inc.

Prathima Agrawal, Chair
Samuel Ginn Distinguished Professor
Electrical and Computer Engineering

Chwan-Hwa "John" Wu
Professor
Electrical and Computer Engineering

Chris Rodger
Professor
Mathematics and Statistics

Joe F. Pittman
Interim Dean
Graduate School

Secure Localization and Node Placement Strategies for Wireless Networks

Santosh Pandey

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
August 4, 2007

Secure Localization and Node Placement Strategies for Wireless Networks

Santosh Pandey

Signature of Author

Date of Graduation

Vita

Santosh Pandey, son of Ghanshyam Pandey and Sita Pandey was born in Mumbai (formally Bombay), India on October 25, 1980. He graduated high school from Kishinchand Chellaram College, Mumbai in March 1998. He graduated from Vivekanand Education Society's Institute of Tech., Mumbai University, with a Bachelor of Engineering degree in July 2002. He began his graduate studies in Electrical and Computer Engineering at Auburn University in August 2002.

Dissertation Abstract

Secure Localization and Node Placement Strategies for Wireless Networks

Santosh Pandey

Doctor of Philosophy, August 4, 2007
(M.S., Auburn University, 2005)
(B.S., Mumbai University, 2002)

135 Typed Pages

Directed by Prathima Agrawal

Localization and placement problems in wireless networks have been widely studied. The adaptation of localization in real world environment can be observed by popular commercial and non-commercial GPS applications. Intelligent placement of base stations in cellular network has led to efficient network deployment. However, with the advent of wireless local area networks (WLAN), ad hoc and sensor networks these problems are again brought into focus. The requirements such as low-cost, nodal resource constrains and multihop characteristics have made difficult problems such as localization and placement very hard to solve for contemporary wireless networks. These problems are considered in this work.

A low-cost secure localization scheme is proposed for infrastructure based WLAN. The scheme and its testbed implementation is based on transmission of special messages at different power levels through different access points (APs). Depending on the set of messages received by a user, his/her location is estimated. Measured results have shown this scheme to perform as well as the traditional signal strength based indoor localization

schemes. Moreover, the scheme is secure and prevents any intruder from spoofing his location.

The problem of node placement is investigated for wireless ad hoc and sensor networks. The placement and interface selection algorithm (PISA) is proposed for placing "drones" (multi-interface devices) that bridge heterogeneous networks. The objective is to place these "drones" to maintain a connected network. Furthermore, the problem of placing sophisticated nodes (SNs) in a hierarchical network is considered, given the placement of lower tier nodes called lite nodes (LNs). A HYBRID algorithm comprising of binary integer linear programming (BILP) and genetic algorithm (GA) is proposed to solve this problem. The placement schemes are validated using MATLAB simulations. Moreover, a mobility scheme is proposed to enable auto-configuration of SN placement without the knowledge of LN locations. The scheme is implemented and tested using OPNET. The results of these placement schemes indicate satisfactory placements of nodes for all test cases. These placement schemes demonstrate an increase in network connectivity, capacity and decrease in nodal energy; thus making the network more efficient.

Acknowledgments

This work wouldn't have been possible without the help and support of my family, my committee members, Auburn University faculty, and my friends and colleagues.

Firstly, I would like to thank my advisor Dr. Prathima Agrawal. Without her encouragement and guidance to investigate many potential ideas, I would not have ended up with such interesting problems. Most of the problems and solutions are a direct result of our discussions. Her enthusiasm in research has been contagious and a driving factor for the completion of this work.

I would further like to thank my committee members, Dr. Farooq Anjum, Dr. Chris Rodger and Dr. Chwan-Hwa "John" Wu. Dr. Farooq Anjum and Dr. Byungsuk Kim have also been very patient and encouraging mentors during my summer internships in Telcordia. Also, my thanks to Dr. Sanjeev Baskiyar for promptly reviewing my dissertation as an outside reader. Also, I would like to thank the Vodafone-US Foundation Fellowship Program that has provided financial support throughout my Ph.D. program.

The graduate program at Auburn University has been very helpful in building fundamentals for my research. I would like to thank all faculty members for their time and support; especially, Dr. Ramesh Ramadoss, Dr. Nedret Billor, Dr. Thomas Denney, Dr. Richard Jaeger and Dr. Lloyd Riggs. The Electrical and Computer Engineering staff members have made work a lot easier by their prompt support and help in many regards. I am very grateful for the help from Shelia Collis, Les Simonton and Joe Haggerty.

My colleagues in the Wireless Research Laboratory have always been there to discuss research ideas. The discussion ranged over a large range of topics and has brought out

interesting outlooks and solutions for various technical and non-technical problems. My special thanks to Pratap Simha, Priyanka Sinha, Ravi Paruchuri, Sowmia Devi, and Dr. Shaoqiang Dong. My other friends in and outside Auburn have occasionally provided a welcome distraction from research and studies.

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliograpy follows van Leunen's *A Handbook for Scholars*.

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `aums.sty`.

Table of Contents

List of Figures

Chapter 1

Introduction

Today's wireless applications like cellular phone services and television broadcast are a part of the day to day life of many people. Wireless communication has evolved immensely from the time it was first implemented. The ease of setting up a wireless network, tetherless communication, and low cost of deployment (as compared to a wired network) are some of the key reasons for its popularity. Also, the reliability of wireless communication has improved significantly and is reflected in its application in areas such as police radio, military communication, and disaster recovery services. This reliability is not only reflected in such public safety applications, but also in many civilian applications. Currently, many people have wireless cellular phones as their primary means of contact. Similarly, it is common to carry out a secure transaction over the Internet through a wireless local area network (WLAN). City wide WLAN deployments are planned in many metropolitan areas. The above examples indicate the widespread implementation of wireless networks, making wireless communication a technology that is available "anytime" and "anywhere". Wireless communication, thus, is no longer a luxury, but a necessity.

Apart from the infrastructure based networks, there is increasing interest in wireless ad hoc and sensor networks. Current research in wireless sensor networks envisions random deployment of inexpensive sensor nodes over a region of interest to monitor multiple parameters. The sensor nodes will autonomously configure themselves into a network that would monitor and deliver relevant data to a single collection center called *sink* [52]. Depending

on the Region of Interest (RoI), parameters to be collected, and the resolution of the required readings, hundreds or thousands of low-end simple sensor nodes such as Smart Dust [29] may be deployed. Moreover, monitoring multiple parameters may need deployment of specialized nodes. For example, in an intrusion detection system there may be laser sensors and cameras to detect and identify the intruder. It would be inefficient to send high bandwidth data through low-end sensor nodes. Not only will the energy of these sensor nodes deplete rapidly, the data itself may not be delivered due to the unreliable nature of the sensor network. Due to these scalability and reliability issues, a hierarchical architecture has been proposed [1].

In order to deploy efficient and service oriented wireless networks, it is essential to intelligently locate and place wireless devices. Location of wireless devices are in focus for realizing many commercial applications such as asset tracking (tracking appliances or people)[1], pervasive computing (differentiating services based on a user's location)[23] and location based access control [16]. On the other hand, the problem of optimal placement of wireless devices is frequently realized in cellular networks where minimum base stations are required to be placed for complete coverage and other service requirements [36]. Also, Access Points (AP) when placed inefficiently in an indoor WLAN would require to be redeployed adding to the cost of network maintenance [42]. Similarly, random deployment of nodes require 6 to 24 times more transmission power than intelligent placement in sensor networks [10]. This is especially costly for sensor nodes which work on low power. Determining optimal placements for such devices is required for many scenarios but is usually an NP-hard problem.

[1] http://www.pango.com/pango.aspx?id=638

1.1 Problem Definition Overview

We consider three scenarios where such device localization and placement is to be determined and propose algorithms for the same. These scenarios are:

- Secure localization of wireless users: We consider a 802.11b based indoor wireless LAN network where location of wireless users is to be estimated. Assuming the presence of an intruder in the network, a secure localization scheme to determine locations of wireless clients in WLANs is proposed.

- Drone placement in heterogeneous networks: A heterogeneous network, comprising of several homogeneous networks of different interfaces is considered. We determine the interfaces and placement of "drones" (multi-interface nodes) which enable bridging between different networks.

- Sophisticated node placement in hierarchical networks: We consider a two-tiered hierarchical network where the resource-constrained Lite nodes (LNs) form the lower layer and sophisticated nodes (SNs) form the upper layer. We determine the placement of SNs based on multiple design constraints.

In the subsequent chapters we briefly describe each of the scenarios, define the respective problems in detail along with the proposed solutions. Chapter 2 will discuss the secure localization scheme for wireless networks. It also provides simulation results with MATLAB[2] and measured results from a testbed deployment. Chapter 3 considers the drone placements required to maintain connectivity in heterogeneous networks. The simulated

[2]http://www.mathworks.com/

results using MATLAB are presented. Chapter 4 discusses the sophisticated node placement problem in hierarchical networks and simulation results from MATLAB are discussed. The placement problem for drones and sophisticated nodes are solved assuming that the underlying network is static. The mobility of nodes for ad hoc network or hierarchical network is further discussed in Chapter 5. The proposed node mobility scheme is tested using OPNET[3].

[3]http://www.opnet.com/

Chapter 2

Secure Localization

Location based services are expected to be the next "killer" application[1]. One such service that has not received much attention is a location based authorization service [14]. In this case, an entity will not only have to prove its identity but also provide evidence of being in the right location in order to get access to the network resources. For example, a user might have to be present in his office in order to access top-secret documents over WLAN or to participate in a conference call. In addition, location information can also be expected to be used for validating some mobile e-commerce transactions. Such services, further described in detail in Section 2.1.1, in which location determination is a major component would attract the attention of adversaries whose goal would be to try to deceive the localization system. An adversary could seek to achieve this while making use of special hardware, power variation etc. Given such an adversary (also referred to as an "intruder" or "attacker"), it would be necessary to design schemes that have some built-in security and are resilient to the attempts by the intruder to cheat the localization system. We refer to localization techniques that achieve this objective as *secure localization techniques* and the applications that depend on these techniques as *secure location based applications*.

In this work, we focus our attention on the problem of *secure low cost location determination* of a wireless client in an indoor WLAN. The normal range of a 802.11b Access Point (AP) spans from about 80 ft to 1750 ft in an office environment [51]. In many situations, the localization resolution requirements are much smaller than this area.

[1]Source: IDC 2004

We next describe the motivation of this work in Section 2.1 by pointing out various location based applications currently being developed and the shortcomings of the previous work to address security issues in this area. We describe the details of our proposed secure localization scheme in Section 2.2 along with the testbed implementation in Section 2.3. The simulated and measured performance of the scheme are reported in Section 2.4. Section 2.5 discusses the security of the proposed scheme.

2.1 Motivation

The motivation of this work stems from the various commercial applications which require location of the wireless user and their vulnerability to location attacks. We next point out the need for secure localization by describing the various location applications and the related research in this area.

2.1.1 Location Based Applications

As wireless networks have gained popularity, many location based services are being deployed. Localization plays a key role in ubiquitous computing which refers to the embedding of computers "everywhere" and also enables their easy interaction with people. Different services can be offered based on the location of the user [23]. For example, the nearest printer may be selected to print the user's document. In an enterprise network, access to the network or certain secure files can be restricted to users present only in specific locations. Such location based access filtering and authentication simplifies the key distribution and management in a large enterpise [16]. Localization also plays a key role in the emerging mobile services. Mobile services such as location specific advertising, location

sensitive information service, and tracking services are proposed [74]. Location information is also important in network planning; the location and performance of the user devices can be used as a low-cost alternative for network maintenance and planning [28]. Planning tools such as AirMagnet[2] and WiSE[3] are available for WLAN networks. Information about location has also been used to enhance the operation of 802.11 MAC leading to an increase in the system throughput by about 22 percent while also ensuring better fairness [43]. Asset management (also known as fleet management) relates to the management of assets such as personnel and vehicles of a large company (for example, workers and trucks of a construction company). In this case, location of various entities is required to efficiently track and manage company resources. Thus, localization is important in these upcoming areas such as ubiquitous computing, enterprise networks, mobile services, network planning, and asset management.

The widespread deployment of wireless networks in enterprise and commercial establishments has also accelerated the development of location based services for these private networks. The main goal of these services is to utilize location information in order to increase user efficiency or customer satisfaction. For example, PanGo Networks is working on a wireless location based asset-tracking project for Rockford Memorial Hospital[4]. The project aims at locating and tracking equipment in order to reduce operating cost and increase efficiency in the workplace. For achieving this, each equipment is tagged with a small 802.11 radio device and WiFi is used as an underlying technology to estimate its location. The tags send updated location information to a central database whenever the equipment

[2]http://www.airmagnet.com/
[3]http://www.bell-labs.com/org/wireless/wisext.html
[4]http://www.pangonetworks.com/News_Events/News/10_19%20Networld.htm

is moved. The project also plans to extend the technology to monitor Alzheimer's patients and infants within the hospital premises.

Another area of increasing interest is location based access control, wherein a user is granted access to network resources based on his/her location. In such a case, secure localization schemes would make it possible to implement organizational policies that restrict access to network resources, such as database access or higher bandwidth, based on the location of the wireless user. These organizational policies could include guidelines to track malicious users within the organization's physical perimeter or to restrict the locations from where users can access confidential material. In [16] the authors describe a key less authentication scheme for network access using wireless user's location. Location based access control can also facilitate applications such as the interactive dance club [26] where it is necessary to determine that a user is in "valid" locations before allowing her to communicate (interact) with the system. In many cases, such as intruder detection, the location of the data is as important as the data itself and thus secure localization is needed.

2.1.2 Related Work

The most prominent amongst the localization techniques is the global positioning system (GPS) [19]. The GPS receiver estimates its location (latitude, longitude, and altitude) based on satellite broadcasts. Although, GPS is widely used in outdoor environments, it is not suitable for indoor or cluttered environments. Also, incorporating GPS receivers would be costly (in terms of expense, power consumption, and resources) for standard laptops and PDAs. Designing secure location based applications to depend on GPS or similar technologies is not advisable. The GPS signals commonly used are also not secure [30]. It is

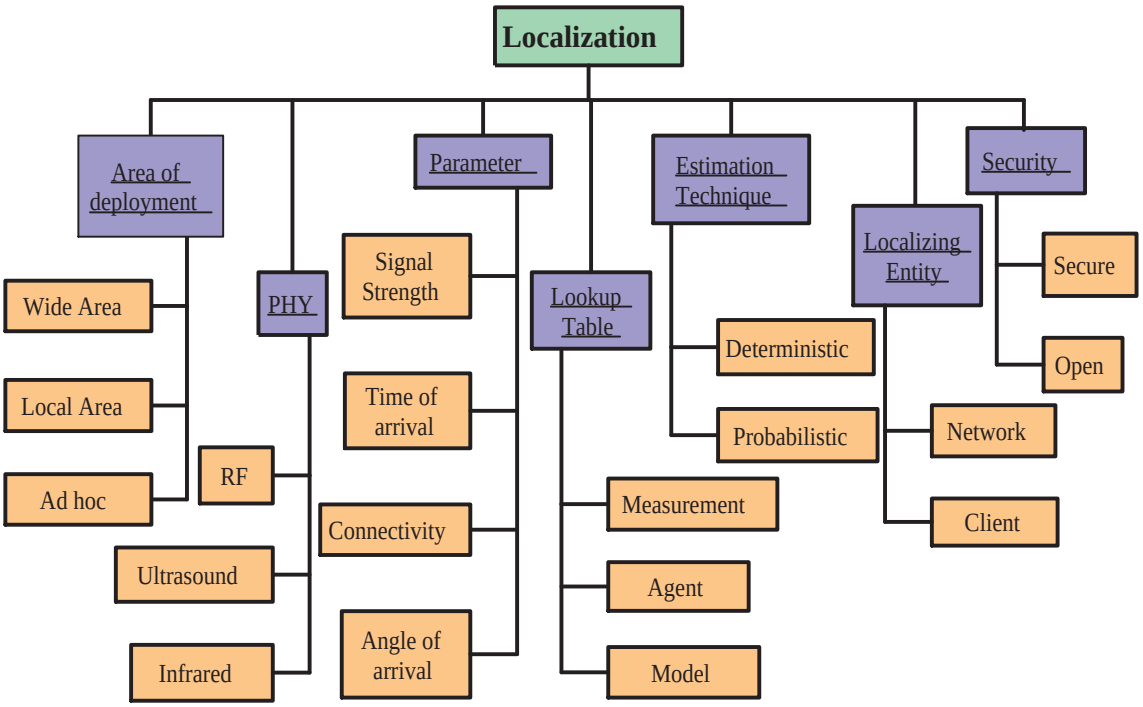


Figure 2.1: Localization Classification

very easy to spoof GPS signals thereby causing the GPS receivers to report wrongly on the location. Further, with GPS, the network will have to trust the end device to report the correct location. This would require the use of tamper proof GPS receivers. Tamper proof GPS devices while adding to the costs are also not foolproof [3]. Hence, it is necessary to investigate alternative technologies for securely determining the location of an end-user.

There have been many localization schemes proposed for wireless networks. These schemes are typically based on the features of the underlying physical layer. For example various schemes based on ultrasound [50], infrared [69], Bluetooth [21], 802.11 RF networks [17, 21, 5] have been proposed. These schemes infer the location of users by measuring various parameters such as received signal strength indicator (RSSI), time of flight [57], and angle of arrival [32]. Some of these schemes are *client based schemes* where the user

can determine his location and the network would have no knowledge of the user's location [17, 50] while others are *network based schemes* where the network infrastructure is used to determine the location of the user [8, 5]. Note that the former approach might be preferred when user privacy concerns abound. The classification of the schemes is shown in Figure 2.1. Interested readers may refer to [44] for details.

Localization schemes proposed for WLAN (802.11) systems are normally based on measuring the signal strength (SS) parameter [17, 21, 5, 38, 73]. The idea is to initially determine the SS map representing the SS at various locations. The system then tries to determine the location of the user based on the best match between the observed signal strength and the SS map. The match can be done based on deterministic or probabilistic techniques so as to improve location accuracy. Many of these cases though do not consider the presence of any malicious user.

There has been some work recently on secure localization mainly in the context of sensor networks [32, 57, 58, 31, 55]. However, many of these techniques would not be appropriate for the purpose of secure localization in 802.11 networks. This is due to the need for special hardware (such as directional antennae or hardware with very tight time constraints) by these techniques and the fact that such special hardware not only increases the cost but also is not preferred with 802.11 networks.

Schemes for secure localization in 802.11 networks based on the signal strength measurement approach are provided in [45, 64]. In [45] the authors propose a secure localization scheme using the SS values. The SS lookup table is built efficiently but this scheme assumes an enterprise like environment with cooperating users. The paper indicated that using a simple trilateration based on averaged signal strength lookup table, an accuracy of 85 %

with a location error range of about 10 ft was obtained. Using the data from [45], we studied the impact of a simple attack that can be easily launched by a single attacker. We emulated the scenario of an intruder transmitting at higher power levels by increasing the received SS values by 25 % and using the regular matching techniques based on least mean squares error. We observed that the accuracy of the localization scheme dropped to 19 %. A way to address this is to use the difference approach as in [64] or to use more robust mapping techniques. In [64] a "difference method" was developed that could detect the location of an intruder transmitting at different power levels. However, the accuracy of estimated locations is poor with 70 % probability of correct location estimate with a resolution of about 10 ft. Robust mapping techniques are introduced in [35] which proposes statistical methods for secure localization in wireless sensor networks. Here the authors propose to determine the location based on a mapping which minimizes the median squared error which is more robust in the presence of malicious users. But the performance of the median scheme is worse than that of the mean scheme under normal conditions.

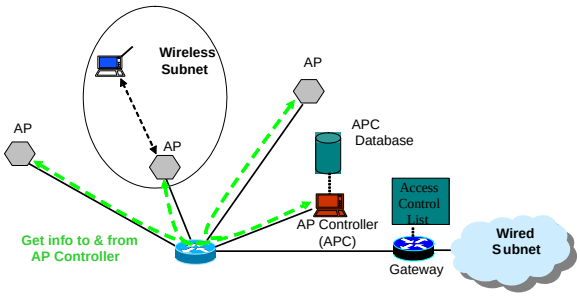
In general, previous research proposes using additional hardware for improving localization accuracy (e.g. Bluetooth interface [21]) or to provide secure localization (e.g. UWB interface [57]). In many cases especially those geared towards non-critical applications, the cost of this tradeoff might not be justifiable thereby making such systems impractical. What might be desired is a lower degree of reliability in the estimated location at a far lower cost. Attention has not been paid to this problem at all. We consider a *single attacker threat model*. This consists of an intruder (malicious user) acting alone, thereby ruling out collaborative attacks. We assume that the intention of the intruder is to convince

the location determination system that he is present in a region different than his actual physical location. In addition, our focus in this work is on locating static users.

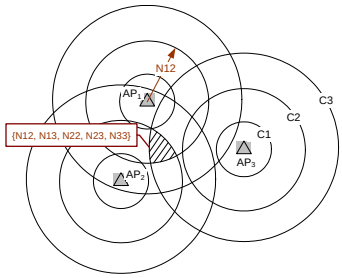
From the above arguments we can draw the following observations: (i) The SS approach is not robust when the intruders can vary their power levels. (ii) The SS values are both highly time varying and device specific. In order to accommodate for such time varying nature, localization schemes relax the localization criteria which makes it easier for intruders to bypass such localization schemes. Note that greater the time varying nature of the parameter, a more relaxed localization criteria will be used and the attacks would hence be easier. (iii) The attacker can easily collect localization critical information (in this case, the various SS values) from different locations within the deployment site. For example, the attacker who wants to falsify his location can park himself at the various access points and determine the SS values from devices at several known locations. As a result the malicious insider can build an SS map which he/she could use to his advantage. Thus, it is clear that the current SS based 802.11 localization schemes are not resilient even to simple attacks launched by a single attacker.

2.2 Secure Localization Scheme

We consider a WLAN system as shown in Figure 2.2(a) with several access points (APs) and a single AP controller (APC) that manages all the access points in the system. This fits the current trend towards controlling cheaper and lightweight access points via standard protocols from a central location. We consider that each access point has a capability to vary the power level. For example, six different transmission power levels 1mW, 5mW, 20mW, 30mW, 50mW, and 100mW are available on the Cisco AP1100 [12].



(a) Architecture of the proposed localization system



(b) Three access points with three transmission power levels each

Figure 2.2: The proposed scheme

We first explain the scheme with respect to an ideal environment[5]. Each power level is assumed to correspond to a different transmission range characterized as a circle of radius r. Thus r is different for different power levels. We assume that all the APs have identical capabilities in terms of the number and value of each of the available power levels. The system components such as the various APs and the APC are all trusted. The various APs are expected to communicate with each other via the APC over wireline links; hence such communication is considered secure from the intruder. In addition, each AP shares a symmetric key with the APC. The need for localization could be in response to detection of malicious traffic or at instances as decided upon by the APC.

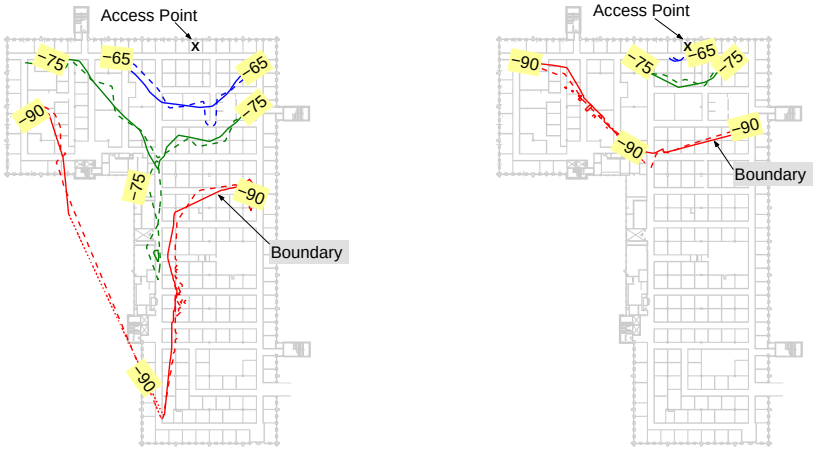
We next describe the secure localization algorithm[6]. Localization of a user is carried out by transmission of special messages at different power levels by neighboring APs. When a single AP is used, a user device (also referred to as a client) can only be located to within an annulus. To locate the user device more precisely, we would need more than a single

[5]Note that for any practical deployment there is a significant deviation from the ideal environment assumptions. This is later taken in account during actual deployment described later in this Section.

[6]Our description assumes two dimensional space and extends trivially to the case of three dimensional space.

AP. Consider three APs as shown in Figure 2.2(b). Let each AP have three transmission power levels. We consider the area of interest (AOI) to be the region where messages transmitted at maximum power level from all three AP can be received. The user device's location would hence be based on the set of messages from all the three APs received by it. Each message contains a nonce, the value of power level at which the nonce is transmitted and the identifier of the transmitting AP encrypted using the key shared between the AP transmitting the message and the APC. Thus, a client will not be able to decrypt this message.

Let Nij represent the message corresponding to the j^{th} power level from the i^{th} AP. For example, N12 represents the message corresponding to the second power level (circle C2) of the first AP (AP1). A user at a distance d from an AP would receive all the messages transmitted using power levels corresponding to transmission range greater than d from the AP. Thus, a user device present in the shaded region of Figure 2.2(b) will receive the set {N12, N13, N22, N23, N33} of messages from different APs. It can be easily verified from the figure that the set of messages heard at the shaded region also called a subregion (as also at the other subregions) is unique to the region. The user device would then respond by retransmitting the set of messages (received by it from all the three APs in this case) to the APC via the AP the user device is associated with. The APC decrypts all the messages received. Following this, the APC determines the location of the user device either using geometric properties (for the idealized case) or using a "message map" created beforehand which contains information about the set of messages that can be received at each location.



(a) Indoor Contour at 100mW

(b) Indoor Contour at 1mW

Figure 2.3: Indoor signal strength contours (in dBm) at different transmission power

2.2.1 Preliminary Results

In order to validate the practicality of the proposed scheme, some preliminary experiments were carried out. For these experiments we used Cisco AP1100 which has six different transmission power levels [12].

The first set of experiments was intended to characterize the behavior of the maximum transmission range in practical deployment when using different power levels. Specifically, we wanted to investigate if different power levels do indeed correspond to different maximum transmission ranges. These measurements were carried out indoors at a 150 x 120 ft. (18,000 sq. ft.) site by placing the AP in a room as shown in Figure 2.3. We used power levels of 1mW and 100mW. The measurements shown were carried out on two different days at different times. The measured readings from day 1 and day 2 are indicated by solid and dotted lines respectively in Figure 2.3. The measurements over different days are really

pertinent to answer the questions related to the time varying aspects of the maximum transmission range boundary. We address this later.

Given a transmission power level, the signal strengths (in dBm) at various locations (a total of 40 and 70 points for 1mW and 100mW respectively) were measured using a Toshiba PDA with internal WiFi card. Using these points the signal strengths at various locations were interpolated and contours of -65dBm, -75dBm and -90dBm are shown in Figure 2.3. The minimum signal strength recorded by the PDA was -90 dBm and is thus considered as the range boundary in these experiments. It can be observed that the signal reaches much larger distances in corridors as compared to offices due to the waveguide effect created by the corridors. Figure 2.3(a) and 2.3(b) represent the signal strength contour with the AP transmitting at 100mW and 1mW respectively. We observe from this figure that there is a significant difference in the range boundary at different power levels.

The next set of measurements seek to answer the question: What is the shape of the transmission range boundary in both the closed spaces such as an an office environment as well as in open spaces such as airport hotspots, shopping malls, etc? Figure 2.3 provides the answer for closed spaces which indicates that the shape of the range boundary is irregular when considering closed spaces. Further, it may also be observed from Figure 2.3 that the range boundary with 1mW transmission power is similar to the boundary with 100mW transmission power. Thus the adjacent range boundaries from an AP are similar in shape. We next focus on answering this question for open spaces.

In the absence of access to locations representing open spaces such as airport hotspots, large halls etc we conducted our measurements in an open field using a Cisco 1100 AP. The measurements of the range boundary were taken at a 1mW transmission power level. The

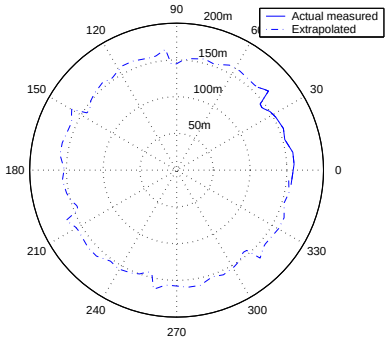


Figure 2.4: Outdoor Contour with 1mW

measurements were again carried out using the Toshiba PDA. Figure 2.4 shows the contour of the -90dBm signal at different angles around the AP. We conducted the measurements for only one quadrant and extrapolate for the rest. The actual measured range is plotted in solid lines in this figure. Using the measured values, the range for the other angles is extrapolated and is shown by dotted lines. We can see from the figure that the transmission range boundary for a 1mW power level can be approximated as a circle for an open space. We also observe from this figure that for realistic localization especially in open spaces, we will have to use power levels far smaller than 1mW and this might entail the use of attenuators with the current APs.

The third set of experiments seek to determine the variation of the maximum transmission range boundary as a function of time and the AP transmission power level. As explained earlier during the discussion on the first set of experiments, we conducted experiments within the building over several days. Results from two consecutive days are shown in Figure 2.3(a) and 2.3(b). We see from these figures that the range boundary (-90dBm) corresponding to different power levels remains fairly constant over different days as opposed to some of the SS measurements. Intuitively, the SS within the coverage region

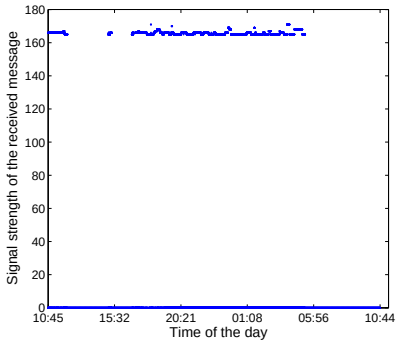


Figure 2.5: Message received at farthest distance during the day

may be affected due to local changes in environment, such as people gathering in a room, however these effects are averaged out at the boundary which is thus more stable than SS within the coverage region. This indicates that the proposed scheme based on transmission range boundaries would be more robust in terms of time variation, as compared to the signal strength based schemes.

It can be argued that the robustness with time of the measurements in Figure 2.3(a) and 2.3(b) might also have been influenced by the times we conducted our experiments. Our experiments were conducted at discrete times during the day. Hence in order to investigate the variation of the boundary over an entire period of 24 hours, we placed a client at a distance identified as the boundary for a 30mW transmission power level. We then let the AP transmit messages continuously to this client. Figure 2.5 shows the signal strength of the messages received throughout the day at this client. A signal strength of zero indicates that the nonce was transmitted by the AP but was not received at the client. It can be observed from the figure that at a distance just outside the range boundary, the client does receive messages during distinct periods of time. One such period in figure is from 10:45 to about 12:00. The fact that this reception is not random allows us to address this by placing

monitoring agents at some points outside the boundary. This is similar to the approach in [17] except that the authors there use the monitoring agents to monitor the SS values at different locations. The reports from these monitoring agents can be used to adjust the power levels of the access point. Thus we see that while the transmission range boundary does not change very much with time, we can also address the minor variations observed by making use of monitoring agents. Hence, we can assume that the maximum transmission range boundary corresponding to a transmission power level does not vary with time.

Summarizing based on the experiments, we can conclude the following:

1. The different transmission power levels will result in different maximum transmission ranges provided that the values of the power levels are sufficiently different. The current power level options on APs are indeed sufficient for this purpose.

2. The maximum transmission range is almost circular in open spaces and irregularly shaped in closed spaces.

3. The maximum transmission range for a given power level can be monitored to ensure that it does not vary with time.

4. We propose that each AP should transmit a message at each power level as N sub-messages. We consider a message to be successfully received if the client receives any k out of the N sub-messages. We call this the 'k out of N' scheme. By using this scheme, it is possible to achieve a sharper transmission range boundary that is also robust against packet losses.

5. The reception of messages at a given location is more robust than the SS measurement at that location.

2.2.2 Secure Localization Algorithm

Based on the scheme description and preliminary results, the Secure Localization Algorithm (SLA) is given as follows:

1. When the APC wants to determine the location of a user, it determines the AP, call it **X**, the user device is associated with. Further let the number of attempts (N_t) at localizing this user be initialized to 0.

2. The APC increments N_t, notifies the user and requests **X** as well as the neighboring APs of **X** to transmit messages securely at each of the power levels (using the 'k out of N' scheme).

3. The client switches to proper mode on receiving a notification from the APC. In this mode the user device receives a set **U** of messages from at least three APs. The client will then transmit all the received messages back securely to **X** which forwards **U** to the APC. The set **U** corresponds to the union of the messages received from each AP by the client.

4. The APC on receiving the set **U**, decrypts all the messages. It then determines the location L_p of the user from the information in the set **U** of messages received and using a "message map" that has been constructed previously.

5. Let N_c denote the maximum number of trials to locate a user.

   - If $N_t \leq N_c$ and number of successive values of L_p that are equal is less than δ then the APC goes to step 2.

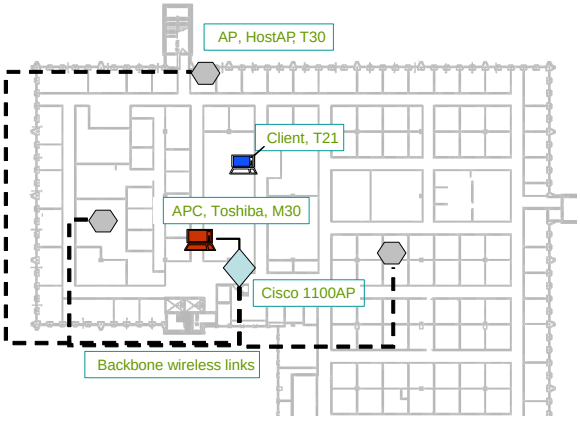
- else if $N_t \leq N_c$ and number of successive values of L_p that are equal is greater than δ then the user location is L_p
- else the APC infers that the user cannot be securely located and terminates the algorithm.

δ, k, N and N_c are parameters above that can be decided based on policies. It should be pointed out that δ represents the threshold number of times (out of N_c) that the same location L_p has to be estimated for successful localization. We would like to remark here that the "message map" will have to be obtained offline during the pre-deployment stage by manually measuring the set of messages received at various locations and may be maintained in a database (lookup_table). A more efficient method of obtaining this map in an enterprise environment is to make use of the plenty of desktop machines as proposed in [4]. This approach can also ensure that the "message map" is obtained in real time thereby taking care of any variations in time that exist. Note that an adversary cannot create this "message map" by walking around unless the adversary controls the APC.

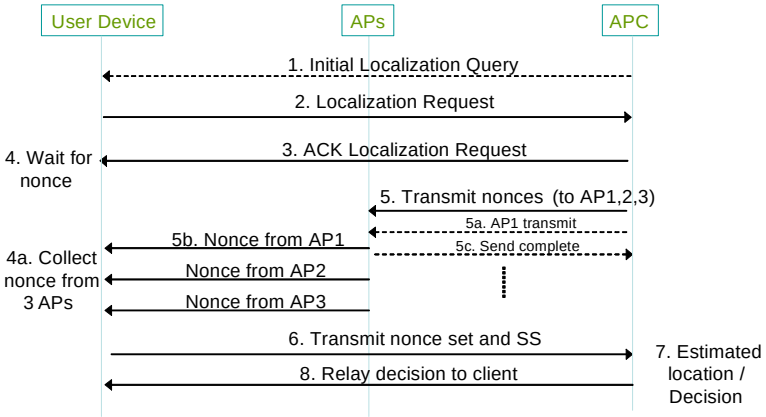
2.3 Testbed Implementation

The localization scheme described earlier was based on the system shown in Figure 2.2(a). We now discuss the detailed functionality of each entity.

The AP controller (APC) is a central entity that manages all the APs and clients of the network. The APC is assumed to have detailed information about the client and the APs. This may be obtained via repeated SNMP queries to the APs. The APC controls the gateway router in order to setup the access control list for access to network resources. The



(a) Testbed entities



(b) Call flow of SLA implementation

Figure 2.6: Testbed setup

APC is also assumed to have access to the "message map" defined in the lookup_table of the APC database.

The APC along with the "message map" was implemented on a Toshiba laptop running RedHat 9. In Figure 2.2(a) the APC is connected to the various APs in the network via the backbone wired network. For the current implementation, the APC and APs are connected via a backbone wireless link instead of wired links. This was done to provide the flexibility that we needed for placement of APs during the testing phase. These backbone wireless links allowed us to study the effect of AP placement on localization. Note that the backbone wireless link is provided by the Cisco 1100AP connected to the APC.

The APC can control several parameters such as:

- The number of APs to be used simultaneously. Currently at most 9 APs can be simultaneously used.
- The number of power levels to use on each AP. Currently 9 power levels are supported.
- The number (N) of sub-messages to be transmitted. This can be varied between 3 to 9.
- The power level at which each sub-message is to be transmitted. This is a value between 127 and 0 (described later). This value is unique for each AP.
- Other parameters such as k, N_c, δ which are decided based on the policies at the APC.

The AP can be any commercially available access point with the capability to transmit at various power levels. This feature of multiple power level transmission is currently inbuilt in various APs such as Cisco AP1100 [12], Dlink DWL-2100AP [13], etc. For the current

deployment though, each AP is implemented on an IBM T30 ThinkPad running RedHat 9 operating system equipped with two wireless interfaces (one external interface using a Linksys WPA 11 card and another internal inbuilt interface). A hostAP driver[7] is employed in order to convert this Linux (RedHat 9.0) laptop as an AP. The AP is expected to act as a bridge between the internal wired and wireless domains. The laptop's wireless and wired interface can be easily bridged using the inherent `brctl` command in Linux. However in our case, the external wireless interface was used for the backbone wireless connectivity to the APC. The internal wireless card was used in ad-hoc mode to transmit the messages, obtained from the APC via the external wireless interface, to the client. The ad-hoc mode was used instead of the AP mode without any compromise in functionality. The power level of the AP transmission was varied for each localization message. For this implementation the message just comprises of a nonce (random number).

The user device can be any device with a 802.11 interface. It may be associated with any one of the APs (as shown in Figure 2.2(a)). For the user device, the gateway is set as the APC controlled router. The user device should be able to receive messages from the non-associated APs and report these back to the APC via the associated AP. This is achieved by putting the client interface into monitor mode to receive messages from non-associated APs and to read the prism 2 headers to obtain the received SS value of the corresponding message. Thus for the current implementation, the client would require a Prism 2 based wireless card that can operate using hostAP drivers. In addition, in the current implementation the client is connected directly to the APC via the backbone wireless

[7] http://hostap.epitest.fi

network setup between the APC and the APs. This is okay since an AP acts as a bridge in the network and would be transparent to the clients.

The resulting testbed setup for deployment is shown in Figure 2.6(a). For the testbed, the area of deployment (Figure 2.6(a)) is about 150 x 120 ft. (18,000 sq. ft.). This figure shows one position of APs. Figure 2.6(b) shows the messages exchanged during the current implementation of the SLA. The control packet exchanges shown in the figure were implemented for the client, AP and APC using Python (www.python.org). The implementation is carried out in the form of a state-based implementation. These details are skipped here for lack of space. Using this testbed we have carried out extensive measurements to investigate the various characteristics of the proposed scheme. We report these findings next.

2.4 Performance Analysis

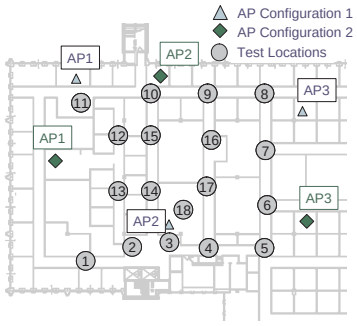
Using the testbed described in the previous section, the proposed scheme was deployed over the site as shown in Figure 2.7(a). We consider two configurations for the AP locations as indicated in the figure. For each of these AP configurations, a client was used to test the localization scheme at 18 different locations. This is indicated as test locations in the figure. These locations were chosen to lie approximately within the AOI defined by the three APs. The client was moved around using a cart but was kept stationary during the measurements at each of these locations. The messages are collected at each of these locations for the entire site in a single 'run'. In all 10 such 'runs' were carried out at different times during a single day for the test setup. The orientation of the client machine was randomly chosen during each measurement.

In this test setup, messages are transmitted at 3 different power levels from each AP. Each of the messages are transmitted as $N = 9$ sub-messages. Further, $k = 6$. Thus if at least 6 (k) of the 9 (N) sub-messages were received, the message is considered as received, else not. As pointed out in the previous section the power levels can be set at any value between 127 (min) and 1 (max). We arbitrarily choose the 3 power levels to be 127, 50, and 5 for this case.

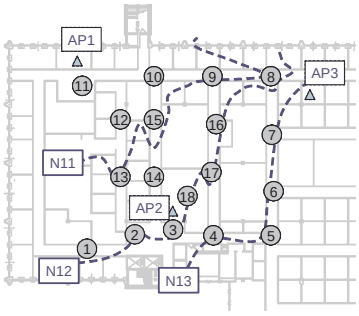
We intend to compare the performance of the proposed scheme to that of a client based SS localization scheme in the absence of any adversaries. Recall that for a client based SS localization scheme, the APs usually transmit at a fixed transmission power level. A client monitors the received SSs at a given location from these APs for localization. In our test setup, during each 'run', the received SS of the messages transmitted at the largest power level (close to actual maximum power level) is considered for measuring the performance of the SS based schemes. This is used to compare the performance of the message based scheme to the SS based scheme under identical test setup.

Building the lookup table

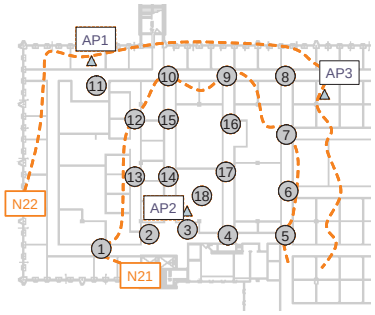
Prior to localization, the lookup_table is created for both the message and SS based schemes. For the message based scheme, the lookup_table defines the unique set of messages (in terms of N11, N12 etc.) that each location receives. In this test setup each AP transmits at 3 different power levels. In order to build the lookup table we consider multiple measurements of the received set of messages at each of the test locations. Using these measured data and the results from preliminary measurements (waveguide effect of corridors) the coverage boundary was approximately estimated. The estimated coverage boundary of



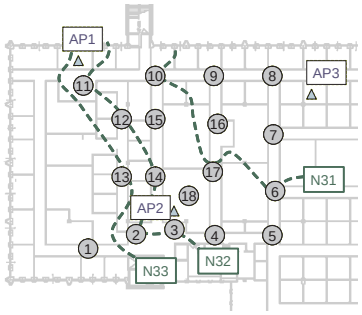
(a) Test locations and AP configuration



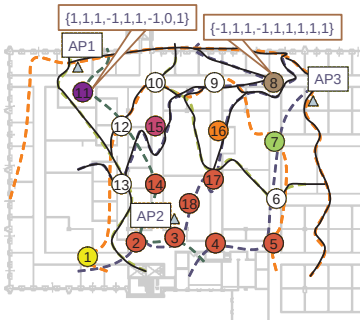
(b) Transmission boundary region of 3 power levels from AP1 (N11,N12,N13)



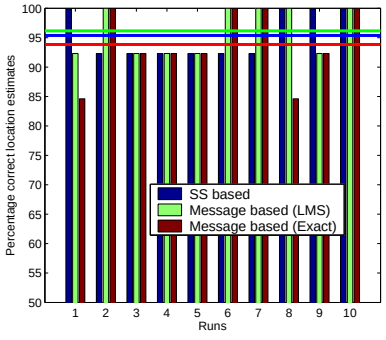
(c) Transmission boundary region of 3 power levels from AP2 (N21,N22,N23)



(d) Transmission boundary region of 3 power levels from AP3 (N31,N32,N33)



(e) Sub-regions formed within the site based on the set of messages received



(f) Percentage correct estimates for the test setup

Figure 2.7: Test setup 1

each transmission power level for different APs is shown in Figure 2.7(b), 2.7(c), and 2.7(d). Figure 2.7(c) does not show the boundary of N23.

Using these boundary contours, we can estimate the set of messages that will be received at each location. For example, location 11 is interior to messages N11, N12, N13, N22, N23, N33, exterior to messages N21, N31 and boundary location for N32. The lookup_table entry for location 11 is thus given by the vector $S_{11} =< 1, 1, 1, -1, 1, 1, -1, 0, 1 >$. Here -1,0,1 represent that the corresponding message from $M =$\{N11, N12, N13, N21, N22, N23, N31, N32, N33\} is not received, may be received, or is received respectively. The 0 in this case indicates a 'don't care' implying that the corresponding message would be ignored. The lookup_table would consist of such vectors for each location considered in the test setup.

A vector S corresponding to each location may be generated based on the boundary of each message. The 0 in S for different locations may make them indistinguishable for message based localization. Such locations are grouped together to form a sub-region. For example, consider a client at any one of the locations corresponding to the vectors $S1 =< 1, 1, 1, -1, 1, 1, -1, 0, 1 >$ and $S2 =< 1, 1, 1, -1, 1, 1, 0, 1, 1 >$. Based on the messages received, the client may be localized to either one of the locations. However if a sub-region made by combining these locations is considered, then the client would be localized to the correct sub-region. The lookup_table would also map these individual locations into various sub-regions.

Figure 2.7(e) shows the 7 sub-regions that can be identified based on the set of messages received at each location. 6 of these sub-regions contain just a single location while locations 2, 3, 4, 5, 14, 17, 18 form a single sub-region. Some of the locations (6, 9, 10, 12, 13) lie on the boundary of these sub-regions and are ignored for this test setup (indicated with white

background in the figure). The lookup_table for the test setup is thereby created. Note that the client is estimated to various sub-regions and not to individual locations.

For the SS based scheme, the vector S would comprise of the average received SS of N13, N23, and N33 at a location. The lookup_table for the SS based scheme would be the vector S corresponding to various locations. In this case too, the localization is carried out with respect to the sub-regions defined earlier and ignoring the locations on the sub-region boundaries.

Location Estimation

In both the schemes (message based and SS based) the location may be estimated based on the least mean square (LMS) estimate. Let the current measurement vector be S_m, while the vector for a lookup_table location be S_i, where i varies from 1 to number of locations L_N in the lookup_table. The estimated location is the j^{th} entry in the lookup_table such that S_j corresponds to the LMS estimate given by

$$S_j = min_i \sum((S_i - S_m)^2) \tag{2.1}$$

For the message based scheme, the vectors S_i would represent the messages received, not received or ignored ($< 1, -1, 0, ... >$), while for the SS based scheme they would represent the average SS of N13, N23, and N33 at location i. The S_m are the measured readings at an unknown location obtained during the 'runs' described earlier. For the measurement at a location, the client may or may not receive a particular message. Hence, the vector S_m cannot contain a 0. However, while comparing S_m to a S_i, entries of S_m, corresponding to

0 entries in S_i, are forced to be 0. This ensures that the corresponding entries from S_i are ignored from the measured reading.

For the message based scheme, the LMS estimation would be able to localize the client to the "closest" sub-region (in mean square sense). The LMS estimation has greater tolerance for error and thus may allow the attackers to spoof their locations by sending incorrect but "close" set of messages. Such an estimation would increase the performance but would compromise the security of localization. In order for the message based scheme to be secure, an exact match of S_m with the S_i would have to be considered. In this case too, entries of S_m, corresponding to 0 entries in S_i, are forced to be 0. Hence, we consider the exact matching scheme as representative of the security capabilities of the proposed scheme. Note that if no exact match is possible with any of the locations, then the localization cannot be carried out.

Accuracy of localization

We now estimate the location of the client for the measured data collected during different 'runs'. We consider LMS estimation for the SS based scheme and both LMS estimation and exact matching for the message based scheme.

The percentage correct estimation for different runs is shown in Figure 2.7(f). The horizontal line in the figure represents the average percentage correct estimate for each scheme over all the runs. As seen from this figure, the LMS estimate for both the schemes are almost equal while the exact matching for message based scheme is slightly worse. This indicates that in terms of accuracy the message based scheme is comparable to the SS based scheme. Note that this experiment corresponds to N_c value of 1.

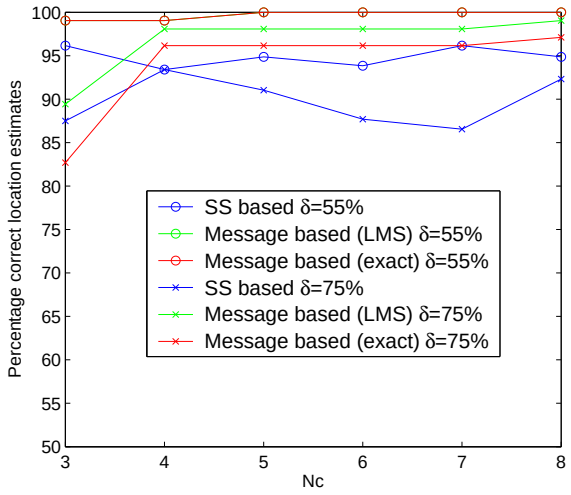


Figure 2.8: Percentage correct location for SS, message (LMS) and message (exact) for different values of N_c and δ

We also study the effect of change in N_c and δ for this test setup. The percentage correct location estimates are shown in Figure 2.8. In the figure, the plot for message based LMS and exact matching schemes coincide for $\delta = 55\%$. As the value of N_c increases, more 'runs' are used to estimate the result. The value of δ is represented as a percentage of N_c for this test setup. The increase in δ implies that more location estimates from the N_c estimates have to be identical in order to obtain an effective location estimate. If the maximum number of identical location estimates (from the N_c estimates) is less than δ, then the effective location cannot be determined and is considered as an error. The figure indicates that as δ increases the accuracy of the estimates drops. However, it should be noted that a higher value of δ and N_c would imply more confidence in results. Note that the security of the scheme is strengthened for higher values of N_c and δ. As pointed out earlier, the parameter values of δ and N_c would have to be decided based on the trade-off between security and performance.

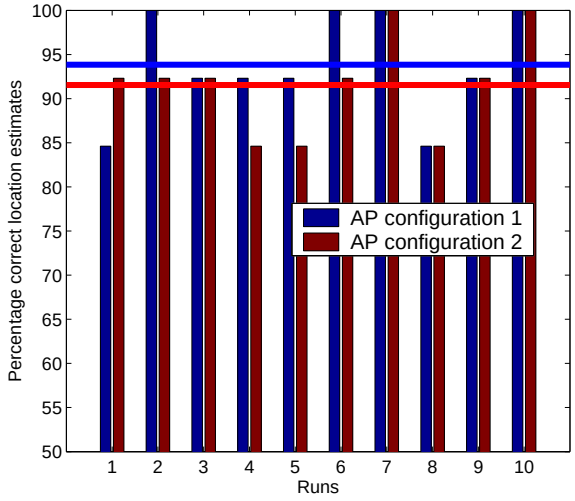


Figure 2.9: Percentage correct estimates for the test setup with AP configuration 1 and 2 for the exact message based scheme

Effect of change in AP configuration

The placement of the APs is important for any localization scheme. The optimal deployment strategy with respect to any localization scheme would be to ensure best performance with minimum number of APs. We study the performance of the scheme for AP configuration 2 of Figure 2.7(a). The entire localization procedure described above was repeated for this new AP configuration considering the same test positions. In this case too, the lookup_table is built the same way as previously described. However, only 4 unique subregion s could be formed for this AP configuration. The area of the sub-regions for this configuration is much larger than compared to the previous case. Thus the AP configuration affects the size of the sub-regions, i.e. the resolution of the localization scheme.

Figure 2.9 shows the percentage accurate estimates when exact matching is used for both the configurations. It can be seen from this figure that the current AP configuration performs slightly worse than the previous configuration. The AP configuration therefore

Table 2.1: Time delay for various localization operations

Operation	Time Delay (msec)		
	Min	Max	Avg
Transmission of one sub-message	3.9	15.5	8.1
Get in promiscuous mode	92	103	99
Get out of promiscuous mode	90	104	93

does affect the accuracy of the scheme. These arguments demonstrate the need to place the APs at optimal locations for the localization scheme. The optimal placement of AP is part of our future work.

Miscellaneous Characteristics

We also studied two other characteristics of SLA. The first is related to the time required for localization while the second is related to the decrease of system throughput due to the use of SLA. In order to estimate the latency of localization we obtain the time required for various aspects of the scheme. The various times averaged over numerous iterations are shown in Table 2.1.

The time required to transmit a sub-message from an AP is approximately 8 msecs. Thus if a scheme with 3 messages with $N = 9$ is considered then we have $3 \times 9 = 27$ sub-message transmission from each AP. As a result the time taken for a single localization iteration at each AP is approximately 216 msecs (648 msec for 3 APs). From the table if we consider the time taken to enter in and out of the promiscuous mode as 100 msec, the localization process on an average can be completed in about 1.5 secs (accounting for the transmission delays). Note that further optimizations are possible such as by transmitting messages that do not overlap simultaneously. This could be in addition to avoiding operation in the promiscuous mode.

We next consider the decrease in throughput at the various APs due to the localization scheme. Such a decrease is caused since an AP has to transmit localization related control packets. This adds on to the overhead of localization for the network. An 'Iperf'[8] server was setup at one of the APs in the testbed. Prior to any localization process, the throughput between this AP and a wireless laptop was measurement using Iperf for 10 secs and was found to be 4.8 Mbps. The second client which is to be localized is then introduced in the system. Due to the control packets transmitted under SLA, the throughput between the AP and first client was found to drop to 4.7Mbps. This is not a significant drop. Thus, SLA does not introduce heavy load on the network. Additionally, SLA can also be used to localize multiple users simultaneously thereby spreading the overheads among multiple users. We would also like to remark here that the interference in the system can be reduced by ensuring that, of the multiple APs that cover an area, only one is used for data transfer while the other APs are only used for localization. This might be justified given the very low costs (we can get an AP for $10 these days) associated with APs.

2.5 Security Analysis

We next investigate the possibility of an attacker spoofing his location when the proposed localization scheme is used. Note that for the purpose of security analysis we have assumed circular transmission regions for the APs.

We have pointed out that with the SS based scheme, an attacker can build a SS map. In order to do the same in the message based scheme, the attacker would have to get the APC to send a localization query. Frequent queries to the APC may be flagged. Further,

[8] http://dast.nlanr.net/Projects/Iperf/

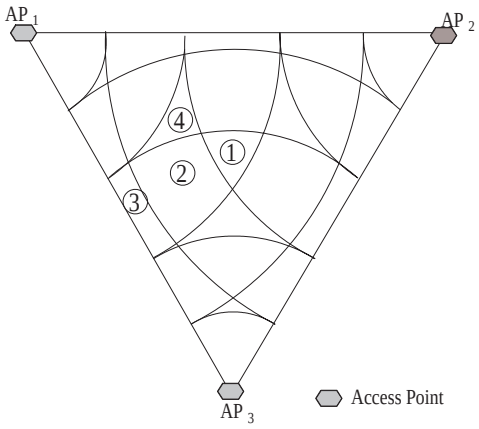


Figure 2.10: The concept of AOI and sub-regions.

the APC chooses the set of power levels and corresponding unique set of messages for each AP for each localization query. Thus for each query the attacker may read different power levels and hence will not be able to build a complete lookup table. This can be further complicated by having the APs spoof a common MAC address for the messages that they transmit.

We have also observed from measurements that the messages from different APs transmitted at different power levels may be received with the same SS at the client. Thus the origin of the received messages cannot be obtained by studying their SS. Hence it is difficult for the attacker to identify the source AP of the message. This makes it more difficult to build the lookup_table for the messages. Further, even if the attacker does obtain such a lookup_table he would have to report the correct set of messages. We later show that the probability of obtaining a correct set of messages for location spoofing is low (simulation and analytical bound) especially given the fact that all these messages are encrypted.

2.5.1 Simple attack with no additional hardware

Consider an attacker with no special hardware such as high gain antenna. One way by which an attacker can spoof his location is by skipping some of the received messages. For such a case, the proposed scheme is secure in that it prevents an attacker acting singly from spoofing his location provided the number of power levels is large enough. This is because by skipping some of the messages received a user device can only increase its distance from the AP. And given that the user device is covered by at least three APs, it cannot increase its distance towards all the APs simultaneously. However in reality, due to the limited power levels available at each AP, a single attacker may be able to spoof his location. For example, consider the AOI as shown in Figure 2.10 where 5 power transmission power levels are used from each AP. Here a user device present in sub-region 1 hears the set of messages {N13, N14, N15, N23, N24, N25, N33, N34, N35}. Such a user device can drop message $N23$ from the set of messages it hears and thereby appear to be in sub-region 2 which corresponds to the message set {N13, N14, N15, N24, N25, N33, N34, N35}. If the user device is present in sub-region 2 though, it cannot pretend to be in sub-region 1[9] although it can pretend to be in sub-region 3 by skipping message N24. On the other hand a user device present in sub-region 4 can hear a set of messages {N13, N14, N15, N24, N25, N34, N35} and such a user device cannot spoof its location at all. Similarly, a user device in sub-region 3 also cannot spoof its location. When the number of power levels increases, such non-spoofable locations dominate.

It should be noted that even when limited number of power levels are utilized, not all subsets of the received messages are valid. Hence the attacker may risk detection by

[9]This is not true if the intruder while being in sub-region 2, can also obtain N23 by using special hardware.

reporting a subset of received messages. Consider an intruder lying in the sub-region (say β sub-region) where the largest number of messages (β) can be received. An attacker located in β sub-region will have access to the largest set of messages and hence can be considered to be at an advantage as compared to the other sub-regions in the AOI. As a result we obtain an upper bound on the maximum probability of spoofing which is given by,

P_{max} = (Prob of being in the β sub-region) $\times$ (Prob of guessing the correct message set for a non-β sub-region from the β received messages)

$$P_{max} = \frac{1}{N_r} \times \frac{N_r - 1}{N_n} \tag{2.2}$$

where N_n is the number of message sets that can be generated using $M_1,\ M_2,\ M_3$ messages that are received by the attacker from the three APs respectively. Since we assume that the intruder is in the β sub-region, $M_1 + M_2 + M_3 = \beta$. The number of valid message sets is equal to the number of sub-regions N_r in the AOI. This is true since every sub-region has a unique set of messages associated with it. Thus the intruder would be successful in spoofing his location if he can guess any one of the $N_r - 1$ message sets. Note that we are considering the case where the attacker wants to prove to be in any sub-region other than the actual. If the attacker is interested in proving to be in only a certain sub-region then our analysis extends trivially.

Figure 2.11 gives the probability that the attacker may be able to guess correctly the set of sub-messages from the collected β messages. In the worse case, represented as *Case 1* in the figure, the attacker is assumed to be able to determine the source AP and the order of the transmitted messages. Thus messages can be dropped intelligently thereby reducing N_n and increasing P_{max}. The analytical and simulated plots are shown for this case. However,

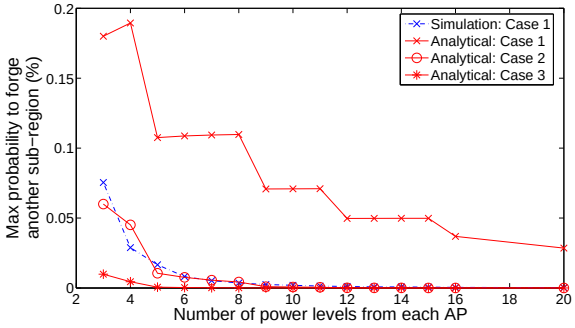


Figure 2.11: Maximum probability of spoofing

if the attacker can only obtain the source AP by reading the MAC, but cannot monitor the power level in order to determine the message order, his probability of spoofing decreases as shown by *Case 2* in the figure. Further, if the APs use MAC spoofing while transmitting the messages the source AP would be hidden and hence the subset of messages N_n increases thereby decreasing P_{max} even further. This is represented as *Case 3* in the figure. Note that as remarked earlier, *Case 3* is a viable implementation option.

2.5.2 Attacks with high gain antenna

Attackers using high gain antenna may be able to obtain more than β messages. To prevent against such attackers from obtaining all the messages transmitted from all APs, a simple yet effective modification to the proposed scheme can be carried out. Messages corresponding to non-overlapping regions of different APs (at different power levels) can be transmitted simultaneously. Although these messages would appear to collide for the attacker due to his high gain antenna, honest clients will be able to properly receive messages transmitted in their corresponding regions. The attacker may also use high gain directional antenna to obtain messages from a single AP. In this case. we assume that the attacker will

be able to successfully orient the antenna to obtain the lower power level messages from an AP. This though is not trivial due to the multipath effects.

In this case too, it may be difficult for the attacker to distinguish amongst messages corresponding to different power levels and APs. This also holds true for directional antenna, since directional antennas have back lobes and side lobes that enable reception from directions other than that pointed by the main lobe. Thus for these attacks, we consider a scenario similar to *Case 3*. Let us assume that the attacker is able to collect more than β messages. Thus, the sample space observed by the attacker will now have more messages from which he has to choose a correct message set of size less than or equal to β. Due to this increase in sample size the probability of spoofing will be lower than that obtained in the previous subsection.

It should be pointed out that this result is not valid for *Case 1* and *Case 2*. For these cases, if the attacker collects more messages then he also gains the corresponding information related to the source AP or the order of messages. Thus, he will be able to intelligently form the set of messages which will increase his probability of spoofing. Clearly in *Case 3* no such additional information is revealed to the attacker with the increase in sample space. This decreases the probability of spoofing.

2.5.3 Multiple attacker devices

The attacks using multiple devices have not been considered in this work. However we give some preliminary directions to extend the scheme in order to effectively deal with multiple attacker devices. Consider a wormhole wherein the attacker (A) from one subregion would transmit the messages that it has received to a different attacker device (B)

in another sub-region. The intention of (B) could be to spoof his location as C by using all the messages obtained by A and B. Additionally, by placing different attacker devices near each AP the attacker may be able to identify the source AP and transmission power level of all the messages. Thus, the scenario described in *Case 3* cannot be assumed.

However, to defeat such collusion, the scheme can be modified to transmit low power messages from a few client devices or by using covert basestations [58] in the network. The attacker claiming to be in a certain location will have to retransmit this client message along with the received AP messages. The only way for an attacker to obtain the messages transmitted by a client would be by placing an attacker device at the spoofed location. This is impractical and will also defeat the intent of location spoofing. The issues related to this modification such as client delay, secure message transmission and client power level variation are part of our current study and will be reported in near future. Another approach to defeat collusion based attacks is to combine the SS measurements with the messages received by using the signal strength values as a cross-check.

Chapter 3

Bridging Heterogeneous Networks

Heterogeneous networks are realized in many real-world scenarios due to application requirements, difference in standards, or need for collaboration in special situations. Direct communication between heterogeneous nodes (nodes with different physical interfaces) working on different physical layer technologies is not possible. A practical example of such a heterogeneous network can be observed in the area of public safety. For example, the police radio network and the fire department's radio network are both homogeneous wireless networks that may be independent of each other but communicate with some backbone infrastructure. During a disaster when the backbone infrastructure is destroyed, they become isolated networks that can be combined together to form a single heterogeneous network. Direct communication between dissimilar nodes in the resulting heterogeneous network is not possible. This problem of connectivity in a heterogeneous network is not limited to public safety: wireless personal area networks, ad hoc networks, mesh networks, and sensor networks are other areas where this problem may be encountered.

The motivation and related work follows in the next section. Section 3.2 describes the proposed PISA algorithm. The results are further discussed in Section 3.3.

3.1 Motivation

We consider the problem of connectivity when several homogeneous networks in a region combine to form a single heterogeneous network. A brute-force and expensive solution would be to equip each node with all additional network interfaces. Previous solutions

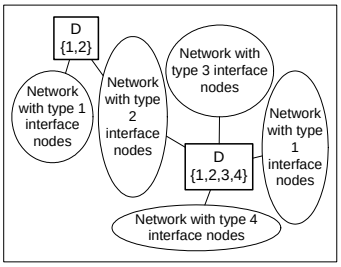


Figure 3.1: Schematic representation of drone placement

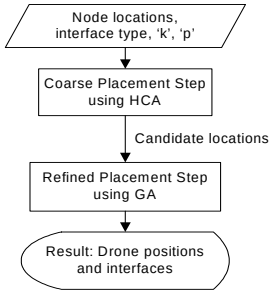


Figure 3.2: PISA flowchart.

for the public safety communication problem use a single high-end base station with all interfaces to relay communication between heterogeneous nodes[1]. However, these solutions are impractical as they do not scale well with the number of interfaces and the number of underlying homogeneous networks. Moreover, in ad hoc, sensor, and mesh networks there is an additional degree of freedom: the individual nodes may reach a 'base station' via multiple hops using other nodes. This work addresses the communication problem in such a heterogeneous ad hoc network.

We propose a scalable low-cost solution for enabling communication in heterogeneous networks by deploying a small number of multiple interface devices called *drones*. Development of such multi-interface devices is currently underway[2]. A heterogenous network comprising of several groups of homogeneous networks is depicted in Figure 3.1. The heterogeneous network may be formed of several groups of 802.11a, 802.11b, ultra-wide band (UWB) and WiMax nodes. It should be noted that all types of nodes may be present throughout the deployment region and not just the areas indicated by the figure. These areas only represent their high density regions. These nodes can communicate within their

[1]http://www.arinc.com/news/2005/09-22b-05.html
[2]http://www.advancenanotech.com/060228_wireless_research.html

own groups but cannot communicate with nodes of a different type (different groups). As shown in the figure, the drones act as bridges between different homogeneous networks. Thus the communication between two different types of nodes would be routed via single or multiple drones.

The set of problems to be addressed are: How many interfaces should the drones have? How to distribute the drones? Where to place them to avoid network partitions? These issues are addressed by a Placement and Interface Selection Algorithm (PISA). In essence, PISA solves the placement problem of drones in two phases, namely: coarse placement followed by a refined placement step. In the first phase, candidate drone locations are identified using heterogeneous clustering. These locations are then used to select drone locations and their interfaces in the second phase. Although we assume an ad hoc network, the proposed solution can be extended trivially to infrastructure based networks, where nodes are one hop away from base stations.

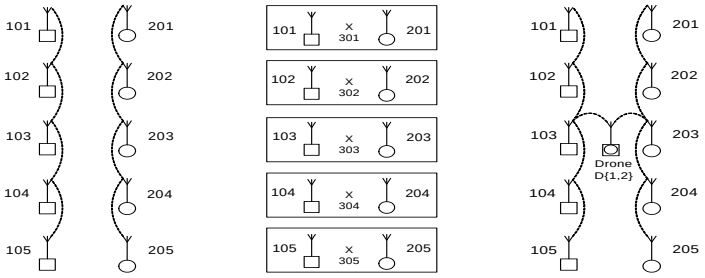
3.1.1 Related Work

For a homogeneous infrastructure-based network, our problem of drone placement is similar to the problem of base station placement in mobile cellular networks. In both these problems, a minimum number of high end devices (drones or base stations respectively) are to be placed to maintain connectivity and coverage for the entire network.

There are many proposed solutions to obtain optimal base station placement for cellular network coverage. One such solution using a combinatorial algorithm was proposed in [41]. The initial base station locations are split into smaller groups and all base station combinations within the group are tested to obtain the best coverage. The base stations

resulting in best solutions from different groups are then merged to form new groups and the process is repeated. However, extensive computations are required as the number of base stations increases. Evolutionary algorithms, which are adaptive heuristic search algorithms based on evolutionary ideas of natural selection and genetics, have also been used. In [53], the authors propose the use of evolutionary algorithms to generate a set of cell sites that satisfy multiple objectives (such as high coverage, low multi-coverage etc) of a coverage problem. In [36], the authors use a genetic algorithm, which is a type of evolutionary algorithm for obtaining the base station placements. An appropriate genetic representation was proposed in order to increase efficiency of the genetic algorithm. A similar problem is also described for sensor networks. In [40] the authors propose techniques based on Voronoi diagrams for placement strategy of additional sensor nodes in a pre-deployed sensor network in order to improve the worst and best-case sensor coverage.

All of the above solutions assume a homogeneous network. Most of the previous work on heterogeneous ad hoc network topology control, such as [33], consider heterogeneous nodes as nodes having different transmission powers but being able to communicate with each other. However, we consider heterogeneous nodes as dissimilar nodes that cannot communicate with each other. Note that for simplicity we assume equal transmission power for all types of nodes and drone interfaces, but this solution is applicable otherwise too.



(a) Original configuration

(b) Coarse placement. Candidate locations marked as 'X'

(c) Refined placement. Drone with interface (1,2)

Figure 3.3: Simple example to demonstrate PISA.

3.2 Placement and Interface Selection Algorithm

We assume that a fixed number of drones are to be deployed throughout the network. Our proposed algorithm, PISA, not only identifies the placements for drones but also determines the interfaces on each one of them. PISA is carried out in two steps as shown in Figure 3.2.

1. Coarse Placement: Using a heterogeneous clustering algorithm, we find regions of nodal heterogeneity within the deployment area. k clusters are formed in this step. The cluster centers are chosen as candidate locations where the drones may be placed.

2. Refined Placement: Using a genetic algorithm, we find the subset of locations from the set of candidate locations for drone placement. We assume p drones are to be placed to improve network connectivity while minimizing network partitions and number of interfaces on each drone.

We present a simple example in order to outline our proposed solution. Nodes 101-105 and 201-205 in Figure 3.3(a), are two groups of nodes with different interfaces. The coarse placement step identifies the heterogeneous clusters and their respective centroids (candidate locations), indicated as 301-305, as shown in Figure 3.3(b). We then use a genetic algorithm to find a refined location that is shown in 3.3(c). It is easy to calculate that the average number of hops between two nodes is minimum when the drone is placed at 303, i.e. the solution obtained by PISA. We explain each of the two steps in detail in the following sections.

3.2.1 Coarse Placement

The first step in the algorithm is to determine the coarse positions (candidate locations) for drone placement. Intuitively, a candidate location should be such that a drone placed at that location can interconnect maximum number of heterogeneous nodes. Thus the problem of determining candidate locations can be translated to finding regions or clusters of heterogeneous nodes in the network.

The regular clustering algorithms such as k-means or hierarchical clustering consider the distance between the nodes for formation of clusters [61]. Along with the distance measure, we incorporate an additional heterogeneity criteria to be considered for cluster formation. This can be formulated as a multi-objective clustering problem in which clusters are formed based on multiple criteria (node distance and heterogeneity in our case). Multi-objective clustering techniques using evolutionary algorithms are described in [22] and [66].

However, in our case, due to the simplicity of the clustering criteria and the relaxed accuracy requirement for this step, we modify the basic k-means algorithm for obtaining

heterogenous clusters. We choose k-means clustering as the underlying clustering technique since it has linear time complexity while hierarchical clustering has quadratic time complexity [61]. The heterogeneous clustering algorithm (HCA) is described next.

Heterogeneous Clustering Algorithm (HCA)

We describe the basic k-means clustering algorithm which forms k clusters. Initially k clusters are formed by randomly classifying all the nodes into one of the k clusters. The algorithm then iteratively reshuffles each node based on its Euclidian distances $(d_1, d_2, ...d_k)$ from all the centroids of k clusters. The node is assigned in the cluster which is "closest" (minimum distance) to it. The algorithm converges if all the cluster elements remain unchanged in successive iterations. The modification of "distance" parameter (or clustering objective function) to incorporate heterogeneity criteria in basic k-means algorithm is explained next.

Consider a network which consists of N_t types of heterogeneous nodes. We will define a measure of heterogeneity as follows. We represent parameter n_i as the heterogeneity due to a node of type t_j for the i^{th} cluster. It is calculated as:

$$n_i = N_i(t_j) - \frac{N_i(t - t_j)}{N_t - 1}, \tag{3.1}$$

where, $N_i(t_j)$ represents the number of nodes of type t_j (same type as the node under consideration) and $N_i(t - t_j)$ represents the number of all other types of nodes (except type t_j) in the i^{th} cluster. Clearly, n_i is positive if cluster i contains more type t_j nodes than

an equal sized perfectly heterogeneous cluster[3] and vice versa. The modified distance of a node to centroid of cluster 'i' is represented as d'_i and calculated as:

$$d'_i = d_i(1 + \alpha n'_i) \quad i = 1, 2, ..., k \ , \tag{3.2}$$

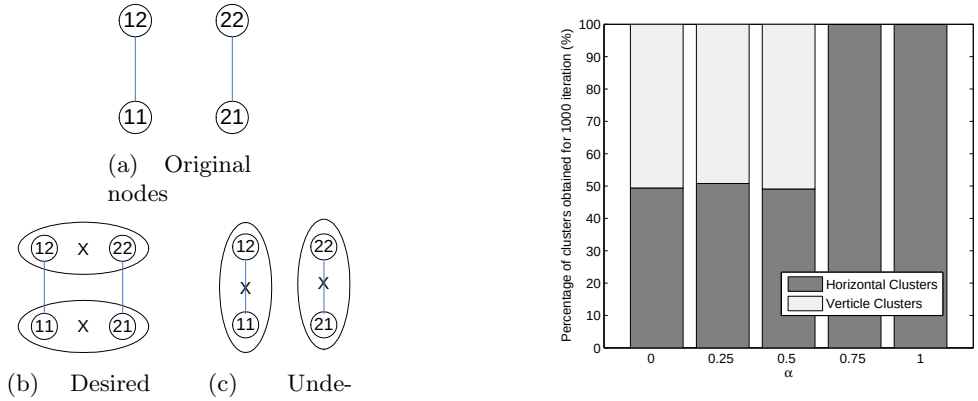
where, d_i is the Euclidian distance, n'_i is the normalized value of n_i and α is the heterogeneity weighing factor. $n'_i \in (0, 1)$ is normalized considering all n_i, $i = 1, 2, ..., k$ clusters. Normalizing n_i to n'_i is required in order to compensate for different cluster sizes in the network. Note that when $\alpha = 0$, the heterogeneity parameter is neglected while clustering, and $\alpha = 1$ gives heterogeneity parameter equal weight as that of distance d_i.

To explain d'_i intuitively, consider a node of type t_j to be added in a cluster 'i'. If the cluster has more type t_j nodes than a perfectly heterogeneous cluster, i.e. $N_i(t_j) > \frac{N_i}{N_t}$, then $n_i > 0$. Similarly, $n_i < 0$ if the cluster has less t_j type nodes. Thus, the corresponding normalized n'_i and hence the resultant d'_i would be greater in the former case than the latter. Note that although $d'_i > d_i$ for all cases, the relative increase in d'_i depends on n'_i. Thus, if the cluster 'i' has more type t_j nodes then d'_i increases and thus decreasing the chances of inclusion of the node in the cluster.

Test Example

In order to validate the heterogeneous clustering algorithm described above, we consider a simple example with 4 nodes of two types placed at the vertex of a square as shown in Figure 3.4(a). When 2 clusters are formed using the original k-means algorithm, the resultant horizontal and vertical clusters are shown in Figure 3.4(b) and 3.4(c) respectively.

[3]All N_j types of nodes are in equal proportion in a perfectly heterogeneous cluster.



(a) Original nodes

(b) Desired horizontal clusters

(c) Undesired vertical clusters

(d) Effect of α on formation of heterogeneous clusters.

Figure 3.4: Heterogeneous clustering example.

Due to the equal distance between horizontal and vertical nodes, both these cases occur with equal probability. However, if heterogeneous clustering is desired, the horizontal clusters comprising of heterogeneous nodes are preferred over the vertical clusters which comprise of homogenous nodes. Figure 3.4(d) shows the percentage of horizontal and vertical clusters formed when clustering was carried out 1000 times using HCA with different values of α. As seen from this figure, when heterogeneity and distance are equally weighted ($\alpha = 1$), the desired heterogenous clusters are obtained for all iterations. This clearly demonstrates that HCA successfully incorporates heterogeneity. We fix the value of α at 1 henceforth.

3.2.2 Refined Placement

From the k candidate locations, obtained from the previous step, we need to select p optimal locations for placing p drones. Each drone may have a different set of multiple interfaces from the total N_t types of interfaces in the network. We use a genetic algorithm in this case for determining both the drone locations and the set of interfaces for each drone.

Genetic Algorithm (GA)

Many genetic algorithm implementations are proposed in the literature for the base station placement problem [36, 53]. We consider the basic genetic algorithm (GA) in this work [18]. In GA, many candidate solutions (population) are considered at a time with each solution (individual of the population) represented as a string of binary sequences (chromosomes). The fitness of each individual is calculated using a fitness function. The individuals that are better (higher fitness value) in the previous iteration (generation) are chosen as parents to generate new individuals in the current iteration. This is done by either swapping random parts of the parent chromosomes (cross over) or by changing random bits in a chromosome (mutation). The resultant children replace the parents only if they have a higher fitness value than the parents. The algorithm terminates after a prespecified number of generations.

We now explain the representation of individual chromosomes for our GA implementation. Our representation is based on the chromosome representation described in approach 3 of [36] since it was found to minimize the execution time and increase fitness of the resultant solution. The drone location and its interfaces are represented as binary strings in an individual chromosome as follows. The drone location is selected from the k candidate locations which can be represented by $l = \lceil log_2(k) \rceil$ bits. Since $k \leq 2^l$, we linearly scale the decimal value (say V_l) represented by 'l' binary bits to map to actual candidate location index (say V_k). Note that $V_l \in (0, 2^l)$ while $V_k \in (1, k)$. Thus the drone will be placed at the ${V_k}^{th}$ candidate location from the k candidate locations. Thus,

$$V_k = INT(V_l \times \frac{k-1}{2^l - 1} + 1), \tag{3.3}$$

Drone	1		2	
	l	N_t	l	N_t
I	11101	110	01110	101
V_l	29		14	
V_k	24		12	
Interpretation	drone to be placed at 24^{th} candidate location with interfaces {1,2}		drone to be placed at 12^{th} candidate location with interfaces {1,3}	

Table 3.1: Individual chromosome representation

where, $INT(x)$ represents the nearest integer to x.

We use an additional N_t bits to represent the multiple interfaces for each drone. Each of these bits correspond to a single type of interface present in the heterogeneous network. The subset of N_t bits that have value '1' represent the interfaces present on the corresponding drone. Thus a single drone can be completely represented by $l + N_t$ bits. Since we consider placing p drones, a possible solution (an individual chromosome) is comprised of p blocks of $l + N_t$ bits, i.e. $p(l + N_t)$ bits in all. For example, consider a heterogeneous network with N_t=3 and 26 candidate locations from the coarse placement step (k=26); thus $l = 5$. If 2 drones are to be placed for this network (p=2), the length of the individual chromosome (I) will be 16. Table 3.1 shows the representation of an example individual chromosome $I = 1110111001110101$.

Next, we explain the fitness function used for the GA. When a multi-interface drone is added to a heterogeneous network, it increases the communication linkages in its neighborhood by interconnecting nodes of different types. Thus, the p drones in a particular solution (individual) are represented with linkages amongst different heterogeneous nodes in their respective coverage regions. The network can be represented as a graph with

edges representing the communication links between nodes. These edges may be due to direct communication between homogeneous nodes or communication between heterogeneous nodes via a drone. Let this resultant graph be represented as **H** with all the nodes as the vertices of the graph. We do not represent drones as additional vertices as this facilitates the comparison of connectivity in the heterogenous network with different number of drones.

The spectrum of graph **H** is used to measure the network connectivity as in [15]. The spectrum of a graph is represented by the eigenvalues and eigenvectors of the adjacency matrix or Laplacian matrix (the difference of the degree matrix and adjacency matrix) [75]. However, unlike [15] that uses the eigenvalue of the state transition matrix to quantify the connectivity, we use the *Kirchhoff index* (Kf) of **H**. *Kirchhoff index* can be explained in terms of *resistance distance* which is defined as,

Definition 1 *Resistance distance (r_{ij}) between the vertices i and j of graph **H** is defined to be equal to the effective electrical resistance between nodes i and j of a network **N** corresponding to **H**, with unit resistors taken over any edge of **N** [6].*

The sum of all the resistance distances over all pairs of vertices is called as Kirchhoff index *(Kf) and is given by [6],*

$$Kf = \sum_{i<j} r_{ij} = n \sum_{i=1}^{n-N_{partitions}} \frac{1}{\mu_i} \tag{3.4}$$

*where, n is the number of nodes and μ_i is the i^{th} largest eigenvalue of Laplacian matrix of **H**. We assume that,*

*$\mu_1 \geq \mu_2 \geq ... \geq \mu_n$. $N_{partitions}$ is the number of partitions in the graph **H** and is equal to the number of μ_i's with value '0' [70]. It should be pointed out that, lower the value of Kf, lower the overall resistance distance in the graph and hence better the connectivity.*

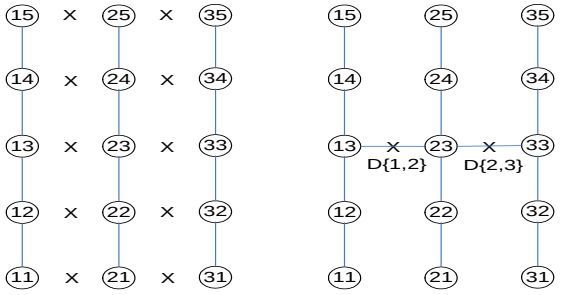
For acyclic graphs r_{ij} corresponds to the shortest distance (number of hops) between vertices i and j and thus Kf will be equivalent to the classical Wiener Index [70, 6]. However, for arbitrary graphs the above representation of Kf corresponds to the overall connectivity within various partitions rather than shortest path connectivity, while ignoring the disconnections amongst partitions. We have validated this measurement parameter for various scenarios but do not report the results here due to lack of space. Thus, we define the objective function, $f(I)$, for an individual (I) as,

$$f(I) = -(N_{interface}) \times (N_{partitions})^{\beta} \times (Kf)^{\gamma} \tag{3.5}$$

where, $N_{interface}$ represents the total number of interfaces from all the p drones in an individual and $N_{partitions}$ represents the number of partitions in the graph $\boldsymbol{H}$. $N_{partitions}$ is equal to the number of unit eigenvalues of the state transition matrix generated using adjacency matrix of $\boldsymbol{H}$ [15]. The exponential factors, β and γ, represent the weight of the respective factors in $f(I)$ and were fixed by trial and error to 2 and 0.5 respectively. The fitness value increases with the increase in the connectivity across the network but decreases with the increase in the number of drone interfaces and network partitions. The fitness function tends to obtain a low-cost solution by minimizing the number of interfaces on each drone.

Test Example

We now consider a test example to demonstrate the validity of the proposed GA. Consider 3 columns of heterogeneous nodes as shown in Figure 3.5. The nodes are placed at regular unit distance from each other, i.e. a node is unit distance intervals away from its



(a) Candidate locations

(b) Refined placement

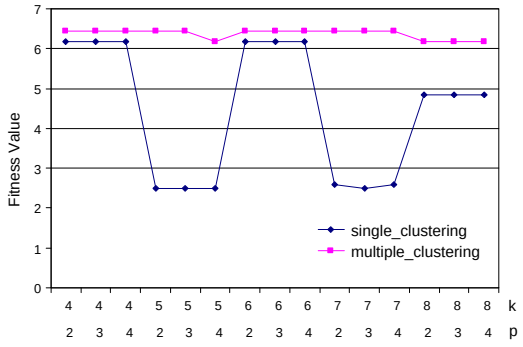
Figure 3.5: Genetic algorithm example

north, south, east and west neighbor. The communication distance for all types of interfaces for both the drone and nodes is assumed to be unit distance. However, since the nodes in adjacent columns are of different types they cannot directly communicate with each other.

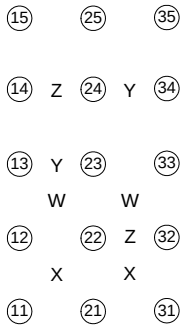
Let the candidate locations be as shown in Figure 3.5(a). We now use this as input to GA in order to place 2 drones (p=2). The result at the end of 300 generations is shown in Figure 3.5(b). The interfaces of the respective drones are also indicated in the figure. As seen from this figure, the solution obtained by GA has optimal placements (minimum average number of hops between any two nodes in the network) and interfaces for both the drones.

3.3 Results

We consider 2 examples to test our proposed PISA algorithm. In these examples, the location and type of nodes are known. PISA gives the solution for placements and interfaces of drones in the network.



(a) Fitness values for different values of k and p



Fitness Value	Drone Position
6.45	W
6.17	X
4.85	Y
2.59	Z
2.49	No drones

(b) Drone placements for different fitness values

Figure 3.6: Effect of multiple clustering iterations for 3 columns example.

3.3.1 Three Column Example

We reconsider the previous example of 3 columns of heterogeneous nodes as discussed in Section 1. In this section we consider the application of PISA for obtaining the complete solution. The number of clusters, k, is varied from 4 to 8 while the number of drones to be placed, p, is varied from 2 to 4. Since a drone should have atleast two different interfaces on it, any drone with less than 2 interfaces, i.e. less than 2 bits set in the N_t chromosome bits, is ignored.

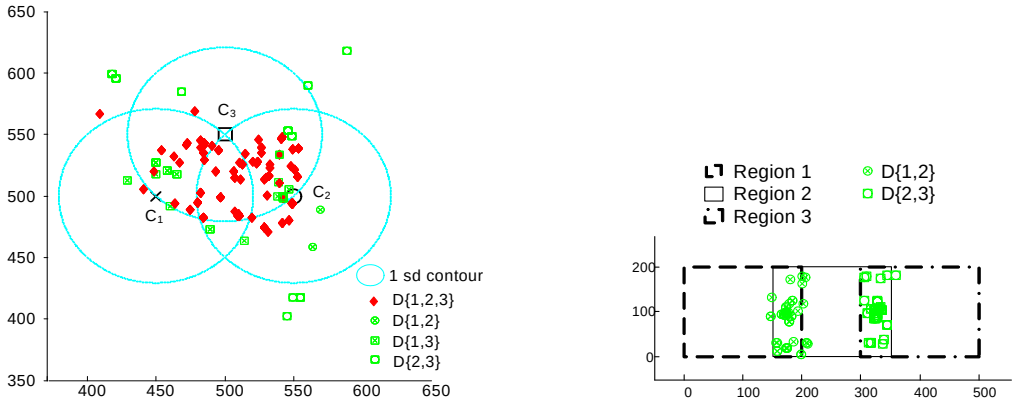
The resultant fitness values are plotted in Figure 3.6(a) under single_clustering. *The drone locations corresponding to different fitness values are represented in Figure 3.6(b). Note that due to symmetry of the node distribution, locations that are mirror images of the solutions depicted in the figure would also result in the same fitness values. These are trivial and not shown in the figure.*

As seen from Figure 3.6(a) under single_clustering, *it is observed that in many cases using k candidate locations that are obtained from single HCA iteration result in lower fitness*

values. In one of the case, k=7 and p=3, no solution was obtained and hence the resultant network is partitioned (Figure 3.6). It is found that this is due to an inappropriate set of candidate locations obtained from HCA. Thus in order to correct this, the coarse placement step is modified to incorporate multiple iterations (5 iterations in our case) of HCA. The resultant set of candidate locations is a union of all the cluster centroids obtained from each HCA iteration. This greatly improves the fitness values of resultant solution for all values of k and p as shown in Figure 3.6(a) under multiple_clustering.

Note that most of the solutions for different values of k and p result in drones placed at location 'W' in Figure 3.6(b). This results in linkages amongst nodes {12, 13, 22, 23} and {22, 23, 32, 33} which are within unit distance from respective drones place at 'W'. This is better than previous solution of Section 1 which only resulted in links amongst nodes {13, 23} and {23, 33}. In fact, it can be easily observed that no other placements of drones anywhere in the network would result in lower average number of hops between two nodes than 'W' (or locations that are its mirror images). Thus, using multiple clustering iterations almost all solutions result in optimally connected heterogeneous network irrespective of the value of k[4].

It is interesting to point out that all solutions, even for $p > 2$*, consist of only 2 drones; the other (p-2) drones had 0 or 1 interface. Moreover, the drones between columns 1 and 2 consist of interfaces '1' and '2' and similarly drones between columns 2 and 3 consisted of interfaces '2' and '3'. No solution had drones with all 3 interfaces.*



(a) Nodes normally distributed with mean at C_1, C_2 and C_3 respectively.

(b) Nodes uniformly distributed within respective regions.

Figure 3.7: Distribution of drone placements for network formed of three type of nodes.

3.3.2 Random Node Distribution

In the previous example, PISA was tested with fixed node locations. We now test PISA for randomly distributed nodes. Consider a network comprising of 3 types of nodes. These nodes are each Gaussian distributed with respective statistical means at C_1, C_2 and C_3 as shown in Figure 3.7(a). The figure also represents a $\times$, 0 and $\square$ with the respective statistical means to indicate the different type of nodes for each distribution. The means are chosen to form a triangular arrangement. Each of the Gaussian distributions has a standard deviation equal to 50 and a cross-covariance equal to 0. We assume that there are 50 nodes of each type; hence, a total of 150 nodes are in the network. The communication distance is assumed to be 50 units for all type of nodes and drone interfaces. Intuitively, a favorable solution would be a single drone with 3 interfaces placed at the center of the triangle formed by C_1, C_2 and C_3.

[4] *Note that this is only valid when non-extreme values of k are considered.*

We run a Monte-Carlo simulation with the node locations randomly selected based on a Gaussian probability density function for each iteration. The interconnections between similar nodes are obtained based on the communication distance amongst nodes. PISA is then applied to obtain the drone locations. Multiple clustering iterations, as explained in previous case, is used in this case too. For each iteration, k and p are varied as {15, 20, 25, 30} and {2, 4, 6} respectively. Note that some of the combinations of k and p result in drones with 0 or 1 interface which were ignored. The number of GA generations for this case was set to 500. In all, 26 Monte-carlo iterations are considered and the resultant drone placements are represented in Figure 3.7(a). As seen from the figure, most of the solutions are drones with 3 interfaces which lie between the 3 means. The drones with 2 interfaces are usually between the means of the corresponding Gaussian distributions. The solution obtained from PISA thus follows the intuitive placement of drones for this simple example.

We also consider uniform distribution of the 3 types of nodes within 3 regions as shown in Figure 3.7(b). The figure shows results obtained following the procedure outlined for the Gaussian distribution case above. As seen from the figure, multiple interface drones were placed at overlapping regions with the correct interfaces. The solution obtained from PISA thus follows the intuitive placement of drone for these simple example.

Chapter 4

Hierarchical Networks

Wireless sensor networks have been envisioned to collect data autonomously for a wide range of applications including habitat monitoring, preventive maintenance of industrial facilities, and security monitoring [52]. Earlier wireless sensor networks were assumed to be wholly comprised of resource constrained sensor nodes to measure the parameter of interest and forward the collected data through multiple hops to a data collection center called sink *[52]. One problem with this architecture is that the sensor nodes need to forward data for other sensor nodes. Since sensor nodes only have limited energy, this architecture is not scalable. To address this problem, a hierarchical architecture [1] was proposed recently wherein the resource constrained sensor nodes (called Lite Node (LN)) will collect the data and forward it to sophisticated nodes (SN) which further aggregate the data and forward it to the sink. Such an architecture has the advantages of being more practical, economical, scalable, and capable of extending the lifetime of the network [63].*

In this work, we consider a two-tiered hierarchical network where the LNs form the lower layer and SNs form the upper layer. The LNs can be regular sensor nodes such as 'motes' [65]. The SNs are nodes, such as Ultra Wide Band (UWB) nodes, which have more computation, energy and communication capabilities. SNs are assumed to communicate with each other using high-bandwidth communication and with neighboring LNs using traditional RF interface. This topology is shown in Figure 4.1.

Section 4.1 discusses the motivation behind this work. The problem is formulated considering practical sensor network characteristics in Section 4.2. Section 4.3 describes the

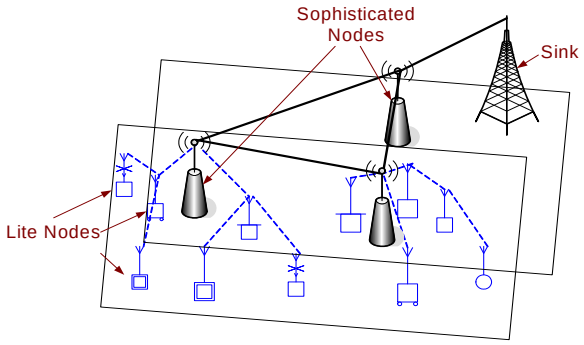


Figure 4.1: Hierarchical network topology.

three candidate algorithms used to solve the problem for random LN deployment. Section 4.4 shows the results of various parameter effects and algorithms on the solution. It also discusses a HYBRID algorithm.

4.1 Motivation

*The specific problem of interest in this work is the optimal placement of SNs considering three placement algoritms. Since the LNs are cheap they would be randomly scattered over the RoI to measure the parameters of interest. We assume their locations can be estimated using any one of the existing localization techniques such as [56, 48]. We intend to place minimum number of SNs to handle the traffic generated by LNs (*traffic constraint*) and ensure that the SNs form a connected network (*connectivity constraint*). This problem of SN node placement has been proved to be NP-hard for both the traffic and the connectivity constraints [62]. We formulate it as a multi-constraint optimization problem and solve it using algorithms traditionally used for such problems; namely, binary integer linear programming (BILP), greedy algorithm (GREEDY), and genetic algorithm (GA).*

Our network architecture differs from existing flat and hierarchical architectures [20, 37, 24], as we consider the practical multi-hop nature of sensor networks wherein the LNs can connect to the SN through multiple hops via other LNs. This is essential to efficiently use the SN's resources when there is large disparity between the LN and SN capabilities. We also consider other phenomena of sensor networks, such as, decreasing probability of successful packet delivery with increasing number of LN hops and the increased load of an LN forwarding other LN traffic to an SN. The key contributions of this work is comparison of the algorithms considering several aspects and formulation of SN placement problem considering various factors such as heterogeneous LN traffic, LN capability to reach a SN via multi-hops and SN connectivity. We also study the effects of the various parameters on the solution. Apart from the comparison of parameters we also propose a new algorithm, HYBRID [47], that improves the results from BILP and GREEDY using GA. Its performance, in terms of number of SNs placed was found to lie between GA and BILP.

Note that although we consider sensor networks in this work, the concepts developed in this work can be extended to any hierarchical mesh network in general.

4.1.1 Related Work

The earlier sensor networks assumed flat network architecture, which causes performance problems when large number of nodes are deployed. In order to improve scalability, two-tiered architecture has generated more research interests. Many relay node placement algorithms have been proposed for this architecture to maintain network connectivity and reliability. We consider such clusterheads or relay nodes as SNs described previously. In some deployment schemes, the nodes are assumed to be homogeneous. For example, an

incremental self-deployment method was proposed in [34] to place some sensor nodes as relay nodes in a homogeneous sensor network. However, the method can possibly result in a sub-optimal deployment. In [9], an integer linear programming method was used to place sensor nodes with different costs in a sensor network. The objective is to find the number and locations of each type of sensors to cover a grid area with minimum cost. In [11], sensor nodes are deployed as relay nodes to maintain global network connectivity. The minimum node placement problem is modeled as a minimum Steiner tree with bounded edge length *problem. Several approximation algorithms were proposed to solve the optimization problem. But it does not consider a hierarchical architecture. The above papers did not consider LN traffic constraints and SN capacity.*

Some placement algorithms assume the sensor network consists of heterogeneous nodes. In [71], the problem of finding minimum number and locations of backbone nodes in a heterogeneous ad hoc network was formulated as a constrained Steiner network problem. *Three approximation methods were proposed to solve the problem. In [63], two algorithms were proposed for relay node placement in a two-tier sensor network. The problem is formulated as a* Minimum Geometry Disk Cover problem *but assumes each sensor node is connected to two relay nodes to form a 2-connected network to provide fault tolerance. Similarly, [37] studied fault-tolerant and minimum number of relay node placement problem in a two-tiered wireless sensor network. The first step solved a* minimum disk cover problem *so that the sensor nodes are connected to the relay node. The second step solves the minimum Steiner tree problem to add Steiner nodes so that the relay nodes are connected. In these papers, the sensor nodes (LNs) are assumed to reach the relay nodes (SNs) in single hop.*

In addition to the connectivity constraint, some research work is concerned about network lifetime. In [24], the joint problem of energy provisioning and relay node placement was studied to increase network lifetime. An efficient heuristic algorithm was proposed to solve the formulated mixed integer nonlinear programming problem. This can be considered complimentary to our problem since we intend to find the solution for an initial network deployment rather than provisioning for an existing network as done in [24]. In [72], the problem of optimal placement of relay nodes is modeled as a minimum set cover problem *and a recursive algorithm is proposed to solve the same. The objective is to deploy heterogeneous nodes so that the total network cost is minimized and the lifetime, coverage and connectivity constraints are satisfied. The research in [67] and [68] extends the work in [72] and proposes a two-phase relay node placement algorithm. The relay nodes placed in the first phase are directly connected to sensor nodes. In the second phase, more relay nodes are placed to satisfy the lifetime and connectivity constraints. However, the above papers consider a single hop from a LNs to SNs. This may not be cost effective, especially when there is a greater disparity between the capabilities of LNs and SNs.*

In this work, we consider a more practical case where the LNs can reach SN via multiple hops. We formulate the SN node placement problem in the next section.

4.2 Problem Formulation

In this work, we consider heterogeneity in terms of node capabilities and traffic generated by different LNs . Considering the notations given in Table 4.1, we make the following assumptions :

Parameter	***Description***
$N(LN)$, $N(SN)$	*Number of lite nodes (LN) and sophisticated nodes (SN) respectively*
R_{LN}, R_{SN}	*Communication range of LN nodes and SN nodes respectively*
LN_i,SN_j	*The i^{th} LN and j^{th} SN respectively*
$d(X_i, X_j)$	*Distance between node X_i and X_j*
$h(X_i, X_j)$	*The minimum number of hops required for the traffic from node X_j to reach node X_i*
h_{max}	*The maximum hops of LN traffic that an SN will handle (design parameter)*
w_{hi}	*Weighing factor for i^{th} hop LN traffic considered at an SN (design parameter)*
$N(LN_{hi}, SN_j)$	*The number of LN that are i hops away from SN_j*
$T(LN_i)$	*The traffic generated (data rate) by node LN_i*
O_f	*The overprovisioning factor for traffic generated by LN*
$C(SN_j)$	*The capacity (data rate) of node (SN_j)*
$C(LN_k, SN_j)$	*The capacity of SN_j offered to a single LN_k*

Table 4.1: Table of parameters and notations.

- *Different LNs are used to measure different parameters (such as temperature, humidity and video feed) and hence generate different amounts of traffic.*
- *Two LNs can communicate with each other if* $d(LN_i, LN_j) \leq R_{LN}$.
- *An SN* (SN_j) *will have two interfaces: one to communicate with neighboring* LN_i*'s (if* $d(LN_i, SN_j) \leq R_{LN}$*) and other to communicate with neighboring* SN_i*'s (if* $d(SN_i, SN_j) \leq R_{SN}$*).*
- *All the SNs need to be connected to the sink directly or via other SNs to send data to sink.*
- *The sensor network is static where the locations of LNs are known and do not change after deployment. The SNs can be placed at calculated locations manually or by mobile robots.*
- *The LNs can forward their data via multiple LN hops to the most feasible SN.*

We now develop a model to define our SN placement constraints. Two constraints namely; the traffic constraint *and* connectivity constraint *are developed. The traffic constraint guarantees that all LN traffic can be handled by SNs, which implies that all LNs can reach at least one SN. Furthermore, the connectivity constraint guarantees that all SNs can reach the sink. The actual routing of packets between LNs, SNs and sink may be carried out via any of the ad hoc routing protocols, such as, AODV or DSR [2]. In practice, depending on the network dynamics, the routing algorithm may deliver LN traffic to a SN different than that considered during design.*

The traffic constraint considers various practical characteristics of sensor networks. It would be desirable that every LN be close to some SN to avoid long paths from a LN to its

closest SN. Therefore, we limit the maximum number of hops (h_{max}) of LN traffic. Thus, the set of LNs that a SN_j can serve is given by $Serve_{SN_j}$ as follows:

$$Serve_{SN_j} = \{LN_i | h(SN_j, LN_i) < h_{max}\}$$

This is just a design restriction so that appropriate number of SNs are deployed. h_{max} serves as a measurable metric for characterizing the number of LNs that can be connected to a SN. The greater the h_{max}, the more LNs can be served by a single SN and hence fewer SNs will be required. This is based on assumption that any SN_j can handle the traffic generated by the LNs in $Serve_{SN_j}$. However, this may not be the case if h_{max} increases above a threshold and hence we consider LN traffic next.

In an ad hoc network, the probability of a packet to reach a destination decreases as the number of hops required to reach the same increases [59]. This may be due to packet collision, intermediate node failure or other characteristics of ad hoc network. In order to reflect this in our constraints, we assume that the reliability of an SN to handle traffic generated by an LN, $T(LN)$, decreases with the number of hops it takes to reach the SN, i.e., $w_{hi} > w_{h(i+1)}$, where w_{hi} are weight indicating reliability of LN traffic reaching a SN in i hops. Adding such weights is another design restriction to accommodate the unreliable nature of the wireless link. The weights can force LNs further from SNs to have access to multiple SNs within h_{max} hops and hence increasing the robustness of the deployment. We next show how the capacity of SNs can be distributed amongst LNs in $Serve_{SN_j}$ based on LNs respective weights.

Let $C(SN_j)$ denote the capacity in terms of data rate that the node SN_j can handle. We need to allocate this capacity to different nodes in $Serve_{SN_j}$ to consider the weights w_{hi}. LNs with higher weights will be granted greater proportion of $C(SN_j)$. Let $N(LN_{hi}, SN_j)$

denote the number of LNs that are i hops away from SN_j. Also, for simplicity, we assume that traffic generated by each LN is equal to $T(LN)$. These nodes will all have weights w_{hi} assigned to them. Thus the total weight for the cumulative traffic generated by nodes i hops away will be:

$N(LN_{hi}, SN_j) \times w_{hi} \times T(LN)$.

Note that the traffic generated from LNs further away from SN_j will be relayed by LNs closer to SN_j. Thus the effective traffic (E_i) from all i^{th} hop LNs will include the traffic generated by LN_ks in $Serve_{SN_j}$ and $h(LN_k, SN_j) > i$. This is given by Equation 4.1.

$$E_i = \sum_{k=i}^{h_{max}} (N(LN_{hk}, SN_j) \times w_{hk} \times T(LN)) \tag{4.1}$$

Thus the proportion of $C(SN_j)$ allocated to LN nodes which are i hops away is denoted by P_{hi} in Equation 4.2.

$$\begin{aligned} P_{hi} &= \frac{E_i}{\sum_{i=1}^{h_{max}} E_i} \\ &= \frac{\sum_{i=hi}^{h_{max}} (N(LN_{hi}, SN_j) \times w_{hi})}{\sum_{i=1}^{h_{max}} (i \times N(LN_{hi}, SN_j) \times w_{hi})} \end{aligned} \tag{4.2}$$

Note that, $P_{hi} = 0$ for $hi > h_{hmax}$ which indicates that no portion of $C(SN_j)$ is allocated to LNs that are further than h_{max} hops away. Also, $\sum_{i=1}^{h_{max}}(P_{hi}) = 1$, indicating that the entire $C(SN_j)$ is divided amongst the LNs in $Serve_{SN_j}$. From the above equation, we can easily find the capacity ($C(LN_k, SN_j)$) of SN_j offered to a single LN_k that is i hops away as:

$$C(LN_k, SN_j) = \frac{1}{N(LN_{hi}, SN_j)} \times P_{hi} \times C(SN_j) \tag{4.3}$$

Considering the above constraints, the placement problem can be formulated as the following optimization problem.

We want to find the number and locations of all SNs for minimizing the number of SNs, denoted by

$$min\{N(SN)\} \tag{4.4}$$

that is subject to,

$$\sum_{j=1}^{N(SN)} C(LN_k, SN_j) \geq O_f \times T(LN_k),$$

$$\forall k = \{1, 2, ..., N(LN)\} \tag{4.5}$$

and

$$All\ SN_j\ are\ connected \tag{4.6}$$

The traffic constraint *in Equation 4.5 specifies that the capacity allocated to LN_k via different SNs, which is represented as $\mathbb{C}(LN_k) = \sum_{j=1}^{N(SN)} C(LN_k, SN_j)$, is sufficient to handle the over provisioned traffic $(O_f \times T(LN_k))$ of LN_k. The overprovisioning factor (O_f) gives additional control over design by accommodating the possibility of higher LN traffic loads. The* connectivity constraint *in Equation 4.6 specifies that the SNs have to form a connected network. In general, finding minimum $N(SN)$ under these constrains is NP-hard [62].*

Simple Regular Grid Example

In order to understand the above problem and its various design parameters, let us consider a simple grid network where an LN is placed at each of the grid intersections. We intend to place SNs on some of the grid points such that the constraints in Equations 4.5 and 4.6 are satisfied. Since in the grid network an LN forms a link only to the closest vertical and horizontal LNs, R_{LN} will be equal to the horizontal or vertical spacing between alternate LNs on the grid.

We formulate the SN placement problem as a binary integer linear programming (BILP) problem. Consider the binary variable X_i where $i = \{1, 2, ..., N(LN)\}$. Here $X_i = 1$ indicates that a SN is placed at the location of LN_i and vice versa. Thus the BILP can be formulated as,

$$min \left\{ \sum_{i=1}^{N(LN)} (X_i) \right\} \tag{4.7}$$

subject to,

$$C \times X \geq O_f \times T \tag{4.8}$$

$$C = \begin{bmatrix} C(LN_1, SN_1) & \cdots & C(LN_1, SN_{N(LN)}) \\ C(LN_2, SN_1) & \cdots & C(LN_2, SN_{N(LN)}) \\ \vdots & \ddots & \vdots \\ C(LN_{N(LN)}, SN_1) & \cdots & C(LN_{N(LN)}, SN_{N(LN)}) \end{bmatrix}$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_{N(LN)} \end{bmatrix}, \text{ and } T = \begin{bmatrix} T(LN_1) \\ T(LN_2) \\ \vdots \\ T(LN_{N(LN)}) \end{bmatrix}$$

Here, $C(LN_i, SN_j)$ indicates the capacity provisioned to node LN_i when the SN is located at location of node LN_j. Although this would result in optimal solution (for the grid deployment) with respect to the traffic constraint, it does not consider the connectivity constraint of Equation 4.6. In order to satisfy this constraint, we obtain the value of h_{max} as follows.

$$h_{max} = \lfloor \left(\frac{R_{SN}}{R_{LN}} - 1 \right) / 2 \rfloor \tag{4.9}$$

Theorem 4.1 *For the regular grid network, a solution obtained with h_{max} given by Equation 4.9 will result in a connected SN network.*

For a given h_{max}, there will be at least 2 SNs for every set of connected $2h_{max}$ LNs. This is easy to deduce since we limit the $Serve_{SN_j}$ to LNs that are at most h_{max} hops away

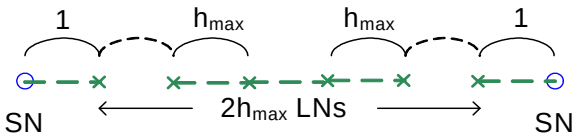


Figure 4.2: SN placement for connected $2h_{max}$ LNs.

from SN_j i.e, $h(SN_j, LN) < h_{max}$. Furthermore, there may be more than 2 SNs required to satisfy the traffic constraint.

The 2 SNs will be farthest away if the $2h_{max}$ LNs lie along a straight line and the 2 SNs lie at the two end points as represented in Figure 4.2. It can be easily observed that any other arrangement of LNs will bring the SNs closer. Also, if the traffic constraint is not satisfied by 2 SNs, the additional SNs required will further decrease the adjacent SN distances. Thus, the maximum distance between SNs serving a connected set of $2h_{max}$ LNs will be $(2h_{max} + 1)R_{LN}$. Thus if,

$$R_{SN} \geq (2h_{max} + 1)R_{LN} \tag{4.10}$$

then the SN placement obtained from satisfaction of the traffic constraint, as formulated in Equation 4.7 and 4.8, will result in a connected SN network. Equation 4.9 follows from Equation 4.10 and hence the proof.

We use CPLEX [27] to solve the BILP. Using the above formulation, we solve the SN placement problem for 81 LNs distributed on a grid over a 20 by 20 unit area. The SN placement is represented in the Figure 4.3. The various parameter values are indicated with the figure. It should be noted that using h_{max} value from Equation 4.9 resulted in a connected SN network. The placement for grid networks is further discussed in Section 4.4.

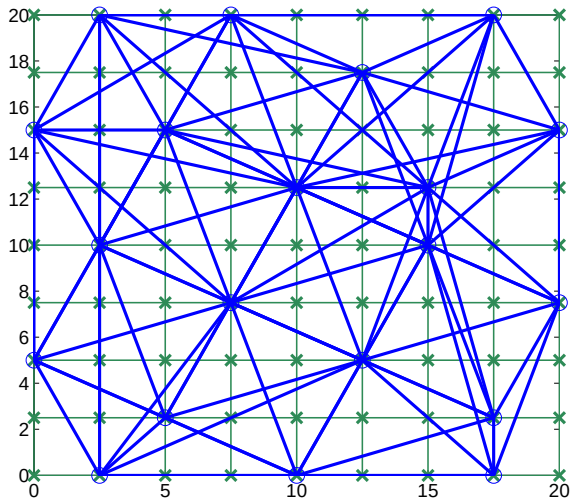


Figure 4.3: A BILP solution for a grid based LN deployment. $R_{LN} = \frac{20}{\sqrt{N_{LN}}}, R_{SN} = 12 units$, $h_{max} = \lfloor (\frac{R_{SN}}{R_{LN}} - 1)\ 2 \rfloor$, $T(LN) = 1$, $C(SN) = 10T(LN)$.

4.3 Methodology

In the previous section, a grid based LN deployment was considered. In this secton, we describe three algorithms to solve the SN placement problem for randomly deployed LNs. Such deployments are more realistic than a grid based deployment. They can be realized by scattering the LNs randomly from an aircraft as envisioned in many future sensor network applications [29].

4.3.1 Binary Integer Linear Programming

Many previous solutions for placing high-end nodes reduced the problem to a binary integer linear programming (BILP) problem under the assumption that SNs can be placed at grid points [39]. The LNs are randomly distributed throughout the RoI. The SNs are to be placed at grid intersection of an assumed grid over RoI. These placements are obtained by solving a BILP problem. Let $N(G)$ be the total number of grid points. The binary variable

X_i represents whether an SN is placed at the i^{th} point of this grid. The formulation and notations are thus similar to the one described in Section 4.2. However, in this case a set of constraints, with $N(LN)$ equations and $N(G)$ variables, is created.

Also, due to random LN placement and a fixed R_{LN} value, setting a value on h_{max} cannot guarantee connectivity of SN nodes. Thus, the SN solution obtained by BILP will satisfy traffic constraint in Equation 4.5 but may not satisfy connectivity constraint in Equation 4.6. In order to connect the SN nodes obtained by BILP we use a greedy implementation of Steiner tree problem as proposed in [7]. Unlike the implementation of [7], we add a single edge at a time and re-evaluate the various distances after adding additional Steiner nodes along the edge. This modification considers shorter links that may terminate on Steiner nodes. In the worst case, when the added edge terminates on original nodes, the performance of our greedy algorithm will be equivalent to the algorithm in [7]. Our algorithm is given as follows:

Greedy Steiner Tree Algorithm

1. *Find connected components of SN graph.*
2. *Find the shortest edge required to connect two disconnected components.*
3. *Add Steiner nodes (additional SNs) along the edge to connect the components.*
4. *If resultant graph is disconnected*
   (a) *Consider all SNs (from BILP and Steiner nodes) in the SN graph and GOTO Step 1.*
   (b) *Else END.*

Thus, the solution steps using BILP can be summarized as:

BILP Algorithm

1. *Find SN locations using BILP on the LN location. The resultant solution is called pure BILP solution.*

2. *Using greedy steiner tree solution place additional nodes to obtain connectivity amongst SN from pure BILP solution.*

4.3.2 Greedy Placement Algorithm

In this section, we describe a greedy algorithm for the SN placement problem. We first obtain candidate SN locations by clustering and the choose the best candidate locations for SN that can satisfy the maximum LN traffic and also maintain connectivity with at least one previous SN. The algorithm is given below.

GREEDY Algorithm

1. *Initialize by obtaining clusters and placing the first SN at the cluster center that can serve maximum number of LNs.* $N(SN) = 1$.

2. *While the traffic of all LN is not provisioned (traffic constraint), i.e.* $(\mathbb{C}(LN_k) - O_f \times T(LN_k)) \geq 0$ *for all* $k = \{1, 2, ..., N(LN)\}$

   (a) *Consider LNs in the neighborhood* $(2R_{SN})$ *of existing SNs in order to find the candidate locations to place the next SN.* $\{LN_i | d(LN_i, SN_j) < 2R_{SN},$ *any* $j = \{1, ..., N(SN)\}\}$

   (b) *Remove* LN_k *from the above set whose traffic is already provisioned, i.e.* $(\mathbb{C}(LN_k) - O_f \times T(LN_k$

   0

(c) *If empty set then goto Step (2g).*

(d) *Cluster the remaining nodes. The clustering algorithm is executed multiple times to obtain several different number of clusters.*

(e) *If no cluster center is within distance R_{SN} from atleast 1 SN then goto Step (2g).*

(f) *Place the additional SN at cluster center which serves the most nodes and is within distance R_{SN} from at least 1 SN. This ensures that the additional SN will be connected to previous SNs. $N(SN) = N(SN) + 1$. Goto Step 2.*

(g) *Find LN_i with $\max(\mathbb{C}(LN_k) - O_f \times T(LN_k)) \geq 0$ closet to any SN_j. Place additional SN at distance R_{SN} from SN_j towards LN_i. $N(SN) = N(SN) + 1$. Goto Step 2.*

The k-means clustering [61] algorithm is used to obtain clusters. We run the clustering algorithm multiple times with different values of k, which represents the number of desired clusters in the k-means algorithm. The value of k considered are 10%, 15%, 30% and 50% of the total number of nodes to be clustered. For step (2d), k=100% of total number of nodes is also considered, so as to include individual node locations as candidate locations for placing an SN. As the greedy algorithm is a heuristic algorithm, it cannot guarantee to find the best solution during a single iteration. Thus, we obtain multiple solutions by running the greedy algorithm a preset number of times (20 times in this work) and choose the best amongst them as the final solution.

4.3.3 Genetic Algorithm

The general description of a genetic algorithm is available earlier in this dissertation in Section 3.2.2. We now explain the chromosome representation used for our problem. Let the RoI be an area between [0,0] and [xmax,ymax]. The chromosome is represented as a vector of 100 real valued variables representing (x,y) coordinates of 50 possible SNs in an area between [-xmax,xmax] and [-ymax,ymax]. However, we only consider SNs within [0,0] and [xmax,ymax] as valid SNs. Thus, $N(SN) <= 50$. In GA terminology, the group of 100 variables is the genotype representation while the valid SN set with less than 100 variables is the phenotype representation of the solution [18]. We evaluate the fitness of a gene after decoding its genotype to a valid phenotype, i.e. discarding locations with negative coordinates. Fitness function (fit) is given by Equation 4.11.

$$\begin{aligned} fit = &-(N(SN)+1) \times (rem_t + 1)^{\alpha} \\ &\times (N(partitions)+1)^{\beta} \end{aligned} \tag{4.11}$$

$$\begin{aligned} rem_t = &\sum_{k=1}^{N(LN)} (abs\{\mathbb{C}(LN_k) - O_f \times T(LN_k)\} \\ &\times (\mathbb{C}(LN_k) < O_f \times T(LN_k))) \end{aligned} \tag{4.12}$$

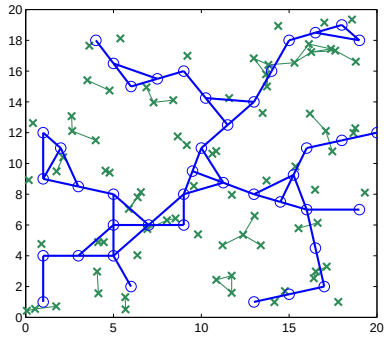
where, rem_t is the total LN traffic that is not satisfied by the SNs. The second term in rem_t calculation is a binary variable which is 1 if $\mathbb{C}(LN_k) < O_f \times T(LN_k)$ and 0 otherwise. Thus, this term forces only unsatisfied LN traffic to be considered in rem_t. $N(partitions)$ is

the number of partitions formed by the graph $\mathbb{G}(V, E)$ representing SN, where $V = \{SN_j\}$ and $E = \{(SN_i, SN_j) | d(SN_i, SN_j) \leq R_{SN}, i \neq j\}$, $(i, j) = \{1, ..., N(SN)\}$. This can be easily obtained by the number of zero eigenvalues obtained from the laplacian matrix of the graph $\mathbb{G}(V, E)$ [70]. The exponents α and β add weights to individual factors in Equation 4.11. To ensure that all individual factors of $fit >= 1$ we add 1 to each of them. This is required to ensure that none of them brings down the fit value to 0 and that the exponents will always increase the value of their respective factors. Note that the individual factors will never be negative.

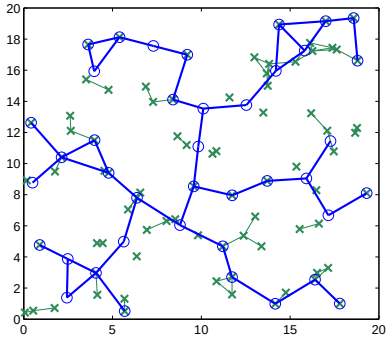
The GA toolbox for MATLAB, GAOT [25], was used to obtain the GA solution. Default parameters for the float representation from [25] were used. The population size was fixed to 30 individuals and maximum iteration (MAX_{gen}) was fixed to 10000. We run the GA a predetermined number of times (20 times in this work) and use the best solution as the solution obtained by GA.

GA Algorithm

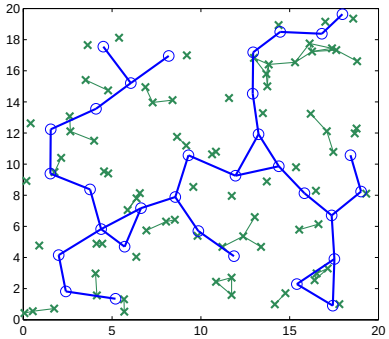
1. *Generate initial random population and calculate the fitness value of each individual.*
2. *Initialize generation: gen = 1.*
3. *While gen < MAX_{gen}*
   (a) *Select parents from current population.*
   (b) *Create children from these parents by mutation and crossover.*
   (c) *Evaluate the fitness of children.*
   (d) *Replace parents if $fit_{child} > fit_{parent}$.*
   (e) *gen = gen + 1.*



(a) BILP results. N(SN)=42 nodes.



(b) Greedy results. N(SN)=40 nodes.



(c) GA results. N(SN)=32 nodes.

Figure 4.4: SN placement obtained by different algorithms. $R_{LN} = 1.5units$,$R_{SN} = 2R_{SN}$, $h_{max} = 3hops$, $T(LN) = 1$, $C(SN) = 10T(LN)$.

4.3.4 Test Case

We obtained the SN placement solution by the algorithms for a single test case. The test case consist of 80 LNs distributed in an 20 × 20 unit RoI. The results are shown in Figure 5.9. All the algorithms provide solutions that satisfy the traffic and connectivity constraints. Also, the GA obtains a solution with the smallest number of $N(SN)$ *with 32 nodes. The detailed performance of the three algorithms is discussed in the next section.*

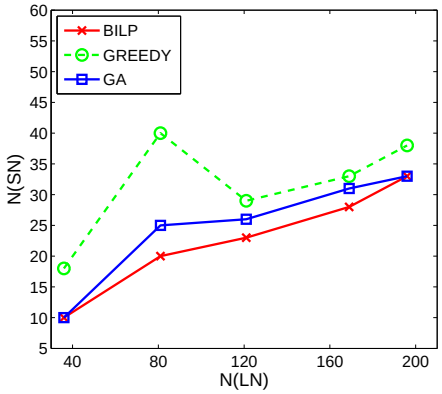


Figure 4.5: Performance of different algorithms. LN placed over a grid. $R_{LN} = \frac{20}{\sqrt{N_{LN}}}, R_{SN} = 12units$, $h_{max} = \lfloor(\frac{R_{SN}}{R_{LN}} - 1)\ 2\rfloor$, $T(LN) = 1$, $C(SN) = 10T(LN)$.

4.4 Results

The three algorithms proposed in previously are compared in this section. Also, the performance effects of various design parameters are discussed. The simulation was carried out in MATLAB over a 20 × 20 unit area RoI.

4.4.1 Algorithm Performance

We first investigate the performance of the three algorithms on a grid network described previously in Section 4.2. $N(LN)$ *is varied but takes only perfect square values due to the grid based arrangement of LN in the 20 × 20 RoI. We use the BILP formulation and* h_{max} *value from Section 4.2. It should be noted that, as derived in Theorem 4.1, the SN placements from the BILP solution satisfied the connectivity constraint. The outcome of BILP thus represents the optimal solution considering SN placements over the grid. The GREEDY and GA described in Section 4.3 are also used to obtain SN placements. As seen from Figure 4.5, the BILP solution results in lowest number of SNs for the grids. The*

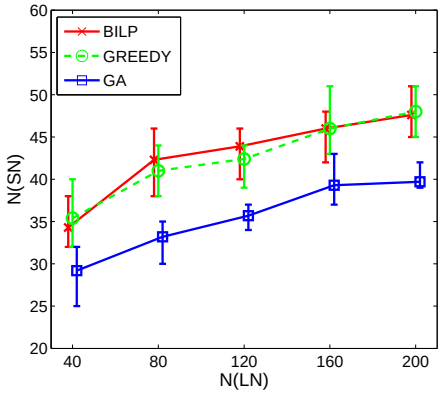


Figure 4.6: Performance of different Algorithms. LN (uniform) randomly distributed. $R_{LN} = 1.5units$,$R_{SN} = 2R_{SN}$, $h_{max} = 3hops$, $T(LN) = 1$, $C(SN) = 10T(LN)$.

solutions obtained by GREEDY and GA algorithms also satisfy both the constraints but result in higher number of SNs for the same LN grid. The GA solution is closer to the BILP solution as compared to the GREEDY solution.

A uniform random distribution of LNs is considered next. The number of LNs was varied from 40 to 200 nodes in steps of 40. Since LNs are randomly distributed, a single distribution cannot be used to characterize the performance of an algorithm. Hence, we consider 10 distinct cases of node distribution for every value of $N(LN)$. *Thus, in all* $5 \times 10 = 50$ *cases are considered. The three algorithms, from Section 4.3, are validated over these 50 cases. The* $N(SN)$ *along with the design parameters are shown in the Figure 4.6. Note that for a given algorithm and* $N(LN)$, *10 cases are plotted as error bars. In the figure, each algorithm's performance represents the average number of SNs required for different values of* $N(LN)s$. *As seen from the figure, the GA outperforms BILP and GREEDY in this case.*

In summary, we can see that BILP gives the best result for a grid-based deployment with value of h_{max} *from Equation 4.9. However, for randomly distributed LNs BILP solution*

will require Steiner nodes to ensure connectivity thus making the solution sub-optimal. The heuristic approaches, do not guarantee optimal solution for both type of LN distributions. GA, however, does perform better than BILP for random LN distribution. However, it takes much longer to run as compared to the BILP implementation.

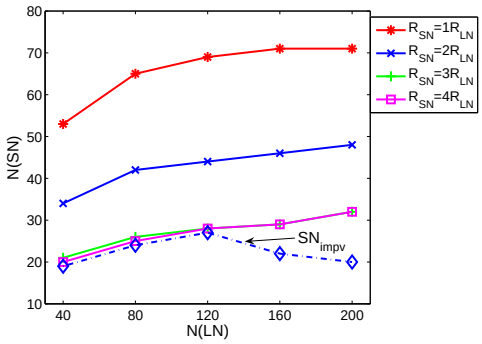
4.4.2 Effect of Design Parameters

In order to study the effect of different parameters, we consider only a single distribution of LN for different $N(LN)$ values. The distribution we consider for a given $N(LN)$ is the one that is closest to the average $N(SN)$ obtained using BILP in Figure 4.6. We vary different parameter values for these distributions and use BILP to solve for the SN placement. Although the effect of the parameter is studied by varying just a single design parameter, we also consider an improved SN SN_{impv} discussed later in this subsection.

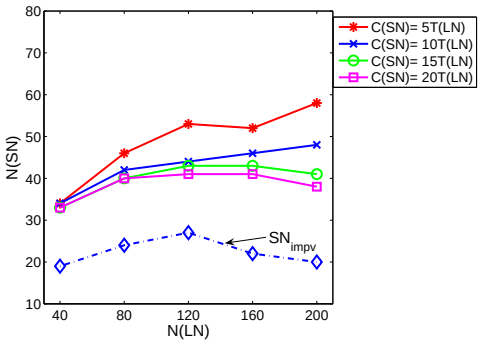
We use BILP to obtain these placement solutions rather than GA since we are interested in studying the effects of various parameter and not the optimal solution for a given parameter value. BILP executes faster and results in a trend similar to GA. Also, in some cases (details in [46]), results in solutions comparable to GA. Hence, we choose BILP to solve the placement problem for different parameter values [46].

Effect of R_{SN}

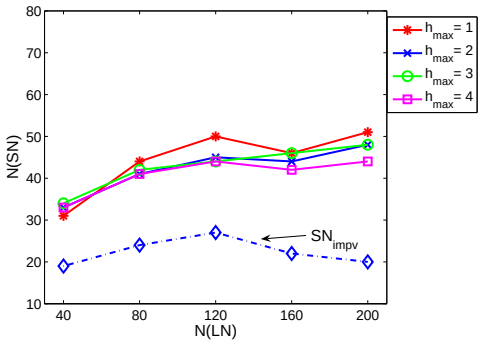
Figure 4.7(a) shows the number $N(SN)$ of SNs required when R_{LN} is fixed and R_{SN} is varied, i.e. in the case when SNs are improved compared to LNs. The figure indicates a drop in $N(SN)$ as R_{SN} increases. A significant improvement is observed when $R_{SN} \geq 3R_{LN}$. This improvement is mainly due to the reduced number of Steiner points required to connect



(a) Effect of R_{SN} on $N(SN)$. $h_{max} = 3hops$ and $C(SN) = 10T(LN)$.



(b) Effect of $C(SN)$ on $N(SN)$. $R_{SN} = 2R_{LN}$ and $h_{max} = 3hops$.



(c) Effect of h_{max} on $N(SN)$. $R_{SN} = 2R_{LN}$ and $C(SN) = 10T(LN)$.

Figure 4.7: Effect of design parameters on $N(SN)$. $R_{LN} = 1.5units$ and $T(LN) = 1$.

the SNs obtained by pure BILP solution (Step 2 of BILP algorithm). This makes the BILP solution closer to the optimal placement solution assuming grid placement of SNs.

In related experiments [46], varying R_{LN} while keeping R_{SN} constant, it was also observed that for dense networks it is more beneficial to improve R_{SN} than R_{LN}.

Effect of $C(SN)$

When SNs with higher data handling capabilities are considered, there is a decrease in the number of $N(SN)$ required as shown in Figure 4.7(b). The performance improvement is more pronounced for higher values of $N(LN)$. This is because for large value of $N(LN)$, increase in capacity $C(SN)$ results in more LNs whose traffic can be completely handled by a single SN. For lower $N(LN)$, the sparse network may limit the set $Serve_{SN_j}$ for a given SN_j due to small number of LNs that are less than h_{max} hops away. Thus, increase in SN capacity may not be useful in this case. Note that this difference between $C(SN)$ and $T(LN)$ can be induced by controlling O_f.

Effect of h_{max}

Figure 4.7(c) shows the effect of h_{max} on $N(SN)$. As h_{max} increases, $N(SN)$ decreases. In fact, $N(SN)$ reduces by 14% when a multi-hop network with $h_{max} = 4$ is considered as compared to a single hop network ($h_{max} = 1$) for $N(LN) = 200$. This clearly shows the advantage of considering multi-hop from LNs to reach SN. Also considering the observed trend, it can be deduced that as higher density network with more resourceful SNs are considered this multi-hop advantage will be more pronounced.

However, there appears to be an exception to this trend for $N(LN) = 40$ nodes. The $N(LN) = 40$ case actually resulted in $N(SN) = 20$ nodes from pure BILP solution for all

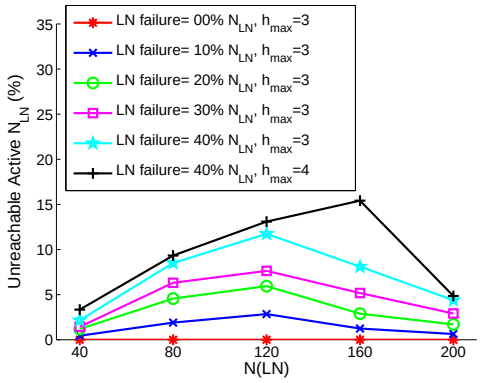


Figure 4.8: Effect of LN failures on residual network. $R_{LN} = 1.5units, R_{SN} = 2R_{LN}$, $T(LN) = 1$, $C(SN) = 10T(LN)$.

values of h_{max}. *This is due to the reason cited earlier as there is little effect on* $Serve_{SN}$ *due to increase in* h_{max} *for lower* $N(LN)s$. *Thus, the small differences in* $N(SN)$ *is due to the Steiner SN nodes that are added using the Greedy Steiner Tree algorithm.*

Combined Effect (SN_{impv})

The combined effect of the improvements were also studied. We considered an SN with $R(SN) = 4R(LN)$, $C(SN) = 20T(LN)$ *and* $h_{max} = 4$. *As seen from the figure 4.7 the combined effect usually results in a much lower* $N(SN)$. *This indicates that as the SN gets more sophisticated fewer SNs will be required. Also, from Figure 4.7(a) it can be seen that* $C(SN)$ *and* h_{max} *have negligible effect for low LN density. They however decrease the required* $N(SN)$ *by 30% when LN distribution is dense (*$N(LN) = 200$ *in the figure).*

4.4.3 Node Failures

In this section, we consider a different performance criteria rather than the number of SNs, i.e., the robustness of the residual network after LN node failures. As SNs are high end devices they are considered to be stable for this work.

The performance of the network after LN failures is evaluated in terms of the percentage unreachable active LNs of the residual network. Figure 4.8 shows the percentage unreachable active LNs for different failure rates. As shown in Figure 4.8 (for $h_{max} = 3$) as the LNs failure increases, the percentage of unreachable active nodes in the residual network increases. We have considered that during the operation phase a SN will serve any reachable LN traffic, i.e. design constraint of h_{max} is ignored. Each point on the plot represents the average result of 100 random cases of node failures. As seen from Figure 4.8, in the worst scenario, when 40% of LNs fail only 10% of the remaining LNs cannot reach a single SN. The results thus indicate a highly robust placement solution for the sensor network.

For lower $N(SN)$, the sparse network results in SN placements close to individual LNs. Thus the failure of nodes has little effect on the connectivity of the remaining nodes. Also note that for higher $N(SN)$ there is drop in the percentage of unreachable nodes. This is due to the proximity of LNs to alternate SNs via multiple hops through different LNs. Figure 4.8 also shows the plot for 40% failure and $h_{max} = 4$. As compared to the 40% failure and $h_{max} = 3$ case, this case results in a higher number of active LNs being affected. Thus, h_{max} can be used during design phase to control the robustness of the deployed solution.

As another point of comparison between the three algorithm, we plot the percentage of unreachable nodes after 40% LN failures for the solutions obtained by the three algorithms. As observed from the Figure 4.9, the GREEDY algorithm outperforms both BILP and GA

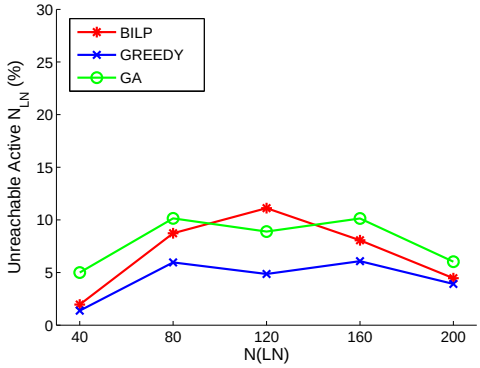


Figure 4.9: Effect of 40 % LN failures on residual network using solutions from different algorithms. $R_{LN} = 1.5units, R_{SN} = 2R_{LN}$, h_{max}=3, $T(LN) = 1$, $C(SN) = 10T(LN)$.

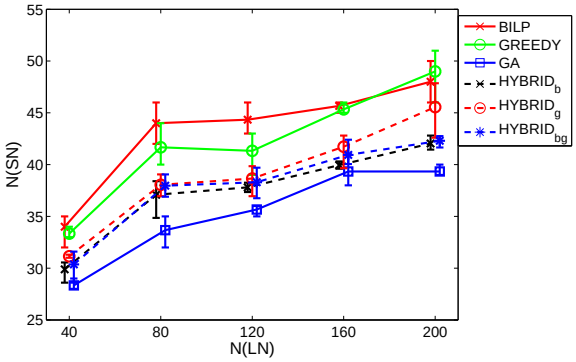


Figure 4.10: Performance of HYBRID Algorithm. LN (uniform) randomly distributed. $R_{LN} = 1.5units, R_{SN} = 2R_{SN}$, $h_{max} = 3hops$, $T(LN) = 1$, $C(SN) = 10T(LN)$.

solutions. In fact, considering node failures, GA solution is almost twice as bad as the solution obtained by GREEDY algorithm.

4.4.4 HYBRID Algorithm

Consider the simulation setup as described in Section 4.4.1 above. The GA solution, although better than BILP and GREEDY solutions, is computationally more expensive than both. For a single case, the GA takes an average of 24 hours, while BILP and GREEDY take an average of 0.75 hours and 0.25 hours respectively on a 2.4 GHz AMD machine.

Thus in order to get the performance of original GA (as described in Section 4.3.3) but at a considerably lower execution time, we propose a hybrid algorithm (HYBRID).

We introduce the solution obtained by BILP or GREEDY or both BILP and GREEDY in the initial population of a modified GA. These algorithms are represented as HYBRID_{b}*,* HYBRID_{g}*, and* $\text{HYBRID}_{\text{bg}}$ *respectively. We also increase certain types of mutations and crossovers (specifically multiple non-uniform mutations and heuristic crossovers [25]) of the GA in lieu of the solutions present in the initial population. Although this increases the number of* function evaluations *(number of times the fitness is calculated) from 24 in original GA to 40 in the HYBRID approach per generation, it results in faster convergence to the optimal solution. We also replace the 2 worse individuals of current generation with the 2 best ones of the previous in order to guarantee retention of the best solutions throughout all generations. The maximum generation is reduced from 10000 to 500 but the population size is kept the same as before (30 individuals). As in the original GA, we run the GA 20 times for each case. However, unlike the original GA, HYBRID converges all the time (due to a solution in the initial population) and hence we consider the average* $N(SN)$ *from these runs as the representative solution for each case. These results are plotted in Figure 4.10.*

The number of functional evaluations for HYBRID approach is 8% of the original GA. Considering the time taken to obtain BILP, the average time taken to obtain a HYBRID_{b} *solution for a single case is about 2.75 hours. This indicates an execution time reduction of 88.54% from that of the original GA. The Figure 4.10 shows the improvement due to the different HYBRID approaches.* HYBRID_{b} *seems to give the best performance. Although original GA results (the best out of 20 GA iterations) are slightly better,* HYBRID_{b} *out performs the original GA when all 20 GA results are considered for each case.*

	$HYBRID_p$	$HYBRID_b$	$HYBRID_g$	$HYBRID_{bg}$	GA	$BILP$	$GREEDY$	GA_{all}	$GREEDY_{all}$
$HYBRID_p$	*N/A*	*1*	*1*	*1*	*1*	*1*	*1*	*0*	*1*
$HYBRID_b$	*-1*	*N/A*	*-1*	*0*	*1*	*-1*	*-1*	*-1*	*-1*
$HYBRID_g$	*-1*	*-1*	*N/A*	*1*	*1*	*-1*	*-1*	*-1*	*-1*
$HYBRID_{bg}$	*-1*	*0*	*-1*	*N/A*	*1*	*-1*	*-1*	*-1*	*-1*

0: no statistical difference between the row and column algorithm
-1: row algorithm better than column, i.e. statistically different and lower mean
1: column algorithm better than row, i.e. statistically different and lower mean

Table 4.2: Statistical (signtest) comparisons of different algorithms.

Apart from the algorithms described above we also consider $\mathrm{HYBRID_p}$, $\mathrm{GA_{all}}$ *and* $\mathrm{GREEDY_{all}}$. $\mathrm{HYBRID_p}$ *is a control for the HYBRID algorithms and does not include any solutions introduced in the initial modified GA population.* $\mathrm{GA_{all}}$ *and* $\mathrm{GREEDY_{all}}$ *indicate the average* $N(SN)$ *obtained from all the GA and GREEDY iterations (20 iterations, refer Section 4.3) for a single case. Note that the GA and GREEDY results represent the best solution form all these iterations. Thus* $\mathrm{GA_{all}}$ *includes raw GA results and is thus representative of overall performance of the original GA.*

For the statistical comparison of different algorithms we consider the average $N(SN)$*'s required for a single case over multiple iterations. As the same case (LN distribution) is solved by all the algorithms, we require to perform paired comparison of the 15 case results over all the algorithms. We consider the sign test [54], which is a non-parametric test designed for distribution-free paired comparisons of small data sample size. Table 4.2 represents the test result of the null hypothesis that row algorithm's result has the same median as that of the column algorithm. A* 0 *indicates that there is no difference while a* ± 1 *indicates that there is a statistical difference at 5% significance level. Furthermore,* $+1$ *indicates that column algorithm has lower mean than the row algorithm, i.e. lower average* $N(SN)$ *and hence better performance. Similarly,* -1 *indicates row algorithm is statistically better than column algorithm.*

As seen from the table, $\mathrm{HYBRID_b}$ *is superior to all other algorithms except GA. However, as indicated previously it takes considerably less execution time than GA. It is also better than GA when all GA iteration results are considered (*$\mathrm{GA_{all}}$*). It guarantees convergence on all iteration and thus a single iteration may be sufficient. Clearly,* $\mathrm{HYBRID_b}$ *performs slightly worse than GA but at considerably lower execution time. Note that although,* $\mathrm{HYBRID_b}$*'s performance is similar to* $\mathrm{HYBRID_{bg}}$*, the former is preferable since it does not require the additional GREEDY result.*

Chapter 5

Mobility

In the previous chapters, the "drones" and sophisticated nodes (SN) were placed assuming a static network, i.e., there is no node mobility. However, node mobility is one of the fundamental characteristics of ad hoc and sensor networks. Many protocols designed for static networks fail when mobility is considered. For example, the service discovery protocol in MANET fails due to node mobility which may lead to network partitions and mergers [60].

In this chapter, the two-tier hierarchical network discussed in Chapter 4 is considered. However, this scheme can easily be extended to the heterogeneous network case of Chapter 3. This chapter should be considered as an extension to the previous two chapters as we consider the problem of mobility of the underlying nodes. Section 5.1 reports the mobility scheme developed in this work. Section 5.2 discusses performance of the proposed mobility scheme considering SN and LN node movement.

5.1 Mobility Scheme

The mobility implementation in this section is based on OPNET[1]*. The SN mobility scheme is discussed in this section. The details of the scheme can be studied using a state diagram of its implementation, discussed in the following subsections. However, before the OPNET implementation is described, a brief introduction to OPNET is provided in the next subsection. The following subsections describe the SN node implementation in detail.*

[1] *http://www.opnet.com/*

5.1.1 OPNET Modeler

OPNET modeler is one of the many suites provided by OPNET (Optimized Network Engineering Tool) for network based simulations. Usually references in literature and this dissertation to OPNET implementation, refer to the implementation in modeler. The modeler provides a quick and graphical means to configure, simulate and analyze a network. Various libraries are provided with the modeler which support wireless nodes and ad hoc protocols.

Although, OPNET is not an open source software, various editors are available to modify or create new node or protocol behaviors. A typical network in the modeler comprises of nodes, their interconnections and applications defined for the nodes. These network entities are created using the network editor*. The behavior of the node is based on its node model which can be edited using the node editor. The node model is defined by process blocks and their interactions with each other. These interactions may comprise of data or statistics exchanged amongst the processes. For example, a wireless client's node model will have a transmitter and a receiver process connected to a MAC (Media Access Control) process. The process blocks represent various packet processing hardware or software elements. The behavior of the process is defined by its process model, which can be modified by using the* process editor*. The process is modelled as Finite State Machine and is represented by state diagrams. These models are coded using* Proto-C *which is a modified version of C programming language that contains different OPNET APIs (Application Programming Interfaces). The code executed in a state or during state transitions can be easily changed or added by the users, thus implementing new or modified protocols and processes. The process model is compiled using a C compiler at the beginning of the simulation. The processes*

are capable of creating sub-processes or child processes. This hierarchy of editors provided by OPNET makes it easier to implement any desired node characteristics without parsing through unnecessary details. The parameters from the process can be promoted so that it can be modified as a node attribute, which can be easily changed for a different simulation scenarios.

The Discrete Event Simulator *forms the core simulation engine of the modeler. In this, all simulation events are arranged in discrete time sequences and are executed accordingly. The simulator keeps track of new packets and inserts and modifies events during simulation. The simulation code comprises of simulation kernel along with different object files that define the behavior of nodes, processes, transmission links and other network elements. The simulation supports multiple iteration scenarios wherein some network parameters are changed iteratively to study its effect on the network performance. In order to model the random nature of phenomena, such as packet generation and bit errors, a random number generator is incorporated in OPNET. It can be initialized with different* seed *values. The simulation parameters also include a vector and scalar file in which simulation parameter values are recorded. The parameters to be monitored are specified prior to the simulation. A user specified statistic can also be defined and collected in these files. The parameters in these files can later be plotted using* Analysis Tool *in OPNET. The modeler also includes an excellent command line debugging tool called OPNET Simulation Debugger (ODB).*

5.1.2 Sophisticated Node Model

The network to be simulated consist of two types of nodes; namely SNs and LNs. The LNs can easily be implemented using the mobile MANET nodes available from the MANET

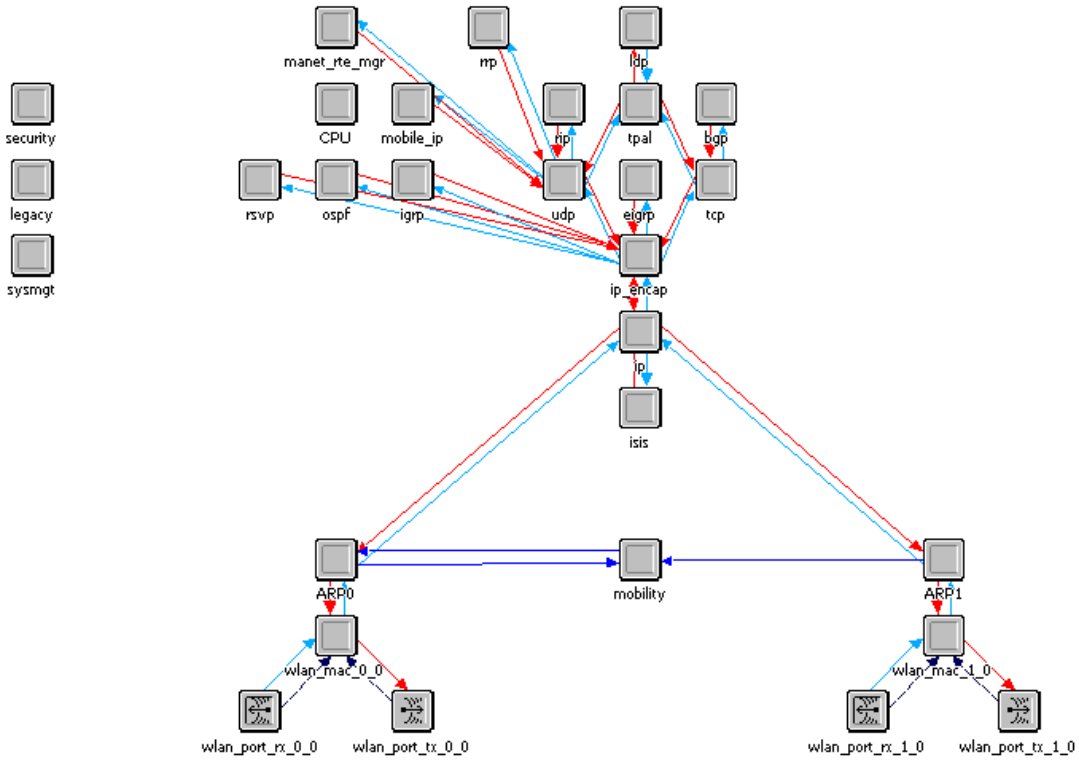


Figure 5.1: Sophisticated node model

object palette in modeler. However, the SNs are nodes with two interfaces; one to collect data from LNs and other to forward it to the sink via neighboring SNs. We use Device Creator *to build such a node. It is used to create different types of network components not available with the built-in object library. Derived models created using* Device Creator *inherit the parent model properties. We use it to create a dual WLAN interface node that will act as SN. Although both the LN and SN interfaces are 802.11 based, they are distinguished by different transmission power and channel assignments. The channel parameter assignments used in our simulation are discussed in the next section.*

The SN node model is shown in Figure 5.1. As seen from the figure there are two pairs of transmitter-receiver interfaces for LN and SN communications. The interface 0 is considered to be an SN interface while the interface 1 is considered as LN interface. Note that SN node will have these interfaces to respectively communicate with neighboring SNs and LNs. The MAC and ARP (Address Resolution Protocol) process models are independent for both the interfaces. The higher layer stack, such as IP (Internet Protocol), TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) models, are same as those in the built-in WLAN node. For the purpose of SN mobility management a new "mobility" process module is added. This module needs inputs from both SN and LN interfaces. The ARP module for both these interfaces is modified to notify mobility process of any packets received with the sink's destination address (assumed to be fixed and known) along with its hop count. The mobility process monitors this information to determine if SN needs to be moved and the intended destination. Along with passive monitoring through ARP processes, the mobility process is capable of exchanging messages with its neighboring SNs. These messages allow SNs to make decisions to move in a distributed manner without losing connectivity with the sink. The messages are generated by the mobility process module and passed to SN interface's ARP module for transmission. Similarly when such messages are received at a SN, the ARP forwards it to the mobility process instead of the IP process.

The SN message format is shown in Figure 5.2. The FD_DST_ID and FD_SRC_ID fields contain the IP addresses of destination and source SN nodes and are 32 bits long. For ease in simulation and without loss of generality, the node IDs (OPNET object IDs) are used instead of IP addresses in this field. The FD_PKT_TYPE defines the type of message to be conveyed and is 4 bits long. The FD_FIELD is a structure that is used to send additional

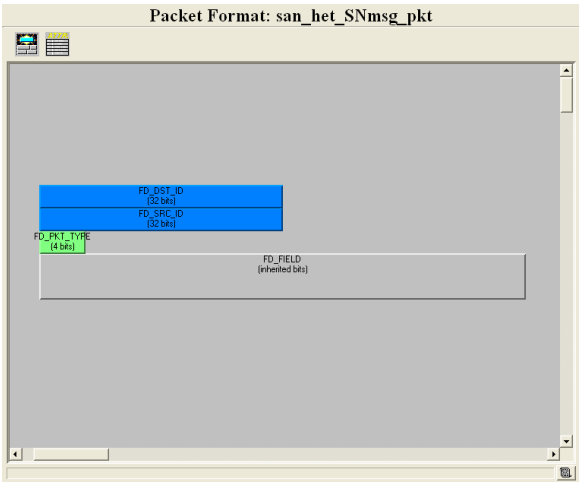


Figure 5.2: SN message format

information for some message types. The size of this field is variable since it is set based on the amount of data to be sent. The message types are described as follows:

SN_msg_intent *This message is sent to announce the intention of the SN to move from its current location to another location. The FD_FIELD holds the intended destination information and the cost-benefit (explained later) associated with the movement. This message is sent only to in-nodes*[2] *and out-nodes*[3] *that will be affected by the intended movement. The SN sending this message and thus requesting node movement is referred to as the* requesting SN.

SN_msg_MvOk *This message is sent after a* SN_msg_intent *is received from an in-node and the current SN is okay with the movement of the neighboring SN. This message indicates that the current SN has found some neighboring SN which is in the vicinity of the intended position of the requesting SN. It then sends this response along with*

[2] *Neighboring SNs whose depend on this SN to forward their traffic to sink*
[3] *Neighboring SN that forwards this nodes traffic to sink*

the ID of the neighboring SN in the FD_FIELD. This message is also sent when the node itself decides to move in response to the requesting SN's SN_msg_intent*. In this case, FD_FIELD contains its own ID.*

SN_msg_MvOkRes *This message is sent after a* SN_msg_intent *is received from an out-node and the current SN needs to check with the requesting node about few candidate out-nodes. This message is generated when multiple alternate paths are available to the sink. However, these paths may depend on the requesting SN in order to reach the sink and thus have to be checked with the requesting SN. FD_FIELD in this message holds the candidate out-node's IDs of different alternative paths. This enables the requesting out-node to select the appropriate path for this SN. The number of out-nodes sent with this request is defined by a variable* SNquerryTbl_max_relObjid*, which is fixed to 2 for this simulation by trial and error.*

SN_msg_MvNotOk *This message indicates that the SN cannot allow the requesting SN to move. Typically this occurs when the SN is already processing another movement request. The FD_FIELD is empty in this case.*

SN_msg_confirm *This message is sent by the requesting SN after it has made sure that its neighboring in-nodes and out-nodes have a candidate out-node to reach the SN after its intended movement. The respective candidate out-node is indicated in the FD_FIELD. This also initiates a cascade movement in which multiple SNs may be moving. After sending this message, the SN initiates the intended movement.*

The locations of neighboring SNs is assumed to be known. The localization of SNs can be carried out by using any one of location estimation schemes proposed for ad hoc

networks as discussed in [44]. However, the proposed mobility scheme uses just the relative locations of its neighbors and thus absolute location information is not needed. The SN is also assumed to have a directional antenna which can be used to know the direction along which the neighboring LN lies. Moreover, by measuring the received signal strength of LN packets, the SN can have a good estimate of its distance to the LN (assuming open field deployment). Thus, the SN can estimate the relative location of its neighboring SNs and LNs. Note that the LNs are not localized, as accurate localization of LNs is not needed and would deplete the energy of the resource constraint LNs. In order to simplify the node model for simulation purposes, the distance and directions are calculated using LN coordinates in our OPNET implementation. The mobility process is explained in the next subsection in detail.

5.1.3 Mobility Process Model

The SN mobility scheme is implemented as a distributed scheme which aims to satisfy the two constraints described for the hierarchical node placement problem in the previous chapter, i.e. handling LN data traffic and maintaining connectivity to sink. The mobility module is shown in Figure 5.3 as a state diagram. The mobility scheme is described next based on the different states in the figure.

"init" State

This is the initial state of the node with a "begin simulation" interrupt. The process is in this state only at the start of the simulation. The data tables to store the information regarding neighboring SNs and LNs are initialized in this state. We maintain three tables in this mobility process; they are described in Tables 5.1, 5.2 and 5.3.

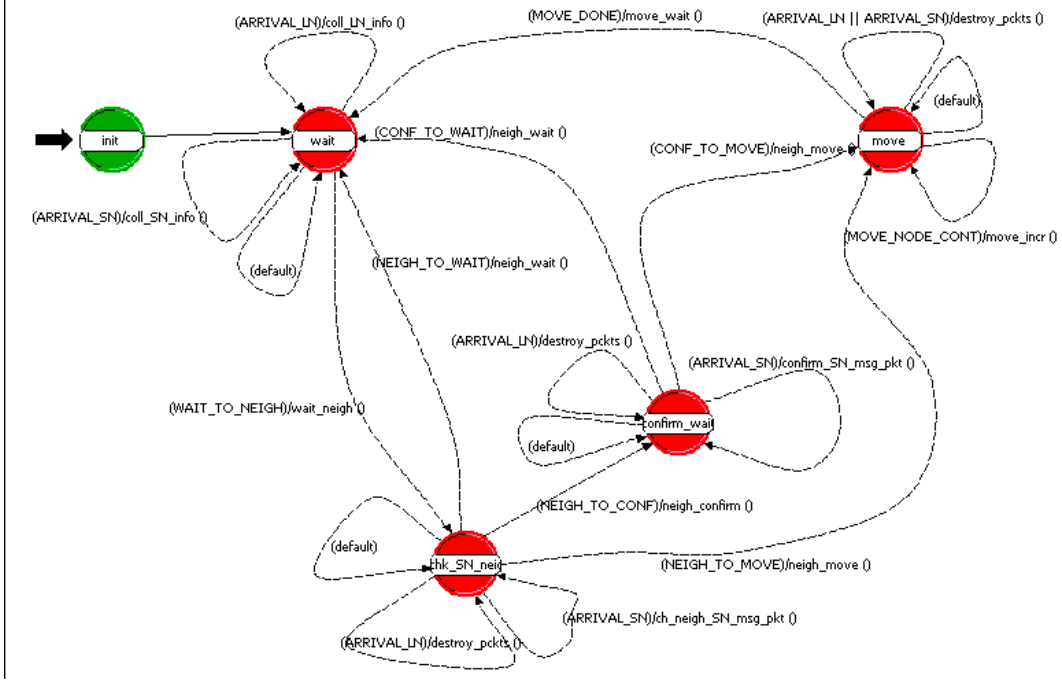


Figure 5.3: Mobility process model

OPNET Data Type	***Table Parameters***	***Description***
Objid	*fwd_mod_id*	*Forwarding node ID*
char	*LN_name[20]*	*LN station name*
double	*x_pos,y_pos,z_pos*	*Position cordinates of LN*
int	*n_pkts, n_hops*	*Total number of packets and hops respectively*
double	*avg_hops*	*Average number of hops*
double	*mv_req_cost*	*Cost to move the node*

Table 5.1: Neighbor LN statistics table (T-LN). It is used to store neighboring LN statistics.

OPNET Data Type	***Table Parameters***	***Description***
Objid	*fwd_mod_id*	*Forwarding node ID*
char	*SN_name[20]*	*SN station name*
char	*node_ip_addr[20]*	*IP address*
int	*status*	*Status of the SN (neigh, in-node, out-node)*
double	*x_pos,y_pos,z_pos*	*Position cordinates*
Boolean	*comp_entry*	*True if the entry are complete from received packet info*
int	*n_pkts, n_hops*	*Total number of packets and hops respectively*
double	*avg_hops*	*Average number of hops*
int	*Nreq*	*Number of intent to move messages*
double	*To_x, To_y, To_z*	*The new position coordinates after intended movement*
double	*mv_req_cost*	*Cost to move to this requested position*

Table 5.2: Neighbor SN statistics table (T-SN).It is used to store neighboring SN statistics.

OPNET Data Type	***Table Parameters***	***Description***
int	*corr_SN_table_id*	*The corresponding T-SN entry ID*
Objid	*node_objid*	*SN node ID*
int	*status*	*Status of the SN (neigh, in-node, out-node)*
int	*msg_response*	*Response to the SN_msg_intent*
Objid	*related_objid [SNqueryTbl_max_relObjid]*	*Related Ids from response*
Objid	*candidate_objid*	*Candidate SN to be sent with* SN_msg_confirm
Boolean	*mv_ok*	*Is it okay to move ?*
double	*time_stamp*	*Last updated*

Table 5.3: Query Table (T-Q). It is used to store neighboring SN responses to intended movement request.

This state is a forced state*, i.e., it transitions to the next state without waiting for any interrupts. Thus, as soon as the tables are initialized the control moves out to the "wait" state.*

"wait" State

In this state SNs are waiting at a location and forwarding LN and SN packets to the sink. The APR processes of both the interfaces are modified to notify the mobility process about the forwarding node IP address and the number of hops they have traveled to reach this SN. Each LN or SN neighbor is identified by its ID in T-LN and T-SN table respectively. Note that for ease in simulation, without loss of generality, the OPNET node IDs are used instead of actual IP address. When a new ARP notification arrives an ARRIVAL_LN or ARRIVAL_SN interrupt is generated and the respective tables are checked for existing entries for corresponding nodes. If an entry is found the table parameters are updated, else a new entry is created before updating the parameters. Along with the message statistics from neighboring SNs, the movement requests from neighboring SN and its intended destination are also recorded. These requests are passed using SN messaging scheme as discussed in the previous subsection and allows SNs to make distributed mobility decisions.

The SN messages are processed as shown in Figure 5.4. When the SN_msg_intent *is received from an in-node the alternate out-node is searched. This node is any neighbor of the current SN that lies in the vicinity of the intended destination. This is the alternate node in Figure 5.4 and can substitute as out-node after requesting SN moves. Depending on the search result, a* SN_msg_MvOk *or* SN_msg_MvNotOk *is sent as a reply. Note that the mobility request is processed by the* `chk_mob()` *function only if the cost-benefit is higher*

```
// Decipher received messages
switch (pkt_type)
{
    case SN_msg_intent :
        Read FD_FIELD to get intended destination and cost-benefit;
        Find if this node is in-node or out-node from T-SN;
        if (in-node)
        {
            if (alternate node that can serve the requesting SN is present)
                Send SN_msg_MvOk and ID of alternate node;
            else
                Log this request to the token table, create new entry if needed;
                if (cost-benefit from FD_FIELD > corresponding cost-benefit from T-SN)
                  chk_mob();
                else
                  Send SN_msg_MvNotOk;
        }
        else
        {
            // This is an out node and it wants to move
            Send SN_msg_MvOkRes with neighbors that are not in nodes or this node in FD_FIELD;
        }
    break;

    case SN_msg_MvOk :
    case SN_msg_MvOkRes :
    case SN_msg_MvNotOk :
    break;

    case SN_msg_confirm :
        Remove the node from T-SN;
    break;

}
```

Figure 5.4: SN message processing in "wait" state.

than the threshold. A default cost-benefit is associated with all new request and is recorded in the T-SN. Based on this entry, when an unsuccessful request is made to neighboring SNs, the value of cost-benefit is incremented, thus increasing the threshold. The next request will only be considered for mobility if the SN_msg_intent *comes with a higher cost-benefit than this threshold. When the* SN_msg_intent *is received from an out-node, alternate out-nodes are suggested by sending* SN_msg_MvOkRes*. If* SN_msg_confirm *message is received the corresponding entry of the source SN in T-SN is deleted as it indicates that the SN will move to a new location.*

As new packets are received, the corresponding tables are updated and mobility decision is made using `chk_mob()` *function. If the decision to move is triggered by LN traffic it is called LN based mobility decision, otherwise if it is triggered by SN request then its called*

SN based mobility decision. For LN based mobility decisions, T-LN is checked to decide if the benefit of moving the SN to LNs location (avg_hops × LN request weight) is greater than the cost threshold (mv_req_cost). Similarly for SN based mobility decisions, T-SN is checked. If benefit is greater than the cost threshold, the process transitions to "chk_SN_neigh" state, otherwise it stays in this state. The intended destination (dest) is the LN location for an LN based mobility decision. For an SN based mobility decision the intended destination for current SN (dest) is calculated by Equation 5.1,

$$dest = Req_{dest} + (Cur_{loc} - Req_{dest}) \times \frac{SN_{Range}}{Dist(Cur_{loc}, Req_{dest})} \tag{5.1}$$

where, Req_{dest} is the intended destination from the requesting SN, Cur_{loc} is the current location of this SN, SN_{Range} is the SN transmission range and $Dist(Cur_{loc}, Req_{dest})$ is the Euclidean distance between Req_{dest} and Cur_{loc}.

"chk_SN_neigh" State

If the SN decides to initiate the movement from one location to another then it transitions to this state from the "wait" state. This state can be initiated due to a LN or SN based mobility decision. During the transition to this state a query table (T-Q) is built, using T-SN, with entries comprising of the SNs in-nodes and out-nodes. The other neighbors are not considered in this table.

Some of the nodes in T-Q may remain connected to this SN after the intended movement is completed, i.e. they may be within SN_{Range} after the SN reaches dest[4]*. While some other*

[4] *Note that any SN nodes within distance of SN_{Range} of each other can communicate directly with each other.*

nodes may be connected through different neighbors after the SN reaches dest. These are also checked by comparing node distances with SN_{Range}. *For these nodes the* mv_ok *in T-Q is set to* TRUE *and the candidate_objid is set to this SN ID. For all other nodes, a* SN_msg_intent *is sent with the FD_FIELD containing dest.*

The process stays in this state to collect SN message responses from the queried neighbors. On receiving the SN_msg_MvOkRes *message, a candidate out-node from the entries in the FD_FIELD is selected and* candidate_objid *and* mv_ok *in T-Q is updated. After every* SN_msg_MvOkRes *and* SN_msg_MvOk *response that changes* mv_ok *entry to* TRUE*, other neighbors entries in T-Q are checked to estimate their connectivity can be maintained through this responding node.*

If all the entries have mv_ok *set to* TRUE *the process transitions to "move" or "confirm_wait" state depending if this state was initiated due to a LN or SN based mobility decision respectively. Note that for SN based mobility decision, the requesting SN confirmation is required before initiating movement. However, for LN based mobility decision, a move can be initiated if all SN neighbors are okay, i.e* mv_ok *is* TRUE *for all T-Q entries. If a timeout occurs or a* SN_msg_MvNotOk *is received, the process transitions to the "wait" state and clears T-Q.*

"confirm_wait" State

This state is invoked in a SN based mobility decision when all neighbors are okay with SN movement. A SN_msg_MvOk *is sent to the requesting SN with this SN's ID in the FD_FIELD. The objective of this state is to wait till the requesting SN has all its neighbors okay for its intended movement.*

On receiving the SN_msg_confirm *from the requesting SN, the SN sends a* SN_msg_confirm *to its neighbors and the process transitions to "move" state. Otherwise, if a timeout occurs or a* SN_msg_MvNotOk *message from requesting SN is received, the SN sends a* SN_msg_MvNotOk *message to its neighbors, purges table T-Q and moves back to "wait" state.*

"move" State

Actual node movement is implemented in this state. It is done by setting up self-interrupts after small increments in time ($\triangle t$). On receiving the interrupt the Cur_{loc} value of this node is changed slightly towards dest as shown in Equation 5.2.

$$Cur_{loc} = Cur_{loc} + (dest - Cur_{loc}) \times \triangle t \tag{5.2}$$

The speed of SN (v) is used to calculate the time t_f it takes to reach dest as $t_f = \frac{Dist(Cur_{loc}, dest)}{v}$. After SN reaches dest, the tables T-LN and T-SN are purged and the process transitions to "wait" state.

5.2 Results

Using the SN model described in the previous subsection, we create a hierarchical network of SN and LNs in OPNET. The hierarchy in the network is enforced by using two interfaces in the network. A simple interface was used for LN communication amongst each other and with the neighboring SN, and a sophisticated interface was defined for communication amongst SNs and the sink. Since the LN would generate the traffic that is to be delivered to the sink, data will be routed from lower-tier LN nodes and then to the higher-tier

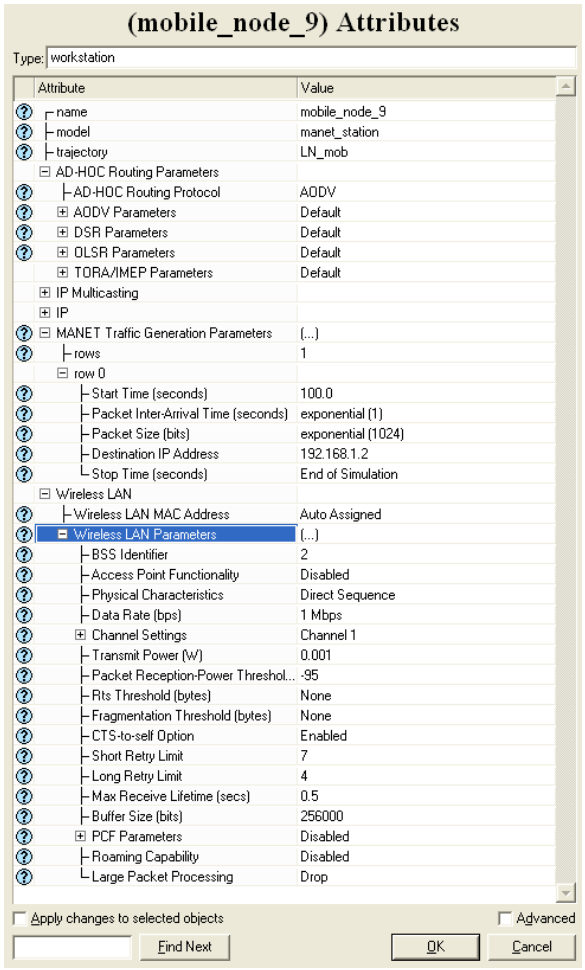


Figure 5.5: LN parameter setting.

SN nodes. AODV protocol [49] is used as the routing algorithm to select packet routes. The LNs and SNs are arranged in different subnets for the simulation.

Although both LN and SN interfaces are 802.11 based interfaces, their parameters are set differently to reflect the difference in LN and SN resources. The LN interface has a maximum data rate of 1 Mbps and a transmission power of 1 mW, while SN interface has these values set to 11 Mbps and 20 mW respectively. The LN are simple MANET nodes and have their traffic configured to be exponential with mean packet size of 1 Kb and mean

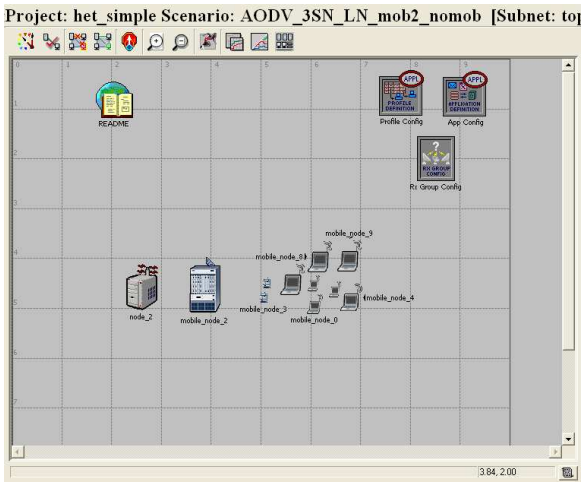


Figure 5.6: Static network with no LN or SN mobility

inter-arrival time of 1 seconds. These LN and SN parameter values can be set using the "Edit Attribute" option for all nodes. As an example, Figure 5.5 shows the parameter set for one of the LNs.

We discuss two sets of experiments to demonstrate the effect of our SN mobility scheme. In both these experiments we consider the IP hop count as the performance metric. Its decrease indicates that the packets from LNs can reach SN with lower number of hops and hence at lower energy. This further indicates an increase in network lifetime. Moreover, the non-zero value of IP hop count indicates that the network is connected, which is an important result when node mobility is considered.

5.2.1 Effect of SN Mobility

In this experiment we assume the network to comprise of static LNs and investigate the performance with and without SN mobility. The above described SN node model is used when mobility is considered. However, to study the effect of no mobility we use the initial

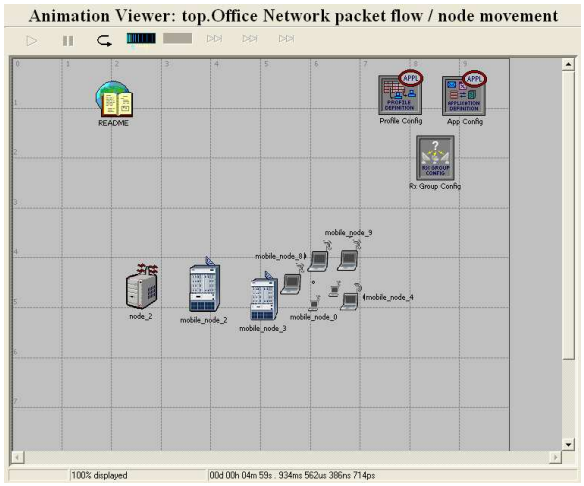
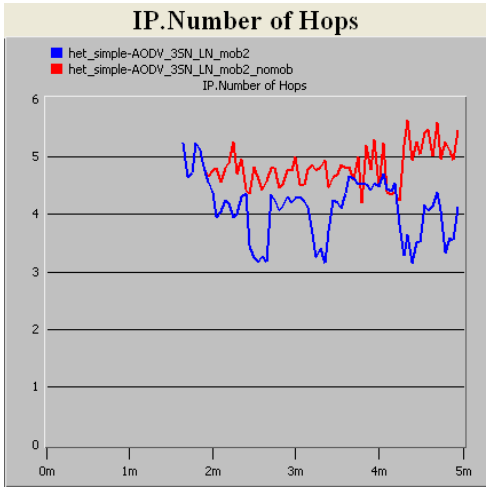


Figure 5.7: Previous network with SN mobility at t=5m

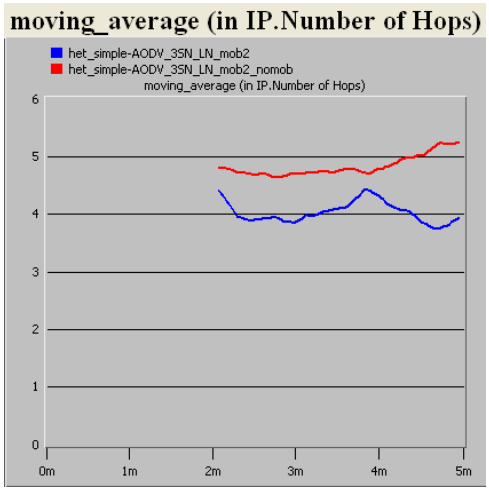
dual interface SN generated using the Device Creator without any modifications. The LNs and SNs at t=0 are arranged as shown in Figure 5.6. The simulation is carried out for 5 minutes. Since no mobility is considered the network at the end of simulation is same as Figure 5.6.

The experiment with the same setup is repeated with our modified SNs which are mobile. The initial network at t=0 is same as the previous case, i.e. Figure 5.6. However, after some time the SNs move towards the LN that forwards packets with greatest average hop counts. This leads to SN movement and at t=5m the resultant network is as seen in Figure 5.7. One of the SN is at the same location as an LN and both of them coexist at a location marked as ◎ in the figure.

The IP hop counts with and without SN mobility are shown in Figure 5.8. While Figure 5.8(a) shows the actual hop count, Figure 5.8(b) shows the average hop count. The average hop count is a moving average with a window size of 50 samples, i.e. the value is an average



(a) Number of IP hop count with and without SN mobility



(b) Average number of IP hop count with and without SN mobility. Moving average with window size=50.

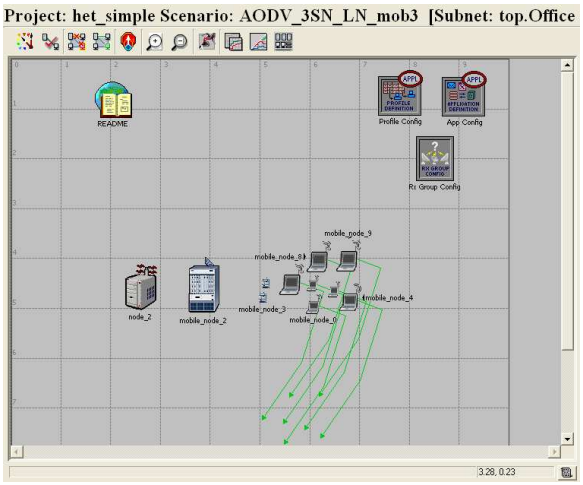
Figure 5.8: Effect of SN mobility on IP hop counts

of previous 50 values. By averaging over time, a clear decrease in the hop count is observed. This indicates that the SN mobility can improve network performance.

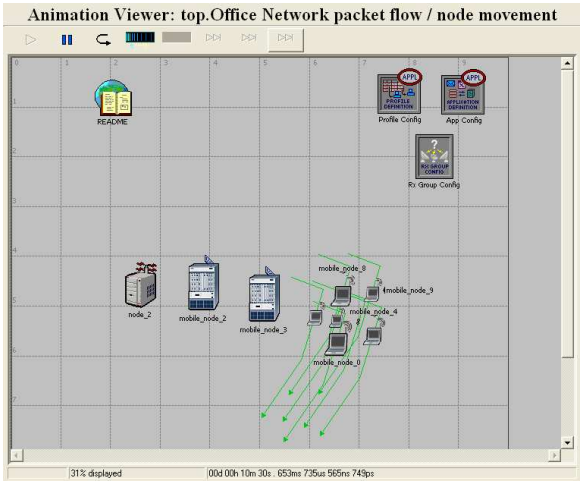
5.2.2 Effect of LN Mobility

Although the previous case indicates that the SN mobility does decrease hop counts in static LN network, it does not demonstrate the effect of LN mobility. Mobility of LN may be caused due to environmental conditions such as wind that may displace sensors. We study the case where LN mobility is enabled.

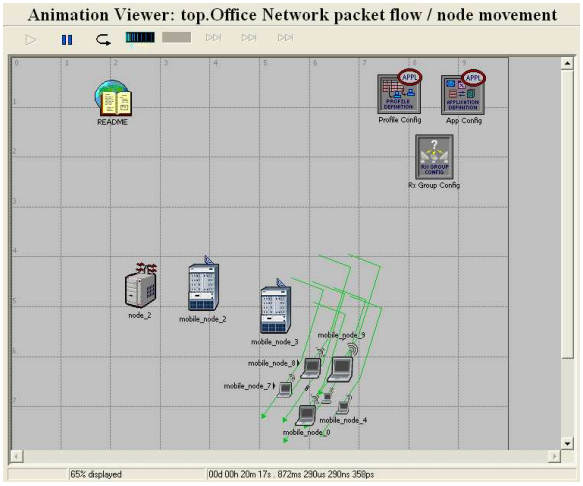
Figure 5.9 shows the network at various time instances. Note that OPNET changes the size of the node automatically depending on the density of the nodes in a region. This is just for clear representation and does not indicate any network related behavior. As mentioned before, if one of the SN is at the same location as an LN, both of them coexist at a location



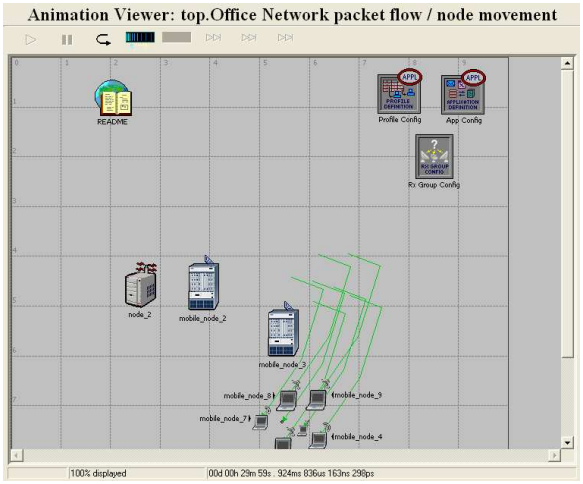
(a) t=0m0s



(b) t=10m30s



(c) t=20m17s



(d) t=29m59s

Figure 5.9: Network with SN and LN mobility

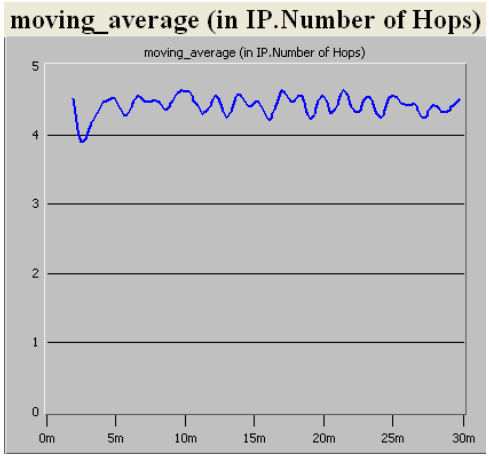


Figure 5.10: Average number of IP hop count with SN and LN mobility case. Moving average with window size=50.

marked as ⊙ in the figure. As observed from Figure 5.9, the SNs successfully track the LNs over time. Note that two SN nodes moved in a distributed manner during the simulation. The average IP hop count is shown in Figure 5.10 which indicates that the network was connected till the end of simulation.

The above two cases validate the correctness of the mobility model. It can be inferred that the mobility scheme is capable of:

1. *Reducing the hop count of the LN traffic and hence conserving nodal energy.*

2. *Maintaining a multi-hop connectivity to the sink.*

The first case demonstrates that without the proposed mobility scheme, more hop counts are required to transmit packets to the sink. For identical network conditions and with our mobility scheme implemented in the SN, the average hop count reduces. Here the hop count metric is used to demonstrate that nodal energy conservation is possible due to our mobility scheme. As the hop count reduces, the number of forwarding nodes reduces. Thus, on an

average the nodes will forward fewer packets and hence conserve their energy. The second case demonstrates the capability of our mobility scheme to react to LN mobility in the network. The hop counts are plotted to indicate continuous LN connectivity to the sink. As the value of hop count is a finite non-zero number, each LN traffic is able to reach the sink via SNs throughout the simulation. Note that in both the cases, all LNs continuously generate packets till the end of simulation. Thus, any change in the connectivity will be reflected in the average hop count metric.

Chapter 6

Conclusion

The dissertation investigates the problem of localization and placement in wireless networks. A secure localization scheme is described and implemented in Chapter 2 considering an infrastructure based WLAN. Chapter 3 describes the PISA placement algorithm to place bridging devices with minimal interfaces in a heterogeneous network. The placement algorithm for hierarchical networks are described in Chapter 4 and a HYBRID algorithm is proposed for the same. Furthermore, in Chapter 5 a mobility scheme is proposed for sophisticated nodes in a mobile ad hoc network to minimize average hop count and maintain network connectivity.

The proposed low-cost secure localization scheme for WLAN networks, described in Chapter 2, is shown to have similar or better performance as compared to the traditional signal strength (SS) based schemes. In this work, our focus has been on a very low cost solution and hence we have not considered more complicated threat models. Note that typically more complicated threat models arise when the benefits to the adversary are correspondingly larger. A localization system in a Starbucks that lets only the patrons connect to the wireless network would not be of interest to complex (and rational) attackers. What is more important here is a low cost solution that works nearly all the time. In this case, the costs of failure are also not catastrophic. But we believe that the current set of techniques can be extended to protect against more complex attackers using techniques mentioned in Section 2.5. We plan to investigate this in the future. Additionally, in the future, we also plan to look at efficient AP placement as well as at efficient methods to build up the "message

map" (lookup table). As mentioned previously, one method to do this is to leverage the various desktops as proposed in [4]. Issues such as preventing unnecessary handoffs between APs during localization also need to be addressed. In the current implementation we have achieved this by increasing the interval used by the APs to transmit the beacons to 4 seconds, much larger than the time needed to complete localization.

Chapter 3 discusses the use of drones (multi-interface devices) to maintain connectivity in a heterogeneous network. The placement and interface selection for these drones is obtained by PISA. PISA applied to different test cases results in desirable solutions. Implementing multiple clustering iterations in the coarse placement step improved the resultant solution. Also, if PISA is given more than the required number of drones to be placed, it returns the excessive drones with zero interfaces. Thus, PISA solutions result in minimum number of drones with minimum number of interfaces on them. As future work, we plan to extend genetic algorithms to automatically determine the number of clusters, k, for the coarse placement step and the number of drones, p, for the refined placement step. Various combinatorial algorithms can be investigated in the future to reduce the execution time for the refined placement step. The proposed work is useful across a wide spectrum of applications dealing with heterogeneous networks.

The problem of SN placement for a two-tiered hierarchical heterogeneous sensor network is presented in Chapter 4. The problem formulation considers many design parameters including LN traffic and SN connectivity. We have compared the performance of the BILP, GREEDY and GA algorithms and found that GA provides better results for randomly distributed LNs. A HYBRID approach that uses the BILP and GREEDY results to seed GA was also proposed. It was observed that HYBRID approach has an execution time of only

11.46% of that of the original GA. It was also shown that HYBRID_{b} *is statistically better than GA when all iterations are considered. After a threshold load over an SN is reached, new SN locations can be calculated and the SNs will be accordingly repositioned. A mobility scheme that considers such an implementation is considered in Chapter 5. The OPNET implementation of the scheme is described using the finite state diagram. The test cases demonstrate the mobility scheme's capability to reduce average hop count in the network and maintain connectivity to sink when LN mobility is considered. Due to the practical difficulties in having mobile SNs, a hybrid SN deployment consisting of some stationary and mobile SNs should be considered in future.*

Bibliography

[1] *Exploratory research: Heterogeneous sensor networks, 2004.*

[2] *K. Akkaya and M. Younis. A survey of routing protocols in wireless sensor networks.* Elsevier Ad Hoc Network Journal, *3:325–349, 2005.*

[3] *R. Anderson and M. Kuhn. Tamper resistancea cautionary note.* Second USENIX Workshop on Electronic Commerce Proceedings, *pages 1–11, November 1996.*

[4] *P. Bahl, J. Padhye, L. Ravindranath, M. Singh, A. Wolman, and B. Zill. DAIR: A framework for managing enterprise wireless networks using desktop infrastructure.* HotnetsIV, *November 2005.*

[5] *P. Bahl and V. N. Padmanabhan. RADAR: An in-building rf-based user location and tracking system.* Proceedings of IEEE INFOCOM, *2:775–784, March 2000.*

[6] *R. B. Bapat, I. Gutman, and W. Xiao. A simple method for computing resistance distance.* Zeitschrift für Naturforschung: A Journal of Physical Sciences, *58a(3):494–498, 2003.*

[7] *J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus. Deploying sensor networks with guaranteed capacity and fault tolerance. In* MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, *pages 309–319, 2005.*

[8] *P. Castro, P. Chiu, T. Kremenek, and R. Muntz. A probabilistic room location service for wireless networked environments.* Proceedings of the 3rd international conference on Ubiquitous Computing, *pages 18 – 34, September 2001.*

[9] *K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks.* IEEE Transactions on Computers, *51(12):1448–1453, 2002.*

[10] *J. Chamberland. A first look at the hidden cost of random node placement in wireless sensor networks. In* EEE/SP 13th Workshop on Statistical Signal ProcessingIEEE/SP 13th Workshop on Statistical Signal Processing, *pages 1164–1169, July 2005.*

[11] *X. Cheng, D. Du, L. Wang, and B. Xu. Relay sensor placement in wireless sensor networks, 2001.*

[12] *Cisco.* Aironet 1100 Series Access Point Installation and Configuration Guide, *2005.*

[13] *D-Link.* DWL-2100AP Data Sheet, *June 2005.*

[14] *D. E. Denning and P. F. MacDoran. Location-based authentication: Grounding cyberspace for better security.* Computer Fraud and Security, Elsevier, *pages 12–16, February 1996.*

[15] *R. M. D'Souza, S. Ramanathan, and D. T. Lang. Measuring performance of ad hoc networks using timescales for information flow.* Proceedings of IEEE INFOCOM, *2:1564 – 1574, April 2003.*

[16] *D. B. Faria and D. R. Cheriton. No long-term secrets: Location-based security in overprovisioned wireless lans.* ACM HOTNETSIII, *November 2004.*

[17] *S. Ganu, A. S. Krishnakumar, and P. Krishnan. Infrastructure-based location estimation in wlan networks.* IEEE Wireless Communications and Networking Conference (WCNC 2004), *1:465–470, March 2004.*

[18] *D. E. Goldberg.* Genetic Algorithms in Search, Optimization, and Machine Learning. *Addison-Wesley Professional, 1989.*

[19] *U. C. Guard. Navstar GPS user equipment introduction (public release version). Technical report, September 1996.*

[20] *G. Gupta and M. Younis. Fault-tolerant clustering of wireless sensor networks. In* Proceeding of IEEE WCNC, *pages 1579–1584, 2003.*

[21] *Y. Gwon, R. Jain, and T. Kawahara. Robust indoor location estimation of stationary and mobile users.* Proceedings of IEEE INFOCOM, *2:1032–1043, March 2004.*

[22] *J. Handl and J. Knowles. Multiobjective clustering with automatic determination of the number of clusters. Technical Report TR-COMPSYSBIO-2004-02, UMIST, Manchester, August 2004.*

[23] *J. Hightower and G. Borriello. Location systems for ubiquitous computing.* IEEE Computer, *34(8):57–66, August 2001.*

[24] *Y. Hou, Y. Shi, H. Sherali, and S.F.Midkiff. On energy provisioning and relay node placement for wireless sensor network.* IEEE Transactions on Wireless Communications, *4(5):2579–2590, 2005.*

[25] *C. Houck, J. Joines, and M. Kay. A genetic algorithm for function optimization: A matlab implementation. Technical Report NCSU-IE TR 95-09, NCSU, 1995.*

[26] *D. Hromin, M. Chladil, N. Vanatta, D. Naumann, W. Wetzel, F. Anjum, and R. Jain. CodeBLUE: A bluetooth interactive dance club system.* IEEE Global Telecommunications Conference, *5:2814 – 2818, December 2003.*

[27] I. ILOG. Ilog cplex 7.5. Mountain View, California, July 2006.

[28] IST-2000-25382-CELLO. Cellular network optimisation based on mobile location: Final report. Technical Report CELLO-WP1-VTT-D36-010-Apr, http://www.vtt.fi/tte/tte35/pdfs/IST-CELLO-Final-Report.pdf, February 2004.

[29] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". In Mobicom*, August 1999.*

[30] M. G. Kuhn. An asymmetric security mechanism for navigation signals. Proceedings of the Information Hiding Workshop*, 2004.*

[31] L. Lazos, S. Capkun, and R. Poovendran. ROPE: Robust position estimation in wireless sensor network. Proceedings of IPSN*, April 2005.*

[32] L. Lazos and R. Poovendran. SeRLoc: Secure range-independent localization for wireless sensor networks. Proceedings of the 2004 ACM Workshop on Wireless Security, WiSe*, pages 21–30, October 2004.*

[33] N. Li and J. C. Hou. Topology control in heterogeneous wireless networks: problems and solutions. Proceedings of IEEE INFOCOM*, 1:232–243, March 2004.*

[34] W. Li and C. G. Cassandras. A minimum-power wireless sensor network self-deployment scheme. IEEE Wireless Communications and Networking Conference*, 3:1897–1902, March 2005.*

[35] Z. Li, W. Trappe, Y. Zhang, and B. Nath. Robust statistical methods for securing wireless localization in sensor networks. Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, April 2005.*

[36] K. Lieska, E. Laitinen, and J. Lahteenmaki. Radio coverage optimization with genetic algorithms. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, 1:318 – 322, September 1998.*

[37] H. Liu, P. Wan, and X. Jia. On optimal placement of relay nodes for reliable connectivity in wireless sensor networks. Journal of Comb. Optim.*, 11:249–260, 2006.*

[38] D. Madigan, E. Elnahrawy, R. P. Martin, W.-H. Ju, P. Krishnan, and A. S. Krishnakumar. Bayesian indoor positioning systems. Proceedings of IEEE INFOCOM*, 2:1217–1227, March 2005.*

[39] P. Maulin, R. Chandrasekaran, and S. Venkatesan. Energy efficient sensor,relay and base station placements for coverage, connectivity and routing. In 24th IEEE International Performance, Computing, and Communications Conference*, pages 581–586, April 2005.*

[40] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless adhoc sensor networks. Proceedings of IEEE INFOCOM*, 3:1380–1387, April 2001.*

[41] A. Molina, G. E. Athanasiadou, and A. R. Nix. The automatic location of base-stations for optimised cellular coverage: A new combinatorial approach. IEEE Vehicular Technology Conference*, 1:606 – 610, May 1999.*

[42] Motorola. Save time and money and reduce uncertainty in your wireless network planning process. Technical report, Motorola, 2006.

[43] T. Nadeem, L. Ji, A. Agrawala, and J. Agre. Location enhancement to IEEE 802.11 DCF. Proceedings of IEEE INFOCOM*, 1:651–663, March 2005.*

[44] S. Pandey and P. Agrawal. A survey on localization techniques for wireless networks. Journal of the Chinese Institute of Engineers*, 29(7):1125–1148, November 2006.*

[45] S. Pandey, B.Kim, F. Anjum, and P. Agrawal. Client assisted location data acquisition scheme for secure enterprise wireless network. IEEE Wireless Communications and Networking Conference, WCNC*, 2:1174–1179, March 2005.*

[46] S. Pandey, S. Dong, P. Agrawal, and K. Sivalingam. Node placement algorithms. Technical report, http://www.eng.auburn.edu/ pandesg/pub/NPlaceAlgo.pdf, August 2006.

[47] S. Pandey, S. Dong, P. Agrawal, and K. Sivalingam. A hybrid approach to optimize node placements in hierarchical heterogeneous networks. In Proceedings of IEEE Wireless Communications and Networking Conference*, March 2007.*

[48] S. Pandey, P. Prasad, P. Sinha, and P. Agrawal. Localization of sensor networks considering energy accuracy tradeoffs. Collaborative Computing: Networking, Applications and Worksharing, 2005 International Conference on*, pages 1–10, December 2005.*

[49] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.*

[50] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, pages 32–43, August 2000.*

[51] Proxim Corporation, CA. ORiNOCO AP-2000 User Guide*, 2004.*

[52] C. Raghavendra, K. Sivalingam, and T. Znati, editors. Wireless Sensor Networks*. Springer Publishers, Boston, MA, 2004.*

[53] *L. Raisanen and R. M. Whitaker. Multi-objective optimization in area coverage problems for cellular communication networks: Evaluation of an elitist evolutionary strategy.* Proceedings of the ACM Symposium on Applied Computing, *pages 714–720, March 2003.*

[54] *F. Ramsey and D. Schafer.* The Statistical Sleuth. *Pacific Grove, CA, 2 edition, 2002.*

[55] *N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims.* Proceedings of the Workshop on Wireless Security, *pages 1–10, 2003.*

[56] *C. Savarese, J. M. Rabaey, and J. Beutel. Locationing in distributed ad-hoc wireless sensor networks.* Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, *4:2037–2040, May 2001.*

[57] *S.Capkun and J. Hubaux. Secure positioning of wireless devices with application to sensor networks.* Proceedings of IEEE INFOCOM, *3:1917–1928, March 2005.*

[58] *S.Capkun, M. Srivastava, M. Cagalj, and J. Hubaux. Securing positioning with covert base stations. Technical report, NESL-UCLA Technical Report, March 2005.*

[59] *S. Shakkottai, R. Srikant, and N. Shroff. Unreliable sensor grids: Coverage, connectivity and diameter. In* Infocom, *March 2003.*

[60] *P. Sinha. Auto-configuration in multi-hop mobile ad-hoc networks. Master's thesis, Auburn University, 405 Broun Hall, Auburn, AL 36849, 2007.*

[61] *M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques.* In KDD Workshop on Text Mining, *2000.*

[62] *J. Suomela. Computational complexity of relay placement in sensor networks. In* Lecture Notes in Computer Science, *volume 3831, pages 521–529. Springer-Verlag, January 2006.*

[63] *J. Tang, B. Hao, and A. Sen. Relay node placement in large scale wireless sensor networks.* Journal of Computer Communications, Special Issue on Sensor Networks, *29(4):490–501, 2006.*

[64] *P. Tao, A. Rudys, A. M. Ladd, and D. S. Wallach. Wireless LAN location-sensing for security applications.* Proceedings of the Workshop on Wireless Security, *pages 11–20, September 2003.*

[65] *M. Turon. Mote-view: A sensor network monitoring and management tool. In* The Second IEEE Workshop on Embedded Networked Sensors, *pages 11–18, March 2005.*

[66] *C. L. Valenzuela. A simple evolutionary algorithm for multi-objective optimization (seamo).* Proceedings of the 2002 Congress on Evolutionary Computation, *1:717 – 722, May 2002.*

[67] *Q. Wang, K. Xu, H. Hassanein, , and G. Takahara. Minimum cost guaranteed lifetime design for heterogeneous wireless sensor networks. In* Proc. IEEE IPCCC*, pages 599–604, April 2005.*

[68] *Q. Wang, K. Xu, G. Takahara, and H. Hassanein. Locally optimal relay node placement in heterogeneous wireless sensor networks. In* IEEE Global Telecommunications Conference*, volume 6, pages 3549–3553, December 2005.*

[69] *R. Want, A. Hopper, V. Falco, and J. Gibbons. The active badge location system.* ACM Transactions on Information Systems*, 10(1):91–102, January 1992.*

[70] *D. B. West.* Introduction to Graph Theory*. Prentice Hall, 2 edition, 2000.*

[71] *Y. Xin, T. Guven, and M. A. Shayman. Topology design for wireless ad-hoc networks with backbone support. In* The IEEE Conference on Information Sciences and Systems*, March 2005.*

[72] *K. Xu, Q. Wang, H. Hassanein, and G. Takahara. Optimal wireless sensor networks (wsns) deployment: Minimum cost with lifetime constraint. In* IEEE International Conference on Wireless And Mobile Computing, Networking And Communications*, pages 454–461, 2005.*

[73] *M. Youssef and A. Agrawala. The Horus WLAN location determination system.* Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218, June 2005.*

[74] *V. Zeimpekis, G. M. Giaglis, and G. Lekakos. A taxonomy of indoor and outdoor positioning techniques for mobile location services.* ACM SIGecom Exchanges*, 3(4):19–27, 2003.*

[75] *P. Zhu and R. C. Wilson. A study of graph spectra for comparing graphs.* British Machine Vision Conference*, 2005. http://www.bmva.ac.uk/bmvc/2005/papers/162/bmvc2005b.pdf.*