View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Illinois Digital Environment for Access to Learning and Scholarship Repository

© 2018 Jerry R. Guo

ANALYSIS OF BLUETOOTH LOW ENERGY BEACONS IN INDOOR LOCALIZATION POLICY AND APPLICATION

BY

JERRY R. GUO

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Adviser:

Professor Carl Gunter

ABSTRACT

Indoor localization systems have yet to be fully realized, but recent research is getting close to providing a cheap, system that will change the way that we interact with indoor computing systems. Internet of Things is growing and providing cheap, alternatives that may help realize this goal. This thesis examines the potential of a localization system using Bluetooth low energy beacons and the application of context-aware policy enforcement on mobile devices through discussion and implementation.

The core of the idea is to develop a basic framework that encompasses the basis of context, policies, and how context can tie into policy enforcement. We show how context-based policies can be formalized and implemented on a mobile architecture with two different case studies.

In addition to the policy engine, research is done into developing a cheap indoor localization system to power the context-based policies in a cheap and low setup cost method based off of existing RSSI distance estimation research. Bluetooth low energy beacons are explored as a cheap method to enable localization indoors by using the beacons to collect signal readings and create distance models empirically. Further techniques are discussed to enhance the absolute error through signal smoothing and filtering. Finally, the system architecture is fully explained and implemented with real-world tests in multiple environments.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I want to express my deepest gratitude towards my thesis Professor, Carl Gunter, for all his advice, time, and support that he has given me throughout my time at the University of Illinois. He has helped me along my path and I only have the deepest respect for him. I would also like to express my gratitude towards all of my friends and family who have continually provided support throughout my life. I would have never made it here without them.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Localization technologies is a rapidly growing field with more and more resources poured into developing it each year. A promising avenue of development is through the Internet of Things, which can help lower the cost of these technologies through the sensor heavy devices produced. This thesis focuses on the intersection between the Internet of Things and localization to develop a framework that allows for context-aware policies to be enforced on mobile devices.

1.1 MOTIVATION

Internet of Things (IoT) is becoming more popular due to their cheap nature and how they integrate sensors to generate massive amounts of data with an ease of access that has never been matched before. In the future, IoT is expected to massively contribute to both home and business applications [1]. Most initial work on IoT has explored radio-frequency identification (RFID) tags but [2] [3] later work explores more complex technologies including Bluetooth. One potential area of exploration is using cheap IoT devices, which include Bluetooth radios, to formalize and develop an indoor positioning system (IPS) that is cheap and easy to deploy in buildings across the globe.

IPS aims to provide an easy way to track user locations while indoors, very similar to how GPS functions outdoors. However, due to the nature of indoor environments, there can be a lot more variation in a more limited space. The importance and difficulty of solving these challenges are further emphasized when Microsoft holds a yearly Indoor Localization Competition [4]. These challenges include the placement of furniture, material of the walls, multipath propagation, loss of line of sight, and more. All of these challenges make it difficult to develop solutions that are accurate in unique environments. To attempt to address these problems, there is a lot of research and development being done. For example, initial work in the early and mid-2000s looked into using RFID and Wifi access points as methods to determine a user's location in a building [5] [6] [7]. Other work has looked into the fusion of phone sensor data with Wifi, RFID, Bluetooth, and more [8] but no system really solves the localization problem [9].

A potentially useful vector to attack this problem is through the use of beacons. Beacons are cheap and energy efficient and also provide plentiful sensor information. If successful, beacons can provide a cheap avenue of deployment with very little setup time and cost. One common type of these beacons is Bluetooth Low Energy (BLE) beacons. BLE beacons

are powered by Bluetooth Low Energy technology and they are relatively cheap while also providing an easy way to stream sensor data (temperature, humidity, etc.) to any listening clients. These beacons and other similar devices being used in various technologies [10]. Using these beacons can potentially allow for cheap localization to enable context-aware applications [11] [12]. This thesis explores the intersection between IPS and IoT to try to develop more context-aware applications by using context-aware policies on mobile devices.

One example could be in the health care industry. Getting doctors the correct patients charts and information is a necessary but time-consuming process. With a context-aware system, a doctor could automatically be given the appropriate charts and information, simplifying the process and getting the patients the help they need faster. In a different context, students in schools could have their phones silenced only in the classroom and during class to help facilitate a learning environment. New buildings are difficult to navigate, but a context-aware system could give auditory guidance to help the visually impaired navigate the building. There are a countless number of different situations where context-aware policies could change the way that we interact with the world and this thesis tries to address some of the current problems.

1.2 CONTRIBUTIONS

This thesis explores the formalization and development of a context-aware policy system and engine for enforcing policies on a mobile platform. In addition, there is exploratory work done on developing a localization system using BLE beacons to ensure a low-cost development and deployment. The main contributions of this thesis includes:

1. We formulated a set of policy actions given application and context. Outlined how to properly enforce the given policies on a myriad of mobile devices.

2. We designed and implemented a context-aware policy enforcement engine using the formalization and demonstrated two different case studies with the system.

3. Did preliminary research into a cheap IPS solution using Bluetooth Low Energy beacons.

    (a) We discuss the development of an appropriate model that approximates the distance from a given device to a BLE beacon using the received signals.

    (b) We tested the overall feasibility of using BLE beacons to determine a user's location within a building in multiple real-world environments.

1.3 THESIS STRUCTURE

The rest of the thesis is structured as follows:

- Chapter 2 introduces the necessary background concepts needed to understand context-aware policies and fundamental technologies used in the localization system.
- Chapter 3 explains the underlying framework and formalization for context-aware policies and their enforcement of mobile devices.
- Chapter 4 discusses the experimental design behind an economical localization system.
- Chapter 5 highlights all the experiments and results from the localization system development and setup.
- Chapter 6 discusses the results and explores potential future work.
- Chapter 7 concludes the thesis.

CHAPTER 2: BACKGROUND

2.1 CONTEXT-AWARE POLICY

This section will detail basic background information about the context-aware policies. The basic background is given for conceptualizing context and policies before a formalized definition is given in Chapter 3.

2.1.1 Mobile Device and OS

A mobile device is a broad term that can refer to a wide range of devices. For this work, we will only be taking a look at smartphones and examining Android in particular. Android is an open source operating system designed for mobile devices. It is a software stack on top of the Linux kernel with a middleware layer and user applications on top [13]. Due to the ease of development, this is the mobile OS used throughout the rest of this work.

2.1.2 Context

Context is made up of three different factors here. However, it can be expanded to include more variables as well.

- Location
- Server Time
- Current User

Location can have multiple levels of detail. The true position is where we know the location of the device in (x, y) coordinates. This level of accuracy is hard to achieve due to a number of factors that will be discussed later. For this thesis, we need the granularity of determining which room a given device is in to derive context. For example, this is distinguishing if a device is a classroom or an office.

Server time is determining when exactly the policy should be enforced. Our policy engine is running on a centralized server setup and the server time is defined as the current system time that the server runs on. Some policies defined may have constraints on when they should be applied and as a result, have a defined time range dictated by the server time. An example of this could be enforcing silenced phones only during class hours to develop a better learning environment.

The user is another factor that plays into context. Only certain users should get specific policies applied to them. For example, only students should their phones silenced during lecture but teachers should not get their phones silenced during lectures due to potential emergencies.

2.1.3 Policy

A policy is a set of statements or actions that will on top of Android. Android has various settings and applications that can be interacted with. The set of actions a policy can perform will directly change the values stored in these settings or send messages/intents to applications to perform complex procedures and methods. For example, a device configuration policy will any setting in Android and deals with any system variable change. Whereas an application policy will deal with user applications to help give context and procedures to various applications.

2.2 INDOOR POSITIONING SYSTEM

The following sections cover information related to indoor positioning/localization technologies.

2.2.1 Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a Bluetooth protocol that is built for low-power wireless technology. Tests have shown that BLE is very energy efficient and can be used for a variety of applications [14]. These applications generally involve short-range control or monitoring applications [15]. It is expected that BLE is to be incorporated into billions of devices in the future. Since this technology is designed to be low energy, BLE devices transmit differently from normal Bluetooth devices[16].

All BLE devices will use the Generic Attribute Profile (GATT) that is documented in the BLE specifications. In GATT there are the following:

- **Client**

  A device that initiates commands and requests. This would generally be a user's smartphone.

- **Server**

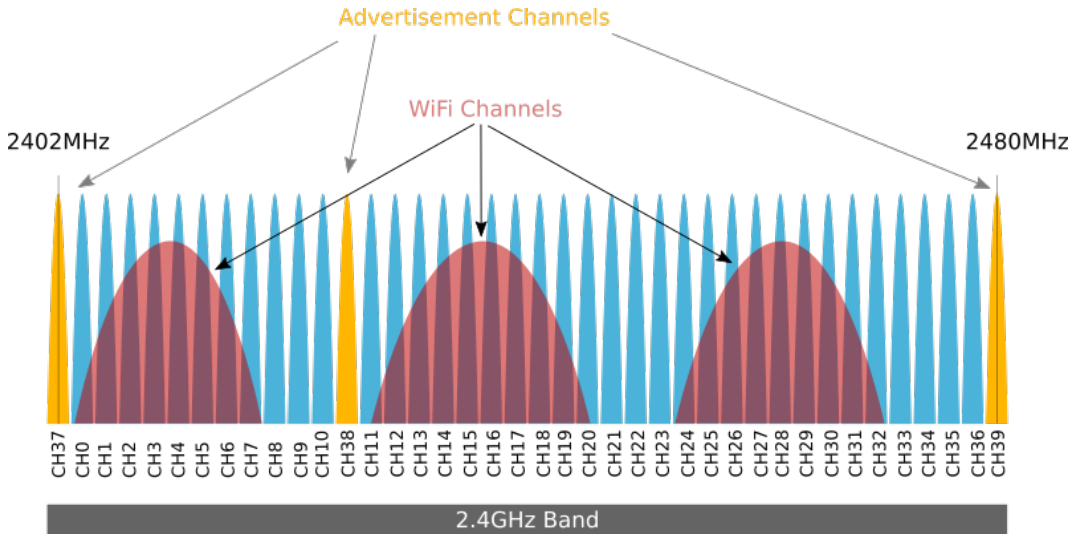

Figure 2.1: Diagram of the Bluetooth Low Energy broadcast spectrum. The three advertising channels are highlighted and can be seen as spread out across the spectrum. [17]

A beacon or other device that receives the client's commands and requests and returns responses. An example would be a humidity sensor.

- **Characteristic**

  A data value that is being transferred. For example, this would be the current temperature read by a sensor.

- **Service**

  A collection of characteristics. An example would be a humidity reading along with a time stamp since the last measurements.

- **Descriptor**

  Provides additional information about a characteristic. For example, a temperature reading could have a descriptor if the measurement was in Fahrenheit or Celsius.

- **Identifiers**

  Services, characteristics, and descriptors are all attributes identified by UUIDs, which are generally assigned by the device manufacturers.

BLE is on the 2.4GHz spectrum, from 2402MHz to 2480MHz, which is the same space that Wifi uses. This can lead to crowding of the bandwidth causing packet drops and loss from the BLE beacons. BLE uses Gaussian frequency shift keying (GFSK) modulation over 40 different channels [18]. GFSK works by smoothing out the frequency shifts with a Gaussian

filter to reduce sideband power and interference with neighboring channels. The 40 channels are 1 MHz wide and are spaced 2MHz apart with three designated advertising channels on channels 37,38 and 39. Despite how they are named, these channels are spread out across the 2.4GHz spectrum to try and prevent interference if an advertising channel is busy. Channels 37,38, and 39 are only allowed for sending advertisement packets and nothing else. All other traffic will be located on other channels which are decided through the client and server handshakes. Beacons will advertise on each advertising channel with a time delay of a fixed interval and a random value to try and prevent collisions. An overview of the BLE spectrum can be seen in Figure 2.1. One detail to note is the positioning of the three advertising channels across the spectrum.

BLE has a single packet format for both advertising and data transmissions. This packet has a preamble, access address, protocol data unit (PDU), and a cyclic redundancy check (CRC). The PDU part of the packet determines if the packet type is for advertising or a data packet. There are different PDU types for advertising based on what the device is trying to accomplish. Below is the list of different advertising packets.

- `ADV_IND`

  This is advertising indications, where a device requests connections to a central device. For example, a smart health device is requesting to connect to any device.

- `ADV_DIRECT_IND`

  Very similar to a `ADV_IND` packet, but has a targeted device that it wants to connect to. For example, a smartwatch is trying to connect to a specific phone.

- `ADV_NONCONN_IND`

  Nonconnectable devices will broadcast this. Essentially, advertising information to any devices that are listening. For example, beacons will broadcast packets with information.

- `ADV_SCAN_IND`

  Similar to `ADV_NONCONN_IND`, but there is additional information via scan responses. For example, a tracking beacon may allow a server to request more information about the payload.

For long-term connections, you will want to use either `ADV_IND` or `ADV_DIRECT_IND`, depending on the exact application. For our case, we use `ADV_SCAN_IND` to only retrieve received signal strength information from them.

2.2.2 Beacons

A beacon is a small BLE radio transmitter. They are made up of a small CPU, radio, and batteries. It generally only will send out advertisement packets on a fixed interval and generally, no direct connections are made. This lets multiple devices scan the advertisement channels for these packets to gain the sensor information from the beacon. Beacons can have a variety of sensors. Some common sensors are temperature, humidity, and signal strength. The unique ID allows for phones to identify which beacon they are communicating with. For this work, two different beacons are tested: Estimote and Feasycom beacons.

Beacons have multiple settings that affect how they transmit packets. First, there are multiple different beacon packet specifications due to different manufacturers. There is iBeacon, Eddystone, Estimote, AltBeacon, and more. These all have different ways that information is encoded in the PDU of the advertising packets and you must follow the specifications to properly decode the information. These packets can be transmitted at different power levels, this power level is known as the transmission level. The higher the transmission level, the clearer the signal will be. However, that can also introduce multiple problems due to a stronger signal strength. A stronger signal strength means it is easier to generate multi-path propagation from bouncing off of obstacles and causing inaccurate signal readings at the destination.

2.2.3 Received Signal Strength Indicator

Received Signal Strength Indicator or RSSI is a measurement of the power present in a received radio signal [19]. RSSI is a signal strength percentage and is a relative measurement that is generally defined by chipset manufacturers and not the Bluetooth specifications. RSSI measurements are negative, where the higher values indicate the more power received in the signal and lower indicates the signal is weaker. However, this measurement can be influenced to a myriad of factors. Some of these factors could include multi-path propagation, other device interference, loss of line-of-sight, and more. RSSI will be the main reading that is used to calculate and model distance in meters from beacons.

2.3 RELATED WORKS

A large amount of work has been done in both policy enforcement and indoor localization systems using various methods. In this section, we will cover a sample of work that is closely related to policy enforcement and indoor localization.

2.3.1 Policy Enforcement

Policy enforcement is done to simplify the task of managing complex networks and systems. There is work from the early 2000s that discuss various policy models and implementations on computer systems [20] [21]. As computer systems evolved and became increasingly more complex and nuanced, more complex policy systems needed to become developed to satisfy the growing requirements to properly manage them. For the Android operating system, there have been many papers that focused on extending its existing permissions model with more user-defined constraints and policies.

Aurasium is described as a practical policy enforcement for Android Applications [22]. It works by repackaging applications into a sandbox and closely monitors the application's behavior for any attempts that violate the user-defined policy. Nauman et al. describe extending the existing Android permissions system to allow for more fine-grained control for specific applications [23]. There is some work done on using context to restrict app run times on Android, but policy enforcement is preventing specific apps from certain permissions [24]. This differs in this thesis since we are enforcing state changes in the system and application level via policies instead of more fine-grained control of resources and permissions in Android.

2.3.2 Indoor Localization

Indoor localization contains a lot of research utilizing many different technologies and techniques. Most solutions are based on a few key technologies. There is work done exploring: GPS, RFID, cellular-based, ultra-wideband, WLAN/WiFi, and Bluetooth [25]. In particular, a lot of work has focused on WLAN/WiFi technologies due to the existing framework that most places have. Some work has looked at exploring received signal strength using Wireless access points to triangulate a users position [26]. Other work has looked at wireless fingerprinting which measures signals in different regions to create a mapping which is later used for localization [27]. Bluetooth has adopted similar techniques to Wifi localization technologies. There are both RSSI methods [28] [29], finger-printing methods [30], and a fusion of methods [11]. While there is a lot of work put into the field, there is still no clear solution that provides cheap and easy indoor localization using BLE beacons [4].

The work in this paper can be seen as developing a basic context-based policy system that looks to enforce system and application changes and settings through the use of user location. The localization system to determine user location stems from background research put into localization systems using Wifi instead of BLE beacons. We try to blend together work from both areas into a unique context and application.

CHAPTER 3: CONTEXT AWARE POLICIES

In this chapter, we will formally give a definition of policy and context. Then we will describe a high-level overview of the context-aware policy enforcement engine and its implementation. Finally, two different use cases will be described and implemented.

3.1 DEFINITIONS

Definition 3.1 ***Action.*** *An action set is defined by* $A(app) = \{a_1, \ldots, a_n\}$. *It describes the set of ALL potential actions* $a_1, \ldots, a_n$ *given an application. An action is defined as* $(app, x \in A(app))$. *An action is a tuple of the application context and an element in the given action set for the application. If any action does not exist in the set, it is not a valid action for that application and cannot successfully be performed.*

For example, an action could be changing the system setting if mobile data is on or off. The action set would defined as $A(\texttt{System.mobile\_data}) = \{\texttt{set on}, \texttt{set off}\}$. Any valid action is contained within that set. Most actions for system settings are relatively simple since they will involve only a couple of states (on/off). However, actions for custom applications can result in larger action sets due to the number of actions that can be performed. We define actions in this relaxed manner to allow for a generalized set of actions. This makes it easy to extend this framework for future work.

Definition 3.2 ***User.*** *User,* U, *is a person that is associated with a physical device.*

The user is the way that we can differentiate different people and this is done through the phone's unique IMEI number. For example, we can tie the user id to a teacher versus a student and then apply the corresponding policies to only students.

Definition 3.3 ***Location.*** *Location,* $L = [(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)]$, *is the location of the device. This is defined as a set of points that when connected in order form a polygon that defines the region in a given coordinate space. The symbol* ! *can prefix the location to indicate not within the defined bounds.*

Location, in this case, can be on multiple levels of granularity. In this definition, we formalize to represent a given boundary that can be checked against by using the results from the localization system.

Definition 3.4 ***Time.*** *Time, $T = [t_1, t_2]$, is a range of time stamps that designate a length of time.*

Time is how long, or when we want the policy to execute. For example, we don't want to silence a phone in the classroom on the weekend since there is probably no class going on during that time.

Definition 3.5 ***Context.*** *Context is defined as a collection of the states of $\{U, L, T\}$.*

This is the heart of the context-aware policy enforcement. As described at a high-level in the background chapter, context takes together user, time and location. With the formalized definition, it is possible to define different contexts where various policies to run in only those instances. For example, now policies can be defined to only run during school hours in the computer lab for specific activities for student users.

Definition 3.6 ***Policy.*** *A policy is defined as $P = (\{A_1, \ldots, A_n\}, C)$. It has a set of rules and given context where the policy will take effect.*

This is the set of actions that are to be taken once given context conditions are met.

Definition 3.7 ***Current Context.*** *The current context is the active context using the current user, location, and current time to generate the context.*

This is the context that is currently used by the system to determine which policies to correctly apply.

3.2 ARCHITECTURE

This section will go into details about how the policy enforcement engine was implemented into Android. The policy engine functions as a high-level application on top of Android's existing permission and applications along with a policy selection server. This makes it so that it is possible to disable the policy engine any time the user deems necessary, but it is possible to implement the architecture on a lower level service to prevent user interference in future works. There are four main components to the policy engine. All of the following parts, except for the policy server, were implemented using Java 8 and the Android SDK since it will run an Android application. A high-level overview of how the various components of the engine interact with each other is displayed in Figure 3.1.

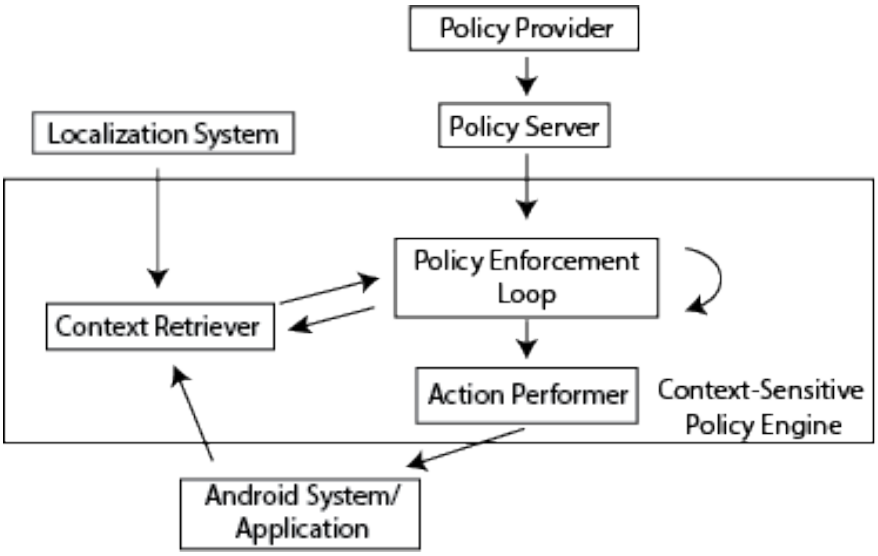

Figure 3.1: This shows a high-level view of the context-aware policy engine's architecture.

3.2.1 Policy Server

The server will hold all of the administrator-defined policies that need to be enforced. This is queried by the `Policy Enforcer` with the current context and it will then return the list of relevant policies that fit the given context for the mobile device to enforce. This removes the storage of various policies on local devices and relies on a server maintained by a policy administrator. Now, all devices using the context-aware policy engine can now synchronize with the server to keep up-to-date on policies as soon as they get updated. Additionally, it makes it easier for system administrators to manage policies company-wide. The server was implemented in Python 3 using multi-threading via `threads` module to allow for multiple clients to connect.

3.2.2 Context Retriever

This is the part of the engine that will generate the current context of the phone. It queries the `Localization System` for the current location, the `Policy Server` for the current time, and the uses the phone IMEI value for the user value. It combines the three results to generate the current context. This generated context is then used throughout the rest of the policy engine to determine what policies it should enforce.

3.2.3 Action Performer

This will try to enforce the actions given. This will use an intent handler to send a request for the application to handle the sent action. This system allows for the applications to handle different state changes and actions making it more flexible in how requests are handled. Now, the policy engine can theoretically handle a wide variety of unique and complex actions. However, this also means that it is up to the developers of the applications to enable these actions which may take some time. Another possible action is to set the state of some phone configuration. This is done through the system settings handled through Android's SDK. Due to resource and time constraints, many of the actions implemented were related to system configuration changes.

3.2.4 Policy Enforcer

This module provides the necessary actions to enforce specific actions to various applications. The `Policy Server` is queried to retrieve the list of relevant policies given the context generated from the context retriever. For each policy, the associated context is checked again through the `Context Retriever`, to ensure that we are in the correct context. If we are not in the correct context, then the policy enforcement will terminate since the conditions are not satisfied.

Otherwise, the context check passes the check and we then move on to the next step. The policy enforcement module will iterate through the set of actions. For each action, the policy enforcement module will pass the action to the `Action Performer`. This entire process will loop, causing these actions and state changes to remain in effect as long as the context remains relevant.

3.3 CASE STUDIES

3.3.1 Silencing Phones

Two different scenarios were implemented to test the policy enforcement engine. The first case study explored the scenario of students in a classroom setting. As students attend class, many of them leave their phones on which can lead to distractions during lecture time if they receive messages or notifications from applications. A way to help prevent this is to develop a set of policies that will silence all the student's phones during lecture time. However, we only want to silence the student's phones during lecture times and when they are physically

present in the classroom. Otherwise, silencing their phones will annoy them and potentially prevent future interest. To address this scenario, the following set of policies are developed:

$$P_1 = (\{(\texttt{System.ringer}, \texttt{set silence})\}, \{\texttt{Students}, [(0,0),(0,5),(5,5),(5,0)], [13:00, 14:00]\}) \tag{3.1}$$

$$P_2 = (\{(\texttt{System.ringer}, \texttt{set ringer})\}, \{\texttt{Students}, ![(0,0),(0,5),(5,5),(5,0)], [13:00, 14:00]\}) \tag{3.2}$$

These two policies will activate as the user leaves and enters the boundary and perform the appropriate action. P_1 will apply when the user is within the room and it will silence their phone. Once the student leaves the room, P_1 will no longer apply and P_2 will take effect. This policy will turn the phone's ringer back on allowing for normal usage. These two policies were implemented on the policy server and the system settings changes correctly applied as the test user entered and left the room with the appropriate user's phone.

3.3.2 Finding Lab Partners

This scenario explores multiple users using the policy enforcement engine. The context, in this case, is students in a class are doing a lab exercise that requires a partner. However, this class is mainly taught in an online environment. For one of the lab exercises, students are required to go to a lab and find a partner from the class to work on the assignment with. The main issue is that the students do not know each other well due to the online environment. To address this issue, we want to develop a set of policies that help students find each other when they are in the lab. To do this, we developed a mock application called `FindAFriend` that will notify users of nearby users once the application receives an intent with the nearby user ids. The policy developed to address this is as follows:

$$P_1 = (\{(\texttt{FindAFriend.Notify}, \texttt{send [User info]})\}, \{\texttt{Students}, [(0,0),(0,5),(5,5),(5,0)], [0:00, 23:59]\}) \tag{3.3}$$

P_1 will send the `FindAFriend` application an intent that a student is currently in the lab. Then the application will handle the rest of the logic of notifying the other user of other students currently in the lab.

CHAPTER 4: INDOOR LOCALIZATION SETUP

In this chapter, we discuss the underlying architecture and implementation for the inexpensive indoor positioning system architecture and describe the experimental setup for different components.

4.1 INDOOR POSITIONING SYSTEM ARCHITECTURE

A high-level overview of the setup can be seen in Figure 4.1. Each part will be described in further detail in the following sections.

4.1.1 Beacon

The beacon provides characteristics in its advertising packets that are used by the clients. Multiple Bluetooth Low Energy beacons are placed throughout the building at fixed, known locations. These locations will generally span the proximity of a given room. There were two different types of beacons tested: Estimote and Feasycom beacons. An example of these beacons can be seen in Figure 4.2. The beacons provide one important attribute to the phones, sending advertising packets that contain received signal strength indicators (RSSI). The sole purpose of the beacons is to provide the advertising beacons for the client to receive and process. In other applications, the beacons can provide a method to get sensor data like temperature.

4.1.2 Client

The client, in this case, is the smartphone. This will scan the BLE advertising channels listening for all the beacons it can hear. Those beacons are broadcasting RSSI information which is recorded by the client. The client is constantly talking with the location server to report RSSI measurements for each specific beacon. There is very little processing done on the client as it mainly functions as a communication device to the server. The client has a unique identifier associated with it which is sent with the RSSI measurements to the server. This design decision will allow for the client to use less computing power and thus it will conserve battery. That is important since we are using BLE beacons, there will be constant scanning for the advertisment packets and that can strain the battery. This client was implemented using Android Studio with Java 8, Android SDK, Estimote SDK, and Android Beacon Library.

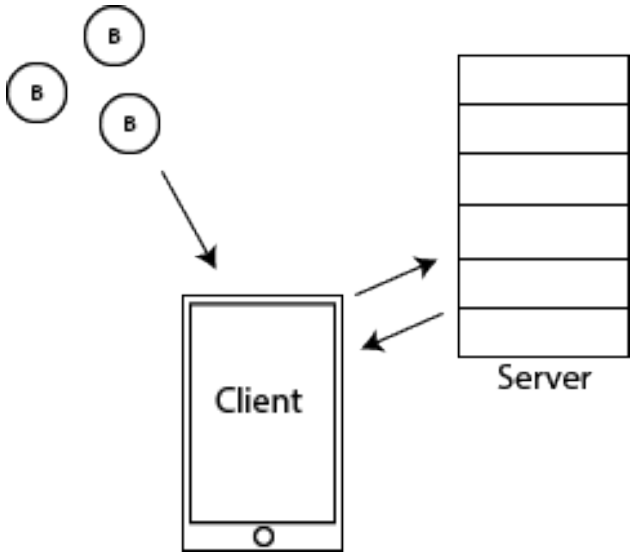

Figure 4.1: This shows an overview of the localization systems architecture.

Figure 4.2: A picture of the two different types of beacons used in this work. The beacon on the left is from Estimote and the beacon on the right is from Feasycom beacon.

4.1.3 Server

The server is where all of the processing is done. This design choice was to lift the computation off of the phone and offload it to a more powerful device. This choice was made to ensure that the calculations are finished within time due to the processing power a server will have over a phone and to help conserve phone battery life. The server will handle the RSSI to distance calculations and eventually perform the location estimation algorithm. Finally, the server will forward the calculated distance/location to the phone. All future actions based on the location is handled by the policy enforcement engine. The IPS server was implemented using Python 3.6. It is multi-threaded using the `threads` module to allow for multiple clients to connect at a time.

4.1.4 Distance Estimation

Distance estimation is performed by the server. It is a method that tries to approximate the distance from a device and the beacon based off of RSSI measurements. This can be done multiple ways and each way was tested to see which one gave the lowest error. This is a commonly implemented distance model for beacons:

$$\text{distance} = A * (\frac{r}{t})^B + C \tag{4.1}$$

where r is the RSSI measured by the device, t is the referenced RSSI, and A, B, C are constants which are calculated by fitting collected data to the model.

Another commonly used model for RSSI drop off uses the transmission power of the beacon RSSI_{TX} [12].

$$\text{distance} = 10^{\frac{\text{RSSI}-\text{RSSI}_{TX}}{-10n}} \tag{4.2}$$

n is a constant that is calculated by fitting collected data to the equation.

Both of these distance models were implemented and fitted on a module on top of the server using `numpy` and `scipy`.

4.2 LOCATION ESTIMATION

There are two methods that were tested for this calculation. The first method takes the given distance estimations from each beacon reported by the client and performs a trilateration to approximate the user's position [31]. This can be done since the beacon positions

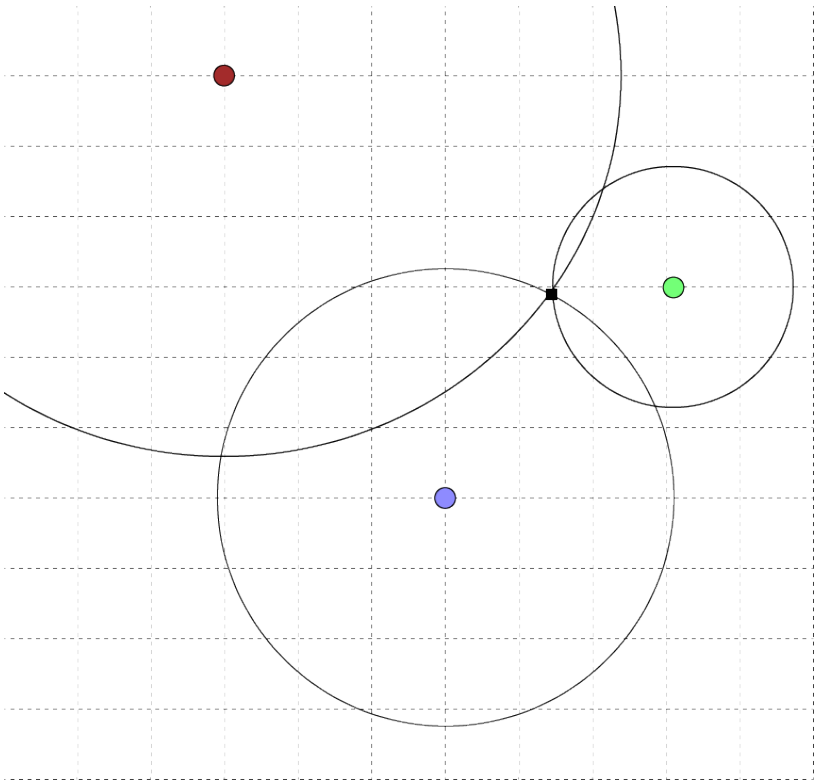

Figure 4.3: This is an example diagram of how trilateration works using three beacons. The three colored circles are the beacons, the large circles are potential locations where the device can be, and the black box indicates their intersection which corresponds to the tracked devices location.

are known. All the algorithms and methods described in this section were implemented in Python 3.6 and the libraries `numpy` and `scipy`, unless specified otherwise.

4.2.1 Basic Trilateration

We will model the given problem the following way. Given that we have n beacons and a single client c, we can model the distance from the client's position (x_c, y_c) to beacon b_i's position (x_i, y_i) by drawing a circle around the device with the radius of the distance from the client to the beacon. Then by finding the intersection between the circles between all of the beacons, we can determine the location of the device. This can be seen in Figure 4.3. The equation for a circle around the given point is:

$$(x_c - x_i)^2 + (y_c - y_i)^2 = r_n^2 \tag{4.3}$$

To linearize these equations, we use two beacons b_i and b_j to create the following expres-

sions.

$$(x_c - x_j + x_j - x_i)^2 + (y_c - x_j + x_j - y_i)^2 = r_n^2 \tag{4.4}$$

$$= \frac{1}{2}[(x_c - x_j)^2 + (y_c - y_j)^2 - r_i^2 + (x_i - x_j)^2 + (y_i - y_j)^2] \tag{4.5}$$

$$= \frac{1}{2}(r_j^2 - r_i^2 + d_{ij}^2) = b_{ij} \tag{4.6}$$

where

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{4.7}$$

Now we can create a linear system of (n - 1) equations that can then be solved through linear algebra. In this case, the system was solved using `numpy`'s least squares method.

$$(x_c - x_i)(x_j - x_i) + (y_c - y_i)(y_j - y_i) = \frac{1}{2}(r_i^2 - r_j^2 + d_{ji}^2) = b_{ji} \tag{4.8}$$

And we convert this to matrix form.

$$\begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_n - x_1 & y_n - y_1 \end{pmatrix} \begin{pmatrix} x_c - x_1 \\ y_c - y_1 \end{pmatrix} = \begin{pmatrix} b_{21} \\ b_{31} \\ \vdots \\ b_{n1} \end{pmatrix} \tag{4.9}$$

which is in the form of

$$Ax = b \tag{4.10}$$

which is then solved by

$$x = (A^T A)^{-1}(A^T b) \tag{4.11}$$

Since there are only two unknowns, we will generally only need three beacons to determine the unique position of a device. However, this method relies on there being very little error to allow for the system of equations to be solved. Using more beacons may provide additional accuracy to the model, but it also introduces more overall error into the system. This error is from the beacon's distance estimation being not entirely accurate due to RSSI fluctuations throughout. These kinds of error are potentially addressed with the next localization method.

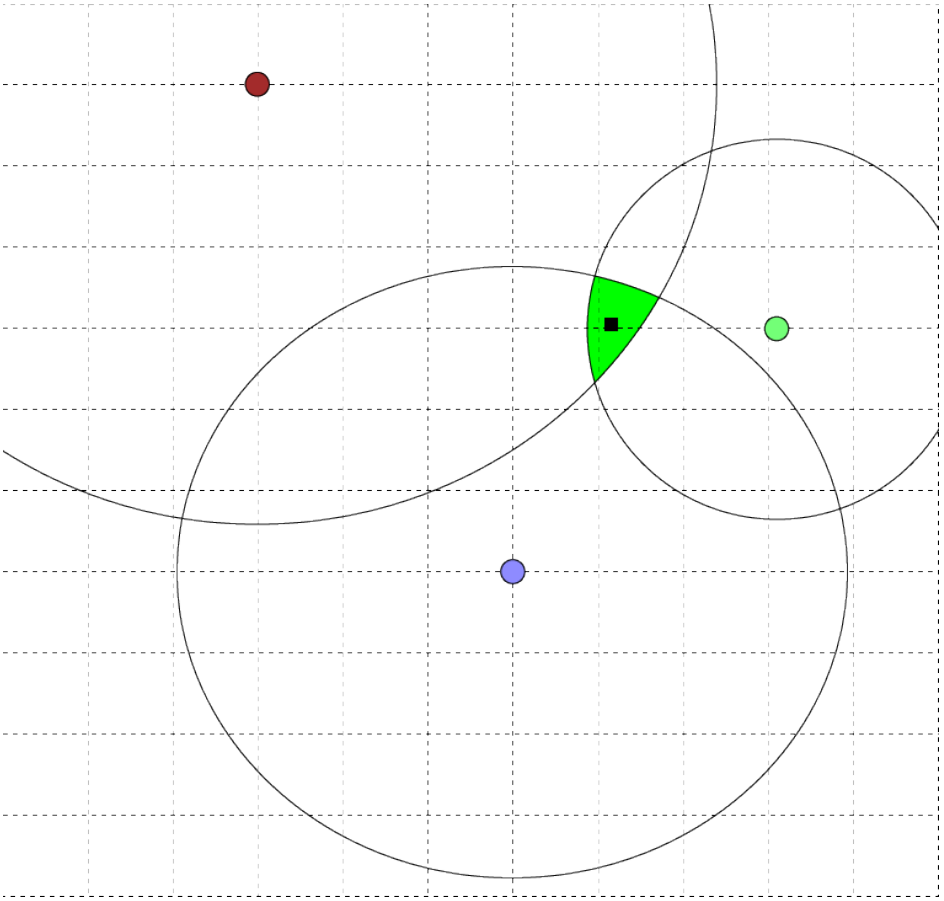

Figure 4.4: This is an example diagram of how trilateration works using three beacons when we try to account for error in the model. The three small circles are the beacons, the shaded area is where the beacon can be located, and the black square is where we approximate the device is by trying to minimize the distance from each beacon. There is no perfect intersection between the circles and instead, we try to minimize the distance for each beacon.

4.2.2 Location Estimation with Error Accounting

The previous method was developed assuming that the distance measurements were accurate causing the circles to intersect on a single point. However as previously discussed, it is very likely that the RSSI measurements will introduce some sort of error when being calculated to distance due to the 2.4GHz spectrum being crowded and multi-path propagation. This is further illustrated in Figure 4.4. The location calculation should try to take into account the error measurements and minimize them. To do this, we can model this problem in the following way:

Let our device c be at given coordinates (x_c, y_c). Given n beacons, for beacon i we have the coordinate (x_i, y_i) and an estimated distance from beacon i to client c by d_{ic}.

$$\sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} + e_{ic} = d_{ic} \tag{4.12}$$

We can rearrange the equation to isolate the error on one side.

$$e_{ic} = d_{ic} - \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} \tag{4.13}$$

The total sum of the errors is now:

$$e_{total} = \sum_{i}^{n} d_{ic} - \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} \tag{4.14}$$

Given that, we can try to minimize the total error to get the position that will approximately be our best guess for where the device is located. To do this, we use a minimization algorithm that will give us the approximate (x_c, y_c) that will produce the lowest total error. For this purpose we looked at Nelder and Mead's Simplex algorithm [32] and Powell's fast algorithm [33] in `scipy`'s minimization method.

4.2.3 Boundary Check

Since the location bounding is highly dependent on generating accurate RSSI measurements, it is not feasible to just use the raw location information generated since it usually will have a high degree of error from the true position. Instead, we are creating a set of boundaries to check if the user is within them to determine if they are in a given room or not. This is done to try and mask the inaccuracies of the entire system.

The error of the system comes from multiple parts. The distance estimation modeling generates error due to signal fluctuations from the beacons to the mobile devices. Further error is compounded during the location estimation due to that being feed in the distances from the beacons. These errors are inevitable in a system like this so, there is a final procedure to try essentially adjust for that. By only detecting within a specific boundary, then the system is built and adjusts to these errors.

To do this boundary check, we use the point in polygon test method [34]. This was implemented using the `Shapely` library in Python. To do the boundary check, we first define the coordinates of each corner of the room (taken from the context's location boundaries) to create a bounding polygon and then the estimated location is used as the point in polygon test to determine if the user is within a room or not. The coordinates are defined in the order as they appear in the polygon. The beacons are given coordinates relative to these coordinates so the generated location can properly determine where a mobile device is.

CHAPTER 5: RESULTS

In this chapter, the results for various different experimental setups for the inexpensive localization setup are shown.

5.1 DISTANCE ESTIMATION

In this section, the distance estimation modeling is discussed. The distance estimation is done based off of the RSSI signals given by the BLE beacons to the phone.

5.1.1 RSSI Measurements

The first thing that we needed to test was how RSSI measurements fluctuate at various distances. This was done to later develop a model mapping from RSSI to distance. The setup involves placing an individual beacon on top of a box and a phone on a box a set distance away. We recorded 100 RSSI values at 0.0m, 0.5m, 1.0m, 1.5m, 2.0m, 2.5m, 3.0m, 3.5m, and 4.0m. The entire room was cleared of obstacles that may cause issues. The phone used in this experiment was the LG G4 running Android 6.0.

Estimote Beacons

There were three Estimote beacons used in this experiment. The figures 5.1, 5.2, 5.3 are the raw RSSI recordings for each of the Estimote beacons plotted versus distance. For each different distance, we recorded 100 continuous RSSI values for each beacon. As seen in the figures, there is a high variation in the RSSI recordings for all three of the Estimote beacons. That generally indicates that RSSI is a somewhat unstable measurement even when testing in a mostly ideal environment.

Afterward, we fit the data to the two models discussed in the earlier to see which set of models gave an overall less absolute error in meters. In Table 5.1, the two models (Equations 5.1 and 5.2) are fitted to the collected data and the absolute error in meters is collected for each different distance measurement (0.0m, 0.5m, 1.0m, 1.5m, 2.0m, 2.5m, 3.0m, 3.5m, 4.0m) along with the average absolute error in meters. According to the results, Equation 5.1's model performs better than equation 5.2's model in most cases by roughly 0.175 meters and overall it has less absolute error.

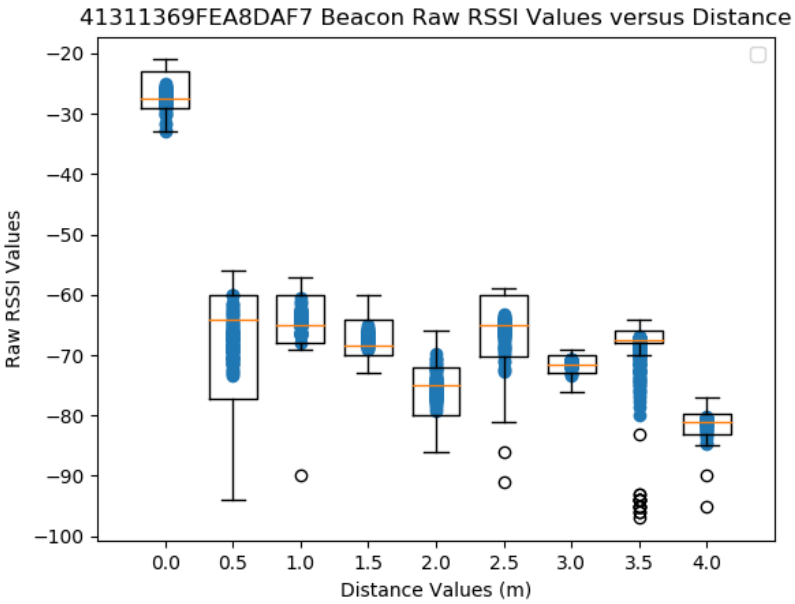

Figure 5.1: This shows the raw RSSI values captured by beacon 1 at various distances.

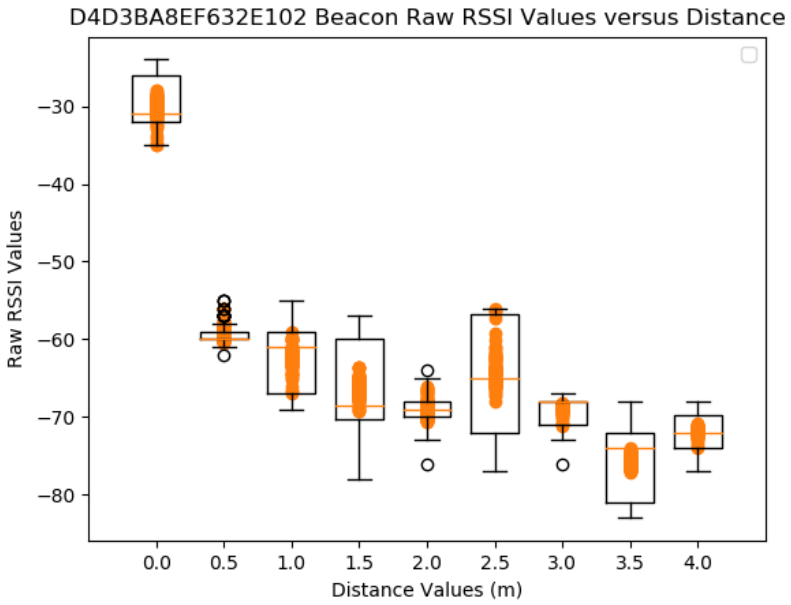

Figure 5.2: This shows the raw RSSI values captured by beacon 2 at various distances.

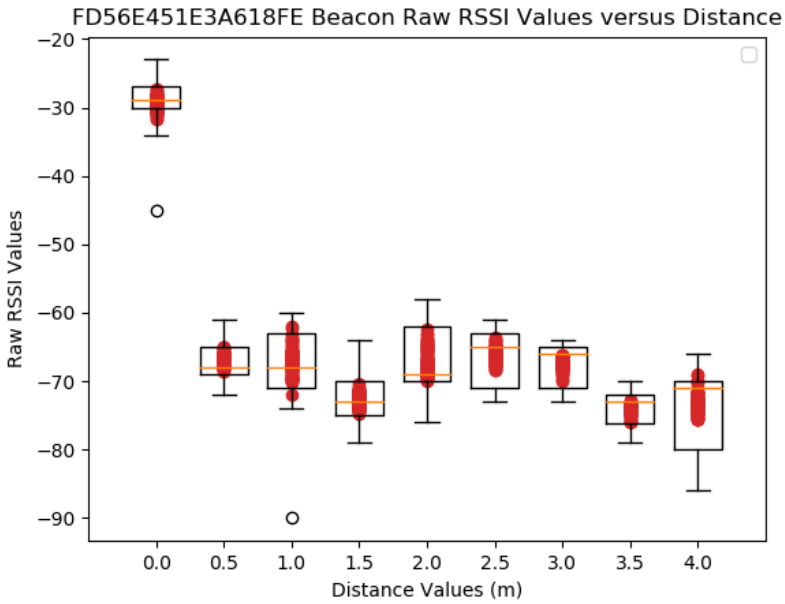

Figure 5.3: This shows the raw RSSI values captured by beacon 3 at various distances.

$$\text{Model 1:} \quad d = A * (\frac{r}{t})^B + C \tag{5.1}$$

$$\text{Model 2:} \quad d = 10^{\frac{\text{RSSI}-\text{RSSI}_{TX}}{-10n}} \tag{5.2}$$

5.1.2 Feasybeacon

For the Feasybeacons, we performed exactly the same setup and data collection as the Estimote beacons. However, there was even more variation in the RSSI readings in the Feasybeacons compared to Estimote beacons.Figure 5.4 shows the RSSI reading results. The RSSI readings show there is even higher variation and the collected readings weren't stable enough to even fit either of the two models to them. Due to this result, we went with the Estimote Beacons for the rest of the experiments.

5.1.3 Signal Smoothing and Filtering

The above results highlight how noisy the RSSI measurements are. We need some way to filter or smooth out the signals so it is more stable and ready to use for more location calculations. The first method we examine is 1-D Kalman filtering. The Table 5.2 shows

	Beacon 1		Beacon 2		Beacon 3	
Error	**Model 1**	**Model 2**	**Model 1**	**Model 2**	**Model 1**	**Model 2**
0.0m	0.042624	1.413688	0.175036	1.492749	0.059736	1.443382
0.5m	1.603021	1.731129	0.558941	1.596474	1.296694	1.683414
1.0m	0.513789	1.132259	0.499714	1.167990	0.955199	1.190438
1.5m	0.444195	0.704195	0.870718	0.778583	1.469435	0.821231
2.0m	1.016804	0.416964	0.515321	0.346743	0.605019	0.183641
2.5m	0.925470	0.329597	1.022401	0.268155	0.840253	0.334541
3.0m	0.693714	0.690804	0.568913	0.655560	1.179656	0.814297
3.5m	2.063840	1.191978	1.027778	0.954635	0.647071	1.144202
4.0m	0.373766	1.430781	0.887066	1.568701	1.554227	1.652690
Overall	0.853935	1.005269	0.680654	0.981066	0.956366	1.029760

Table 5.1: Absolute Error of different models

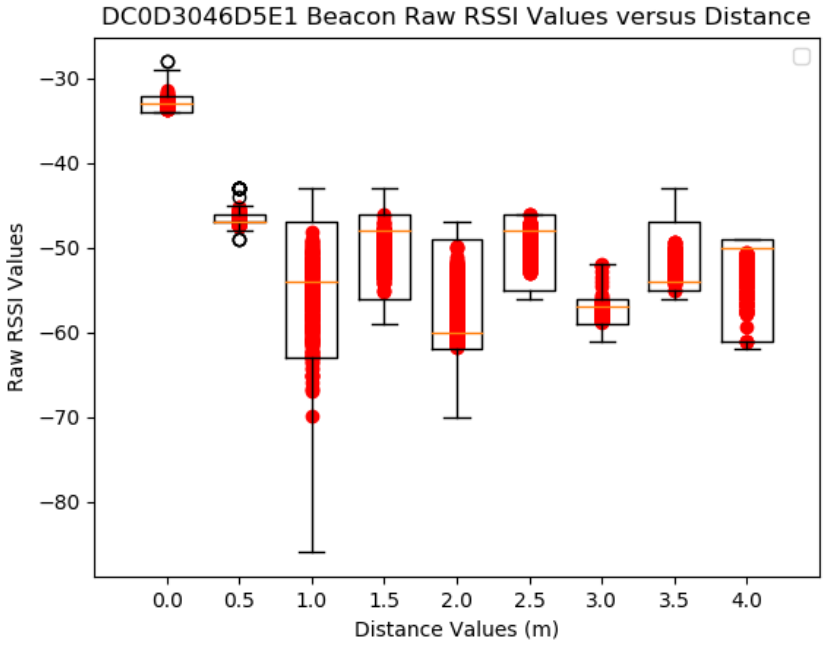

Figure 5.4: This figure shows the RSSI readings from the Feasycom BLE beacon.

	Beacon 1		Beacon 2		Beacon 3	
Error	**Model 1**	**Model 2**	**Model 1**	**Model 2**	**Model 1**	**Model 2**
Overall	0.662921	1.003607	0.548922	0.984523	0.774674	1.029128

Table 5.2: Absolute Error with Kalman Filtering on Each Model

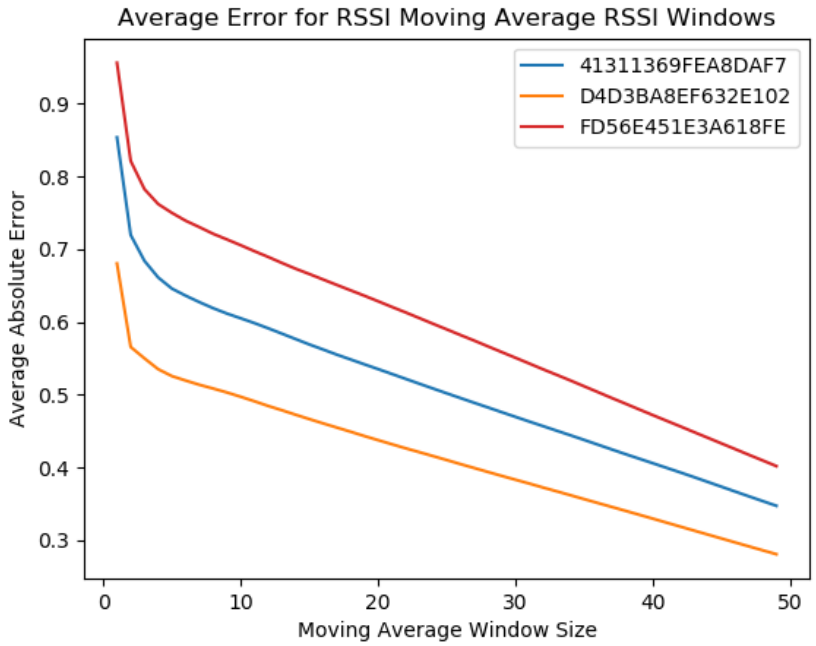

Figure 5.5: This shows the absolute error as the average window changes from 1 to 50.

the effect that Kalman filtering has on the two different models. Model 1 shows a drop in absolute error but Model 2 only shows a slight drop. Another method employed to smooth out the RSSI readings was a moving window average. This basically means that we take the average RSSI value across a window and use it for the distance calculation. Figure 5.5 shows the effect of window size on the absolute error for Model 1. As the window size increases, the absolute error will also decrease. We then look at the effects of using both Kalman filtering and the moving average window in Figure 5.6. After a window size of 10, there is little difference between only using the moving average window and the combination between Kalman filtering and the window.

5.2 LOCATION DETECTION

For location detection, we are testing if the system is able to successfully detect that you are inside a room correctly. This system only works for a single room, but it can be easily

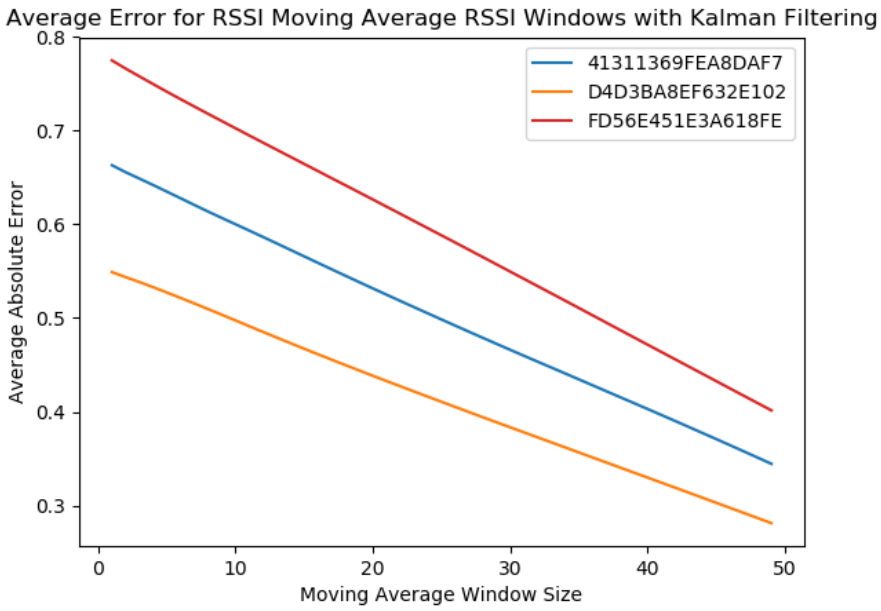

Figure 5.6: This shows the absolute error as the average window changes from 1 to 50 with Kalman filtering.

expanded to multiple rooms using the same server architecture. For the experiment, we will go through each meter of the room and test if the system properly detects if you are in the room along with regions outside of the test room. Two different beacon setups are tested to see how beacon configuration affects the system's accuracy. Finally, two different real-world testing environments are used to see if the system can perform in actual environments.

5.2.1 Apartment Room

This location is done in my apartment room. The Estimote beacons were placed in the corners on the ceiling secured with double-sided tape and all obstacles were left in the same position throughout all of the experiments. This environment is a complex environment since it consists of multiple factors that can potentially cause signal bouncing. This was chosen in order to simulate a real world example as closely as possible. Figures 5.7 and 5.8 show the results found.

The results show that the system seems to work relatively well. The overall accuracy in this environment showed a 91.635% classification accuracy for predicting if the user device was correctly in the room or not. From these results, the system can be used for actual applications - like the context-aware policy engine.

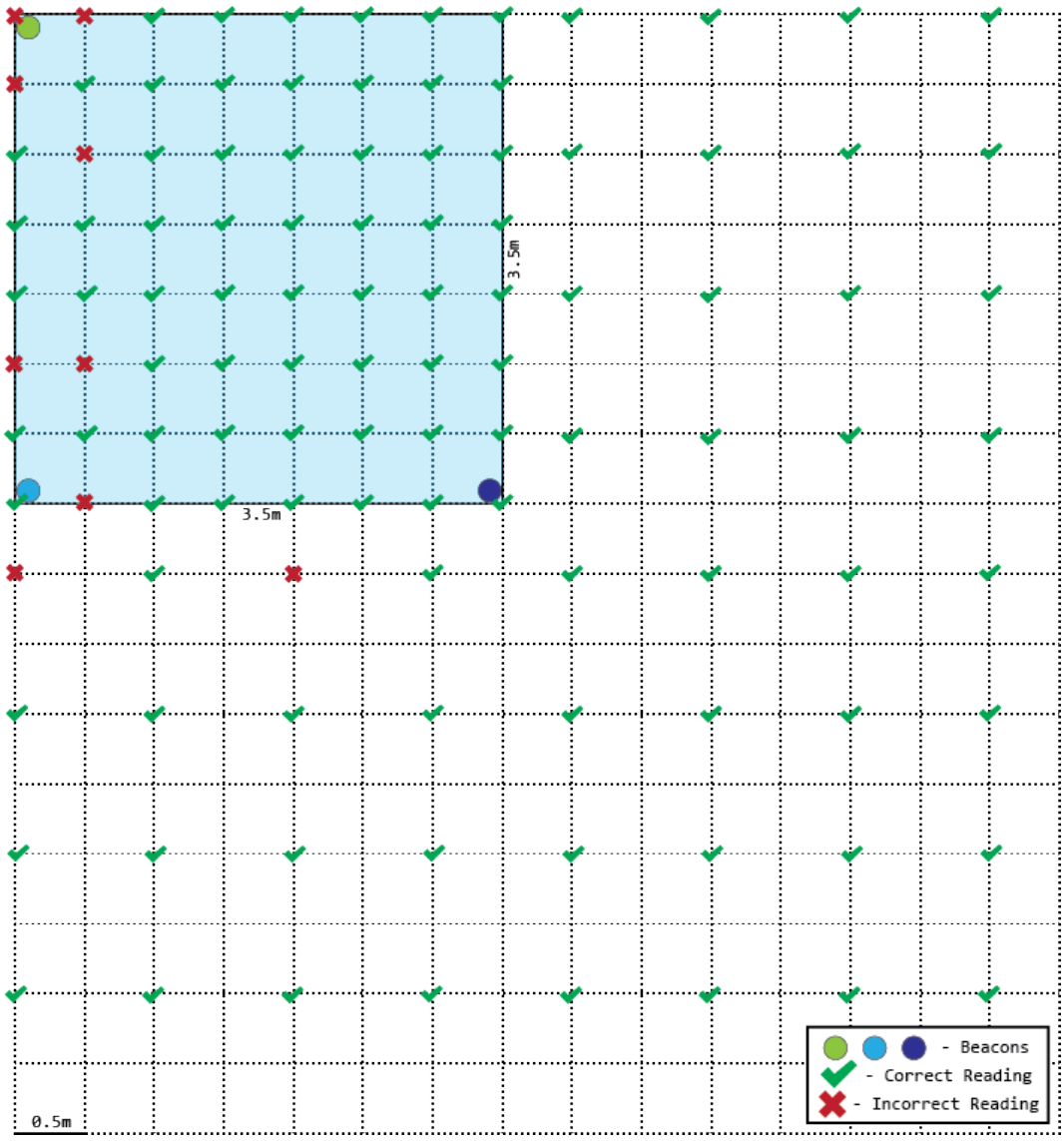

Figure 5.7: This shows the results of the location test. The check marks indicate the location test at that location successfully detected being inside/outside of the room. The cross indicates that the wrong location was determined.

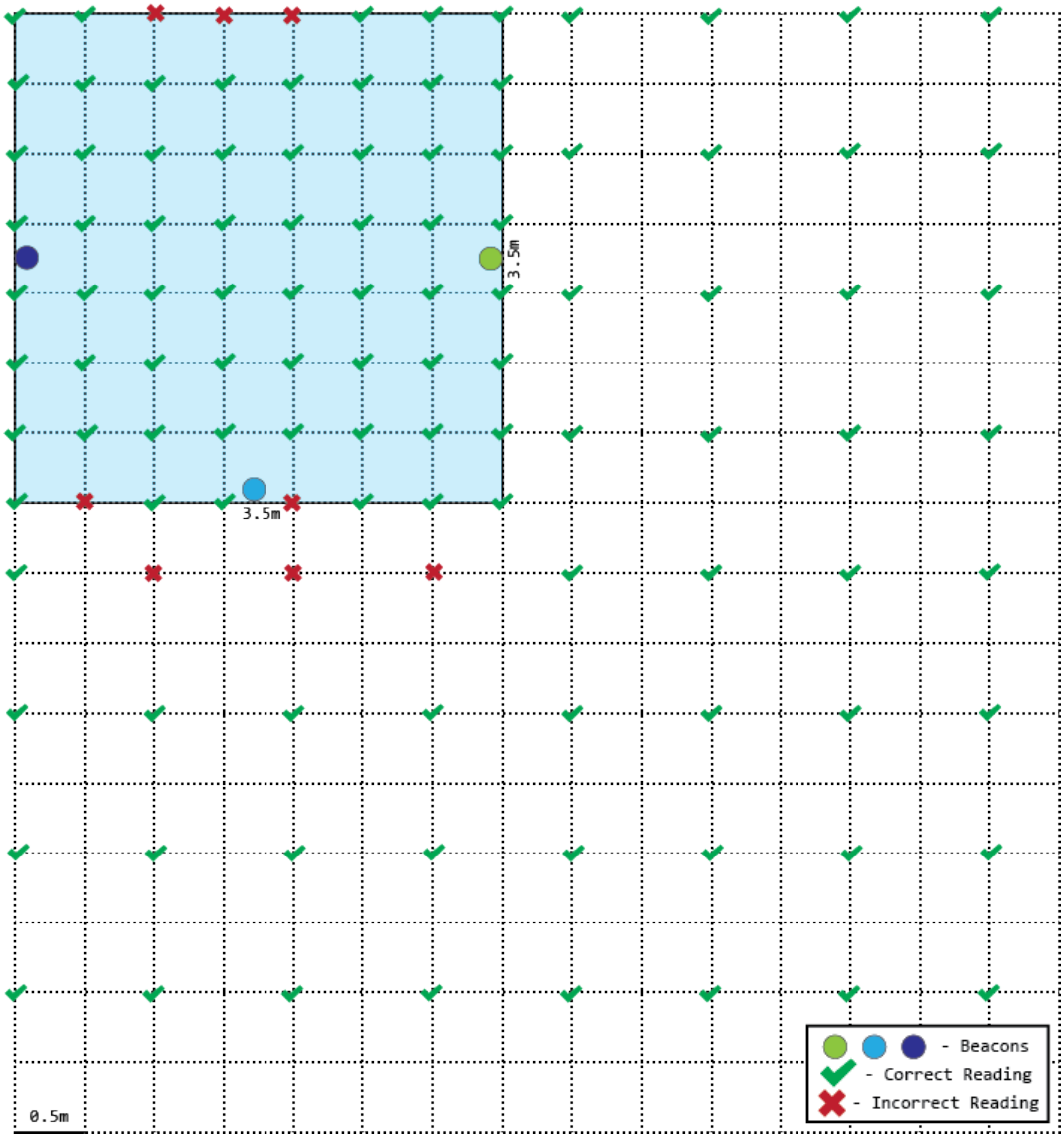

Figure 5.8: This shows the location test results when the beacon configuration changed. There seems to be very little variation on the overall accuracy of the system even when the beacon positions changed.

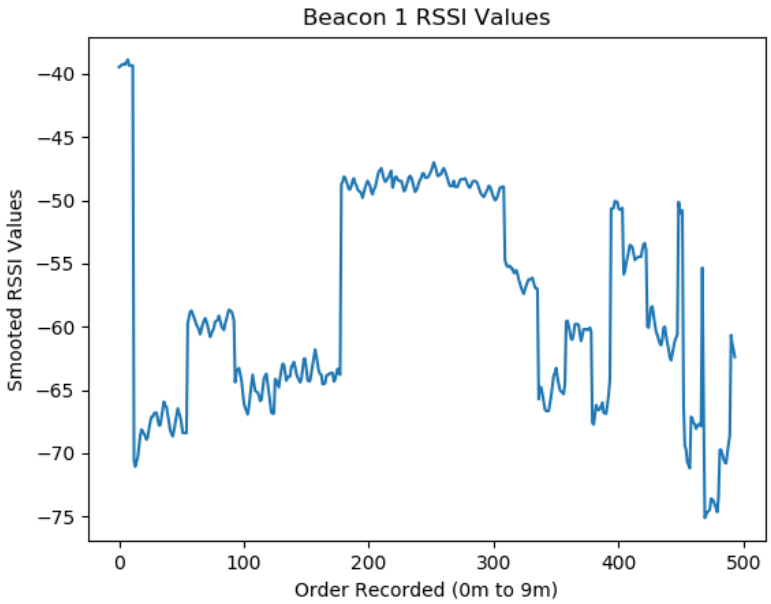

Figure 5.9: This shows how the RSSI values changed as the device moved from underneath the beacon (0m) to the other end of the room (9m). The RSSI changes have no pattern to them.

5.2.2 Siebel Center Lab

This location was done in the Security Lab on the fourth floor of Thomas M. Siebel Center for Computer Science. The beacons were placed on the indicated locations on the ceiling with double-sided tape as well and all obstacles were left as is throughout all of the experiments. This room may be typical of a computer labatory found in other buildings. There are multiple obstacles that can cause potential issues. Figures 5.10 and 5.11 indicate the results found. In both figures, there is a lot of wrong predictions when testing location outside of the room boundaries. This seems to be an issue with the recorded RSSI measurements for both of the setups. The overall accuracy for this environment was only 49.853%, which indicates the system does not function well at all.

Both of the setups were not correctly detecting that the user was outside of the given room boundaries. Looking at smoothed/averaged RSSI values as seen in Figure 5.9, there is a lot of change in the RSSI signal that did not correspond to the appropriate distance change. The RSSI values never dipped below -75 which seemed to indicate that there may be issues with the beacons themselves or there was a lot of signal interference at the lab. This shows that relying on RSSI values is unreliable and as a result causes the rest of the system to not function as intended.

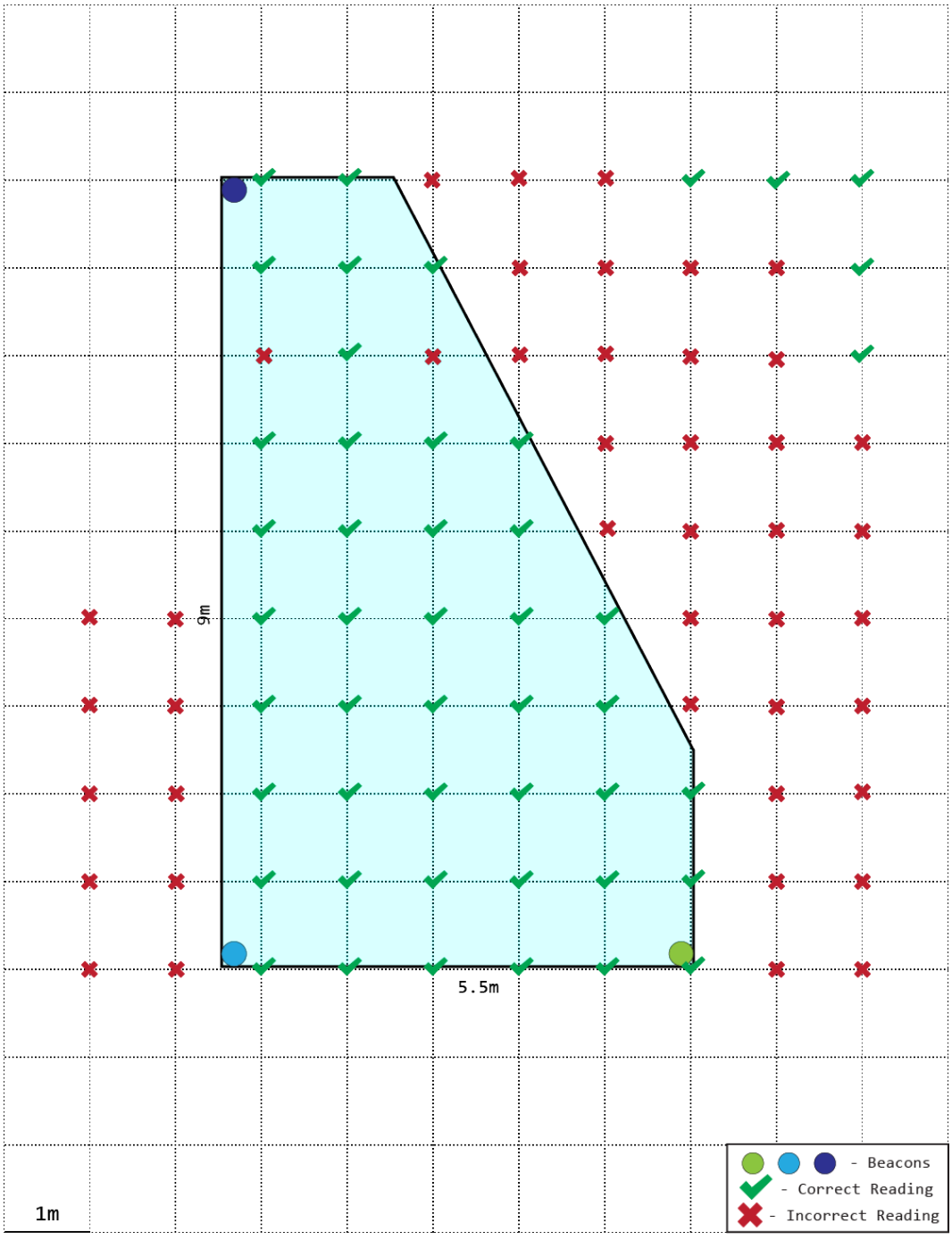

Figure 5.10: The location results of the localization system when being tested at Siebel center.

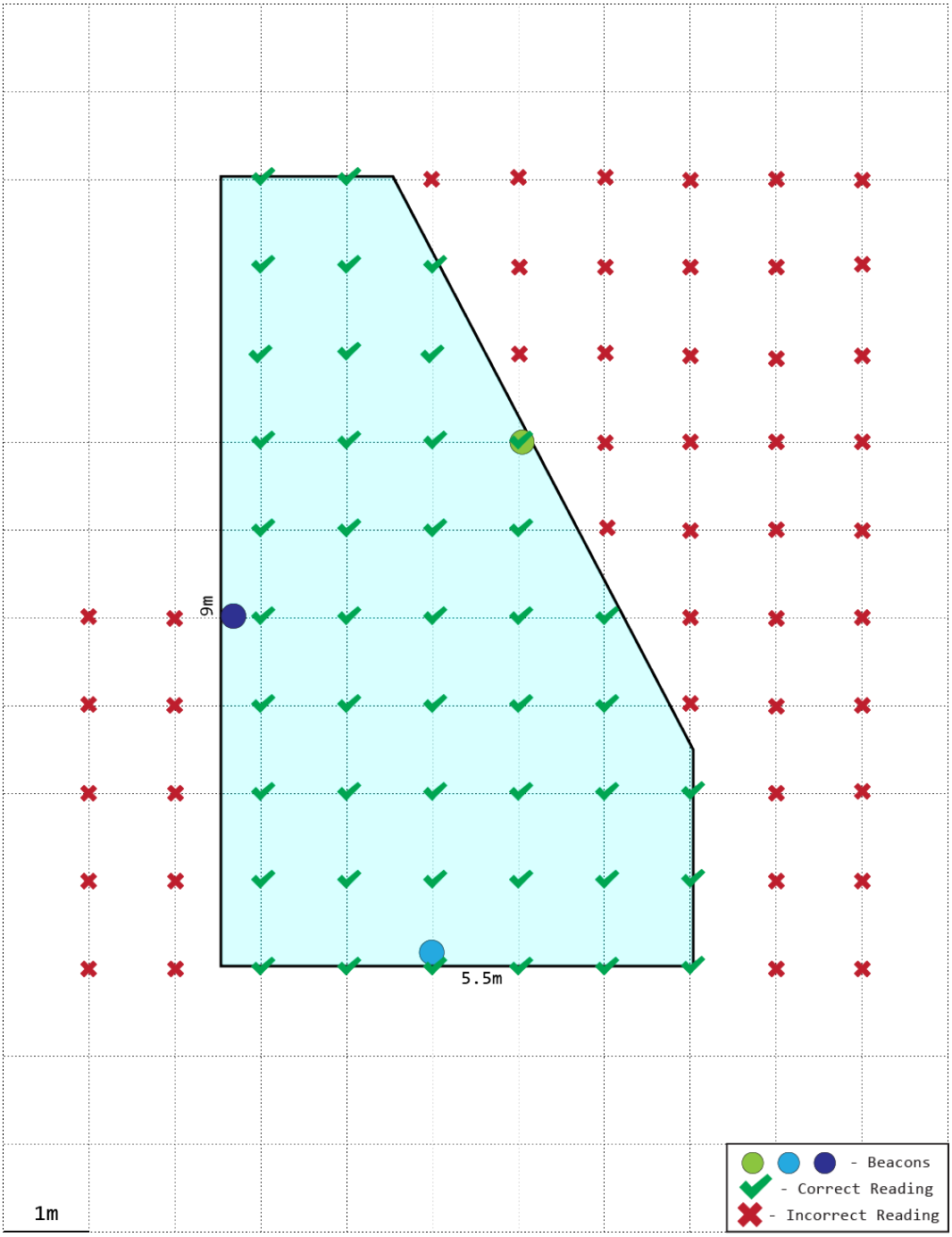

Figure 5.11: Location test results of a different setup at Siebel center. There isn't much change in system accuracy even when the beacon positions changed.

CHAPTER 6: DISCUSSION AND FUTURE WORK

This project illustrated potential architectures of both a context-sensitive policy enforcement engine in addition to a basic indoor localization system using BLE beacons as a cheap alternative to other hardware stacks designed to solve localization. However, there were several issues and areas of concern that came up as research progressed.

6.1 DISCUSSION

The policy formulation and implementation showed that this is a viable system design that works in different scenarios as explored by the two different case studies. The framework explained is easily expandable and shows that it is possible to enforce context-sensitive computing and policies on smartphones in an easy and efficient manner. The two different case studies were easily implemented on the described context-aware policy engine showing the flexibility that the system can bring.

The distance estimation based on the RSSI signals of beacons showed very different results. In the normal testing environment, we showed that there is generally a 1-meter error with distance estimation using either of the models discussed. This by itself is a good result since most indoor localization systems are getting a couple of meters of inaccuracy. This error is even further reduced when introducing signal smoothing. The moving average window and Kalman filter help to reduce the random fluctuations that RSSI measurements normally experience. The combination from the two methods reduces the variation the most causing the distance estimation error to decrease making the system more reliable for location estimation. Any reduction to the noise in RSSI measurements means that the distance estimation will be more accurate allowing for more accurate location estimation.

However, further testing in multiple real-world environments showed that there are a lot of issues with only using raw RSSI values. As shown in Figure 5.9, the RSSI readings in a noisy environment, like a computer lab, shows high variation and sometimes lack any signal drop off even as distance increases to 9 meters or more. This most likely indicates that there are multiple factors influencing the RSSI readings. Some of these factors are crowding of the 2.4 GHz band causing lots of packet loss, a lot of potential propagation paths due to the environment having multiple obstacles, and lack of line of sight due to the aforementioned obstacles. As a result, there is a lot of consistent RSSI readings that cause distance estimation in noisy environments to be very error-prone. These issues may be due to a lack of a solid model that takes into account these different factors.

Location estimation only using the RSSI signals seemed to work relatively well in the first environment, as shown by Figures 5.7 and 5.8. The system demonstrated roughly a 90% accuracy for detecting if the user's smartphone was in the room or not. For most general applications, this accuracy is fine as most of the error is centered around the boundary transitions from entering and leaving the room boundaries. However, this result did not transfer when testing at Siebel Center's computer laboratory environment. This is most likely due to the reasons stated above that cause the RSSI readings in a larger environment not be stable. This indicates that quieter, smaller areas may make it easier to deploy the Bluetooth based systems and that the current approach does not translate to noisier environments well. The range of beacons may be hardware dependent and better BLE radios may affect the results. In the end, the inexpensive localization system ran into issues that dealt with hardware inaccuracies. If these could be addressed, then future development may result in an inexpensive and accurate localization solution.

6.2 FUTURE WORKS

In the future, there are multiple avenues to explore. First, the formal definition can be expanded and more strict. Currently, the definition serves a very general framework but it lacks the coverage of some cases as to how to handle conflicts, context overlaps, access to resources, and more. This could be further explored to make it more comprehensive. The entire framework is implemented on top of Android due to ease of development. However, in the future, the entire system could be brought down a level. The context-aware policy engine can easily be adapted into the Android operating system which could lead to more fine-grained control of system settings and other security benefits.

Another area of interest is different options for modeling RSSI to distance from beacons. This work only explored two options, but there are many possible models to explore that might more accurately account for different variables and factors to help make the distance estimation more accurate and thus more useful. If the distance model can be accurately established, then all future localization systems will also be more accurate as a result. One potential area of interest is the use of neural networks as trained regression models for accurately modeling distance based on time and RSSI readings from beacons.

Finally, the localization setup explored in this work was very naive and basic. Future work could take into account more variables to make it more accurate. Instead of only using RSSI measurements, the system could take into account phone sensors and do a fusion based approach to make it more accurate as some work has started to do.

CHAPTER 7: CONCLUSION

Past work has focused on using custom-made hardware or existing expensive hardware solutions to try and solve the indoor localization problem. However, recently there has been more and more development on the Internet of Things and that could potentially allow for cheaper solutions for the problem. This work looked at potential policy systems that would take advantage of these solutions as a way to provide inexpensive localization system. This system can then be used by other works to enable and allow for context-aware computing and applications.

The first thing that this thesis discusses is the formal definition of basic context-sensitive policies. Based on the formal definitions, the system's architecture and implementation were developed and described in detail. Multiple trade-offs were discussed such as using a centralized policy server to offset all of the CPU intensive work to help conserve battery life on the less powerful mobile devices. Additionally, it is easy to extend the action set to encompass multiple applications for more complex user interactions. Finally, two different case studies on potential policies that were implemented as a demo were explained. The two case studies explored different vectors of applications that context-aware policies can bring to mobile computing and showed that the context-aware policy system discussed is a viable way to enable context-aware computing.

The second part of the thesis explored a preliminary approach to solving the indoor localization problem using cheap Bluetooth low energy beacons. This approach is important because it looks at the feasibility of using BLE beacons in an inexpensive localization system. Previous work has looked at using RSSI as a way to estimate distance and we look at how we develop an inexpensive solution using that work. The past work has shown how BLE and RSSI can be used as a method for determining general user location. These works used BLE beacons since they are extremely cheap and can provide a rough estimate of the distance from a smartphone based on their RSSI readings. These RSSI readings were modeled using various signal to distance equations to generate a real-data model. Then the system uses these distances to try and pinpoint the location of the user through multiple beacons. In practice, this system was shown to work well in one real-world environment. But another real-world test showed high variation in RSSI values. That issue prevented the system from functioning reliably and it showed that BLE beacons have not advanced far enough to mitigate the issues to perform well in multiple, complex environments. In the end, a basic architecture for both the server and client were implemented to help demonstrate the potential of this approach but further research is needed to solve the underlying problems.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] E. Welbourne, L. Battle, G. Cole, K. Gould, K. Rector, S. Raymer, M. Balazinska, and G. Borriello, "Building the internet of things using RFID: the RFID ecosystem experience," *IEEE Internet computing*, vol. 13, no. 3, 2009.

[3] X. Jia, Q. Feng, T. Fan, and Q. Lei, "RFID technology and its applications in Internet of Things (IoT)," in *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*. IEEE, 2012, pp. 1282–1285.

[4] D. Lymberopoulos, J. Liu, X. Yang, R. R. Choudhury, V. Handziski, and S. Sen, "A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned," in *Proceedings of the 14th international conference on information processing in sensor networks*. ACM, 2015, pp. 178–189.

[5] G.-y. Jin, X.-y. Lu, and M.-S. Park, "An indoor localization mechanism using active RFID tag," in *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, vol. 1. IEEE, 2006, pp. 4–pp.

[6] B. D. Ferris, D. Fox, and N. Lawrence, "WiFi-SLAM using Gaussian process latent variable models," 2007.

[7] C.-H. Lim, Y. Wan, B.-P. Ng, and C.-M. S. See, "A real-time indoor WiFi localization system utilizing smart antennas," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 2, 2007.

[8] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 197–210.

[9] D. Lymberopoulos and J. Liu, "The microsoft indoor localization competition: Experiences and lessons learned," *IEEE Signal Processing Magazine*, vol. 34, no. 5, pp. 125–140, 2017.

[10] J. Nieminen, C. Gomez, M. Isomaki, T. Savolainen, B. Patil, Z. Shelby, M. Xi, and J. Oller, "Networking solutions for connecting Bluetooth low energy enabled machines to the Internet of Things," *IEEE network*, vol. 28, no. 6, pp. 83–90, 2014.

[11] P. Mirowski, T. K. Ho, S. Yi, and M. MacDonald, "SignalSLAM: Simultaneous localization and mapping with mixed WiFi, Bluetooth, LTE and magnetic signals," in *Indoor Positioning and Indoor Navigation (IPIN), 2013 International Conference on*. IEEE, 2013, pp. 1–10.

[12] M. Al Qathrady and A. Helmy, "Improving BLE Distance Estimation and Classification Using TX Power and Machine Learning: A Comparative Analysis," in *Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*. ACM, 2017, pp. 79–83.

[13] A. Developers, "What is android," 2011.

[14] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4," in *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*. IEEE, 2012, pp. 232–237.

[15] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[16] B. S. Proprietary, "Bluetooth Core Specification v5.0," pp. 1–2822, 2016.

[17] "A BLE Advertising Primer," http://www.argenox.com/a-ble-advertising-primer/, accessed: 2018-11-18.

[18] S. H. Gerez, "Implementation of Digital Signal Processing: Some Background on GFSK Modulation."

[19] M. Sauter, *From GSM to LTE: an introduction to mobile networks and mobile broadband*. John Wiley & Sons, 2010.

[20] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy Core Information Model–Version 1 Specification," Tech. Rep., 2001.

[21] L. Kagal, T. Finin, and A. Joshi, "A policy language for a pervasive computing environment," in *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*. IEEE, 2003, pp. 63–74.

[22] R. Xu, H. Saïdi, and R. J. Anderson, "Aurasium: Practical Policy Enforcement for Android Applications." in *USENIX Security Symposium*, vol. 2012, 2012.

[23] M. Nauman, S. Khan, and X. Zhang, "Apex: extending android permission model and enforcement with user-defined runtime constraints," in *Proceedings of the 5th ACM symposium on information, computer and communications security*. ACM, 2010, pp. 328–332.

[24] M. Conti, V. T. N. Nguyen, and B. Crispo, "Crepe: Context-related policy enforcement for android," in *International Conference on Information Security*. Springer, 2010, pp. 331–345.

[25] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1067–1080, 2007.

[26] M. Sugano, T. Kawazoe, Y. Ohta, and M. Murata, "Indoor Localization System using RSSI Measurement of Wireless Sensor Network based on ZigBee Standard." in *Wireless and Optical Communications*, 2006, pp. 1–6.

[27] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 269–280.

[28] A. Thaljaoui, T. Val, N. Nasri, and D. Brulin, "BLE localization using RSSI measurements and iRingLA," in *Industrial Technology (ICIT), 2015 IEEE international conference on*. IEEE, 2015, pp. 2178–2183.

[29] M. Kaczmarek, J. Ruminski, and A. Bujnowski, "Accuracy analysis of the RSSI BLE SensorTag signal for indoor localization purposes," in *Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on*. IEEE, 2016, pp. 1413–1416.

[30] R. Faragher and R. Harle, "Location fingerprinting with bluetooth low energy beacons," *IEEE journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2418–2428, 2015.

[31] W. Murphy and W. Hereman, "Determination of a position in three dimensions using trilateration and approximate distances," *Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado, MCS-95*, vol. 7, p. 19, 1995.

[32] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[33] M. J. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in *Numerical analysis*. Springer, 1978, pp. 144–157.

[34] K. Hormann and A. Agathos, "The point in polygon problem for arbitrary polygons," *Computational Geometry*, vol. 20, no. 3, pp. 131–144, 2001.