Abstract

Bhargava, Vishal. Localization for Intrusion Detection in Wireless Local Area Networks. (Under the guidance of Prof.Mihail Sichitiu) Keywords: Wireless LAN, Localization, Security, 802.11b, parking lot attack, active attacker.

On a wired network, authentication is implicitly provided by physical access; if a user is close enough to the network to plug in a cable, he must have gotten past the receptionist at the front door. While this is a weak definition of authentication, and one that is clearly inappropriate for high-security environments, it works reasonably well as long as the physical access control procedures are strong. Wireless Networks are attractive, in large part, because physical access is not required to use network resources. Therefore, a major component of maintaining network security is ensuring that stations attempting to associate with the network are allowed to do so. In this paper, we present a method to identify the location of wireless LAN users, and thus defend a "parking lot" attack.

LOCALIZATION FOR INTRUSION DETECTION IN WIRELESS LOCAL AREA NETWORKS

BY

Vishal Bhargava

A THESIS SUBMITTED TO THE GRADUATE FACULTY OF

NORTH CAROLINA STATE UNIVERSITY

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

RALEIGH

Aug 2003

APPROVED BY:

Dr. Mihail Sichitiu

CHAIR OF ADVISORY COMMITTEE

Dr. Peng Ning

Dr. Wenye Wang

Biography

Vishal Bhargava is a graduate student at the Department of Electrical and Computer Engineering, North Carolina State University. He commenced his education at Porbandar, the hometown of Mahatma Gandhi. His schooling years were spread all across India, over ten schools during twelve years, before finally joining the prestigious Indian Institute of Technology, Bombay. He graduated with a Bachelor of Technology in Electrical Engineering, in 2001, before joining North Carolina State University for a Master of Science in Computer Networking. He is currently working towards the completion of his degree.

Acknowledgments

I would like to thank Dr. Mihail Sichitiu, my advisor and committee chairman, for his support, encouragement, and invaluable guidance throughout the course of this work. I would also like to thank my colleagues Xinbing, Wei, Lee and Amit for the great time we spent in the lab. I would like to thank my roommates Siva, Gaurav, Alok and Girish for being my family away from home.

Last but not the least, I express my deep gratitude to my family for moulding me into the person I am today with their constant love, support and guidance.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction and Motivation

1.1 Wireless LANs

Wireless LANs are fast becoming the networks of choice. With the availability of cheap wireless solutions, they are seeing exponential growth which is expected to continue in the years to come. The lack of cables makes Wireless LAN easy to install for LAN administrators and at the same time offer tremendous portability for the users. Users are free to move about in the officespace without getting disconnected from the network. The bandwidth offered by wireless LANs is sufficient for most user needs. This kind of portability at a reasonable price, without a noticeable drop in bandwidth, has been mainly responsible for the growing popularity of WLANs.

1.2 Security

With the advent of wireless LANs, arises the issue of security in a wireless environment. Security in WLANs takes centerstage, because of its inherent broadcast mechanism. Every packet that is transmitted in the air can theoretically be captured by any receiver in its range. Most WLANs do offer some form of security, but as their popularity has increased, a host of flaws have been identified both in the standards, as well as in the implementation, of the standards [2–5].

1.2.1 Wired Equivalence Privacy (WEP)

WEP, which is 802.11's optional encryption standard that most radio network interface card (NIC) and access point vendors support, has serious vulnerabilities as exposed by [3] and confirmed by the implementation of [5]. WEP employs RC4 encryption, which is shown to be vulnerable [3]. WEP uses a static 40-bit or 128-bit key for encryption, which has to be manually configured on the NIC as well as the access point, which makes it cumbersome to change periodically. It concatenates the shared secret key supplied by the user with a random 24-bit initialization vector(IV), to generate the seed. The resulting seed is then used by a pseudo-random number generator to generate a key stream of the length of the frame's payload, plus a 32-bit integrity check value ICV). The keystream is then XOR-ed bitwise with the payload to produce the encrypted data (ciphertext). The receiver, who has the same key, XORs the ciphertext to retrieve the original plaintext. The IV is included by WEP in the cleartext, which is used by the receiving station to decrypt the payload portion

of the frame body.

This mode of operation makes stream ciphers extremely vulnerable to attacks. A bit flipped by the attacker in the ciphertext leads to the corresponding bit in the plaintext to being flipped, too. Also, if an eavesdropper intercepts two ciphertexts encrypted with the same key stream, it is possible to obtain the XOR of the two plaintexts. The knowledge of plaintexts XOR enables statistical attacks to recover the plaintexts. The statistical attacks become increasingly practical as more ciphertexts that use the same key stream are known. Once one of the plaintext becomes known, it is trivial to recover all of the others. The short IVs and the keys that remain static are the main reasons for WEP's vulnerability. With only 24 bits, WEP eventually uses the same IV for different packets, and the recurrence of IVs increases with the size of the network. If a hacker manages to collect a sufficient number of frames based on the same IV, he can possibly determine the shared values among them, and the shared secret key or the keystream. Consequently, the hacker achieves the ability to decrypt all of the 802.11 frames.

Using a 40-bit or a 128-bit key does not make a big difference when it comes to these vulnerabilities. The IEEE 802.11 workgroup developed a new mechanism, WEP2, to overcome the shortcomings of WEP; but that too did not prove to be as strong as expected [6]. WEP-creacking tools are freely available on the Internet [7, 8].

1.2.2 802.1X

The failure of WEP triggered the development of new technologies, like 802.1X, by the Standards Committee. The use of 802.1X offers some advantages. It provides for an effective framework for authenticating and controlling user traffic to a protected network, as well as dynamically varying encryption keys. 802.1X ties EAP(Extensible Authentication Protocol) to both the wired, as well as the Wireless LAN, while supporting multiple authentication methods like digital certificates, tokens, Kerberos or one-time passwords. Once authenticated through 802.1X, a dynamic WEP key exchange system can be setup. In a typical 802.1X implementation, the client is expected to change the encryption key often, to minimize the amount of time eavesdroppers may have in order to crack the current key. 802.1X has its own vulnerabilities, as shown by [9]. One-way authentication algorithms, like the EAP-MD5 authentication, leave client authentication exposed to man-in-the-middle attacks.

1.2.3 MAC Address Filtering

Another common misconception is that the MAC address filtering feature offered by many access point vendors offers some level of security. To implement MAC address filtering, all access points store a list of valid MAC addresses and allows only those stations with MAC addresses on the list to associate with the WLAN. However, it is possible for the attacker to spoof a MAC address and gain access.

1.2.4 Authentication Gateways

Authentication gateways [10], like the nomad setup at NC State, also allow access to only authorized personnel, by authenticating the MAC address using a password. It is very easy for an attacker to sniff packets, and discovery of an authenticated MAC address allows the attacker to easily spoof the MAC address and gain access to the WLAN.

1.2.5 Virtual Private Networks (VPN)

Virtual Private Networks are an alternative when looking for solutions to get a secure communication over untrusted networks. VPNs are being used successfully to provide trusted access to the corporate LANs from the untrusted Internet. A user establishes a secure VPN tunnel to the VPN server after successful user authorization, and then all the traffic sent through the tunnel is encrypted. If the user is on a WLAN, all the data sent through the wireless link will be encrypted. Nevertheless, new VPN vulnerabilities are often discovered, and it is by no means a fool-proof security measure.

1.2.6 Parking Lot Attacks

The boundary-less nature of wireless LANs creates significant security issues for administrators. Since the signal strength cannot be controlled, it is probable that the signal will extend beyond the normal boundaries created by wired LANs. This leaves the system open to what is commonly known as the parking lot attack (Fig. 1.1). In a parking lot attack, a hacker can eavesdrop on a communication by setting up a laptop with a wireless LAN adapter in a van or car parked just outside the building housing the wireless LAN. This

practice has become extremely prevalent, and a new term, "war plugging", has been created to describe these activities. With software programs such as *Netstumbler*, wireless LAN hackers called "war drivers" can listen for wireless communications on the 2.4 or 5 GHz frequencies to determine specific system information that would allow them to gain access to the LAN. The attacker may also be able to gain access to the wired network and potentially sensitive information.

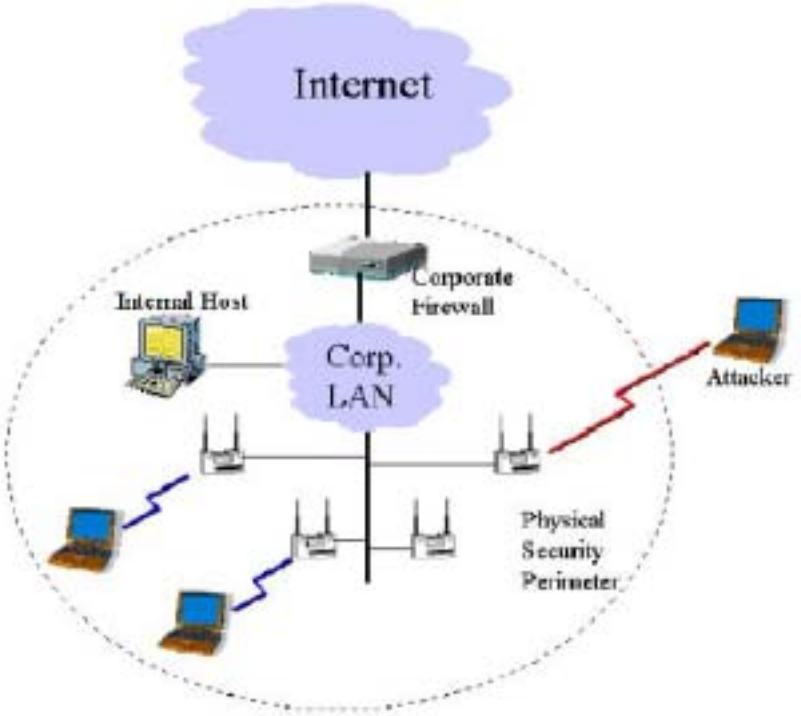

Figure 1.1: The Parking Lot Attack

1.3 Our Proposal

One important feature missing from all of the existing proposals for security, is the ability to distinguish between the users located within a physical security perimeter and those outside this perimeter. Such a feature would be quite helpful in implementing access restriction based on the physical location of the user. If noone outside the security perimeter is allowed to connect to the organization's WLAN, it would become much more difficult for the attacker, or the "war-driver", to break into the network from outside the security perimeter.

In this thesis, we propose an algorithm to localize and track a user on the basis of the signal strengths of the packets that he transmits, and thus be able to block an attacker based on his physical location. The main drawback of this approach is that it can be used to track only an active attacker. If the attacker is passive i.e, just sniffing onto packets, it is impossible to be able to detect him. But, we would be able to identify his location as soon as he starts transmitting any packets. While passive attackers can be extremely dangerous by gathering sensitive information, it is the active attackers who cause the maximum amount of damage.

The proposed approach aims to restore the properties of the physical security perimeter to wireless networks: only users inside the perimeter would be allowed access, while all other users would be denied access. The solution presented does not require any custom equipment; in some cases, only a firmware upgrade of the access points is enough. In other cases, special monitoring stations might be deployed. Also, the proposed approach does

not require any modification to the user side equipment(either hardware or software). The emphasis of our work is not on achieving fine grained localization, since it is not essential for authentication. The focus of our work is on achieving highly reliable coarse localization, which can answer the question, "Is the user inside or outside the security perimeter?"

1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 discusses the related work in the area of localization. Chapter 3 covers our proposal of a localization algorithm. In Chapter 4, we discuss issues pertaining to the implementation of a proof-of-concept, working model based on our algorithm. In Chapter 5, we present the results of our tracking and localization experiments. Chapter 6 provides the conclusion and future work. Finally, in the Appendix, we provide a step-by-step approach to settip up a test-bed and running the localization software developed for the evaluation of our approach.

Chapter 2

Related Work

Localization has been a growing area of research, triggered by the growth in Wireless Networking and Robotics (Mobile Robot Localization). With the deployment of wireless broadband networks, there has been an increased commercial interest in more evolved services. Location-aware services are thus becoming more attractive. These services rely on knowledge of user location. Considerable research has been done in the area of WLAN localization [11–21]. Sizeable research has also been done in robot localization [22–25], where localization is essential for the robot to be aware of its environment and for its movement. Most localization algorithms developed so far, require the participation of the user himself in the localization process. In contrast, our aim is to determine the location of the user in a WLAN for security issues; therefore, we cannot seek the user's co-operation in the process. Our goal is to find the intruder's location in order to control his access. In this report, we propose a novel technique to reliably find the intruder's location using external measurements.

2.1 Localization techniques

Related work in the area of localization and tracking systems can be categorized into four broad categories in terms of the infrastructure involved :

- RF-based systems
- IR-based systems
- Ultrasonic systems
- Other types of localization (optical, magnetic)

2.1.1 RF-based Systems

There is a broad variety of proposed localization schemes in RF-based systems, depending on the frequency range and the environment. We discuss RF-based localization techniques in detail in Sect. 2.2.

2.1.2 IR-based Localization

The Active Badge system as reported in [18] is an IR-based location system. In this system, an active badge worn by the user transmits a unique IR signal every 10 seconds. Sensors placed at known positions within the building pick up the unique identifiers and relay these to the location manager software. While this system provides accurate location information, it suffers from several drawbacks: (a) it scales poorly due to the limited range of IR; (b) it incurs significant installation and maintenance costs; and (c) it performs poorly in the presence of direct sunlight, which is a likely problem in rooms with windows. This technique

estimates the location of the user to be the same as that of the AP to which it is attached. Consequently, the accuracy of the system is limited by the (possibly large) cell size.

2.1.3 Ultrasonic-based Localization

Active BAT [19, 26] uses trilateration, involving ultrasonic transmission by a transmitter attached to the object to be located, which is received at multiple receivers. The time of arrival is used to calculate relative distances of the transmitter from the receiver, and thus trilateration can be used to accurately calculate the position of the sender. The direction in which the person is facing can also be deduced by analyzing the pattern of receivers that detect the ultrasonic signals from the transmitter.

The Cricket Location Support System [12] for in-building, mobile, location-dependent applications is similar in which the mobile nodes are able to determine their own physical location using listeners that can hear and analyze the information from beacons spread throughout the building. CRICKET involves both ultrasonic and RF-based localization.

2.1.4 Optical Localization

Research has been done on Optical tracking of the user [27, 28], but such a scheme requires the use of costly hardware. The user is tracked by recognizing his environment (Landmark based localization), using cameras and/or light sensors.

2.2 RF-based Localization

2.2.1 External

Global Positioning System

Global Positioning System (*GPS*) [29] estimates distance from the RF signal time of flight using the time difference of arrival (TDOA) [30], or the absolute amount of time. With a high probability, the user is able to calculate his own location. GPS uses satellites, and triangulation is done using 3 satellites. The synchronized atomic clocks of the satellites are used to find the 3-dimensional position of the user. As the clocks are synchronized, the user can accurately find his location by measuring the time difference of the received signals from the known satellites.

Wide-area Cellular-based Localization

Several location determination techniques have been proposed [31–33] in the wide-area cellular based localization. The various technological alternatives for locating cellular telephones are based on measuring the signal attenuation, the angle of arrival (AOA), and/or the time difference of arrival (TDOA). Though these systems are promising in outdoor environments, the performance in indoor environments is limited by the multiple reflections suffered by the RF signal. Also, inexpensive off-the-shelf hardware is limited in its ability to provide fine-grain time synchronization.

Sensor Networks

Lately, there have been a considerable number of publications [34–36] in the field of sensor networks localization schemes. These schemes are based on both RF and acoustic-based localization, of GPS-less nodes assisted by GPS-enabled beacons.

2.2.2 Indoor

The most relevant work for the problem under consideration includes the proposals for indoor RF-based localization in a WLAN, using inexpensive off-the-shelf hardware to provide fine-grain localization. The range of a WLAN is small, typically a building. There are essentially two ways in which a user in a wireless LAN can be localized, depending on the parameter we use to localize.

- By measuring time delays between transmission and reception of the packet. For such an approach, we would have to calculate the exact arrival time of a broadcast packet from a source, and then compare it with the times at different locations to be able to find out the location of the transmitter. In an indoor environment, this approach does not work because the obstructions cause multiple paths between the transmitter and the receiver. Even in an environment devoid of obstructions, one would require extremely synchronized clocks on the receivers, which are not easily realized as the time stamped at the reception can vary because of multiple reasons, e.g. processor load, clock speed, state of kernel buffer, etc. The distances involved in a building environment are very

small, and considering multipath reflection, it is almost impossible to localize using such a technique.

- The other measurable quantity about the received packets is the received signal strength (RSS), which is a function of the location, but it varies with a large number of factors. As the distance between the transmitter and the receiver increases, the RSSI decreases. We discuss RSS-based methods in greater detail in Sec. 2.4.

2.3 What is RSS?

Received Signal Strength or RSS [37] is a measure of the power of a received signal. The IEEE 802.11 standard defines a mechanism by which RF energy is to be measured by the circuitry on a wireless NIC. This numeric value is an integer with an allowable range of 0-255 (a 1-byte value) called the Receive Signal Strength Indicator (RSSI). Each vendor can choose its own specific measure of the Signal Strength Indicator. Thus, each vendor's 802.11 NIC will have a specific maximum RSSI value, *RSSI_Max*. Cisco chooses to have 101 different levels of RF energy and, thus their RSSI_Max is 100. Lucent Orinoco cards have an RSSI_Max set to 127. The RF energy level reported by a particular vendor's NIC ranges between 0 and RSSI_Max. Nothing has been said about measurement of RF energy in dBm or mW. Thus, RSSI is an arbitrary integer value, defined in the 802.11 standard and intended for internal use by the microcode on the adapter and by the device driver.

2.4 RSS-Based Techniques

A number of proposals to use an RSS-based technique for localization have been proposed, but as far as we know, they have all used the signal strength information of the Access Points at the mobile node, to locate the user's position. The problem definition has been stated as: "Given a list of known values of signal strengths related to the different access points and the current values, can you/I determine where I am?" There are a few solutions to this problem. Either a deterministic or a probabilistic approach is used to solve the problem, though the deterministic approach has proven to be less reliable [11, 16, 20, 38]

2.4.1 Path-loss Models

Recieved Signal Strength can be modeled mathematically [11, 39, 40], but such path-loss models are not very accurate in an indoor environment. Inside a building, the variation of the RSS with distance is significant due to obstructions and multipath fading effects (the inaccuracy of the path-loss model). It is usually not reliable to use the RSS method as such in this manner. The RSS method has been used along with training and interpolation to improve accuracy in [41]. Another alternative is to use the path loss models to compile an artificial database of RSS values [11], and the measured RSS values are then compared to this database to obtain the mobile node's location. This seems counter-intuitive as it retains the inaccuracy of the path loss models and the delay associated with the database search.

2.4.2 RADAR

The RADAR [**?**, 11] system is built using off-the shelf LAN technology. RADAR is based on the fundamental idea that the RSSI measurements are a function of the mobile user's location, which can be calculated with some accuracy. This implies that location of the user is an inverse function of the Signal Strength. In RADAR, a *Radio Map* of the building is created, which is a database of locations in the building and the corresponding signal strength of the beacons from the various access points.

To locate the position of the mobile user in real-time, the mobile node measures the signal strength of each of the Access Points within range. It then looks up the Radio Map for the closest match of the signal strength's tuple. This location is approximated to be the location of the mobile node. RADAR uses the nearest neighbor in signal space *(NNSS)* algorithm to find the location of the mobile user. In this technique, the closest tuple is found out by minimizing the length of the vector defined by the measured signal strengths (ss_1, ss_2, ss_3) and the signal strength vectors in the database (ss'_1, ss'_2, ss'_3). The length of the vector is the same as its euclidean distance in the signal space given by $\sqrt{((ss_1 - ss'_1)^2 + (ss_2 - ss'_2)^2 + (ss_3 - ss'_3)^2)}$. A variant of this method, *NNSS-AVG* algorithm picks up a small number of the closest matches, and then averages the locations between them to estimate the user's position. In an enhancement to the RADAR, a Viterbi-like algorithm [42] is suggested, with *k-NNSS* (k nearest nodes in signal space) and a history depth of h [13]. Thus, the collection of these h k -NNSS sets can then be imagined as a graph, with edges connecting different sets of nodes, representing the likelihood of change

from one state to another. It works like a Viterbi algorithm to find the right location in time, along with error correction, due to the introduced redundancy.

2.4.3 Neural Networks-based Localization

The signal strengths of the Access Points are related to the location of the user. But the complexity of finding an inverse solution, i.e. to find the position of the user given a set of signal strengths while lacking complete information, makes it a suitable candidate for a cognitive model, a neural network. Such a solution was tried [17] but did not prove very promising, as the results were only comparable to a relatively much simpler algorithm of the k-closest neighbors.

2.4.4 Bayesian Filters

In order to analyze and make inference about a dynamic system, at least two models are required. First, a model describing the evolution of the state with time (the system model), and second a model relating the noisy measurements to the state (the measurement model).

In the Bayesian approach to dynamic state estimation, one attempts to construct a posterior probability density function (*pdf*) of the state based on all available information, including the set of received measurements. Since this pdf embodies all available statistical information, it may be said to be a complete solution to the estimation problem. An optimal estimate of the state can be obtained from the pdf. For many problems an estimate is required each time a measurement is received. In this case, a recursive filter is a convenient solution. A recursive filtering approach means that the received data can be processed

sequentially rather than as a batch, so that it is not necessary to store the complete data set or to reprocess existing data if a new measurement becomes available. Such a filter consists of essentially two stages: prediction and update. The prediction stage uses the system model to predict the state pdf forwards from one measurement time to the next. Prediction generally translates, deforms and spreads the state pdf. The update operation uses the latest measurement to modify the prediction pdf. This is achieved using Bayes Theorem, which is the mechanism for updating knowledge about the target state in the light of extra information from new data.

A number of algorithms based on the Bayes principle have been proposed for localization. Some of the relevant ones are:

- Markov Localization

  The Markov assumption is that the updated probability depends only on the robot's expected position and current sensor readings, and not their history.

- Monte Carlo Localization

  Monte Carlo Localization(*MCL*) [23, 43] is a family of algorithms, derived from Markov Localization, for localization based on particle filters, which are approximate Bayes filters that use random samples for posterior estimation.

2.5 Issues with RSS-based Localization

2.5.1 Variation in Measured RSSI

RSSI is not a very reliable measure, and it depends on a large number of factors.

- Location: The main variable behind the variation of the signal strength received by the base stations is the location of the intruder. This assumption is the basis of our work.
- Transmission power: A change in the transmission power of the user's wireless NIC will cause a change in the received signal strength at the receiver. But, as the received strength is directly proportional to the transmission power, a similar drop in RSSI will be noticed at all the base stations. The fact, that RSSI is measured on a logarithmic scale aides in observing relative changes in signal strength.
- Directionality of the wireless NIC: A typical 802.11 card is not omnidirectional in nature. Thus, with neither the transmitter nor the receiver being omnidirectional in nature, we are bound to have variations in measured RSSI. A variation in the directionality of the wireless NICs, compared to the NIC used for fingerprinting, could change the perceived position of the user.

The signal frequency, as well as the range between the end points, affects the amount of attenuation. Attenuation increases with both frequency and range. Unlike open outdoor applications based on straightforward free space loss models, attenuation for indoor systems is very complex to calculate. The main reason for this difficulty is that the indoor signals bounce off obstacles and penetrate a variety of materials that offer varying effects on attenuation. The signal strength falls exponentially as the range increases.

Typical office obstacles such as doors, windows and walls offer fairly known levels of attenuation. These values of attenuation are in addition to the path loss mentioned earlier. Table 2.1 provides some examples of the attenuation values of common office construction [1].

Plasterboard wall	3dB
Wall with metal frame	6dB
Cinder block wall	4dB
Office window	3dB
Metal door	6dB
Metal door in brick wall	12.4dB

Table 2.1: Signal Attenuation due to common office construction [1]

As a result, a typical small office having several plasterboard walls could equate to an additional 9 to 12dB of attenuation. Metal doors and glass walls could sometimes be in the way of the propagation of the signal, causing even larger amounts of attenuation. The attenuation varies continuously as people move across the officespace, and with any small change in the environment, e.g. the shifting of a table or some other piece of furniture or restructuring.

2.5.2 Effect of Multiple Floors

Multiple floors lead to multiple locations having a similar RSSI finger print. All the experiments in the RADAR system were conducted on a single floor. A problem with multi-floor environments is that two or more very different locations could potentially have the same RSS-location fingerprint. In [14, 44], instead of storing average RSS values, the joint or

marginal distribution of the RSS are used in fingerprinting locations. In most cases, the accuracy is about 5 to 20 feet.

2.5.3 Effect of Multiple Channels

There are 14 channels in an 802.11b network on which the mobile node might be transmitting. We would, thus require each of the base stations to be able to hop across all the channels to sniff any packet from the mobile node, which in effect is impossible. The Cisco Aironet cards have an in-built hopper, while the Wavelan/Orinoco cards need a software hopper to be able to do the same.

2.5.4 Granularity

The database entries are collected on a grid of points inside and outside the building. The granularity is based on the number of cells in which the map is divided. For a fine-grained localization, the cell size would need to be extremely small. But, in reality, the signal strength hardly varies over small cells of a few feet, in free space. Thus, there is no point in making the cells smaller in size. The localization, using RSSI, cannot practically be too fine-grained.

2.5.5 Fault Tolerance

Finally, there is a possibility that one of the base stations crashes and be unable to capture packets. Therefore, the base station would be unable to contribute in the localization of the mobile node. Even in such a case, the results obtained should not be completely off the

mark.

Chapter 3

Proposed Approach

3.1 Overview

In Chapter 2, a number of localization techniques in a WLAN environment were discussed. All of those techniques measure the received signal strength of the *beacons* at the user's end for localization. The problem of localizing a user in a WLAN environment without his awareness or co-operation is a very interesting one. It has applications in security where a "parking lot attacker" can be blocked from accessing the network because of his physical location. Such a feature could be particularly useful in high-security, defense-related zones. In this chapter, a simple algorithm for solving this problem is proposed. This chapter is organized as follows: Section 3.2 formulates the problem in a more rigorous manner; In Sec. 3.3, the fingerprinting technique to effectively map the test-bed and to obtain sample data which could then be used for localization is explained. Finally, in 3.4, a simple algorithm is proposed to solve the localization problem, as well as provide examples to further illustrate the concepts.

3.2 Problem Formulation

The problem of self-localization, or localization of a user using the access points signal strengths as a reference, has been a topic of research in many publications [11, 14–16, 20]; but, the problem of localizing the user without the users co-operation has been mentioned in passing reference in [14, 20] and never been discussed seriously in any of the papers. We aim to solve this problem, though we cannot assure a fine-grained solution to the general localization problem. Our aim is to be able to implement at least a coarse-granular localization scheme to effectively track an intruder and determine if he is inside or outside a defined security perimeter. Only the users inside the perimeter are allowed access to the network, but anyone outside this security perimeter must be identified and blocked with a high degree of accuracy. This would prevent the so-called "parking lot attacks" on a WLAN [2]. The only physical quantity available for the determination of the user, is the RSSI measurement of each packet transmitted by the users.

For our algorithm, we follow a Bayesian approach for dynamic state estimation. The state of the system is represented as a probability distribution function and each state changes according to observed data. Our approach is based on the Markovian principle i.e. the predicted state is a function of only the present estimated state and the new measured data. We have used a *recursive filtering* approach, in which each new measurement is processed individually. The received data is processed sequentially rather than as a batch. We also take some ideas from the K-nearest neighbors approach for filtering possibly erroneous data.

The problem of localization and tracking of an object can be divided into two parts: (1) Calibration (or finger-printing phase), and the (2) Localization phase. The purpose of the finger-printing phase is to collect labeled data (containing their own location) from different points spread across the test-bed, and then filter this data into a form which can then be used in the Localization phase to solve the inverse function of discovering the location of unlabeled data (a normal 802.11 packet).

3.3 Calibration

3.3.1 Goal

A key step in our research methodology is the finger-printing phase which can also be termed as calibration or the data collection phase. In this phase, we collect relevant labeled data namely the location along with the RSSI, and process it into a form that is used in the localization phase to derive the location of unlabeled data. The known variables during the localization phase are the RSSI of the mobile node and the identity of the Base Station(BS) which receives the signal. Thus, the aim of this phase is to be able to generate the probability density function (pdf)of the user's location, given the RSSI measurements at the Base Stations. For this purpose, we broadcast a Mobile Nodes location and each of the Base stations capturing these packets measures the RSSI. Thus, the collected information consists of the location (x,y coordinates), RSSI and the Base Station Identifier ($BSID$) of the BS which captures the packet. Finger-printing of the whole map requires that labeled packets be collected from a number of uniformly distributed data points across the test-bed.

If the base-stations are uniformly spread over the test-bed, and each has approximately an equal range, they will each receive an almost equal number of labeled packets with varying RSSI, during the process of finger-printing. We then construct a model that allows us to obtain the *pdf* for the user's location, given the Base Station identifier *(BSID)* and the RSSI. We store all the packets received from the different BS's in a single database and then filter it using a combination of the BSID and measured RSSI.

3.3.2 Proposed Approach

Let us assume the test-bed is divided into an NxN matrix, where the (x,y) co-ordinates vary from (1,1) to (N,N). Suppose our finger-printing covers uniformly spread datapoints over the test-bed. Each of the elements in the database would look like:

$< x_i >, < y_j >, < BS_{bsid} >, < \lambda_{rssi} >,$

where x_i and y_j define the location,

BS_{bsid} = the Base Station identifier of the station that captured the packet,

λ_{rssi} = measured Signal Strength.

For a collection of data elements with a constant BSID=BS_1 and a constant RSSI=λ,let

η_{ij} = number of elements at location (i, j) on the X-Y plane; then:

$$p(x, y) = \frac{\eta_{xy}}{\sum_{i=1}^{i=N} \sum_{j=1}^{j=N} \eta_{ij}},$$

where $p(x, y)$ represents the probability that the user's position is (x, y).

Representative data is collected during finger-printing, but it would take extremely diligent work to collect representative data from the entire set of possible locations. Instead we collect data from a number of uniformly distributed points and generate data for the rest of the points, for which we have no measurements. This is achieved by smoothing the measured data using a Gaussian filter [45] as described in Sec 3.4.1.

3.3.3 Example

Assuming that Table 3.1 shows a subset of the elements in the database.

x	y	$BSID$	$RSSI$
1	1	BS_1	80
1	1	BS_2	50
1	1	BS_1	80
1	5	BS_1	77
1	5	BS_1	80
1	5	BS_1	75
3	1	BS_2	50

Table 3.1: A section of the Database

Suppose Table 3.2 represents the data corresponding to $BSID = BS_1$ (Fig.3.1), and $RSSI = \lambda = 80$, with N=5.

x	y	η_{xy}
1	1	3
1	5	5
3	1	5
3	3	15
3	5	6
5	1	3
5	5	2

Table 3.2: A sample datafile with BS_1 and $RSSI = \lambda$

Assume that we covered only the cells (1,1),(1,5),(3,1),(3,3),(3,5),(5,1),(5,5) during the

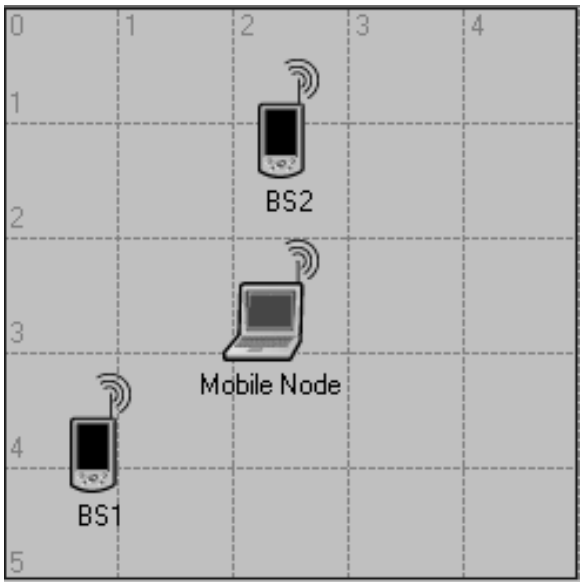

Figure 3.1: Example Test-bed

finger-printing process. Even though we did not actually collect any data for some points (e.g (1,2),(1,4)) this does not mean that we would use a value of zero at these points. We generate values for such points by passing the measured dataset through a Gaussian filter with standard deviation $\sigma_{loc} = 1$ and kernel size 5. Even though RSSI does not follow a strictly Gaussian distribution over a region, it still proves to be a good approximation. Table 3.3 represents the filtered dataset. It has non-integer values representing number of packets received from a particular position because the graph is smoothed out and the peaks spread out. Figures 3.2 and 3.3 show how the filtering helps in generating new datapoints. According to the definition of conditional probability, which in simple terms is given by,

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

we compute the probability density matrix for the user's location when a packet arrives at

X	Y	η_{ij}
1	1	1.257773
1	2	1.101763
1	3	1.054937
1	4	1.514021
1	5	1.937472
2	1	1.936445
2	2	2.693292
2	3	3.392411
2	4	3.044591
2	5	2.515639
3	1	2.496199
3	2	4.018065
3	3	5.324303
3	4	4.237258
3	5	2.857588
4	1	1.936445
4	2	2.693292
4	3	3.314026
4	4	2.693292
4	5	1.936445
5	1	1.257773
5	2	1.101763
5	3	0.925702
5	4	0.934827
5	5	0.982542

Table 3.3: Gaussian filtered datapoints for BS_1 and $RSSI = \lambda$

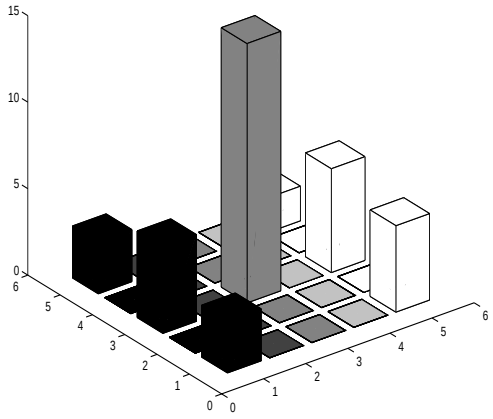

Figure 3.2: Datapoints for BS_1 and RSSI=λ

BS_1 with $RSSI = \lambda$ (Table 3.4). The probability density is then mapped on to the X-Y plane in Fig.3.4.

The probability of finding the user at a position (x, y) is dependent on the number of labeled packets with (x_1, y_1) being collected during the finger-printing process and it is also directly proportional to the number of packets transmitted from the position (x_1, y_1) during finger-printing. Therefore, for an accurate estimation of the pdf, we need to evenly distribute our data-points over the test-bed as well as to transmit a constant number of packets from each data-point.

3.3.4 Summary

The finger-printing process can be summarized as follows:

1. Select a large number of uniformly distributed data points over the test-bed.

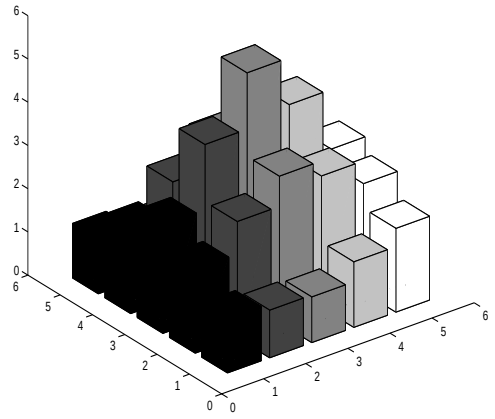

Figure 3.3: Probability density for Filtered data

2. Store elements in the form "$< BSID >< RSSI >< X >< Y >$".

3. Filter the elements into tables, based on the combination of BSID and RSSI, and store the identical locations (x,y) by their number of occurences. Thus, elements would be of the form "$< X >< Y >< \#\ of occurences >$".

4. Filter the tables using a Gaussian filter, spreading the data-points to account for the lack of representative data from the remaining cells.

5. Form pdfs using the formula

$$p(x, y) = \frac{\eta_{xy}}{\sum_{i=1}^{i=N} \sum_{j=1}^{j=N} \eta_{ij}},$$

where $p(x, y)$ represents the probability that the user is at position (x, y).

X	Y	$p(x,y) = \eta_{ij}/\Phi$
1	1	0.0220052
1	2	0.0192758
1	3	0.0184565
1	4	0.0264884
1	5	0.0338968
2	1	0.0338789
2	2	0.0471202
2	3	0.0593516
2	4	0.0532663
2	5	0.0440121
3	1	0.043672
3	2	0.0702976
3	3	0.0931508
3	4	0.0741325
3	5	0.0499946
4	1	0.0338789
4	2	0.0471202
4	3	0.0579802
4	4	0.0471202
4	5	0.0338789
5	1	0.0220052
5	2	0.0192758
5	3	0.0161955
5	4	0.0163552
5	5	0.01719

Table 3.4: Probability Density Function for BS_1 with $RSSI = \lambda$

3.4 Localization

We define the state estimate as the probability density function (pdf) of the position on the X-Y plane. It can be represented as a two-dimensional matrix as shown in Fig. 3.5 where $p(x_i, y_i)$ is the probability of finding the user at location (x_i, y_i). We have to define a model that relates this *pdf* with the new tuplet (BSID, RSSI) received from the base-station, as well as change in state, because of movement of the mobile node. Any algorithm concerning

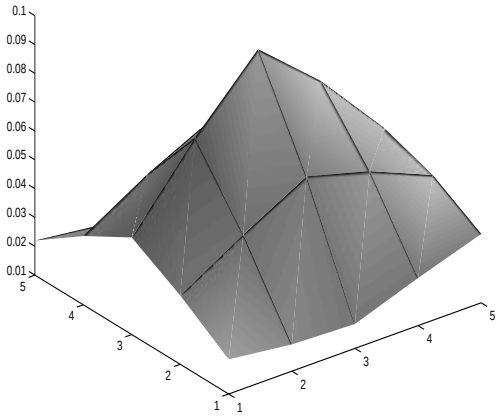

Figure 3.4: Probability Density over the X-Y plane

$$\begin{pmatrix} p(x_1, y_1) & p(x_2, y_1 & \dots & p(x_N, y_1) \\ p(x_1, y_2) & p(x_2, y_2 & \dots & p(x_N, y_2) \\ \vdots & \vdots & \ddots & \vdots \\ p(x_1, y_N) & p(x_2, y_N & \dots & p(x_N, y_N) \end{pmatrix}$$

Figure 3.5: The state representation.

a dynamic system consists of at least two models.

1. System model - a model describing the evolution of the state with time.

2. Measurement model - a model relating each measurement with the change in state.

3.4.1 System Model

We assume that the mobile node that we aim to track is moving, and the random movement has a normal distribution, centered at the starting location. Thus, we periodically smoothen our state representation with a Gaussian filter. In 2-dimension, an isotropic Gaussian

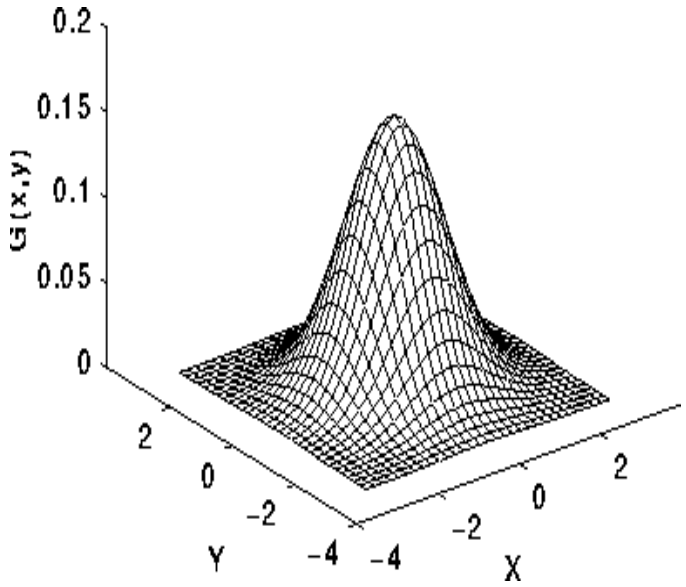

Figure 3.6: 2-D Gaussian distribution with mean (0,0) and $\sigma = 1$

(Figure 3.6) is of the form

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

The idea of a Gaussian function is to use this 2-D distribution as a "point-spread" function, which is achieved by convolution. Since the probability density matrix is discrete in nature, we need to find a discrete approximation to the Gaussian function before we can perform the convolution. Although in theory, the Gaussian is non-zero everywhere; but in reality, it is effectively zero at coordinates more than three standard deviations from the mean. We, therefore, truncate the kernel at this point. The matrix below shows a suitable integer-valued convolution kernel that approximates a Gaussian with a σ of 1.0 and kernel of size 5.

$$\frac{1}{273}\begin{pmatrix}1 & 4 & 7 & 4 & 1\\ 4 & 16 & 26 & 16 & 4\\ 7 & 26 & 41 & 26 & 7\\ 4 & 16 & 26 & 16 & 4\\ 1 & 4 & 1 & 4 & 1\end{pmatrix}$$

Table 3.5: Gaussian Kernel of size 5 with $\sigma = 5$

3.4.2 Measurement Model

The measurement model defines the relationship of the state estimate to successive measurements. Section 3.3 describes the algorithm to compute the *pdf* over the X-Y plane for a given pair of RSSI value and BSID. This 2-dimensional pdf matrix that we compute acts as a constraint for our measurement model. We propose a simple algorithm for the update process.

The position estimate is updated as follows,

$$MyNewPositionEstimate = MyOldPositionEstimate \cap NewConstraint,$$

where *MyOldPositionEstimate* is the pdf for the user's location estimate. *NewConstraint* is the pdf derived, from the new measurement received, using the finger-printing process.

An important factor that we need to keep in mind while developing an algorithm is the reproducibility of the experimental data. There is a variance in the RSSI with time, as well as other external factors as discussed in Section 2.5.1. If we assume that the standard deviation of the RSSI distribution over the X-Y plane is β and the measured RSSI is λ, then we can assume that most RSSI values λ_i are such that,

$$\lambda - \beta <= \lambda_i <= \lambda + \beta$$

Therefore, we calculate the NewConstraint by averaging the probability densities calculated in the finger-printing phase for all λ_i in the range of $\lambda \pm \beta$.

3.4.3 Example

The position estimate is initialized to an equiprobable density over the complete test-bed. Assume the test-bed is of size NxN and for simplicity, assume N=7(Fig. 3.7).

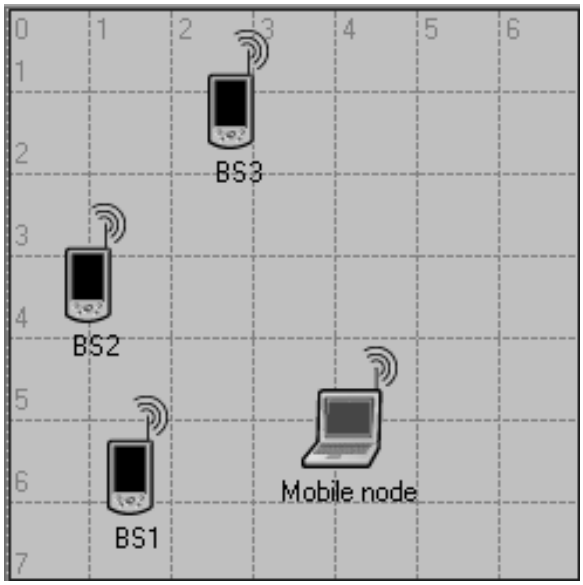

Figure 3.7: Example Test-bed

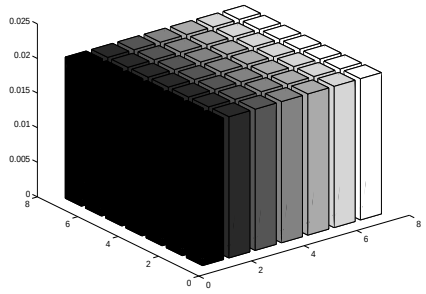

Figure 3.8: Initial Probability Densities

Thus, the initial state estimate ϕ_{init} is given by

$$\phi_{init} = \frac{1}{49} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Now, let us assume that a packet arrives from BS_1 with RSSI value of λ_1. The updation matrix is derived from the pdfs obtained during the calibration process. Assume the updation matrix (ϕ_{update}) which is evaluated by averaging the probability density matrices corresponding to all λ_i where, $\lambda_1 - \beta =< \lambda_i < \lambda_1 + \beta$ is,

$$\phi_{update} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 0 \\ 1 & 5 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$\phi_{k+1} = \phi_k \cap \phi_{update}$, and ϕ_{init} has all 1's. Therefore, $\phi_2 = \phi_{update}$ after the first packet is received. Assuming that the next packet is received from BS_2 with RSSI λ_2, and the new ϕ_{update} is given by

$$\phi_{update} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 4 & 2 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

The resulting state estimate ϕ_3 would then become (Fig **??**)

$$\phi_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 & 0 & 0 & 0 \\ 1 & 15 & 4 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

In this representation, we have omitted normalizing the matrices because dividing the matrix by a constant does not change its representation in a normalized form. The matrix can be normalized at any state, and the probability distribution would remain unchanged.

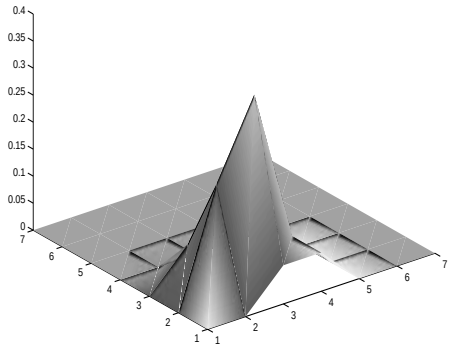

Figure 3.9: Probability density after the first packet, $\phi_2 = \phi_{update}$

Once a second, a Gaussian filter as defined in the System model (Sec. 3.4.1) is applied over the estimated state matrix. Fig. 3.11(b) and Fig. 3.11(c) illustrate the effect of the

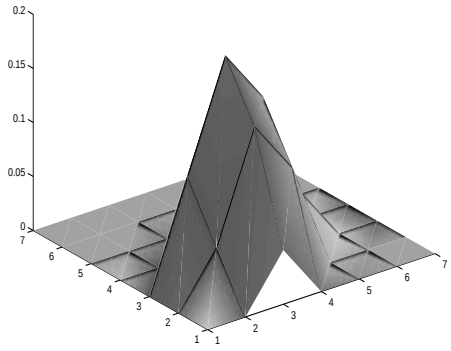

Figure 3.10: The pdf of the updation matrix, derived from the second packet

Gaussian filter with varying standard deviation σ over this probability density. We use a Gaussian filter with $\sigma = 0.5$ and $\sigma = 1$ and of kernel size 5 in the two figures.

3.4.4 Summary

The localization algorithm, including the system model and the measurement model, can be summarized as :

1. Initialize the position estimate (ϕ_{init}) as the entire space.
2. For each packet received :
   - if the packet is received from base station BS_i with RSS λ_i, calculate the ϕ_{update} (the updation matrix) by averaging over the probability densities for λ where $\lambda_i - \sigma < \lambda < \lambda_i + \sigma$; and
   - update the position estimate $\phi_{new} = \phi_{old} \cap \phi_{update}$.
3. At every 1-second interval, apply the system model (a Gaussian filter as in Sec. 3.4.1)
4. Goto Step 2

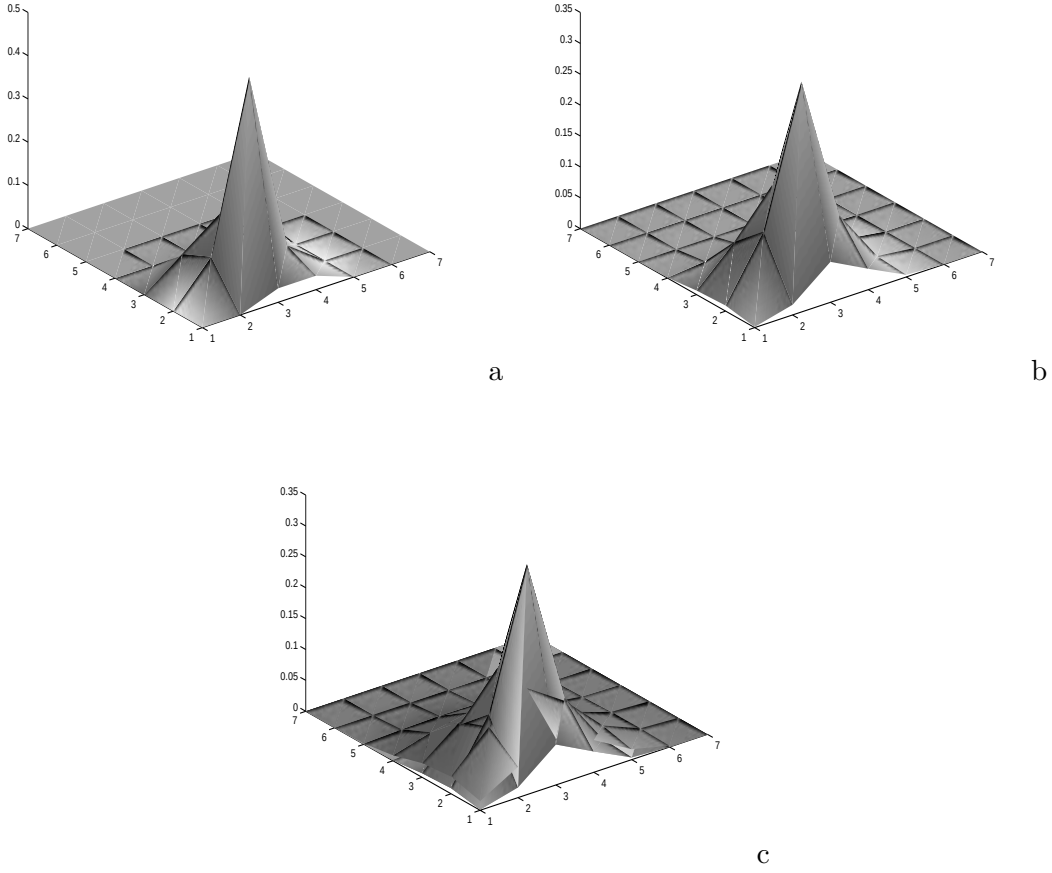

Figure 3.11: (a) The pdf representation of ϕ_3, the state representation after receiving the two packets. Fig. (b) is the representation after filtering with a Gaussian having $\sigma = 0.5$, and (c) represents the pdf after filtering with a Gaussian having $\sigma = 1$

.

Chapter 4

Implementation

4.1 Overview

In Chapter 3, we proposed an algorithm for localization and tracking a user in a WLAN. Since no networking simulator has realistic indoor propagation models, we chose to do a proof-of-concept implementation. The proof of concept validates the proposed approach and allows us to evaluate the performance of the proposed approach using a realistic environment. This algorithm can be implemented as a small firmware upgrade on the Access points in the WLAN. As the hardware information is proprietary, a firmware upgrade was no longer a viable option. Therefore, we decided to come up with a proof-of-concept demonstration using a network of base stations. Section 4.2 gives a brief description of our infrastructure. Section 4.3 is a discussion of RSSI and its extraction from captured data. Sections 4.4 and 4.5 describe the implementation issues in finger-printing and localization, respectively.

4.2 Experimental Setup

4.2.1 Overview

The experimental setup consists of base stations(BS), mobile nodes(MN) and a server. The MNs are on the wireless network, and any packet being transmitted by the mobile node is sniffed by the BSs. Each BS has a wireless as well as a wired interface. It captures all packets on the wireless interface and transmits the RSSI to the server, using the wired network. The server which is connected to the base stations on the wired network, receives and processes the information to estimate the location of the MNs.

4.2.2 Test-bed

The experimental test-bed is located on the third floor of the Electrical and Computer Engineering Department in Daniels Hall. We have limited the experimental area to two perpendicular corridors of the Department (Fig. 4.1). We set up a network of 5 base stations relatively uniformly distributed over the test-bed as required in Sec. 3.3. Each base station captures wireless packets, extracts the RSSI and sends it to the server for computation of the user's location. The base stations are placed in such a way that their range covers the entire test-bed. Our aim is to track WLAN users anywhere along the two corridors, and in the adjoining rooms. The shaded area, Room No.361, acts as the experimental "exterior". It is marked as "outside the security perimeter" for the purpose of our experiments. Fig. 4.1 shows a floor-map of our test-bed. The map in Fig. 4.1 is divided into a grid of 100x100, whereas the dimensions in reality are 217ft x 240ft. Each cell is thus 2.17 ft on the x-axis

and 2.40 ft on the y-axis.

- Base Station: The setup consists of an iPAQ (Model No. H3870) connected to a computer, via a serial link supporting data transfer rates of upto 115 Kbps. Each iPAQ is equipped with an Orinoco gold Classic PC card for capturing the packets of the tracked users. The iPAQs run Familiar Linux 0.6 (Opie) with kernel 2.4.18-rmk3. Each of the computers attached to the iPAQ has a 333 Mhz Pentium II processor, running Redhat Linux 7.3 (kernel 2.4.18-3).
- Server: Our server is a desktop with a 2.4 Ghz Intel Pentium 4 processor, 512 MB RAM, running Redhat Linux 7.3 (kernel 2.4.18-3).
- Mobile Node: A laptop with a 1.8 Ghz Intel Pentium 4m processor, 256 MB RAM, running Redhat Linux 8.0 (kernel 2.4.18-14) acts as the mobile node. A Cisco Aironet 350 series PC card is used as the transmitter.

4.3 Received Signal Strength Information (RSSI)

The IEEE 802.11 standard [46] states that

> "The received signal strength indicator (RSSI) is an optional parameter that has a value of 0 through RSSI_Max. This parameter is a measure by the PHY sublayer of the energy observed at the antenna used to receive the current PPDU. RSSI shall be measured between the beginning of the start frame delimiter (SFD) and the end of the PLCP header error check (HEC). RSSI is intended to be used in a relative manner. Absolute accuracy of the RSSI reading is not specified."

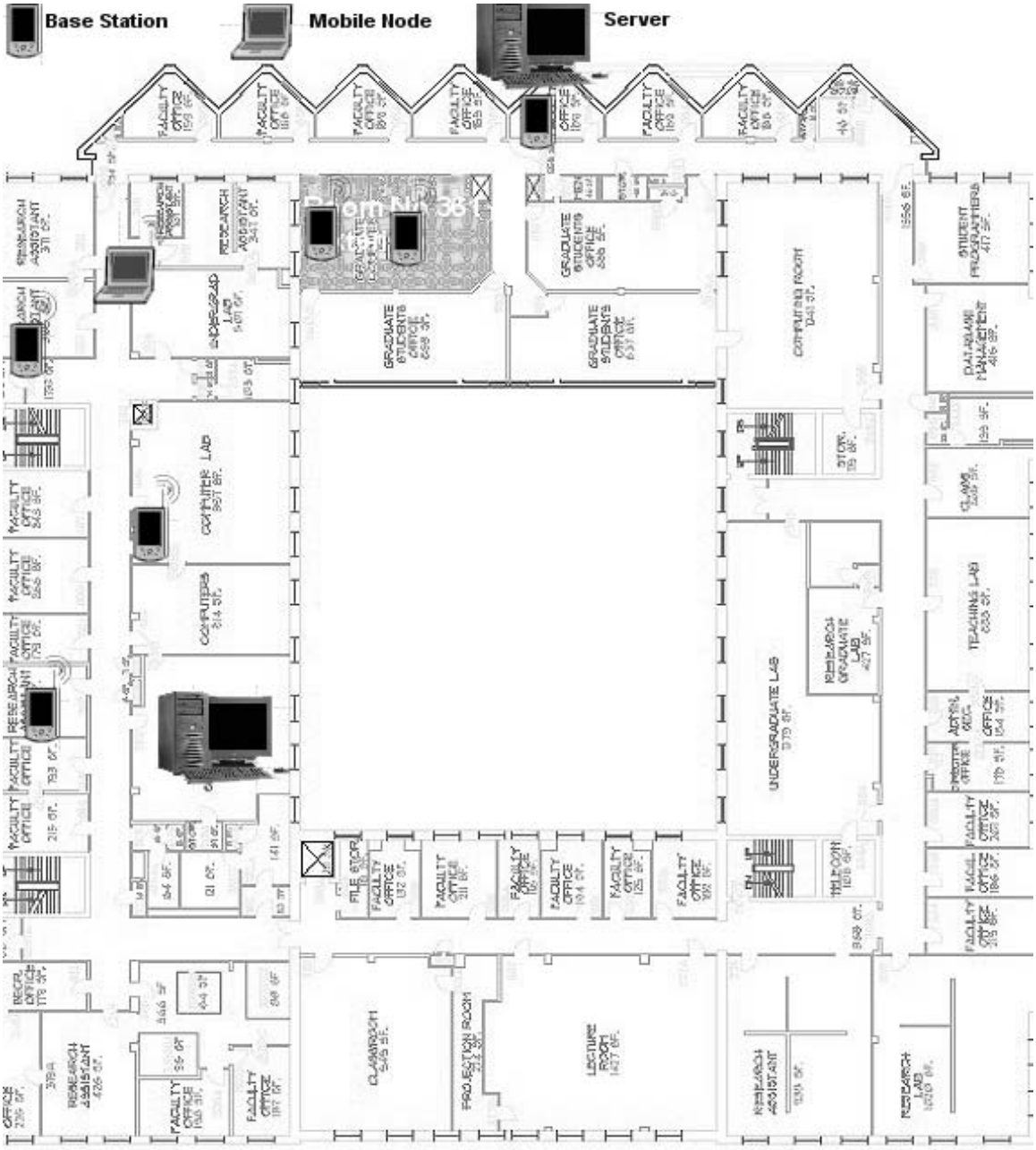

Figure 4.1: The Experimental Test-bed

The last two sentences are especially important. They signify that there is neither a stipulated accuracy required for the RSSI, nor a relationship with any particular energy level which is measured in mW or dBm. Thus individual providers can choose to provide their own levels of accuracy, granularity or range for the RSSI values.

Measurement of RSSI is one of the basic requirements for the working of our algorithm. It is an 8-bit field in the 802.11 header. It is a vendor-specific quantity and each vendor has the liberty to choose a unique scale for its measurement. In the case of Orinoco cards, it is measured on a scale of 0-127. The complete 802.11 header is visible only when packets are captured in the monitor mode. The Orinoco monitor-mode patches from the Shmoo Group [7] enables an Orinoco NIC to be put into monitor mode. When an Orinoco NIC is in monitor mode, it receives raw packets and makes them available to an application-layer program via the PF_PACKET interface used by the packet-capturing library. The monitor mode is necessary in order to capture packets from all wireless nodes in its range. Without it, the network interface will convert the 802.11 header to a pseudo 802.3 (regular Ethernet) header. Fig 4.2 shows a typical packet captured in the monitor mode with the ARPHRD_IEEE80211_PRISM header (PRISM2 Header) prepended to the 802.11 header.

We developed a packet sniffer using the *libpcap* library provided in *C* to capture the packets on the wireless LAN. A cross-compiler is used to compile the sniffer for the *ARM* architecture on the iPAQs.

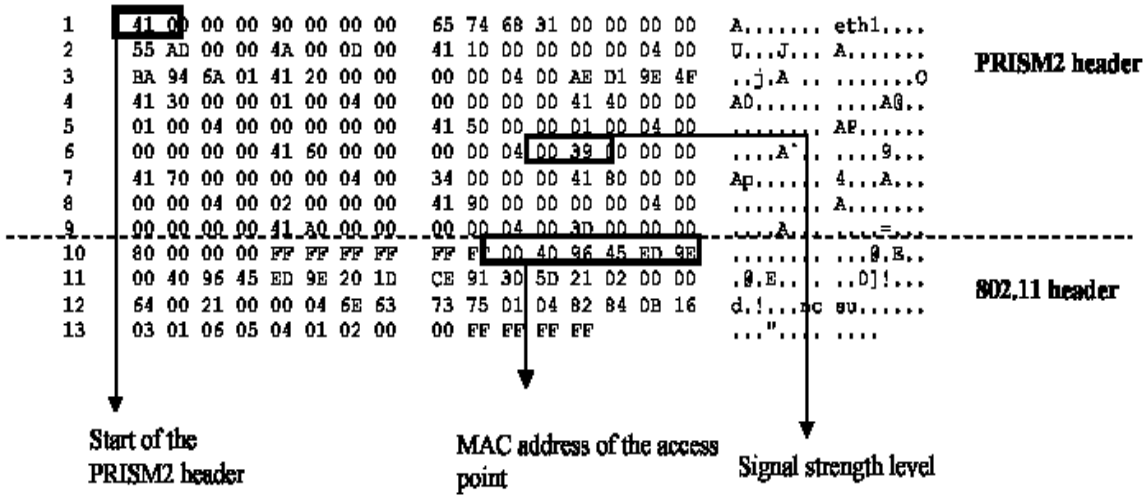

Figure 4.2: A captured Beacon Frame

4.4 Calibration

Calibration, as described in Section 3.3, is required for gathering representative data about the RSSI distribution of a mobile node in the test-bed. Filtering the data into a form that is usable by the Localization algorithm is also a part of the finger-printing process. Figure 4.3 shows the finger-printing map on our test-bed.

4.4.1 Data Collection

Data collection is an essential part of the calibration process. Representative data is collected from the network, which helps us to finger-print the entire test-bed. This part can be viewed as a multi-step process involving the following steps :

1. On the mobile node (laptop), the user clicks on his location on a map of the floor.

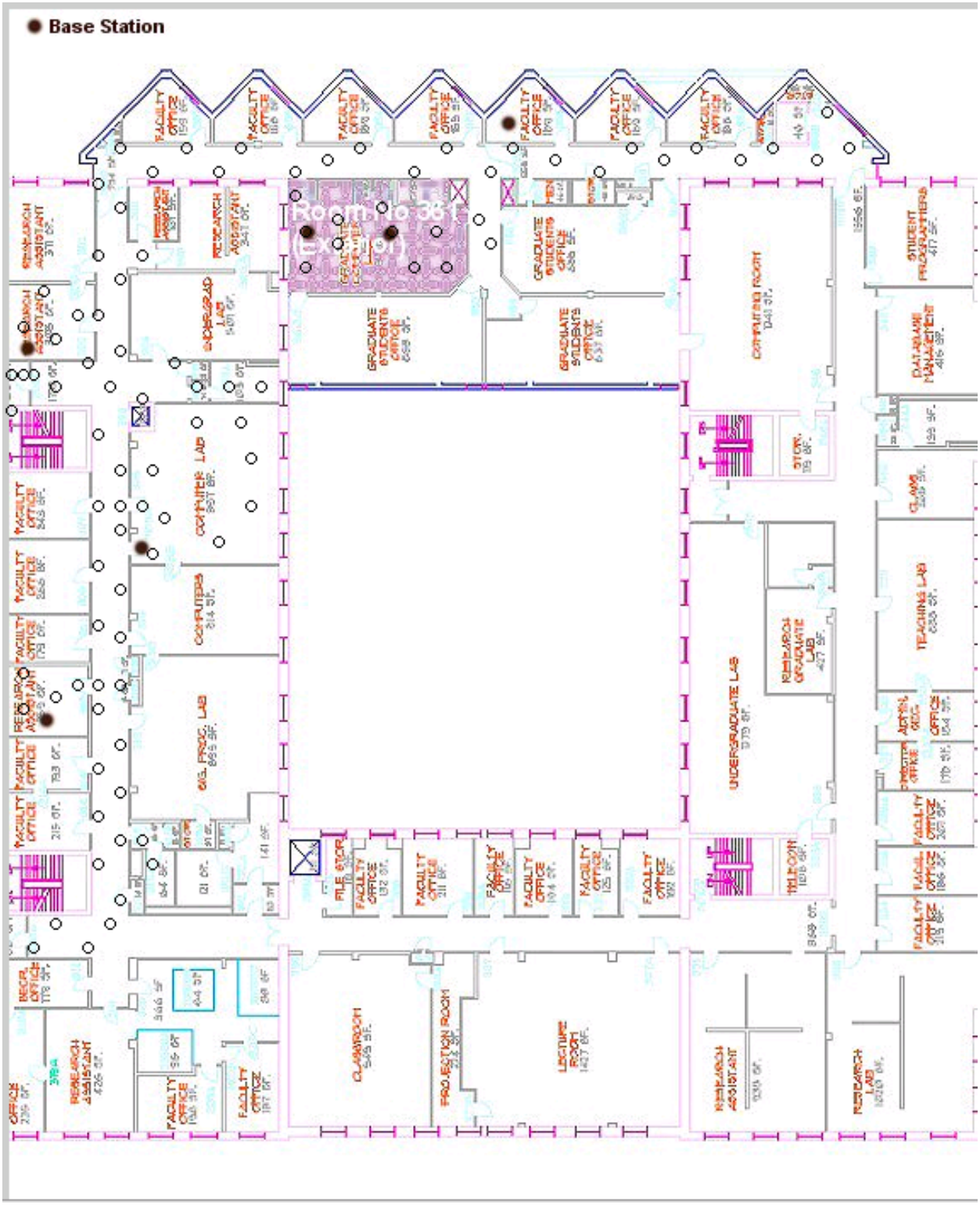

Figure 4.3: Finger-printing Map. The circles represent points for which data was collected.

The co-ordinates are embedded in a UDP packet and broadcasted by the mobile node. A hundred identical packets are transmitted at regular intervals of 20 ms each. Thus, the transmission lasts for 5 seconds, during which time the laptop is rotated around the point in order to capture packets transmitted in all directions (360°), to get a better estimate in the variation of RSSI with direction. We gathered data using approximately 100 datapoints (Fig 4.3).

2. Each base station is set in the monitor mode, running a sniffer to capture all packets broadcasted by the user. The base stations in the range of the user are able to capture the packets and extract the location of the user from the data field of the packet, as well as the RSSI from the PRISM2 header.

3. The sniffer also transmits UDP packets to the server, on the wired network, containing the location as well as the RSSI.

4. The server stores the labeled packets in a database where each entry looks like

   `<BSID> <RSSI> <X> <Y>`

   where BSID = base station identifier which, in our case, is the IP address of the base station.

During the data collection phase, the user has to be cautious in choosing points equidistant to each other and uniformly spread over the test-bed. This is needed for a fair representation of the system. The map is divided by a grid of 100x100.

4.4.2 Data Filtering

The Data Filtering process is undertaken once the data collection process has been completed. During this process, the database, which has entries as shown above, is filtered into separate tables with a common BSID and RSSI. This process can be described as a multi-step process:

1. The database is filtered into tables with common BSID and RSSI. An *awk* script is used for the process.

2. These tables, which might have multiple duplicate entries corresponding to the same co-ordinates, are filtered again by another *awk* script, to group duplicate entries by a single entry followed by their frequency. For example, a table named 1_80 (BSSID=1 and RSSI=80) looks like

   ```
   40 50
   38 48
   40 50
   ```

   This is converted to a table with the following entries:

   ```
   40 50 2
   38 48 1
   ```

3. A Gaussian filter is applied to smooth the data points in the tables to generate more entries for datapoints which might not have been covered during the data collection process. We use $\sigma_{loc} = 1.5$ as the standard deviation for the Gaussian filter, with a kernel size 7. This value has been obtained by making reasonable assumptions in

Sec 5.3.1. On filtering the two entries, they get converted to a table with 73 entries. Some of the sample datapoints are as follows:

```
40 48 0.174481
40 49 0.273134
40 50 0.306853
40 51 0.234434
40 52 0.116321
40 53 0.038292
41 45 0.002591
41 46 0.007871
```

4.5 Localization

Finally, the goal of our implementation is to estimate the location of the mobile userand show him on the map. This task consists of two parts, computation of the probability density of the user's location based on the the observed RSSI, and visualization of the user on the map. Section 4.5.1 describes how we estimate the probability density of the mobile user's location. In Section 4.5.2 describes the estimation of an exact location and plotting on the map.

4.5.1 Location Determination

The aim of the localization process is to compute the probability density function for the mobile node over the X-Y plane. Localization is a multi-step best described as follows:

1. All packets on the WLAN are captured by the Base stations on their wireless interface. The RSSI, as well as the MAC address of the Mobile node, is extracted from the

PRISM2 header, and transmitted as a UDP packet to the server through the wired interface.

2. The server attaches the IP address of the sender (called BSID).

3. A lookup for the BSID_RSSI file, which is created in the finger-printing process, leads us to the probability density function related to the particular set of BSID and measured RSSI.

4. The updation matrix ϕ_{update} is computed by averaging the matrices from the files between BSID_(RSSI-β) to BSID_(RSSI+β). We make reasonable assumptions for β in Sec. 5.3.2.

5. The localization algorithm described in Sec. 3.4 is implemented at the server.

6. The measurement model in Sec. **??** is implemented for each new packet. The location state estimate is updated as follows

$$\phi_{new} = \phi_{old} \cap \phi_{update}.$$

Assume that C represents the new estimate ϕ_{new}, A represents the old estimate ϕ_{old} and B which represents the updation Matrix ϕ_{update}; then an element $[i, j]$ is given by $C[i, j] = A[i, j] * B[i, j]$, $\forall\ i = 1, \ldots,$N and $j = 1, \ldots, N$.

7. The new location state estimate is normalized.

8. At every one-second interval, the location state estimate is filtered with a Gaussian to enable tracking of the mobile user. This is in accordance with the system state model

in Sec. 3.4.1.

9. The location estimate for the Mobile Node is updated every 1 second. This estimate is the expected value of its (x,y) from its pdf.

4.5.2 Visualization

The visualization tool (Fig. **??**) uses the output of the localization program to plot the location of the person on the Floor Map. The coordinates of the mobile node $(\hat{x}, \hat{y})$ are estimated by using

$$\hat{x} = E(X) = x_1 p_1 + x_2 p_2 + \ldots + x_k p_k = \sum_{i=1}^{k} x_i p_i,$$

and similarly,

$$\hat{y} = E(Y) = y_1 p_1 + y_2 p_2 + \ldots + y_k p_k = \sum_{i=1}^{k} y_i p_i,$$

where x_i is the sum of the elements of the entire column of the state representation, where the distance from the y-axis is x_i. The standard deviation of the estimated value of x, σ_x is calculated by

$$\sigma_x = \sqrt{E(X^2) - E(X)^2}.$$

For the purpose of visualization, we plot an ellipse with its center being location $(\hat{x}, \hat{y})$ and its width and height being $3\sigma_x, 3\sigma_y$.

4.5.3 Salient Features

- We maintain a cache of recently opened files, to improve the performance of the algorithm. Most packets from a mobile node in a particular time frame

have the RSSI values in a small range. Thus, there is a very high probability of finding the required file in the cache. This greatly reduces the processing time per packet.

- Any number of Mobile Nodes can be tracked simultaneously.
- All the processing is done in C for efficiency, and JAVA is used for the visualization.

4.6 Practical Problems Faced

A number of practical problems occured during the implementation of this algorithm. Some of them are listed below:

1. The Orinoco NICs on the iPAQs freeze up sending an *orinoco_lock_hardware* interrupt, when a large number of packets are captured, due to the limited buffer length on the iPAQ. We solved this problem by setting up our network on a channel which is not being used by any of the access points around the test-bed.

2. The Orinoco cards lock up after capturing packets for a duration of about 10-15 minutes. We solved this problem by running a parallel process which resets the NIC every 200 seconds.

3. We tried using Aironet PCI NICs and modified the driver to incorporate the signal strength for each packet in the monitor mode [47]. But due to other practical problems, we had to change to Orinoco gold NICs on iPAQs.

4. The Lucent Orinoco cards are not able to capture packets broadcasted with a Cisco Aironet card, if the Aironet card is associated with an Access point. This is because the Access points are also from the same vendor, Cisco, and communication between them has a shortened preamble. To avoid association with the access points, the mobile node was set up in ad-hoc mode.

4.7 Suggested Improvements

1. For demonstration purposes, and because of a lack of computing power, we used a fixed transmission power at 100mW. But, if the user varies his power, we would not be able to successfully track the user. In such a case, our implementation will either alert us of an intrusion, or result in an incorrect perception of the location. To avoid this, we would have to finger-print the test-bed with varying transmission power and add another dimension to our algorithm by estimating the transmission power in addition to the X and Y coordintes of the location. Considerable computing power would be required to estimate the position in real-time with a continuous flow of packets.

2. Human errors are caused by clicking on the wrong position or the finger-printing not being uniform. If these errors are minimized during the finger-printing phase, better results might be obtained.

3. More elaborate finger-printing with additional datapoints would increase the accuracy of the results.

4. The number of cells may be increased for greater accuracy, but this would also increase the processing by $O(n^2)$, while not guaranteeing significant improvements.

5. In our implementation, we use only channel 4 of the spectrum and can locate users only on that channel. To be able to block all intruders, we would have to scan all the channels on the WLAN, by using an effective hopper.

6. The iPAQs freeze up on receiving a large number of packets. Instead we should use more powerful machines such as a base station equipped with Orinoco PCI wireless NICs.

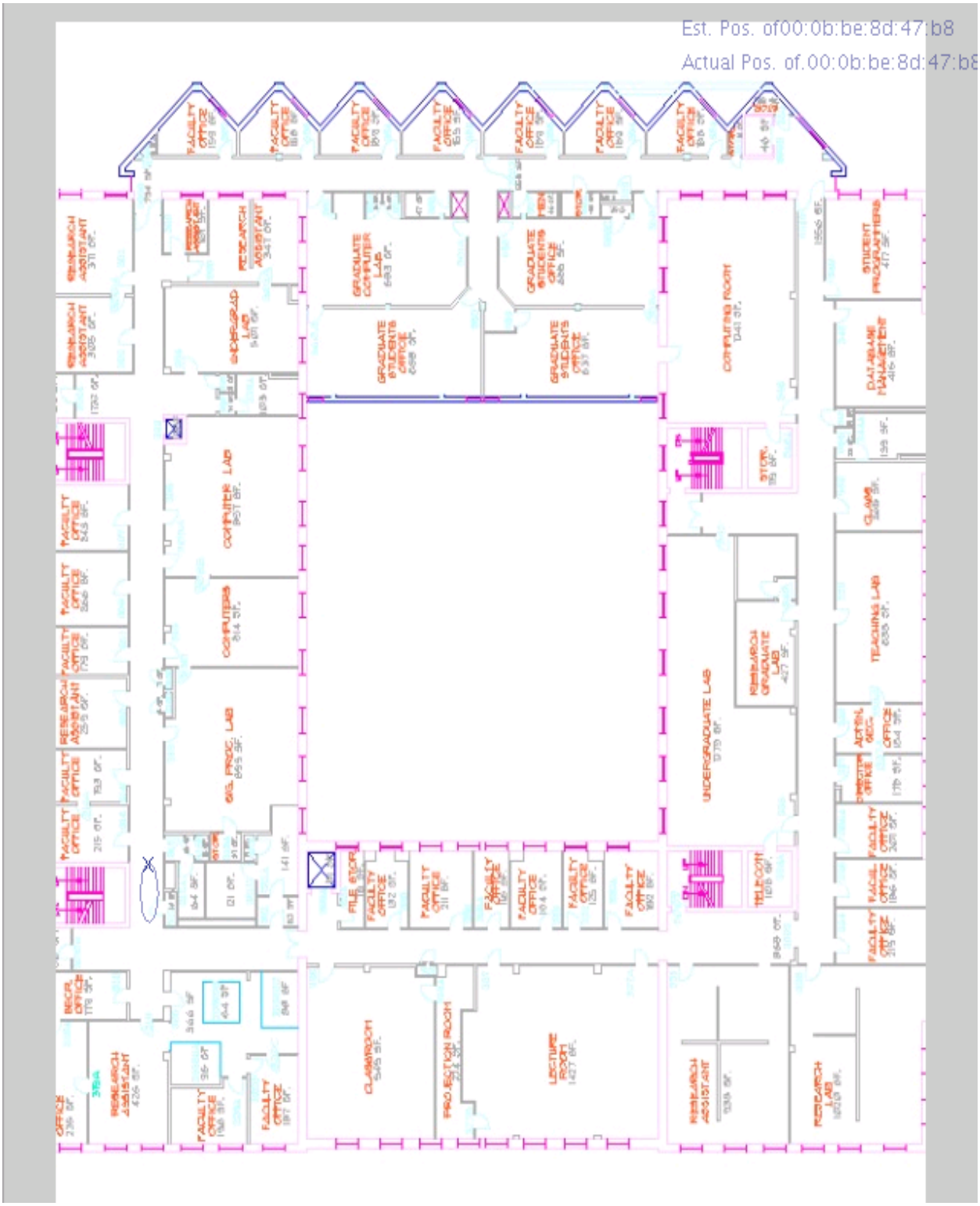

Figure 4.4: The Visualization Tool. The ellsipse represent the estimated region of the user

Chapter 5

Experimental Results

5.1 Overview

In Chapter 4, the proof-of-concept implementation of our algorithm was discussed. In this chapter, we discuss the results obtained from our implementation. The goal of our implementation is to locate an intruder and be able to identify if he is inside or outside a given security perimeter with a very high reliability. For our implementation, we set the security perimeter as the boundary of Room No.361(Fig. 4.1). A user inside the room is considered an intruder. In order to compare the estimated position of the user with his real co-ordinates, the real position of the user has to be known. A utility was implemented using which the user can move with the laptop and click on his own location on the map. This location is taken as the real location of the user while each packet that is transmitted from the laptop is used for calculating an estimated position.

5.2 Metrics

The goal while localizing a user is to minimize the average error in estimating the position of the mobile user and to reliably estimate if the user is inside or outside the specified security perimeter. The following metrics are used to judge the efficacy of our implementation.

5.2.1 Error in location estimate

The euclidean distance between the real position of the user and the estimated position given by our implementation, is defined as the error in location estimate. The performance of the algorithm with respect to the error in estimation can be visualized by plotting the percentile of the error estimate against the error in distance. Another measure that reflects the performance is the average error in estimation of the user's location.

5.2.2 Misinterpretation of position

When the actual position of the user is different from the estimated position, with respect to the security perimeter, we call it a misinterpretation. Misinterpretation of position can be of two types (a) False Positives and (b) False Negatives.

- A false positive is the case when the estimated position of the user is inside the security perimeter(positive) though the real position of the user is outside.
- A false negative is the case when the estimated position of the user is outside the security perimeter(negative) though the user is actually inside the security perimeter.

The probability of false positives is a very important metric, as the objective is to detect the intruder effectively (a user outside the security perimeter). It the more important to be able to detect and block intruders without any exceptions than to allow a legitimate user close to the security perimeter(False negatives). The probability of false positives should ideally be zero. The probability of false negatives is also an important measure which would reflect the inconvenience caused to a legitimate user in the vicinity of the security perimeter. It is a measure of the probability that if a user has been detected outside the perimeter, what is the chance that he is actually a legitimate user inside the security perimeter.

5.3 Design Parameters

In this section, we discuss the design parameters that were used for the implementation. As discusssed in Chapter 3, the problem can be divided into two parts, Calibration and Localization. Sec. 5.3.1 discuss the design parameters used for the calibration process, and Sec. 5.3.2 details our approach in finding reasonable values for design parameters used in the localization process.

5.3.1 Calibration

One of the primary calibration parameters is σ_{loc}, which is the standard deviation for the Gaussian filter used to generate new data-points. During the data collection process, selected data-points spread uniformly over the test-bed were chosen for the sake of convenience and to reduce the time taken to finger-print the test-bed. A Gaussian filter is then applied

over the collected data, to generate new data-points over the entire test-bed. Some of the factor to be considered while choosing σ_{loc} are:

- As discussed in Sec.2.5.1, the indoor signal strength does not have a normal distribution by nature, as indoor obstructions (e.g walls) cause a significant attenuation in signal power. Thus σ_{loc} should be sufficiently small (< 2 feet) considering the fact that actual RSSI measurements can vary significantly across very short distances (e.g across a wall).
- RSSI variation is very small over short open spaces (e.g open corridors), and we did not observe noticeable change in measured RSSI between locations 7-8 feet apart. This would lead us to set σ_{loc} to as much as 7-8 feet.
- In our implementation, our cell size is about 2.17 feet on the X-axis and 2.40 feet on the Y-axis. When we apply the Gaussian filter, we scale the Y-axis correspondingly to make the filter circular in nature, and thus keeping a unit equivalent to the unit on the X-axis, which is 2.17 feet.

Considering these factors, we choose the value of $\sigma_{loc} = 3.5$ feet which is a compromise solution considering the above factors. Thus $\sigma_{loc} \approx 1.5$ cell units.

5.3.2 Localization

System Model

Section 3.4.1 describes our system model, which helps us in continuous tracking of the mobile user. The main parameter in the system model is σ_{sys}, which is the standard deviation of

the Gaussian filter that is applied to the probability distribution matrix of the user at one-second intervals. The Gaussian filter is applied to account for the movement of the user and to track the user. Thus, assuming that with a 97% probability that the mobile user does not move at a speed more than 10 feet/sec, we keep $3\sigma_{sys} = 10$feet, therefore $\sigma_{sys} = 3.33$feet. Thus, $\sigma_{sys} \approx 1.5$ units.

Measurement Model

The measurement model (Sec. 3.4.2) describes our approach at finding the updation matrix to update the estimated position every time a new packet is received. The updation matrix is calculated by averaging probability density matrices for a range of RSSI values over which we can assume the RSSI to vary for the same location. During finger-printing, a large number of packets were collected for every position rotating the laptop around the point to account for variation in RSSI due to the directionality of the wireless NIC. This does cause a variation in the RSSI, but it might not be sufficient to reflect the variation in RSSI over time, because of other external factors. By looking at the collected packets from a single point, we assume the variation in RSSI(β) to be about 3 units. Thus averaging $2\beta = 6$ different RSSI values, which is similar to the deviation in RSSI assumed in [13].

5.3.3 Summary of Assumptions

In this section, three main design parameters, which could affect the results of our algorithm, were identified and reasonable assumptions about their values were made. The parameters are as follows:

1. σ_{loc}: The standard deviation of the probability density over the location of a person with a particular RSSI. A gaussian filter is used to generate new data-points from the data obtained during the Calibration process.

2. σ_{sys}: In the system model, we use a gaussian filter to emulate the movement of a person. σ_{sys} is the standard deviation used for the Gaussian filter applied to the state estimate matrix at every one-second intervals.

3. β: The assumed variation in the measured RSSI. We use β to average out the probability density functions corresponding to the RSSI. when the RSSI(λ) varies between $\lambda \pm \beta$, to find out the position estimate update function given the new RSSI value λ_i measured at BS_i.

We make reasonable assumptions and fix the parameters used in our implementation at $\sigma_{loc} = 1.5$, $\sigma_{sys} = 1.5$ and $\beta = 3$.

5.4 Results

Figure5.1 shows the error as a percentile value i.e the value on Y-axis marks the probability that the error is less than the distance x, which is marked on the X-axis. We observe that there is a 30% chance that the estimated location is less than 3.14 feet from the real position. Also, atleast 50% of the estimates are less than 5 feet off, and over 90% of the estimates are less than 10.8 feet from the real position. The average estimation error is *5.69* feet. Other metrics to observe are:

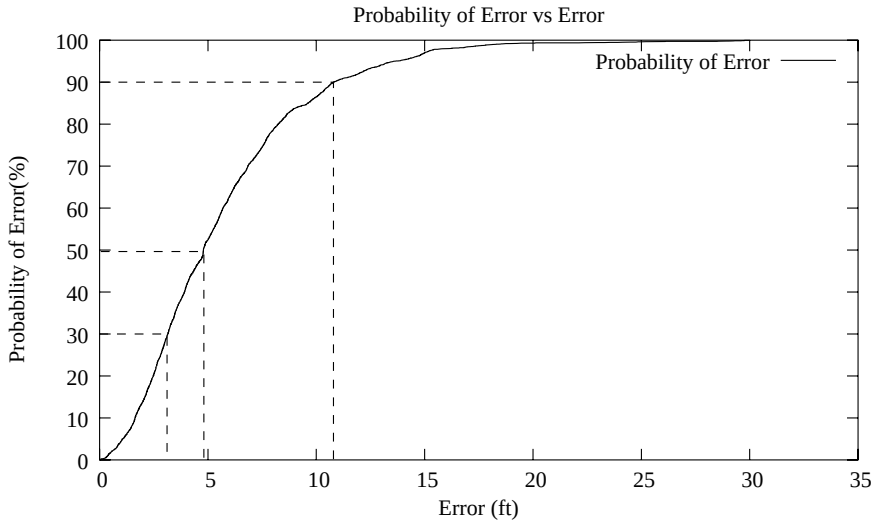

Figure 5.1: Percentile error with respect to the error in estimating the location. $\sigma_{loc} = 1.5, \sigma_{sys} = 1.5 and \beta = 3$

- False Positives = 0. This means a 0% error in reliably estimating if the user is outside the security perimeter. Thus, we are reliably able to determine if the user is outside the security perimeter.
- False Negatives = 10.4%. This translates to an approximately 10% chance that the user, who has been estimated to be outside the security perimeter, is actually inside the perimeter. When the user is close to the edge of the security perimeter, or standing close to an exit, he could be mistakenly identified as being outside the perimeter.

5.5 Effect of Variation in Design Parameters

In this section, we evaluate our choice of design parameters by varying the design parameter and observing the error in position estimates, while fixing the other parameters. In

Sec. 5.5.1, we observe the effect of varying the standard deviation of the Gaussian filter used for generating data-points during the Finger-printing process. Sec. 5.5.2 considers variation in the standard deviation of the Gaussian filter used in the system model. The system model aims to emulate the movement of the user to enable tracking. In Sec. 5.5.3, we observe the effect of varying the assumed variation in measured RSSI (β).

5.5.1 Effect of Varying the Standard Deviation (σ_{loc}) of the Gaussian Filter used for Finger-printing

Error in Position Estimate

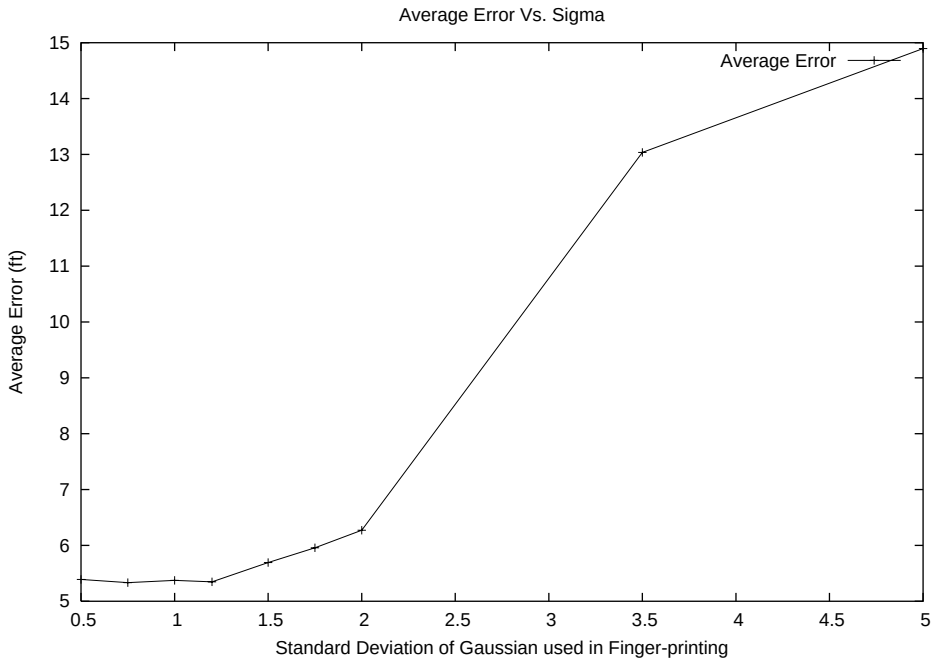

Figure 5.2: Variation of average error in estimation vs.variation in standard deviation(σ_{loc}) of the Gaussian filter used for finger-printing, with $\sigma_{sys} = 1.5$ and $\beta = 3$

From Fig. 5.2, we observe that the average error in estimation increases with an increase

in the standard deviation (σ_{loc}) for the Gaussian filter used in the finger-printing process. One might expect the error to be high for extremely small standard deviation, but this is not seen in the results. It is due to a dense finger-printing (Fig. 4.3) with respect to the area in the test-bed. This observation can be explained easily. For the experimental phase, we move in approximately the same locations as marked during the data-collection phase for the calibration process. The state estimate matrix ϕ (Sec. 3.3.2) becomes very sparse when σ_{loc} is reduced. Thus, the probability of the matrix becoming zero increases when the updation matrix is multiplied to it. Therefore, the stability of the system decreases when σ_{loc} is lowered, which is also observed in our experiments. The system could not estimate a solution with values of σ_{loc} lower than 0.5. On the other hand, when σ_{loc} is increased, we reduce the resolution of the system. The possibility of a wrong estimate increases as multiple locations might have a similar set of RSSI values. The average estimate increases on increase in σ_{loc}, which we can observe in both Fig.5.2 and Fig. 5.3. The sharp increase in average error for $\sigma_{loc} > 2$ would be because of the system estimating the user at a location with a somewhat similar set of expected RSSI at various BSs. Many such solutions are eliminated on using a small σ_{loc}. The minimum average error of approximately *5.3* feet is for $\sigma_{loc} \approx 1.2$.

False Negatives

In Fig. 5.4, we observe that the false negatives increase with an increase in the standard deviation of the Gaussian filter used for σ_{loc}. Fig. 5.4 closely correlates well with Fig.5.2, which is the average error in estimating the position. As the error in estimation increases,

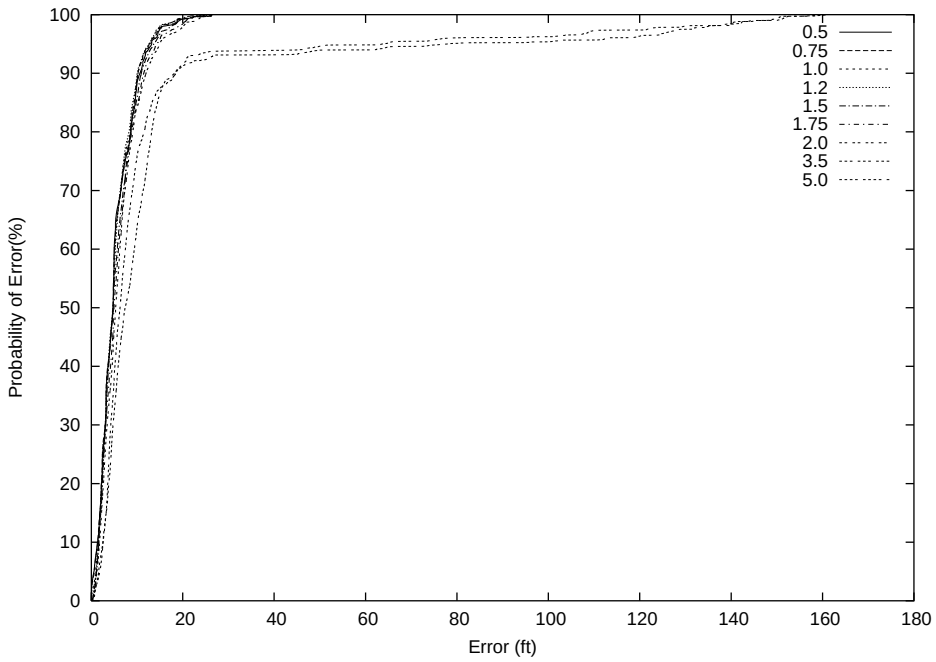

Figure 5.3: Effect of varying the standard deviation (σ_{loc}) of the Gaussian filter used in Finger-printing, on the Error distribution, keeping $\sigma_{sys} = 1.5$ and $\beta = 3$

we see an increase in the number of false negatives.

False Positives

An interesting observation is that the probability of false positives remains 0 on varying σ_{loc}. This could be due to the large difference in measured RSSI when the user is outside the security perimeter(i.e in room No.361 for our implementation) and when it is inside, because of the presence of two Base Stations in Room 361. None of the points outside Room 361(i.e inside the security perimeter) would provide an alternate solution for the set of RSSIs because of the large difference in RSSI inside and outside the room.Also, the effect on the estimated position is minimal because we use a circular Gaussian filter which spreads the values circularly over the region, and the mean remains the same. If the probability

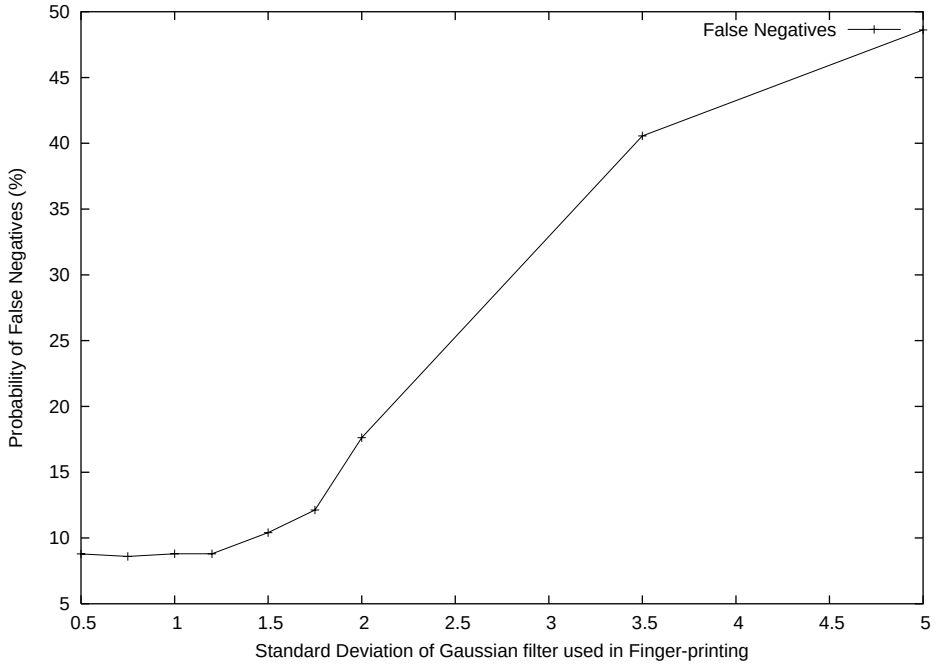

Figure 5.4: Effect of varying the Standard deviation of the Gaussian filter used in Finger-printing(σ_{loc}) on probability of False Negatives, keeping $\sigma_{sys} = 1.5$ and $\beta = 3$

density is extremely high inside a particular area, the estimated position would remain in the area even on applying a Gaussian filter with a high standard deviation.

5.5.2 Effect of varying the standard deviation(σ_{sys}), used for Gaussian filtering in the system model

Error in Position Estimate

In Figure 5.5, we observe that the average error in estimation has a minima for $\sigma_{sys} \approx 0.8$. The value of the minima is dependent on the average speed of the user during the test-run. While assuming $\sigma_{sys} = 1.5$ in Sec. 5.3.2, we had assumed the maximum speed of a user to be 10 feet/sec for calculating a σ_{sys}, which would lead to a stable system enabling continuous

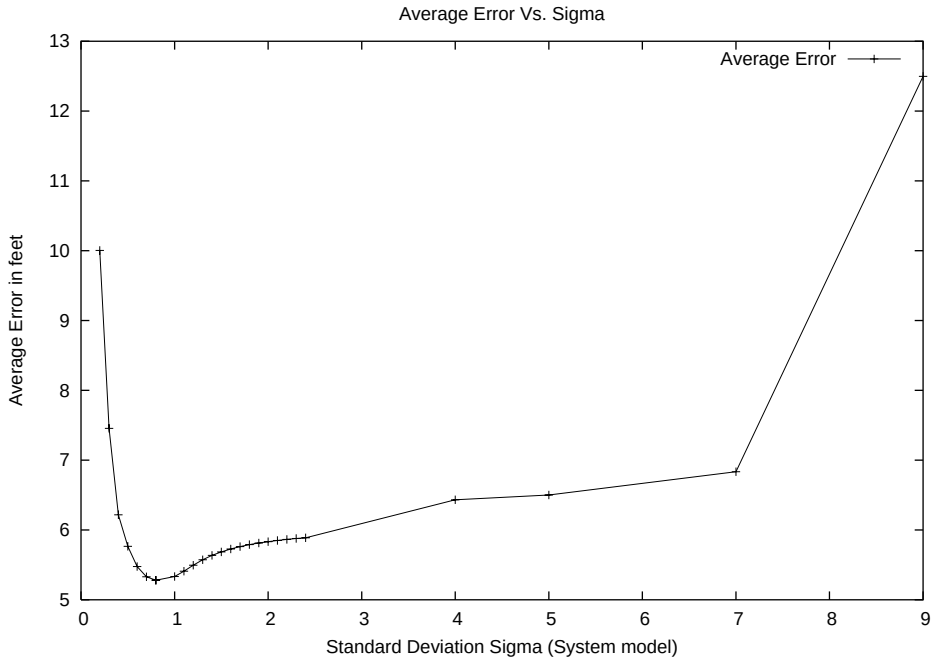

Figure 5.5: Effect of varying the standard deviation of the Gaussian filter in the system model(σ_{sys}) vs Average Error, with $\sigma_{loc} = 1.5$ and $\beta = 3$

tracking of the user. In reality, the average speed is much lower in normal conditions. In this case, even though the minima is clearly around $\sigma_{sys} = 0.8$, it would be better to keep σ_{sys} high to maintain a stable system, which can track mobile users moving at a higher speed. Initially, the average error in estimate increases very slowly with an increase in the standard deviation of the Gaussian filter. There is only a slight change in the error in estimation because a circular spreading does not shift the average or the estimated value of a probability distribution. As σ_{sys} is increased to values greater than 7, we see a marked increase in the average error. This is because the system starts converging to a solution which is otherwise eliminated when σ_{sys} is lower. We also observe that the average error estimate is higher for a very small value of σ_{sys} (e.g 0.2). This is due to a decrease in the

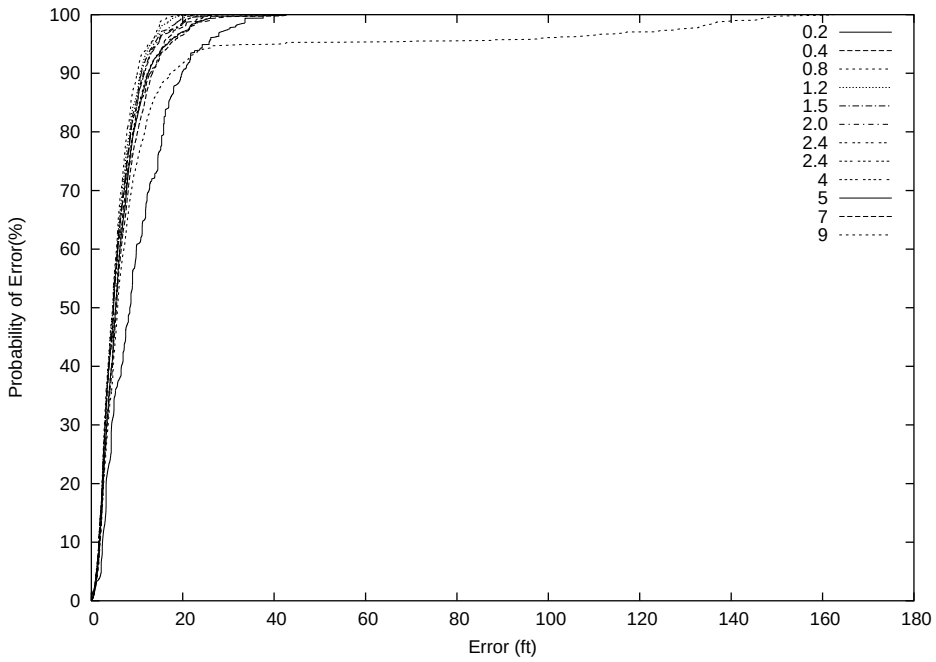

Figure 5.6: Effect of varying σ_{sys} on the Error distribution, $\sigma_{loc} = 1.5$ and $\beta = 3$

tracking speed as we decrease σ_{sys}. For values of σ_{sys} less than 0.2, the system fails to track the mobile user.

False Negatives

In Fig. 5.7, we see that the probability of false negatives increases with the increase in the standard deviation of the Gaussian filter used in the system model. We observe that the probability of false negatives first decreases on varying σ_{sys} from 0.2 to 0.3 and increases for values of $\sigma_{sys} > 1$. The decrease for values between 0.2 and 0.3 can be attributed to the difference in the tracking speed with respect to the actual average speed of the user during the course of the experimentation. The increase in the probability of the false negatives with an increase in σ_{sys} can be attributed to the spreading of the probability densities by

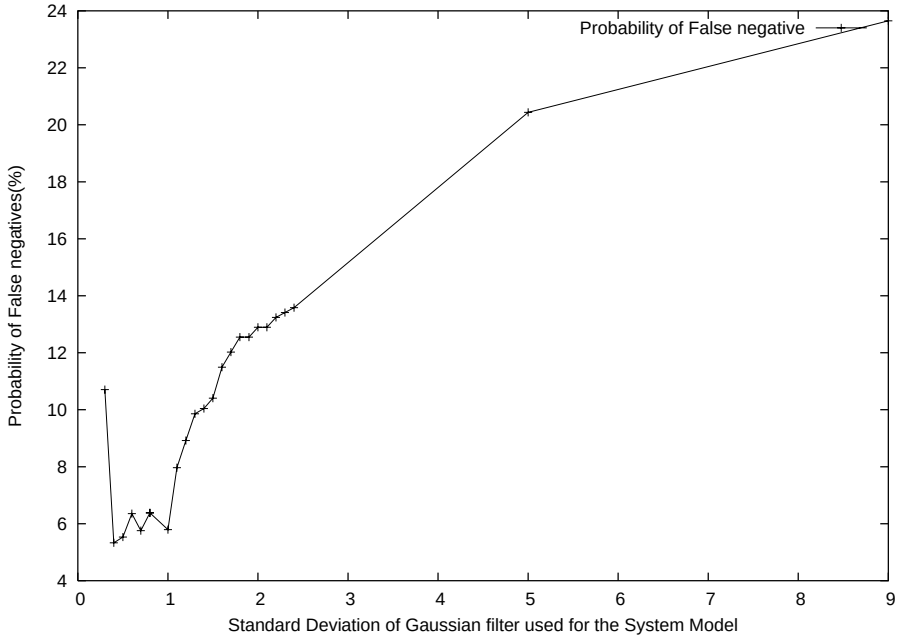

Figure 5.7: Effect of varying the standard deviation of the Gaussian filter used in the System model (σ_{sys}) on the Probability of False Negatives, with $\sigma_{loc} = 1.5$ and $\beta = 3$

the gaussian filter. Even a slight change in the estimated position can cause a sharp increase in the number of false negatives because of the rigid boundary.

False Positives

Our model performs extremely well in detecting the user to be outside the security perimeter if he is actually outside. The probability of false positives is always less than a 1%; in fact it is exactly equal to 0%, for $0.4 < \sigma_{sys} < 1.9$. Thus, we are able to locate the intruder who is outside the security perimeter with 100% accuracy.

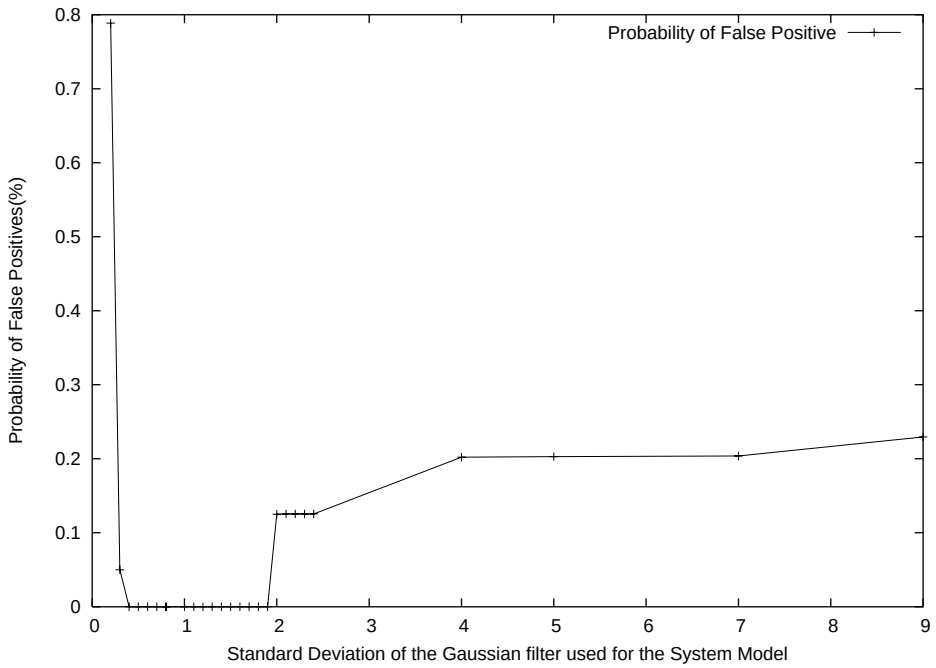

Figure 5.8: Effect of varying the standard deviation of the Gaussian filter used in the System model (σ_{sys}) on the probability of False Positives, with $\sigma_{loc} = 1.5$ and $\beta = 3$

5.5.3 Effect of varying the assumed variance(β) of the measured Received Signal Strength

Error in Position Estimate

In Fig. 5.9, we observe that by varying the assumed variance in the RSS (β), the average error starts increasing after $\beta > 6$. For $1 < \beta < 6$, the average error in estimate is almost constant. This means that for a specified location, the received signal strength at a particular BS does not vary by more than 6 RSSI units. And, in case we vary it more, the error in estimate starts increasing. Another interesting observation is that the average estimated error even on assuming the variance of just $\beta = 1$ is not higher than when we assume a more realistic variation in β of around 3. This can be explained by the fact

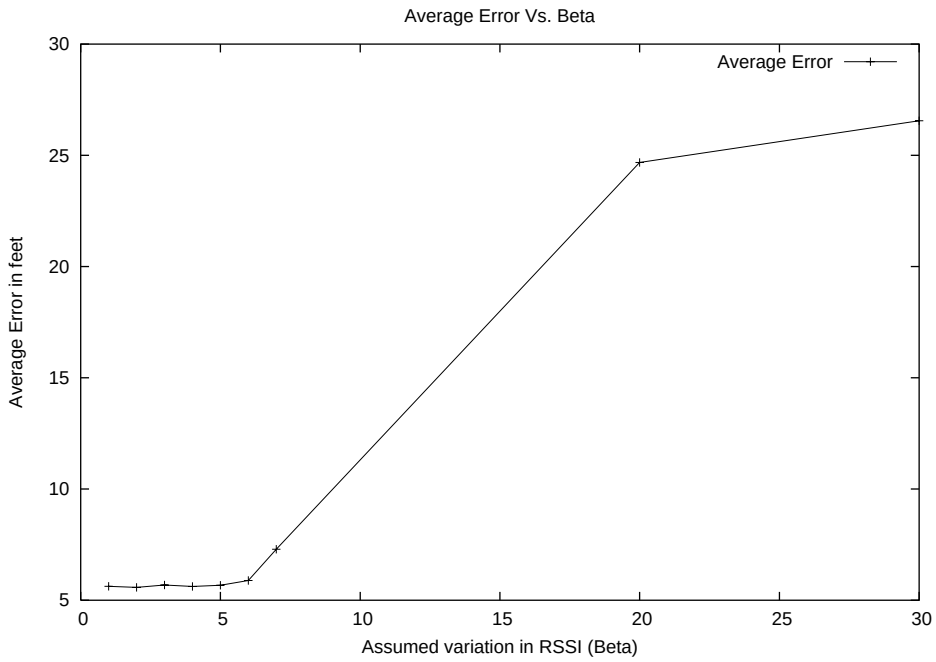

Figure 5.9: Effect of varying the assumed Variance in RSS(β) on the Average Error, with $\sigma_{loc} = 1.5$ and $\sigma_{sys} = 1.5$

that during the data collection process (Sec. 4.4.1), we obtained data for finger-printing by sending a large number of packets transmitted in different directions. The natural variation in the RSSI of these packets might be satisfying almost the entire set of RSSI values that can be recorded at a BS from a particular position. But, if we consider a zero deviation in RSSI i.e for $\beta = 0$, the system is not able to locate the user and becomes unstable. In Fig. 5.10, we can see the variation of error in estimation.

False Negatives

Fig. 5.11 shows the effect of varying β on the probability of false negatives. We observe that even though the average error in estimate hardly varies for $1 < \beta < 6$, there is a drop in the number of false negatives for the same region. This justifies our assumption of a

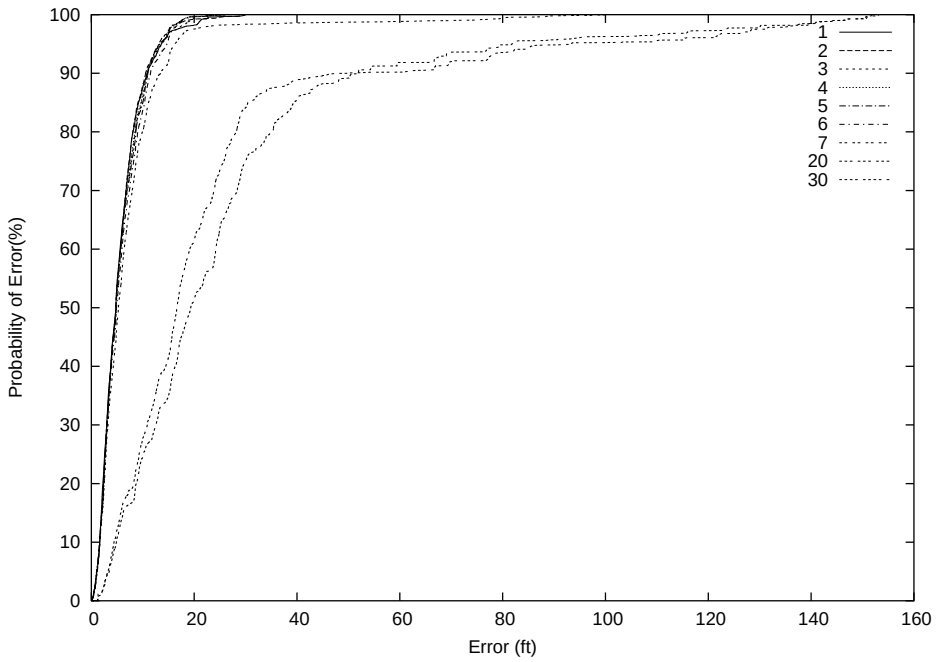

Figure 5.10: Effect of varying β on the Error distribution, $\sigma_{loc} = 3$ and $\sigma_{sys} = 3$

variation in observed RSSI. The number of false negatives rises dramatically as we increase β and it also corresponds with an increase in the average estimated error.

False Positives

The probability of a false positive is zero for all values of $2 < \beta < 30$. At $\beta = 1$, there exists a small probability of 0.2%, of estimating the user inside the security perimeter even though he is actually outside.

5.5.4 Effect of Varying Transmission Power

In the case when Finger-printing process is conducted at a particular transmission power level, and the intruder enters the network keeping his transmission power set at a different

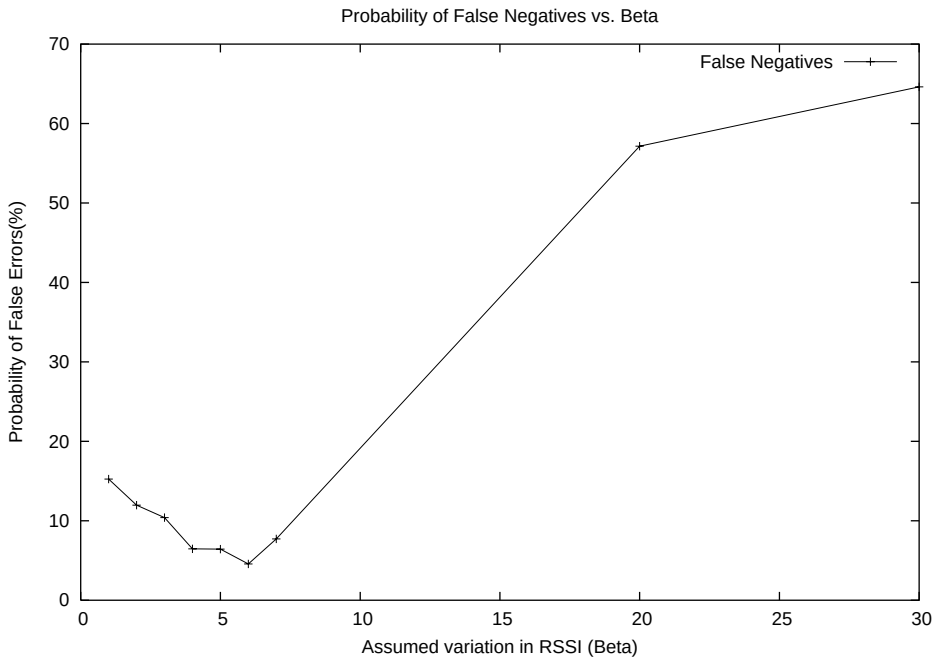

Figure 5.11: Effect of β on False Negatives, $\sigma_{loc} = 1.5$ and $\sigma_{sys} = 1.5$

level, his location estimate would not necessarily be close to his actual position.

The Cisco Aironet wireless NIC that we were using allowed us to vary the transmission power from 1mW to 100mW. The RSSI values are logarithmic in nature as they are proportional to the received power in dBm. We observed a variation of about 10-12 units of RSSI between power levels of 1 mW and 100mW. This deviation is not much compared to the deviation due to location. Thus the received signal strength changes by greater orders of magnitude with change in location as compared to change in the transmission power.

Thus, we decided to try out an experiment, taking a power level of 20 mW for the finger-printing process and then varying transmission power levels of the mobile node from 1 to 100mW and observing the error in position estimation.

Error in Position Estimate

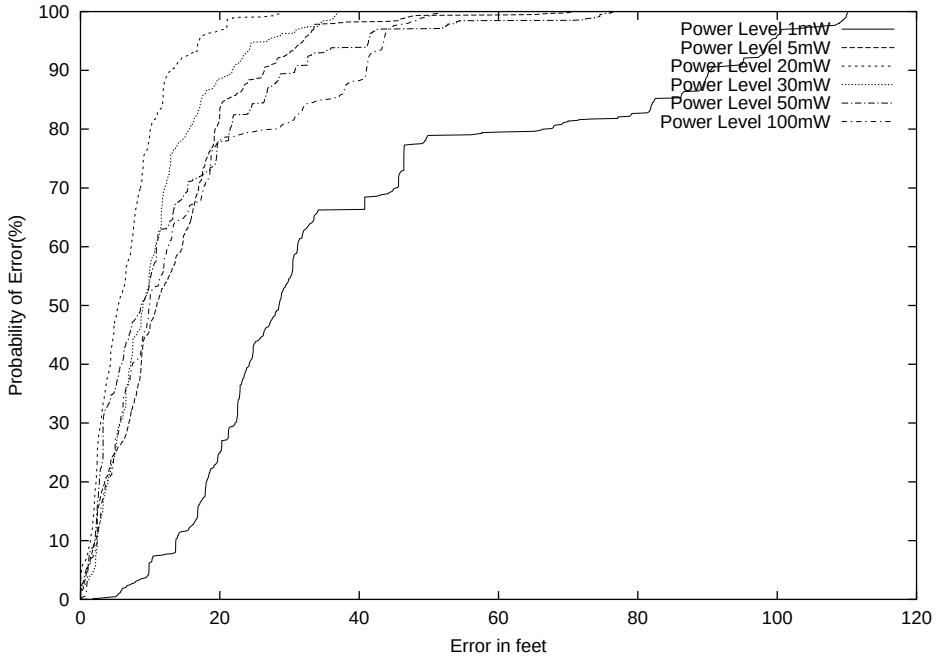

Figure 5.12: Effect on Error in Estimation with Varying Transmission Power, with $\sigma loc = 1.5$, $\sigma_{sys} = 3$ and $\beta = 7$

In Fig. 5.12 and Fig. 5.13, we see that the average error in estimation is least for 20mW which is the same power at which the finger-printing is performed. This is as expected. We had to keep the $\beta = 7$ for the estimation to work in the case of 1mW and 100mW runs.

For a transmission power level of 1mW, the average error in position estimate is about 39 ft.; it decreases to 13 feet for 5mW and 6 ft for 20mW, before increasing to 14 feet at 100mW.

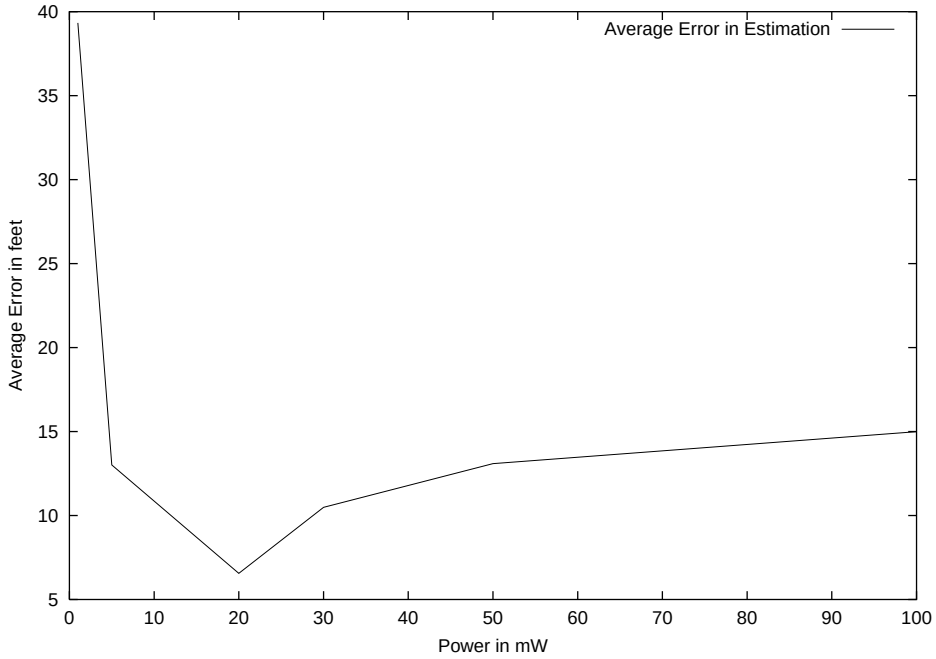

Figure 5.13: Average Error vs Transmission Power, with $\sigma loc = 1.5$, $\sigma_{sys} = 3$ and $\beta = 7$

False Negatives

The probability of a false negative for 20mW is about 10%, which means that 10% of the estimated positions outside the security perimeter are actually inside the perimeter. This number increases to 60% at a power level of 50mW.

False Positives

The probability of a false positive for 5mW is around 5%, which is still pretty low, but not 100% accurate. At power levels of 20mW, 30mW and 50mW, the probability of a false positive is 0%, thus indicating that the intruder is always detected.

5.6 Fault tolerance

To observe the effect of the failure of the base stations, we conduct our experiment by switching off base stations, one at a time. The BS's are shown in Fig. 5.14. We take two different cases (a) BS_3 failed and (b) BS_3 and BS_4 failed.

Figure 5.14: Base Station Map

From Fig. 5.15, we can see that the error in approximation increases as there is a fault in the network. But the localization process does not fail. Even when two out of the five BSs fail, we are able to locate the user with considerable accuracy.

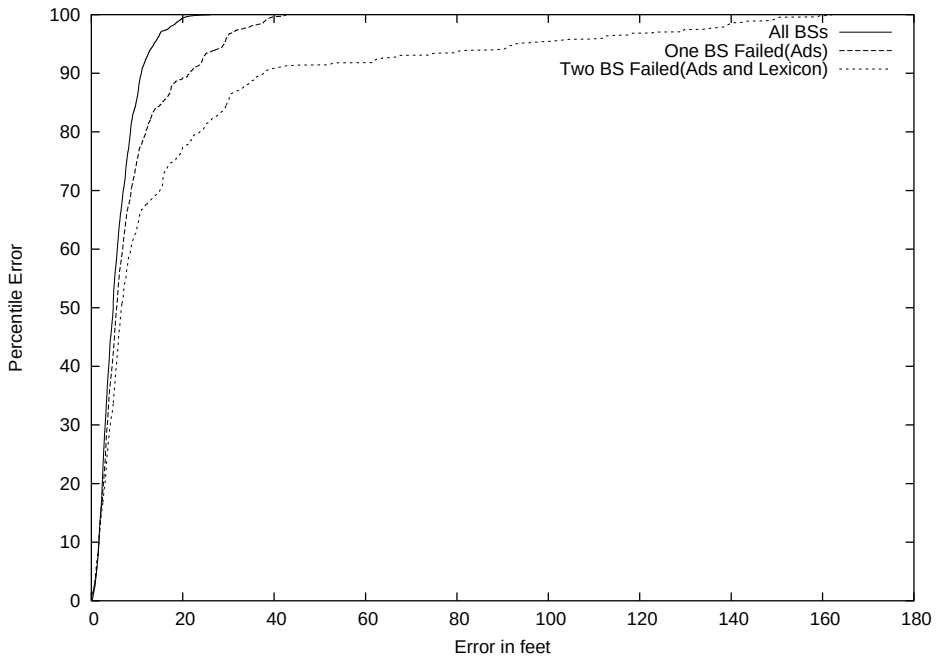

Figure 5.15: Effect on Performance with the Failure of BSs, with $\sigma loc = 1.5$, $\sigma_{sys} = 3$ and $\beta = 3$

- BS_3 failed:

  Average Error = 8.4 feet, False positives = 0.16%, False Negatives = 10.8

- BS_3 and BS_4 failed:

  Average Error = 18.1249 feet, False positives = 0.25% and False negatives = 25.55%

We observe that the average error increases considerably when the BSs are removed from the network, but the localization process still works. If more than two BSs develop a fault, the system becomes unstable and users only in a small part of the test-bed can be located.

5.7 Summary

In this chapter, we started by defining the metrics to gauge the performance of our implementation. The error in location estimation and the probability of misinterpretation of the user's location (inside or outside the security perimeter) stood out as the main criteria for judging the efficacy of the implementation. Some of the main results are as follows:

1. We achieve a 100% reliability in identifying the position of the intruder(who is outside the security perimeter).

2. The average error in estimation is 5.69 feet, with the design parameters $\sigma_{loc} = 1.5, \sigma_{sys} = 1.5$ and $\beta = 3$. 90% of all the estimates are less than 10 feet off from the real location.

3. On varying σ_{loc}, while keeping other design parameters constant, we find that the average error is minimum for $\sigma_{loc} = 1.2$ and is 5.34 feet.

4. On varying σ_{sys}, while keeping other design parameters constant, the average error is minimum for $\sigma_{sys} = 0.8$. The error is 5.28feet.

5. Variation of β while keeping other design paramaters constant leads to a minimum at $\beta = 2$, which is 5.57 feet.

6. The average error in estimation for $\sigma_{loc} = 1.2, \sigma_{sys} = 0.8$ and $\beta = 2$ is 5.04 feet with 0% False positives and 6.42% False negatives.

7. On varying the transmission power of the mobile node, the average error in estimating

the location increases when the transmission power is not the same as that, at the time of finger-printing. But as the RSSI variation is logarithmic in nature and the difference in RSSI values varies more with distance, rather than with the transmission power, we are still able to roughly locate the mobile node.

Chapter 6

Conclusion

The problem of localizing a user in a WLAN environment without his awareness or co-operation is a very interesting one. It has applications in security where a "parking lot attacker" can be blocked from accessing the network because of his physical location.

In this thesis, we proposed and implemented a novel method to locate the user in a WLAN using only the received signal strength of packets transmitted by him.Even though we aimed at providing only a coarse-grained localization, but an extremely reliable method to identity a user outside the defined security perimeter, our implementation was able to locate and continuosly track users with an average error in estimated position of around 5 feet. Apart from a reasonably fine-grained localization, we were also able to achieve a 100% accuracy in reliably identifying the intruder(user outside a fixed security perimeter).

References

[1] J. Geier, "Beating signal loss in wlans." http://80211planet.com/tutorials/article.php/1431101.

[2] W. A. Arbaugh, N. Shankar, and Y. J. Wan, "Your 802.11 wireless network has no clothes," *IEEE Wireless Communications*, pp. 44–51, Dec. 2002.

[3] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," *Lecture Notes in Computer Science*, vol. 2259, pp. 1–20, 2001.

[4] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: The insecurity of 802.11." http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html, 2001.

[5] A. Stubblefield, J. Ioannidis, and A. Rubin, "Using the Fluhrer, Mantin, and Shamir attack to break WEP," in *ATT Labs Technical Report TD-4ZCPZZ, Revision 2*, August 2001.

[6] W.Arbaugh, "An inductive chosen plaintext attack against wep/wep2." IEEE Document 802.11-01/230, May 2001.

[7] T. S. Group, "Airsnort homepage." `http://airsnort.shmoo.com`.

[8] P. D. A. Rager, "Wepcrack tool." `http://sourceforge.net/-projects/wepcrack`, 2001.

[9] A. M. William, "An initial security analysis of the ieee 802.1X standard," Tech. Rep. CS-TR-4328, University of Maryland, College Park, February 2002.

[10] N. Zorn, "Authentication gateway HOWTO." `http://www.linux.org/docs/ldp/howto/Authentication-Gateway-HOWTO/`.

[11] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *INFOCOM (2)*, pp. 775–784, 2000.

[12] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket location-support system," in *Mobile Computing and Networking*, pp. 32–43, 2000.

[13] P. Bahl and V. N. Padmanabhan, "Enhancements to the radar user location and tracking system," in *Microsoft Research Technical Report MSR-TR-2000-12*, 2000.

[14] M. Youssef, A. Agrawala, and U. Shankar, "Wlan location determination via clustering and probability distributions," March 2003.

[15] P. K. Chrysanthis, "On indoor position location with wireless lans," in *Proc. of the 13th IEEE Int'l Symposium on Personal, Indoor, and Mobile Radio Communications*, (Lisbon, Portugal), September 2002.

[16] M. Youssef and A. Agrawala, "On the optimality of WLAN location determination systems," Tech. Rep. UMIACS-TR 2003-29 and CS-TR 4459, University of Maryland, College Park, March 2003.

[17] R. Battiti, T. L. Nhat, and A. Villani, "Location-aware computing: a neural network model for determining location in wireless lans," Tech. Rep. DIT-5, Universita di Trento, Dipartimento di Informatica e Telecomunicazioni, 2002.

[18] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The Active Badge Location System," Tech. Rep. 92.1, Olivetti Research Ltd. (ORL), 24a Trumpington Street, Cambridge CB2 1QA, 1992.

[19] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The Anatomy of a Context-Aware Application," in *Mobile Computing and Networking*, pp. 59–68, 1999.

[20] A. M. Ladd, K. E. Bekris, G. Marceau, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Using wireless Ethernet for localization," in *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Lausanne, Switzerland), Sept. 2002.

[21] A. M. Ladd, K. E. Bekris, G. Marceau, A. Rudys, L. E. Kavraki, and D. Wallach, "Robotics-Based Location Sensing using Wireless Ethernet," in *Proc. of Eighth ACM International Conference on Mobile Computing and Networking (MOBICOM 2002)*, (Atlanta, Georgia), Sept. 2002.

[22] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[23] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *AAAI/IAAI*, pp. 343–349, 1999.

[24] S. Thrun, "Probabilistic algorithms in robotics," *AI Magazine*, vol. 21, no. 4, pp. 93–109, 2000.

[25] C. Olson, "Probabilistic self-localization for mobile robots," *IEEE Transactions on Robotics and Automation*, no. 1, pp. 55–66, Feb 2000.

[26] A. Ward, A. Jones, and A. Hopper, "A new location techniques for the active office," *IEEE Personal Communications*, Oct. 1997.

[27] J. Wolf, W. Burgard, and H. Burkhardt, "Robust vision-based localization for mobile robots using an image retrieval system based on invariant features."

[28] D. Burschka, J. Geiman, and G. Hager, "Optimal landmark configuration for vision-based control of mobile robots."

[29] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*. Springer-Verlag, 4th ed., 1997.

[30] F. Gustafsson and F. Gunnarsson, "Positioning using time-difference of arrival measurements."

[31] J. Caffery and G. Stuber, "Subscriber location in cdma cellular networks," *IEEE IEEE Trans. Veh. Technol*, vol. 47, pp. 406–416, May 1998.

[32] C. Drane, M. Macnaughtan, and C. Scott, "Positioning GSM telephones," *IEEE Communications Magazine*, vol. 36, pp. 46–54, Apr. 1998.

[33] G. G. Messier, M. Fattouche, and B. R. Petersen, "Locating an is-95 mobile using its signal," in *Proceedings of Wireless*, (Calgary, Alberta), July 1998.

[34] S. Capkun, M. Hamdi, and J. P. Hubaux, "GPS-free positioning in mobile ad-hoc networks," *Cluster Computing*, vol. 5, April 2002.

[35] N. Patwari, A. III, M. Perkins, N. Correal, and R. O'Dea, "Relative location estimation in wireless sensor networks." Submitted to IEEE Trans. on Sig. Proc.

[36] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low cost outdoor localization for very small devices," 2000.

[37] J. Bardwell, "Converting signal strength percentage to dbm values." `http://www.wildpackets.com/elements/whitepapers/Converting_Signal_Stren%gth.pdf`.

[38] M. Helén, J. Latvala, H. Ikonen, and J. Niittylahti, "Using calibration in RSSI-based location tracking system," in *Proc. of the 5th World Multiconference on Circuits, Systems, Communications & Computers (CSCC20001)*, 2001.

[39] W.-C. C. Yang-Han, "A novel prediction tool for indoor wireless lan under the microwave oven interference."

[40] K. Cheung, J. Sau, and R. Murch, "A new empirical model for indoor propagation prediction."

[41] A. Smailagic, D. P. Siewiorek, J. Anhalt, D. Kogan, and Y. Wang, "Location sensing and privacy in a context aware computing environment."

[42] G. Forney, "The Viterbi algorithm," in *IEEE*, vol. 61, pp. 268–278, 1973.

[43] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[44] "A probabilistic location service for wireless network environments," in *Ubiquitous computing*, Sept 2001.

[45] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE Transactions on Automatic Control*, vol. 45, pp. 910–927, may 2000.

[46] IEEE, *IEEE 802.11 Standard*, 1999 ed.

[47] "Modified aironet driver for capturing signal strengths." `http://www4.ncsu.edu/~vbharga/RSSIaironet.c`.

[48] "iPAQ h3870 reference guide." `http://h18007.www1.hp.com/support/files/HandheldiPAQ/us/download/11048.html`.

[49] "Orinoco classic gold pc card." `http://www.proxim.com/products/all/orinoco/docs/ds/gold_pccard.pdf`.

[50] A. Karunanidhi, "Linux on iPAQ in 21 easy steps." `http://www.cfar.umd.edu/~arvind/linuxonipaq.htm`.

[51] J. Hicks and R. Nelson, "Familiar v0.6 installation instructions." `http://familiar.handhelds.org/releases/v0.6.1/install/install.html`.

[52] "Orinoco monitor mode patch." `http://airsnort.shmoo.com/orinocoinfo.html`.

Appendix A

Manual

A.1 Setting up the test-bed

A.1.1 Installation of the iPAQs

System Specification

Hardware:

- iPAQ model no. H3870 [48]
- Compaq PC Card Expansion sleeve Plus
- Orinoco Gold NIC [49]

Software:

- Familiar Linux 0.6
- kernel 2.4.18-rmk3

Linux Installation

The iPAQs come with pre-installed Windows CE. For backing up WinCE and installation of linux on them, a number of references are available on the web. The tutorials at [50] [51] proved helpful while setting up linux on the iPAQs. Familiar Linux 0.6 with Opie was installed on each of the iPAQs. The package *qt-embedded-rotation* had to be installed later to correct the display on the iPAQ.

Setting up the Serial Link

On the Linux host, a new file */etc/ppp/peers/ipaq* is created and it should look like

```
-detach
noauth
nocrtscts
lock
local
user ppp
connect '/usr/sbin/chat -v -t3 ogin--login: ppp'
/dev/ttyS0
115200
192.168.0.1:192.168.0.2
proxyarp
```

The iPAQ is connected to COM1, and thus we use */dev/tty0*. We also assign the IP address "192.168.0.1" to the desktop, and "192.168.0.2". These can be changed according to the required IP addresses. On the iPAQ, create a file named */etc/ppp/options* which looks like

```
-detach
defaultroute
noauth
local
nocrtscts
lock
lcp-echo-interval 5
lcp-echo-failure 3
/dev/ttySA0
```

```
115200
```

To connect to the iPAQ,

```
[root@neo root]# pppd call ipaq
```

Orinoco Driver Installation

The orinoco driver *orinoco-0.13b* works in the *Monitor Mode* on the iPAQs. It has to be patched, using the patch downloaded from [52]. The kernel source is available at [**?**], and the arm-linux compilers are available at [**?**]. The compiled drivers for the iPAQs are available at [**?**]. The files *orinoco.o*, *orinoco_cs.o*, *hermes.o* have to be copied to the drivers directory on the ipaq.

```
[root@neo bin]# scp *.o root@ipaq:/lib/modules/2.4.18-rmk3/kernel/drivers/
net/wireless/
```

The file */etc/pcmcia/config* is also edited to bind the *orinoco_cs* driver to the Lucent Orinoco cards.

```
card "Lucent Technologies WaveLAN Adapter"
version "Lucent Technologies", "WaveLAN/PCMCIA"
bind "orinoco_cs"
```

After changing the driver, reboot the iPAQ and insert the orinoco card into the PC card slot of the sleeve. The lights should start blinking indicating that the card is working.

A.1.2 Setting up the Hosts

While setting up the serial link for the iPAQs, we created a new file */etc/ppp/peers/ipaq*. Next, we create a new entry for the iPAQ in the */etc/hosts* file e.g.

```
192.168.0.2 ipaq
```

A.2 Software Installation

All the relevant code, and executables are available in Final.tar.gz [**?**]. Table **??** lists the files and their respective functionalities.

```
[root@neo Project] tar -zxvf Final.tar.gz
```

A complete directory structure would be created.

A.2.1 Setting up the iPAQs

The required files are present in the ipaq/bin folder. They are copied to the ipaq.

```
[root@neo Project] scp Final/ipaq/bin/* root@ipaq:/
```

A.2.2 Setting up the Server

The required files are present in the Server/bin folder. Copy the files to a new directory which would be used for running the server and keeping the collecting all data.

A.2.3 Setting up the mobile node (Laptop) for Finger-printing

The required files for setting up the Laptop are in the directory Laptop/bin. Copy all the files to a new directory on the Laptop used for finger-printing the area. The file "map2" has to be replaced with the map of the Test-bed in jpeg format.

A.3 Finger-Printing

The finger-printing or the mapping process requires the test-bed to be completely ready. The iPAQs are connected to the wired network through the host computers connected via a serial line. The server and the mobile node(a laptop) should be configured before starting the mapping process.

Once the network is setup, the finger-printing process would consist of the following steps.

1. Starting the server

   ```
   [root@neo Server]# ./server
   ```

2. Starting the iPAQs in mapping mode

   ```
   [root@ipaq ]# ./startmapping
   ```

3. Starting the mapping software on the Laptop

   ```
   [root@Zion Project]# java MapCoordinator
   ```

4. Collecting Sample data-points To finger-print the test-bed, locate your position on the map on the Laptop, and click on it at every position you want to map. The data-points should be uniformly distributed over the test-bed. Data is transmitted for 5 seconds after clicking on a location. One should rotate and collect samples for varying directions during this time.

5. Processing the data On the server,

   ```
   [root@neo Project] sh testwrap
   ```

A.4 Localization

The actual localization process can be started by setting up the network as in the case of finger-printing and then running the following commands.

```
On the iPAQs,

[root@ipaq ]# ./startlocalize

On the server,

[root@neo Project]# java Locator
```

Filename	Location	Mode	Functionality
server	Server	Finger-printing	Collect data
MapCoordinator	Mobile Node	Finger-printing	Starts Map and sends Location
startall	Server	Localization	Script that initializes and starts localizing software on all iPAQs
startlocating	iPAQ	Localization	Starts parallel threads to re-initialize card and the sniffer
startmapping	iPAQ	Finger-Printing	Starts parallel threads to re-initialize card and the sniffer
tablewrap	Server	Finger-Printing	Script to filter the Database created while Finger-Printing, for Localization
Locator	Server	Localization	Visualization tool to see the intruder on the Map

Table A.1: List of files and their functionality