To the Graduate Council:

I am submitting herewith a thesis written by Brandon Jeremy Rogers entitled “The Reliability of Signal Strength for Localization of High-Density ZigBee Wireless Networks.” I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

_____
Dr. Greg Peterson, Major Professor

We have read this thesis and
recommend its acceptance:

_____
Dr. Itamar Elhanany, Committee Member

_____
Dr. Paul Crilly, Committee Member

Acceptance for the Council:

_____
Carolyn R. Hodges
Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

The Reliability of Signal Strength for Localization of High-Density ZigBee Wireless Networks

A thesis
Presented for the
Masters of Science
Degree
The University of Tennessee, Knoxville

by

Brandon Jeremy Rogers
August 2007

Copyright © 2007
by Brandon Rogers

All Rights Reserved.

Abstract

This thesis investigates the reliability of ZigBee signal strength within a high-density wireless network, specifically looking at the Link Quality Indicator (LQI) as provided by the physical layer and accessible at the networking and application layers within the stack protocol. It also investigates methods by which LQI can be used for discovery, identification, and localization of nodes within a ZigBee wireless network. The thesis concentrates on practical approaches specifically as it would pertain to commissioning a high-density network for an application such as lighting control in building automation. There are seven potential algorithms proposed using factors such as minimum distance error, perceived confidence, and triangulation. Experiments, which explore the reliability of signal strength indicators, reveal that the fluctuations of the signal strength indicate that a simple, inexpensive algorithm is insufficient. Simulations, which explore varying conditions such as network layouts, node-count, and node-density, reinforce this notion; however, this thesis also shows that more complicated and expensive methods do show promising results. Using the simplest of methodologies, the experiments and simulations result in 30-35% accuracy. However, with the more complicated methodologies (using techniques such as reiteration, emulation, and cooperation), the results reveal accuracies of 80-90%. This thesis concludes from these results that a simple algorithm and methodology may not be sufficient but that there are approaches that can improve the results. However, the cost of these approaches may be deemed too high for practical use. Further exploration in these methodologies, though, should show promise that more sufficient accuracies without sacrificing too much cost are attainable.

Dedications

This thesis is dedicated to my current and future family – my parents, my sister, and my fiancé; without each of whom, I could not be where I am now, I could not have accomplished what I have accomplished thus far, and my future would not seem so bright without them. Thank you for everything.

Acknowledgements

I would like to thank first and foremost Dr. Peterson as my advisor for his guidance, input, and determination to put me in a position to succeed. Moreover, I would like to thank him for his patience, understanding, and direction that he so willingly gave to a formerly somewhat directionless graduate student. Without his help, this thesis certainly would not be what it is.

I would also like to thank Ron Thompson and his team at Eaton who helped fund, direct, and motivate me in my efforts. They provided all of the equipment, facilities, and opportunity that could not be obtained anywhere else. I would also like to thank the ZigBee research team at the Eaton Innovation Center in Milwaukee, Wisconsin, who provided extensive training and a research foundation which jump-started me in an emerging technology that I otherwise would have had very limited knowledge about. Specifically, I would like to thank Luis Pereira for his extensive help, training, and trouble-shooting to assist in this thesis, and I would like to thank Charles Luebke and Sujit Das for their directional support and allowing me to take time from their research team in order to assist my efforts. In addition, I want to thank Dave Davidson and Alan Wyant out of Sumter, South Carolina, who provided yet another level of support from within the Eaton team. They helped provide the motivation and direction for this thesis as they lead the industry research in the wireless lighting control business. Finally, I would like to thank Eaton's strategic partners, Jim Sekinger and William Sandoval of Philips Advance and Philip Ivester of H I Solutions, for their additional support in helping me understand and see the problems and obstacles that exist in the commissioning, discovery, and operation of a ZigBee network application. There are so many other people just within Eaton who have assisted me in many ways, and I want to thank them all. Eaton, as a whole, has provided me opportunities that I would not trade at all, and all of whom I cannot thank enough.

Most importantly, though, I want to thank my friends, my family, and my fiancé. I want to thank my friends who helped me get through the tough times and were there to celebrate with me during the good times. I want to thank my family for the twenty-four years of endless support to put me in the position I am in today. They have pushed me when I needed to be pushed, they have given me the support I have always needed, and they have provided me the opportunities with which I have been given. In doing all of this, they never failed to show me their love, and for that, I am truly blessed. I will never take their efforts for granted. Finally, I want to thank Terri, my fiancé, who has mostly only known me during this turbulent time in my life but still chooses to love and support me. Those long days and nights of staring at a computer screen for hours at a time would have been just that were it not for her. Instead, these have been the most joyous days of my life with her by my side.

Finally, all glory and honor that I might be afforded for my efforts are given to God, for without His mercy, grace, and peace, my efforts would be fruitless.

"Consider it pure joy, my brethren, whenever you face trials of any kind, for you know that the testing of your faith develops perseverance. Perseverance must finish its work so that you may be mature and complete, not lacking anything."

James 1:2-4

Preface

This thesis was written as the result of conjunct independent research from within the University of Tennessee, Knoxville, and Eaton Corporation. Eaton is an active member of the ZigBee Alliance and one of the forerunners in developing a ZigBee-compliant wireless lighting control solution as well as some residential monitoring applications and other general research. The motivation behind this thesis is based on Eaton's commercial belief that commissioning an application such as lighting control is going to be a difficult task. Therefore, a 'self-commissioning' application would be very attractive and usable.

The motivation for doing this specifically for lighting control is because of the 'control' aspect of the wireless network. In a scenario where monitoring is the only concern, discrete discovery and identification is not of concern because the identity will be inferred when the data is collected. However, in a control scenario, some central node will need the capabilities to control a specific node, which, in the lighting control scenario, is a light ballast. Therefore, it needs to know all of the network's nodal identities in order to know which group of lights to turn on or off for example. Moreover, in the lighting control application, it is likely that in a typical building, ballasts will be placed about 2-3 meters apart, but research shows that most localization algorithms can only achieve accuracies of about 5 meters without requiring additional hardware, which is the motivation for focusing on high-density networks.

The goals of this thesis are to extend and add to specific problems forecasted by Eaton's research efforts and conjunctly add to the general research community within the University of Tennessee as well as the general development of ZigBee applications. The funding for all of the equipment used in the experiments as well as all of the facilities used in both the experimentation and development of this thesis were provided by Eaton.

Table of Contents

Chapter 1: Introduction 1

1.1 Goals 1

1.2 Motivation 2
- *1.2.1 Emergence of Wireless Sensor Networks 2*
- *1.2.2 Emergence of 802.15.4 and ZigBee 3*
- *1.2.3 ZigBee in Real-World Applications 3*

1.3 Background 4

1.4 Related Work 6

Chapter 2: Technical Background 8

2.1 Wireless Technology Overview 8
- *2.1.1 Wi-Fi Overview 8*
- *2.1.2 Bluetooth Overview 10*
- *2.1.3 ZigBee Overview 11*

2.2 Wireless Sensor Networking Overview 12

2.3 ZigBee Wireless Technology 13
- *2.3.1 Protocol Stack Layers 13*
- *2.3.2 Device Types 13*
- *2.3.3 Network Topologies 14*
- *2.3.4 Channels 15*
- *2.3.5 The ZigBee Alliance 18*
- *2.3.6 ZigBee Application Programming 19*

2.4 Wireless Signal Strength 20
- *2.4.1 Received Signal Strength Indicator 20*
- *2.4.2 Link Quality Indicator 21*

2.5 Node Localization 22
- *2.5.1 Reference versus Non-Reference Localization 22*
- *2.5.2 Serial versus Parallel Localization 23*

2.6 Other Localization Techniques and Metrics 24
- *2.6.1 Global Positioning System 24*
- *2.6.2 RADAR 24*
- *2.6.3 SONAR 25*
- *2.6.4 In-Building RADAR 25*
- *2.6.5 Cricket Indoor Location System 26*

2.6.6 Comparison of Techniques 26

2.7 Chapter Summary 27

Chapter 3: Design ***28***

3.1 Goals and Motivation 28

3.2 Assumptions 29

3.3 Approach 30

3.4 Other Considerations 31

3.5 Algorithms 32
3.5.1 Perceived Confidence without Contention 33
3.5.2 Perceived Confidence with Contention 35
3.5.3 Relative Confidence without Contention(Fig. 3-3) 37
3.5.4 Relative Confidence with Contention 39
3.5.5 Minimum Distance Error without Contention 41
3.5.6 Minimum Distance Error with Contention 42
3.5.7 Radial Triangulation without Contention 42

3.6 Expectations and Hypothesis 45

3.7 Chapter Summary 46

Chapter 4: Experiments ***47***

4.1 Equipment and Resources 47

4.2 Approach 50
4.2.1 Data Collection and Modeling 50
4.2.2 Interference Considerations 51
4.2.3 Node Count and Node Density Considerations 52

4.3 Environmental Scenarios 52
4.3.1 Line of Sight with Limited Interference 52
4.3.2 Line of Sight with Typical Interference 55
4.3.3 Multiple LOS with Typical Interference 56

4.4 MatLab Simulations 57
4.4.1 GUI Overview 57
4.4.2 Error Modeling 58
4.4.3 Algorithm Implementations 58

4.5 Chapter Summary 61

Chapter 5: Results 62

5.1 Experimental Results 62
5.1.1 Line of Sight with Limited Interference 62
5.1.2 Line of Sight with Typical Interference 67
5.1.3 Multiple LOS with Typical Interference 70

5.2 Simulation Results 71
5.2.1 Error Modeling 71
5.2.2 Algorithm Results 73

5.3 Chapter Summary 81

Chapter 6: Conclusions 83

6.1 Conclusions 83

6.2 Challenges 85

6.3 Future Work 85

Bibliography 89

Appendix A 95

Appendix B 114

Other Figures and Simulation Results 114

Vita 118

List of Figures

2-1	Wireless technology mapping	**9**
2-2	LR-WPAN stack layer protocol	**14**
2-3	Examples of network topologies	**16**
2-4	Summary of frequencies of operation in all three bands	**17**
2-5	Framework for a generic stack architecture	**19**
3-1	Perceived confidence w/o contention algorithm pseudo-code	**34**
3-2	Perceived confidence w/ contention algorithm pseudo-code	**36**
3-3	Relative confidence w/o contention algorithm pseudo-code	**38**
3-4	Relative confidence w/ contention algorithm pseudo-code	**40**
3-5	Minimum distance error algorithm pseudo-code	**41**
3-6	Minimum distance error w/ contention algorithm pseudo-code	**43**
3-7	Radial Triangulation w/o contention algorithm pseudo-code	**44**
4-1	Chipcon's CC2420DBK development board	**47**
4-2	MaxStream XBee development kit	**48**
4-3	Chipcon CC2420DK packet sniffer	**49**
4-4	Screenshot of Chipcon packet sniffer software	**50**
4-5	Office layout where experiments take place	**53**
4-6	Overview of warehouse experiments	**54**
4-7	Office space single LOS experiments	**55**
4-8	Office space multiple LOS experiments	**57**
4-9	Example screen shot of MatLab GUI	**59**
4-10	Sample Gaussian distributions	**60**
5-1	Experimental results from warehouse 'walking' experiments ranging from 0 to 45 feet	**63**
5-2	Discrete averaging of packet LQI	**64**
5-3	A combined graph of each of the warehouse experiments from 0 to 45 feet	**65**
5-4	Fully modeled data in 'lossless' scenario	**66**
5-5	Experimental results for LOS office experiments	**67**
5-6	Spectrum analysis of various channels	**68**
5-7	Comparing channels 0x12 and 0x16 in office LOS experiment	**69**
5-8	Office layout for multiple LOS experiments	**70**
5-9	Actual data results from nodes placed throughout office	**71**
5-10	Gaussian error model for +/-25 dBm	**72**
5-11	100 random samples of Gaussian distribution in +/- 25 dBm range	**73**
5-12	Results of the actual data run through the simulations	**75**
5-13	Confidence without contention algorithms as node density and error increase	**76**

5-14	MDE with contention algorithms as node density and error increase	**76**
5-15	Accuracy as base point count increases (1 to 12)	**78**
5-16	Accuracy as base point count increases (1 to 20)	**78**
5-17	Accuracy as emulated base point count increases (5x1 to 5x4)	**79**
B-1	Linear modeling of LQI versus Distance	**117**
B-2	Example 4-base (centrally-located) node layout in GUI	**118**
B-3	Individual simulation result for MDE w/o contention	**118**
B-4	Individual simulation result for perceived confidence with contention	**119**
B-5	Individual simulation result for perceived confidence without contention	**119**
B-6	Individual simulation result for relative confidence with contention	**120**
B-7	Individual simulation result for relative confidence without contention	**120**

List of Tables

2-1	Summary of Bluetooth Classes	**10**
2-2	Comparing 3 Primary Wireless Standards	**12**
5-1	Accuracy table for algorithms using real data	**74**

List of Equations

4-1	Gaussian Distribution	**58**
5-1	Average of three trend lines	**63**
5-2	LQI as a function of distance according to the data in Figure 2-2	**64**
5-3	Trend line for data in Figure 5-4	**66**
5-4	Combined trend line for 2 LOS office experiments	**68**
5-5	Expected LQI function of distance based on previous experiments	**72**

Chapter 1

Introduction

1.1 Goals

The goals of this thesis are to inspect, in general, the reliability of ZigBee wireless signal strength in multiple environmental and node-layout scenarios. More specifically, this thesis intends to investigate how ZigBee's built-in Link Quality Indicator, or LQI, can be used as a reliable distance indicator for use in localization and discovery in a high-density wireless sensor network within a real-world application. There have been other similar algorithms ([2], [5], [10], [16], and [20]) used for localization and tracking, but the resolution of those algorithms do not appear to be sufficient for high-density localization, such that one can distinguish between two nodes that are very close to one another (close here is defined as within 5 meters). Moreover, the complexity requirements for some of these algorithms, are too high for most real-world applications. In addition to higher resolution for higher density networks, the other focus of this thesis is to look at how discovery will be needed in a real-world application, which allows us to make some assumptions such as pre-defined layouts but also force us to

remove other assumptions such as interference and environmental conditions.

1.2 Motivation

1.2.1 Emergence of Wireless Sensor Networks

Wireless technologies are slowly improving in their reliability, bandwidth, and generally in their practical use. Slowly we are seeing the wired world transition into a wireless community wherever possible. The power of wireless networks working together in ad hoc, mesh natures are close to limitless except, of course, for their physical limitations. A wireless network will never truly be as reliable and efficient as a wired network for the most part, but there are many applications that the wireless world can make tremendous improvements upon and open up a whole new realm of applications.

One of these emerging applications of networking where the wireless world has opened new possibilities is the concept of wireless sensor networks. A wireless sensor network (WSN), loosely defined, usually consists of a large set of limited functionality sensor nodes that communicate usually in an ad hoc, mesh network through some wireless medium. These wireless sensor networks are usually used for monitoring or control, which typically do not require high data transfer rates which consequently lower the power and cost requirements. These networks have been in existence for several years now, but over the last few years, there has been a movement to standardize these communication protocols.

1.2.2 Emergence of 802.15.4 and ZigBee

One of the newest protocol standards to emerge is the IEEE 802.15, which is the Wireless Personal Area Network, or WPAN, which defines a low-bandwidth communication with low power consumption. The WPAN standard defines the physical (PHY) and medium access control (MAC) layers for such a radio. On top of these standards, ZigBee is an attempt to develop application, security, and networking layers on top of the 802.15.4 standard [28]. In order to accomplish this, several large companies, who were in the forefront of the wireless technology development, created the ZigBee Alliance [26]. Since then ZigBee has slowly developed application profiles for target applications that the ZigBee Alliance has identified as key areas where ZigBee can be utilized.

1.2.3 ZigBee in Real-World Applications

Wireless sensor networks, traditionally, have been used typically in research environments or very specific non-generic job tasks. In the last few years, though, as the wireless standards such as ZigBee are beginning to develop, we are starting to see more and more applications hit the commercial and residential marketplaces with even some products that can be purchased off of the shelf. For example, we are already starting to see some ‘smart home’ applications move to a wireless approach. Eaton Corporation is in early launch production of a home awareness application that can monitor critical points of interest within a home such as electrical outlets and hot-water heaters, and it is utilizing the ZigBee standard [11]. An especially appealing aspect of using ZigBee wireless is

the fact that it allows for a product to be easily retrofitted into a home or building.

In more of an industrial setting, building automation is probably, at this point, the most developed of the application profiles within the ZigBee Alliance [26]. One specific application is wireless lighting control, where Eaton Corporation is also a leading forerunner. This is a particularly challenging application as it pushes the limits of the ZigBee networking layers in terms of node count, node density, and routing algorithms. In a lighting control environment where the node is the light ballasts, there is now a node in every light fixture which is typically only a couple meters apart and spread very thoroughly throughout a building. This type of setting (with high node counts and node density) and application is the primary focus for this thesis. Typically, in this thesis, whenever a "real-world application" is mentioned, it is referring to a scenario where node count is high and densely distributed and where cost and complexity are weighed heavily. While the focus of the thesis is not on these applications specifically, it assumes similar networking and wireless signal strength implications.

1.3 Background

As mentioned in the preface, this thesis is being guided and funded in conjunction with the research efforts of Eaton Corporation, who are one of the leaders in the industrial lighting control business and one of the forerunners in both ZigBee development and the development of a wirelessly-controlled lighting control system. Eaton has formed a strategic partnership with one of the world's

leaders in light ballast development, Philips Advance Ballasts, to lead the research in developing a fully ZigBee-compliant wireless lighting control solution. The research found in this thesis is guided by Eaton and their leading efforts in this emerging technology. The basis of this thesis is on specific problems that have been encountered as the limits of the ZigBee standards are pushed in real-world applications. The goals reflect the need for an industry-level robustness to accommodate this approach to high-density nodal discovery such that a company like Eaton could implement any algorithms and would gain satisfactory results such that it could be placed in a commercial product. This standard is significantly higher than the typical research application and results in a need for high accuracy rates for a wide range of scenario considerations.

Because lighting control requires such a high node count (every light fixture in a building is potentially a wireless node) and a high node density (light fixtures are typically only about 2 meters apart) and since it requires discrete control over each individual node (which most control applications are going to require), there are two primary issues that arise: network traffic and node discovery. Network traffic is a problem that should be handled at the network layer, and it should hopefully be a solvable problem once the standards become more robust. However, there is no element within the stack protocol to handle location discovery. Within the stack protocol, a coordinator node does have the capability of discovering node identities that are in its network, but it has no way of knowing where that node is located, which is what location discovery is. This is critical to

an application such as lighting control, because each node is controlled differently based on its location.

1.4 Related Work

There has been significant research in the area of localization for wireless sensor networks[2][5][10][16][20][24]. The majority of the scenarios investigated, though, include node tracking, robot localization, or small node count with sparse distribution. Chapter 2 discusses more thoroughly the specifics of some similar algorithms, but in general, it is important to identify some of the differences between previous research and what this thesis hopes to accomplish. Many localization techniques that exist in practice, involve tracking a node. In a tracking scenario, one is typically only using two to four nodes (one for tracking and one to three as base points). Even then, one is assuming an initial position, and then it is using directional movement information to augment one's knowledge. Additionally, in a tracking scenario, one is not typically interested in very precise distance measurements. Generally in such a scenario, one is only interested in general location so that a 5 meter error is not a significant issue. The robot localization problem is very similar to the tracking scenario, but the accuracy becomes a slightly more important factor. The robot localization scenario still is using a limited number of nodes in relatively sparse networks. Even in scenarios where the goal is simple localization of a non-moving set of nodes, the node count is typically small and the node density is sparse. Therefore, the goals of this thesis are to investigate specifically, higher node

counts, higher density networks, and obtaining higher accuracy and precision.

1.5 Goals Overview

The primary goals of this thesis are to investigate the signal strength factors present in the ZigBee wireless communication protocol and how these factors can be translated into an estimated distances for the purposes of discovery, identification, and localization of a high node-count, highly dense, network in a relatively feasible manner for a real-world application, such as lighting control. These goals are relevant based on the emergence of ZigBee as a potentially widely used technology in the building automation industry. As the node count an node density increase, these issues of discovery, identification, and localization, become a significant issue. The hope is to gather some real-world data from experimentation, develop some models based on the limited resources, and then develop and simulate some algorithms to solve the discovery issue. The motivation and basis for such design and experiments is from real-world experience in the development of first generation application by companies such as Eaton and Philips.

Chapter 2

Technical Background

2.1 Wireless Technology Overview

Wireless technology has quickly progressed over the years to become an extremely robust method of communication across most bandwidths. Some of the common wireless technologies today range from cellular phone technologies (1G-3G) and Wi-Fi down to lower-level communications such as Bluetooth and ZigBee. These protocols vary in their physical specifications in radio frequencies, cost, complexity, power consumption, and bandwidth. This thesis takes a deeper look at two of the more common wireless protocols, Wi-Fi and Bluetooth, and compare them with ZigBee, specifically with elements that are relevant to the goals of this research. Figure 2-1 below shows a mapping of all of the major wireless communication protocols and where they fall in terms of data rate, cost/complexity, and power consumption [19].

2.1.1 Wi-Fi Overview

The IEEE 802.11 standard protocol, commonly referred to as Wi-Fi defines the set of standards that fall in the category of Wireless Local Area Networks, or

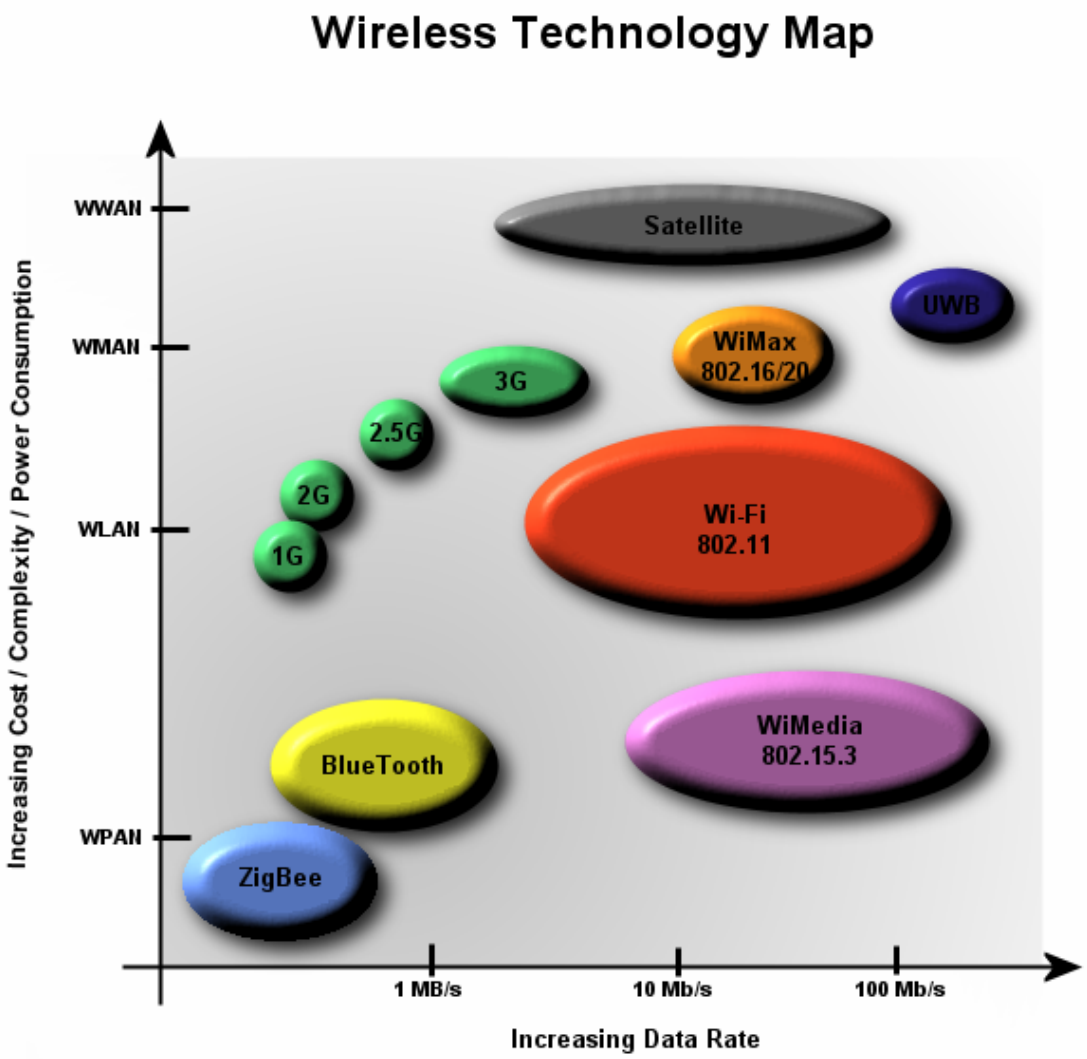

Fig 2-1: Wireless Technology Mapping

WLAN. The WLAN family consists of 802.11b and 802.11g, which operate on the 2.4 GHz range. The 2.4 GHz range is part of FCC Rules and Regulations, which means anything that operates under this frequency will likely incur interference from other devices such as microwaves, cordless telephones, and of course, other wireless technologies that operate under the same frequencies. The 802.11a standard operates on the 5 GHz range, which allows it to avoid the aforementioned interferences. Each of the WLAN standards which are in the 2.4 GHz range operate under 14 channels with center frequencies 5 MHz. The data rate for 802.11b and 802.11g are about 6.5 Mb/s to 54 Mb/s. Newer standard, 802.11n, increases data rates up to 540 Mb/s, which allows for multimedia streaming. The range for Wi-Fi is 25 to 100 meters or up to 125 meters if one includes 802.11n.

2.1.2 Bluetooth Overview

Bluetooth is a specification that operates under the WPAN umbrella, which is defined by the IEEE 802.15.1 standard. The Bluetooth radio is designed for low power applications, but this also results in shorter ranges. There are three classes defined in this standard ranging from a 1 meter range to 100 meter range with corresponding power ranging from 1 mW to 100 mW [3]. Table 2-1

Table 2-1: Summary of Bluetooth Classes

Class	Maximum Power	Approximate Range
Class 1	100 mW	100 m
Class 2	2.5 mW	10 m
Class 3	1 mW	1 m

summarizes these classes of Bluetooth. These low power rates allow Bluetooth to be a perfect fit in low-level applications such as communication between small computer or cell phone peripherals such as headphones, mouse and keyboard, remotes, or similar devices.

2.1.3 ZigBee Overview

This thesis discusses the specifics of ZigBee later, but in general, ZigBee is an application and networking framework on top of the IEEE 802.15 WPAN standard. It is specifically defined under the IEEE 802.15.4 standard which defines LR-WPAN, or Low-Rate Wireless Personal Area Network. It is very closely related to Bluetooth in terms of its data rate, range, and power consumption. However, Bluetooth seems to be more of a better fit for applications such as small electronic devices as previously mentioned. The intentions behind the formation of ZigBee were to create an application and networking layer that would provide reliable, robust low data rate communication with low power consumption such that it could be used in an industrial environment [28]. Bluetooth is limited in the number of nodal neighbors it can allow, and battery life is not very long. ZigBee aims to improve on both of these factors, and at the same time reduce the price of the radio, so that it can be suitable for large wireless mesh networks.

2.1.4 Overall Comparison

IEEE 802.11 WLAN, or Wi-Fi, operates at higher bandwidths, but the cost and

Table 2-2: Comparing 3 Primary Wireless Standards

	ZigBee LR-WPAN 802.15.4	Bluetooth WPAN 802.15.1	Wi-Fi WLAN 802.11
Range	10 – 30 m	~10 – 100 m	~100 m
Data Throughput	< 0.25 MB/s	1 MB/s	~2-11 MB/s
Power Consumption	Ultra Low	Low	Medium
Size	Smallest	Smaller	Larger
Cost	~$1	~$5	~$10

complexity are too high for an embedded sensor application. On the lower end of the bandwidth chart is Bluetooth, which is used primarily for low-level communication between small remote devices. ZigBee wireless falls very closely in the range of Bluetooth in terms of bandwidth and cost, but it is designed for a much more robust industrial use. Table 2-2 summarizes these comparisons [19]. As one can see from these comparisons, ZigBee sacrifices range and data throughput for the sake of extremely low power consumption and cost which is an essential factor for an industrial application using wireless sensors. When comparing the complexity of these protocols, the 802.15.4 standard is significantly less complex than the others in terms of both documentation and code size which affords lower memory requirements and thus less expense in development and implementation [19].

2.2 Wireless Sensor Networking Overview

Wireless sensor networking, in general, is nothing more than the collection of several wireless nodes typically used for monitoring and/or control. They

typically are arranged in a mesh topology with ad-hoc capabilities. Often times, though, there is need for a coordinator to issue the control commands or aggregate the data being collected. Typical topologies for this would be a star or cluster-tree diagram, and the non-coordinator devices are typically also ad-hoc. Typically they also need to be easily distributed and moved around (thus the typical need for ad-hoc capabilities), and low power consumption is very useful so that the life of the node is longer and batteries can be used as opposed to wiring power to the device which defeats some of the advantages of its wireless communication (though power is almost always easier to wire than data communications).

2.3 ZigBee Wireless Technology

2.3.1 Protocol Stack Layers

The ZigBee standard defines the Network Layer (NWK) and Application Support Layer (APS) on top of the IEEE 802.15.4 MAC and PHY layers as shown below in Figure 2-2. The NWK layer holds all of the routing algorithms including some limited routing tables and neighbor information. On top of the APS layer is the application-specific layer. The PHY layer holds two separate layers – one for the 900 MHz range and one for the 2.4 GHz range [19] [28].

2.3.2 Device Types

The ZigBee standard defines two different types of devices: a Fully-Functional Device (FFD) and a Reduced-Functionality Device (RFD). A RFD can only

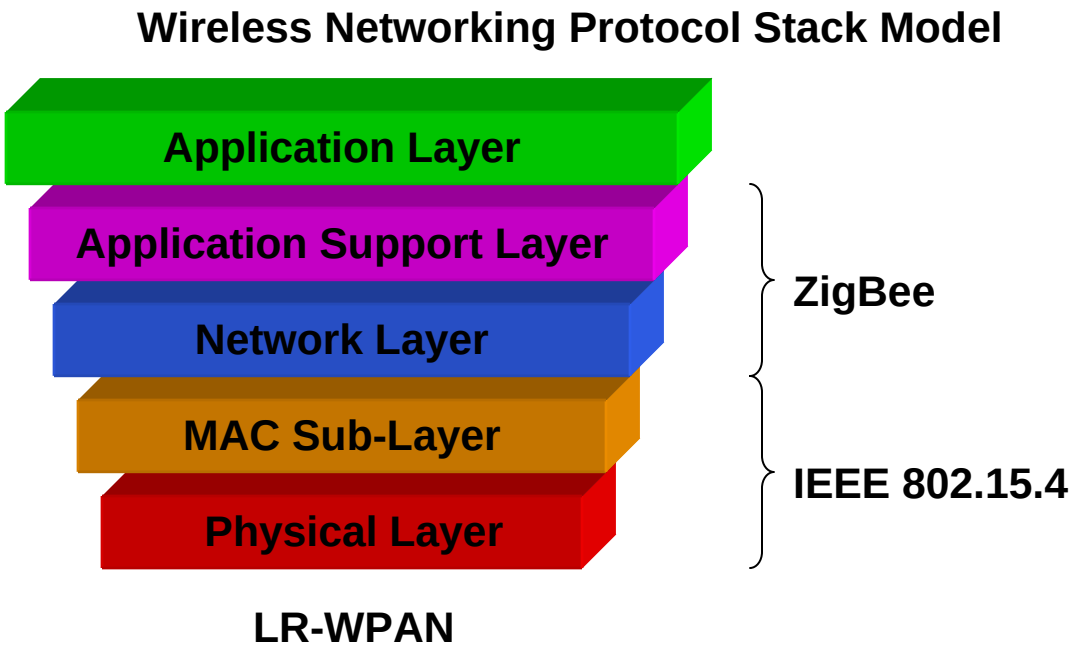

Fig 2.2: LR-WPAN Stack Layer Protocol

transmit and receive data (or create a binding) with only one FFD. FFD's, on the other hand, can transmit and receive data with any number of FFD's or RFD's. Within these two types of devices, we can define three types of nodes: coordinator, router, and end device. The end device is simply an RFD. The coordinator and router are both FFD's. There must be and can only be one coordinator in a network. The coordinator establishes the network parameters such as channel, topology, and other routing and identification parameters for the network. The router is a simple FFD that can act as an intermediary hop if necessary [28].

2.3.3 Network Topologies

There are three primary types of network topologies that are supported: star,

cluster tree, and mesh (or peer-to-peer). The three types of networks are compared in Figure 2-3. The star topology contains one coordinator node, and all of the end devices are then connected directly to the coordinator. The cluster tree contains a central coordinator node, which is connected to several end devices as well as routers. Those routers are then connected to their own set of end devices and routers. The tree continues in similar fashion as needed. The mesh network topology, commonly referred to as peer-to-peer, interconnects every device. Typically, if one wants a fully-connected mesh network, the mesh network topology consists only of a coordinator and router nodes. The star topology is very easy to setup and requires minimal overhead and virtually no routing complexity. The disadvantages of such a topology include a central point of failure, network traffic bottlenecking, and one is unable to perform multi-hop communication. The cluster tree topology allows for multi-hop communication and decreases the other two disadvantages, but it does not completely eliminate them and increases the overhead and routing complexity. The mesh network topology is by far the most complex in terms of routing and overhead, but it virtually eliminates the disadvantages in the previous topologies [19] [28].

2.3.4 Channels

The IEEE 802.15.4 standard defines 27 channels that ZigBee can operate under. There are 16 channels in the 2.4 GHz ISM (Industrial, Scientific, and Medical) band, 10 channels in the 915 MHz ISM band, and 1 channel in the European 868 MHz band. The summary of those frequency bands is given in Figure 2-4. In the

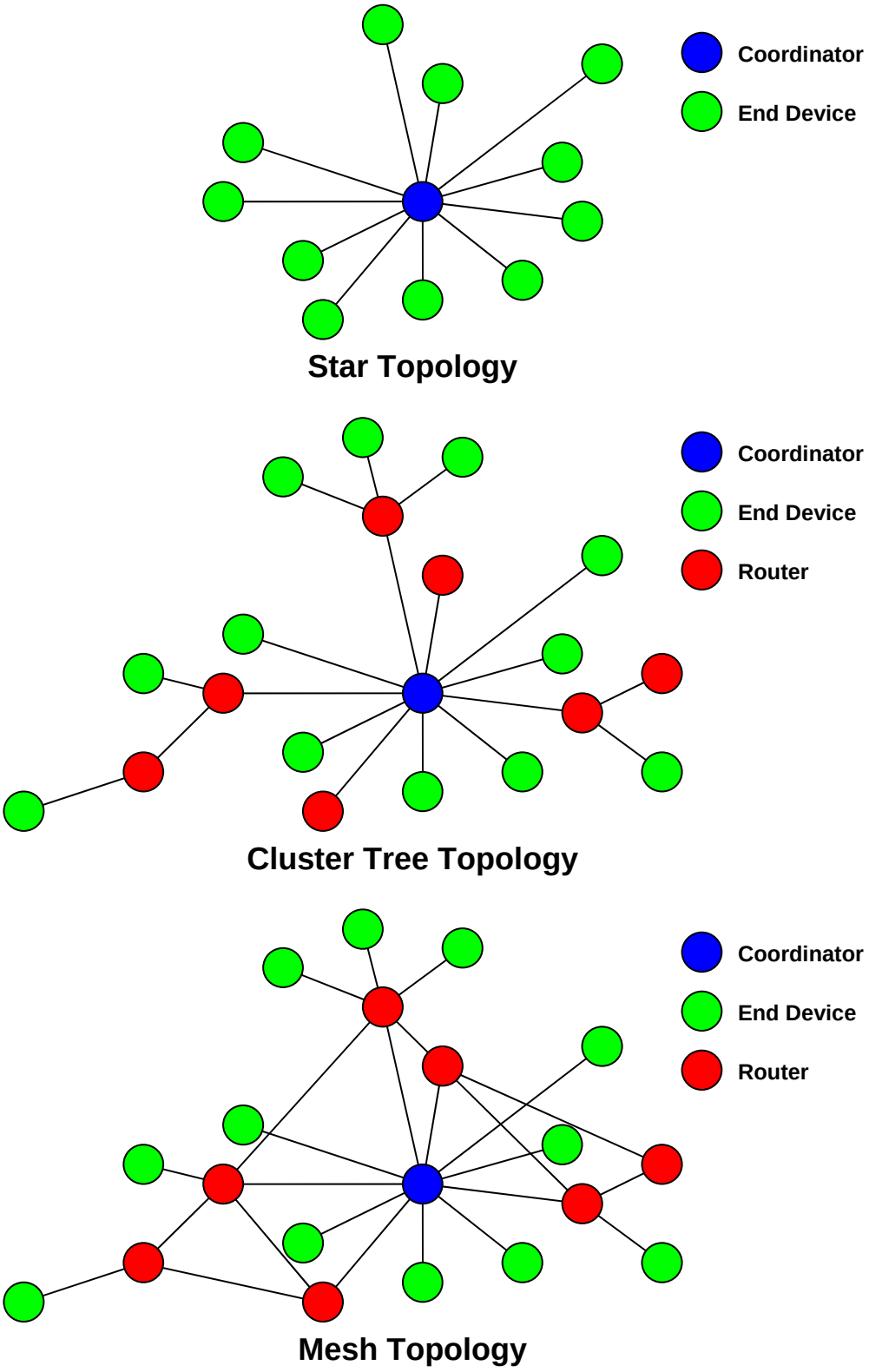

Fig 2-3: Examples of Network Topologies

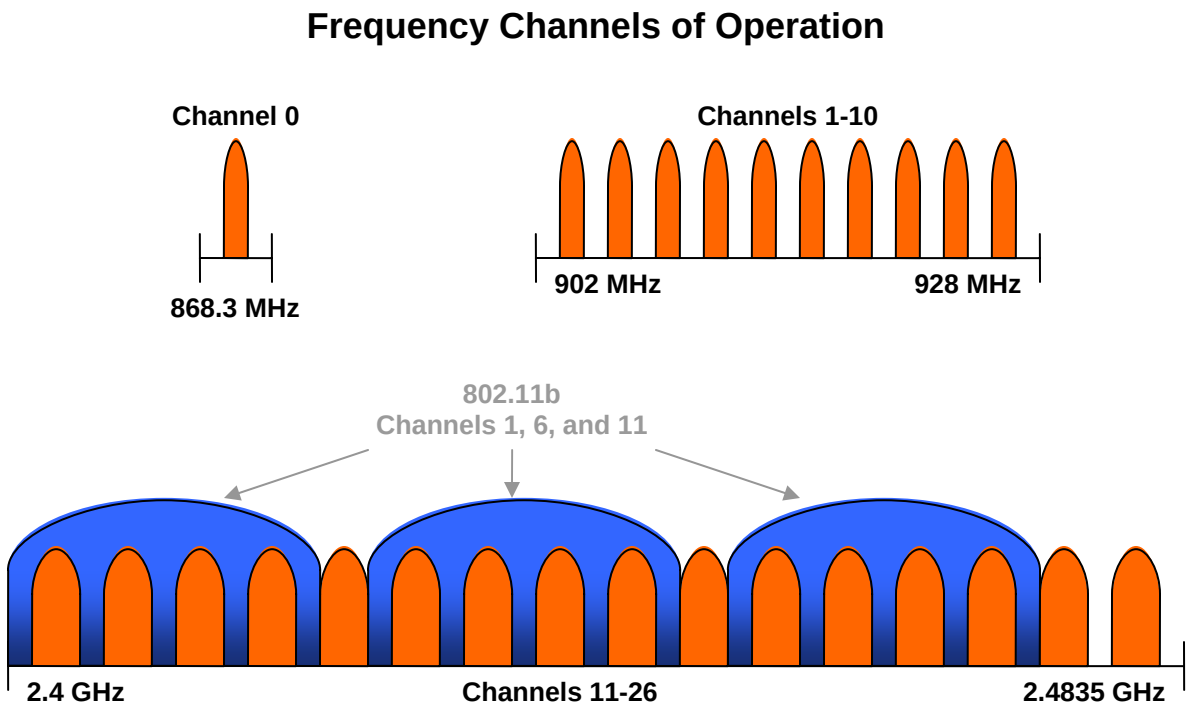

Fig 2-4: Summary of frequencies of operation in all three bands

915 MHz frequency band, the center frequencies are 2 MHz apart, and in the 2.4 GHz frequency band, the center frequencies are 5 MHz apart. The figure also shows how the 2.4 GHz band interferes with 802.11b Wi-Fi. This indicates that most likely channels 15, 20, 25, and 26 should contain the least interference in a Wi-Fi environment. The 868 MHz band is available only in Europe, the 915 MHz band is available only in North and South America, and the 2.4 GHz band is available world wide. There are some key differences between each of the bands, specifically data rate. In the 2.4 GHz band, one is able to achieve 250 kb/s versus 20 kb/s for the 868 MHz band and 40 kb/s in the 915 MHz band [19] [28]. It appears that the most common band that is used in early development is the 2.4 GHz. Despite the potential for interference, the higher data rate and the

fact that it is available world wide makes the 2.4 GHz band the most desirable at this point.

2.3.5 The ZigBee Alliance

The ZigBee Alliance, as previously mentioned, was formed as an independent non-profit organization to combine the work of several companies which lead the development of ZigBee to define an open standard for the layers not defined by IEEE 802.15.4 [26]. These layers primarily consist of the Application Support Layer (APS), the Networking Layer (NWK), and the Security Layer (SSP), which are often referred to as the “upper” layers. The alliance also serves as a bit of a marketing tool for the technology as a whole. Because it is an open standard, anyone can be a member of the alliance (assuming one can pay the fees, of course).

When the alliance was formed, there were six target areas that the alliance felt they should concentrate on for ZigBee development. They are consumer electronics, PC peripherals, personal healthcare, residential and lightweight commercial, building automation, and industrial control [26]. For each application within these categories, the hope is to create a standard profile in order to make development easier as well as to ensure that all devices, no matter the vendor, can inter-operate and sustain an operable network. Figure 2-5 shows a general architecture of what an application stack would look like. The application profiles for building automation and lighting control have seen the most progress and are

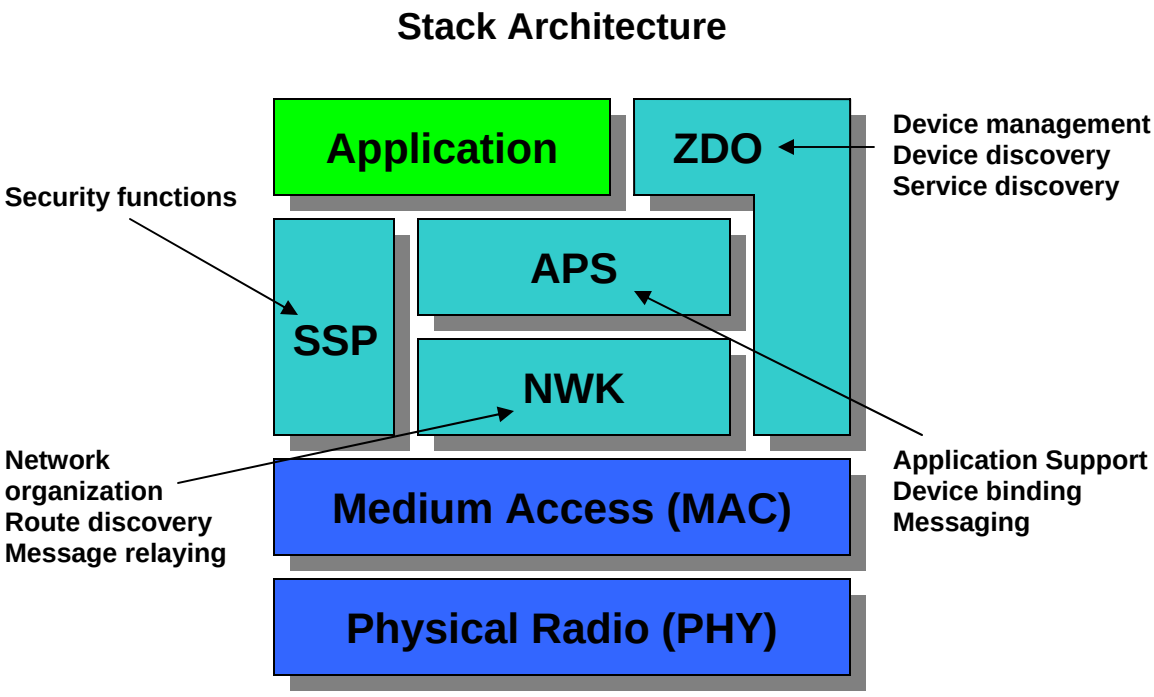

Fig 2-5: Framework for a generic stack architecture

in fact the closest application profiles to completion within the ZigBee Alliance [26]. For every application identified, a committee is formed to begin standardizing that application profile. Even though this is an open standard, because the committee is often formed of competing members in the commercial environment, it can sometimes be counter-productive. Nevertheless, the application profile list continues to grow, and slowly applications are starting to be ready to be pushed out of the door.

2.3.6 ZigBee Application Programming

The programming model for an embedded ZigBee application is dependent on the application profiles developed by the ZigBee Alliance as mentioned in the

previous section. The profile defines a set of functional identifiers called end points, cluster IDs, and attribute IDs. Typically the end point is given to identify the application. Cluster IDs can be used to either group a set of application devices or define a set of functions. The attribute ID then specifies the function desired, or the specific element or information it associated with the function. These IDs form something similar to an address to a function or functionality to make data transfer, requests, and tasks easier and quicker to perform. The lower-level programming such as network creating or joining, device bindings, etc. are handled automatically and assumed from the device type (coordinator, router, or endpoint). For the purposes of this thesis, the Texas Instruments Z-Stack [27] programming profile is used which helps implement the above.

2.4 Wireless Signal Strength

2.4.1 Received Signal Strength Indicator

Radio signal strength has commonly been used for position location. A common indicator found in most wireless radios is the Received Signal Strength Indicator, or RSSI. For this thesis, the Chipcon CC2420 radio transceiver is the primary radio used [27]. It includes a digital direct sequence spread spectrum (DSSS) modem which provides a gain of 9 dB with the maximum data rate being 250 kb/s. It supports 8 discrete power levels including 0 dBm, -1 dBm, -3 dBm, -5 dBm, -7 dBm, -10 dBm, -15 dBm, and -25 dBm. The RSSI value is an 8-bit number averaged over 8 symbol periods. There is also a status bit which indicates when the RSSI value is valid. There are several aspects which

contribute to the inconsistency of the RSSI value other than basic interference. This includes transmitter variability, receiver variability, and the antenna position and direction. In the experiments in this thesis, the same two nodes for all transmissions are used for modeling and simulations; therefore, these variations should not be an issue. Typically, to model RSSI as a function of distance, the assumed equation has a logarithmic relationship with some added Gaussian noise. In other techniques attempting a direct correlation, the error variation seems to be too much to provide any real precision or accuracy high enough for practical use [7] [27].

2.4.2 Link Quality Indicator

The Link Quality Indicator, or LQI, is a variation of the concept of RSSI, which is provided by the IEEE 802.15.4 PHY layer specification. The LQI representations can sometimes characterize the received power, the estimated signal-to-noise-ratio (SNR), or some combination of both. In addition, though, the LQI uses packet error rate as one of its metrics, as opposed to average energy level as the RSSI does. The initial experiments for this thesis suggest that the difference is minimal, and the two are very closely correlated. Most research seems to suggest, though, that LQI is a much more reliable indicator, so it is the indicator used throughout this thesis. The LQI is passed through the MAC layer and then up to the network and upper layers. The LQI result is an integer ranging from 0x00 to 0xff (0 to 255) with the highest value resulting in the highest or best link quality [4] [16] [28].

2.5 Node Localization

The localization of nodes within a wireless network is a basic need for many applications. Node localization is simply identifying the location of a node within a 2-D or 3-D environment. This could involve tracking a single node moving across the plane, or trying to simply identify the location of a static node. It might seem that locating a single static node is easier than tracking a moving node, but this is not always true. With a moving node, one actually has more information. With a static node, the only information one can use is distance information (however it might be obtained) at that one distance. With a moving node, one can obtain directional information and use previous knowledge to help hone in on the accuracy. The precision is also dependent upon the assumed knowledge. In a tracking scenario, the user typically knows the starting point, which again provides an extra element that does not exist when trying to discover a new node.

2.5.1 Reference versus Non-Reference Localization

There are two types of localization algorithms: those that require a reference point (or points), or anchor, and those that do not, or anchor-free. Localization with reference points assume that there are some nodes in the network where the coordinates and distance information are known. From these reference points one can create an absolute coordinate system. In a non-reference algorithm, or anchor-free algorithm, one cannot assume any knowledge of the coordinates for any of the nodes. Therefore, the resulting localization is a

relative coordinate system based on inter-node distances [21]. This thesis primarily explores anchored algorithms because the assumption of pre-configured knowledge is easily afforded in the real-world applications that are being explored.

2.5.2 Serial versus Parallel Localization

Another separation point between localization techniques is whether or not the nodal discovery is done concurrently or one at a time. A serial localization usually requires some reference points used to identify a single node. It then progresses to the next node to identify it. Some algorithms may choose to add a newly localized node as a reference point. This is not always the best option, though. If there is an error in a single node localization and then that node is also used as a reference point, the error then becomes compounded as the algorithm progresses. Instead some algorithms will maintain the same reference points and then just reiterate the process to refine the accuracy until they are confident they can add a node as a reference point. Parallel localization utilizes all of the nodal information at the same time rather than looking at a discrete amount of information at a time. This sharing of information though can create intensely complex algorithms requiring higher on-board memory and network bandwidth. On the other hand, parallel localization helps prevent falling into local minima [20] [21].

2.6 Other Localization Techniques and Metrics

2.6.1 Global Positioning System

Global Positioning System, commonly known as GPS, is the most common used application for localization. GPS can be used for such large scale applications such as aircraft tracking down to tracking an individual. GPS uses a system of 27 satellites that follow well-known orbits around the earth. The satellites send a RF signal synchronized by an atomic clock to the device of concern. The device then calculates its position based on time differences in the received signals. The accuracy of this localization technique is about 15 to 30 meters. Differential GPS (DGPS) is a method to reduce the estimation to about 5 meters of accuracy by using a known position device to calculate potential delays in the atmosphere. GPS, though, does not work indoors, so it is relatively useless for a node discovery in building automation type applications [20] [21].

2.6.2 RADAR

Radio Detection and Ranging, commonly known as RADAR, systems are typically used in large-scale applications such as navigation or location of large vehicles or aircrafts. The architecture of a simple RADAR system includes a radio transmitter and receiver attached to a rotating antenna such that it transmits a short burst of radio signals which will reflect from larger objects back to the receiver. By measuring the time difference, one can use the speed of the RF signal (which is the speed of light) to determine the distance, and the angle of transmission gives one the coordinates of the object. This of course requires line

of sight, which is again relatively useless in an indoor application. This also does not give us any interaction with the actual object being located [20] [21].

2.6.3 SONAR

Sound Navigation and Ranging, or SONAR, is a system that is typically used underwater for navigation. It is virtually the same concept as RADAR except that it uses acoustic signals. It propagates a signal towards an object, receives the signal back, and then calculates the distance based on the time difference. Again, one can assume that knowing the angle of transmission will give one the location coordinates. Like RADAR, this technology is not very applicable to our problem [20].

2.6.4 In-Building RADAR

This system developed by Microsoft researchers utilizes information from a network of RF devices that already exists. RF signal strength is measured from a set of known positions to the unknown device. That signal strength is then translated into distance which builds a database of distances. This database is then processed at a central processor using triangulation methods. There are also versions of this method using 802.11, which provide an accuracy of about 3 meters. This system is probably the closest to what this thesis explores in terms of algorithm approaches. The accuracy is closer than most other systems, but still not close enough [2] [20] [21].

2.6.5 Cricket Indoor Location System

The Cricket indoor location system was developed by MIT specifically for high accuracy location. They claim to be able to achieve precision between 1 and 3 cm. The Cricket Technology combines the use of RF and ultrasound technologies. The node in question sends out a beacon via RF transmission as well as a simultaneous ultrasonic pulse. The receiving node then takes the time difference between the two pulses, and using the knowledge that RF is transmitting at the speed of light and ultrasonic at the speed of sound, the node can calculate an accurate distance. The accuracy of this system is obviously very appealing, but the downside to this application is the requirement needs for extra hardware (ultrasonic transmission and reception) and extra memory and processing capabilities (requires TinyOS implementation) [20] [21] [22].

2.6.6 Comparison of Techniques

There have been many other techniques that research the discovery and localization problem. Many of them though are just combinations of the above mentioned techniques. The GPS, RADAR, and SONAR methods are obviously not practical for an indoor environment, but they do provide a history of some common techniques and approaches. Some of these techniques can be mimicked in an indoor environment; though, while not actually performing the physical aspects of the technique, the concept is translated into a different physical medium. For instance, one could implement a RADAR technique indoors where instead of reflecting the signal off of an object, one simply receives

and relays a new message with the nodal identification and still use signal propagation delays as the estimation tool for distance.

The two indoor methods seem to provide the best results for the type of application we are looking at. The indoor cricket method would seem ideal if it were truly able to accomplish 1-3 cm accuracy. There would never be any need in an application for higher precision; however, the additional costs of memory, complexity, and even hardware are not a viable option. The in-building RADAR approach by Microsoft seems to be the most logical approach for this application. By leveraging the LQI information embedded in the ZigBee PHY layer and passed up to the application layer, if we can get an accurate correlation between distance and LQI, we can assume some pre-configured knowledge that would hopefully generate a higher accuracy than the 3-5 meters most similar algorithms seem to be able to achieve. For our application, we are looking at a nodal density of about 2 meters/node in a 2-D plane.

2.7 Chapter Summary

This chapter examined the technical background that is necessary to discuss and evaluate the algorithms that are used in our localization and discovery techniques. The construction of the ZigBee stack protocol is important in understanding what information is available at the application layer and how interferences might affect the reliability of signal transmission, which is key in many of the localization techniques.

Chapter 3

Design

3.1 Goals and Motivation

As ZigBee becomes a more and more prevalent technology within the industry of building automation, the node count for the wireless networks will begin to grow dramatically. In addition, as more devices become available, the node density will grow just as much. This poses a problem for those who have to install and commission these devices. The easy answer to such a scenario is to keep a record of the device identification numbers as they are being installed in the field. The reality of this scenario, though, is that this is more difficult to accomplish than it would seem. If numbers are being written down manually, they can easily be misinterpreted, nodes may be moved around after the initial installation, and when nodes have to be replaced, the likelihood that an accurate record will be kept is both unlikely and an unnecessary burden. When one compounds this with the fact that one often has multiple contractors and owners having to coordinate, this is just an unfeasible option. Therefore, there is significant need for some automated process to help discover, identify, and locate the nodes

within an indoor facility after the installation takes place.

3.2 Assumptions

Prior to discussing the approach to solving the above issue, it is important to identify the assumptions that must be made to make the problem solvable as well as some aspects that must be factored in to create an accurate solution. The first assumption is that we have full knowledge of the facility layout. The knowledge of the facility, though, does not have to be anything more than a bitmap image of the layout and a feet-per-pixel conversion factor, d_p. With these two factors, we can determine the distance between any two points in the facility.

The next assumption is that we know the locations of the nodes within the facility. It should be noted from the beginning that this is an assumption that is critical to the success of the proposed solution, but it is also an assumption that may not always be true. For instance, if a facility were to be equipped with wireless light ballasts, but not all of them were to be wireless, last minute design changes could create conflicts. Obviously if an incorrect node layout is given, the accuracy of the solution will most likely be completely compromised.

The last major assumption to consider is the environment. For instance, will a commercial retail environment react differently than a corporate office building or factory environment? The likely answer is yes. In general, we cannot make assumptions of a universal environment; however, for the sake of these experiments and the resources available, the scope of the environments considered is limited.

3.3 Approach

The importance in the approach to solving the discovery problem is to approach with reality in mind. In commissioning a complex system like a wireless network in a dynamic environment, conditions can vary widely, which is also why assumptions are important to ascertain accurate results.

The first step in finding a solution is determining the reliability of the signal strength factors. The Link Quality Indicator (LQI) defined by the IEEE 802.15 physical layer (PHY) standard is calculated for each received packet and appended to the packet at the PHY layer [28]. Therefore, it is a trivial task to obtain this information in experiments. By allowing two nodes to communicate at a fixed distance, one can determine how the LQI changes over time, and by moving the two nodes, one can determine how the LQI changes over distance. The primary concern for this design is distance; however, by examining it over time, we can gather some inclination as to how other interferences might affect the signal strength and thus the LQI. It is also important to consider the LQI over time in order to determine how many packets are required to generate an accurate average for the given distance. In other words, if the LQI fluctuates dramatically over time at a fixed distance, then it may require more transmissions to determine a more accurate estimation of the actual or expected LQI. If the packet LQI is fairly consistent, though, then it might only take a few transmissions to get an accurate estimation of the LQI.

Once this is modeled, we can determine some sort of LQI-to-distance conversion factor. This could be a simple scalar factor, or it could be a non-linear function. We will reference this conversion factor as $A_{LQI}(x)$, which is a function of LQI and returns an estimated relative distance. Conversely, we would also like to know the inverse of this function $A^{-1}(d)$, which is a function of distance in order to determine an estimated LQI. This allows us to determine an error between actual results and estimated results, and it is also integral to any location discovery algorithm.

The next step in solving this problem is to identify several algorithms to attempt to discover the locations of several nodes. Of course, in order to get an accurate assessment of the algorithm's validity, many various scenarios need to be considered -- changing parameters such as node count, node density, etc. Each of the algorithms center around matching an estimated distance from LQI data collected with known distances from the floor layout. In order to assist in calculating actual distances and estimating distances, MatLab provides sufficient calculation capabilities as well as simulation and repeatability capabilities.

3.4 Other Considerations

There are other factors in these scenarios that need to be considered in order to validate the design. In the experiments, we investigated multiple environmental scenarios, but even beyond that the immediate local environment can potentially create some significant alterations in the measured signal strength. Examples

would include nodes in an office cubicle where one node is in the open on a table and another node is in a corner underneath metal cabinetry. Though the global environment may be the same, the local environment can potentially cause significant variations from expected values. Other potential obstructions include physical walls (including different types of walls – i.e., dry wall versus a firewall), kitchen equipment, etc. Perhaps, the biggest environmental concern would be other wireless traffic, specifically Wi-Fi. Since one of the target scenarios is an office space, this becomes an important factor to consider in the design [12] [17] [18] [23].

3.5 Algorithms

The first step in each of the algorithms to follow is to set up a group of base points which serve as reference pionts. For most of the algorithms, one technically only needs one base point, but to gain sufficient accuracy one likely needs at least two or three base points. The only algorithm that requires more than one node is the triangulation method, which requires at least three base points. A base point can either be a unique coordinator node placed at each base point or a single coordinator node moved to each of the base points. At this point, the base point(s) send out a broadcast request for each node in its single-hop range to begin sending packets back to the base point. Obviously, one does not want to have every node immediately begin to send packets back, because that would flood the network. Therefore, one needs some sort of scheme to ensure that this does not occur. This could be as simple as keeping a record of

its neighbors and sending a binding request to each individual node or having each node own a time slot based on a random number generator seeded on serial number. The scope of this thesis does not explore various manners to accomplish this task and only assumes that it is easily doable. When a node does send a pre-determined number of packets (how this number is determined is explored later in this chapter), the average LQI is calculated for all of the packets. This average LQI is converted to a distance (how this is done is also be explored later). Once this is accomplished for all of the nodes, the coordinator now has a distance estimation for each node. This process is then completed for each of the base points. The data from each base point is then aggregated, and the discovery algorithms take over from that point.

3.5.1 Perceived Confidence without Contention

The perceived confidence algorithm is a method which uses a probability matrix of *confidence factors* for each node determining which node it thinks it most likely is based on its estimated distance. Figure 3-1 summarizes the algorithm's steps. Once the average LQI between all the nodes and all the base points is determined, we can estimate the distance using a pre-determined conversion factor (step 8). This estimated distance for that node is then compared with the actual distances between the base point and all of the nodes. An error is then computed between those distances, converted to accuracy and then added to that node's accuracy array. We now have a list of the accuracy of the estimated distance vs. the actual distance between that node and the entire list of nodes.

Perceived Confidence w/o Contention

```
1.  For each base b_i ∈ B:
2.      For each node n_j ∈ N:
3.          Transmit x packets from n_j to b_i
4.          LQI_j = 0
5.          For each packet p_k = {p_1, p_2, … , p_x}
6.              LQI_j = LQI_j + LQI(p_k)
7.          LQI_avg = LQI_j/x
8.          d_est_ij = A_LQI(LQI_avg)
9.          For each node n´_1 ∈ N:
10.             d_act_ij´ = ((c_i(0) - c_j´(0))^2+(c_i(1) – c_j´(1))^2)^½
11.             d_err_ij´ = abs(d_act_jj´ - d_est_ij)/d_act_jj´
12.             d_acc_i[j´] =  1-d_err_ij´
13.         d_acc_ij = MaxSort(d_acc_j)
14. For each node n_j ∈ N:
15.     For each node n_k ∈ N:
16.         Confidence_j[k] = 0
17.         For each base b_i ∈ B:
18.             Confidence_j[k] = Confidence_j[k] + d_acc_ij[k]
19.     Confidence_j = MaxSort(Confidence_j)
20.     For each node n_k ∈ Confidence_i
21.         If not assigned_i[k]
22.             Assign n_j as n_k
23.             assigned[k] = true
24.             Break
```

Fig 3-1: Perceived Confidence w/o Contention Algorithm Pseudo-Code

We can then sort that list to create an order of likelihood of which node should be assigned to the actual node according to that base point.

Once this is completed for each of the base points, we then compute what is referred to as a confidence array. This confidence array is a set of confidence factors between that node and all of the other nodes. The confidence factor is calculated by adding the accuracies between the two nodes from each base point. Since the maximum accuracy between a single base point and a node is 1 (or 100%), the confidence factor can range anywhere from 0 to the number of

base points (e.g. – if there are three base points, the confidence factors would range from 0 to 3). If we sort this new list, we now have an order of likelihood for which node that actual node should be assigned as based on the combined confidence of all of the base points. Since there is no contention in this algorithm, we simply go through the list of nodes in order and assign that node to the node with the highest confidence that has not been used.

3.5.2 Perceived Confidence with Contention

This algorithm, as summarized in Figure 3-2, is no different than the previous except that we now introduce the concept of contention. A node is considered to be in "contention" if two or more nodes both rate that node as their highest confidence such that both would want to assign that node identity as their own. For example, let's suppose we have a small network of 5 nodes and 5 base points with the following confidence arrays:

$Confidence_1$ = [4.65 4.32 4.05 3.56 3.10]

$Confidence_2$ = [2.95 4.65 4.70 3.35 3.32]

$Confidence_3$ = [4.45 4.10 4.75 2.45 3.98]

$Confidence_4$ = [3.12 4.05 2.45 4.96 3.34]

$Confidence_5$ = [3.40 2.90 3.45 4.78 4.77]

In this scenario, node 1 would first be assigned to corresponding node 1 (since element 1 has the highest confidence in the array). But then both node 2 and node 3 have node 3 as their highest confidence factor. Without contention, node 2 would be given the priority just because it was the first node assigned. With

Perceived Confidence w/ Contention

```
1.  For each base b_i ∈ B:
2.      For each node n_j ∈ N:
3.          Transmit x packets from n_j to b_i
4.          LQI_j = 0
5.          For each packet p_k = {p_1, p_2, … , p_x}
6.              LQI_j = LQI_j + LQI(p_k)
7.          LQI_avg = LQI_j/x
8.          d_est_ij = A_LQI(LQI_avg)
9.          For each node n´_1 ∈ N:
10.             d_act_ij´ = ((c_i(0) - c_j´(0))^2+(c_i(1) – c_j´(1))^2)^½
11.             d_err_ij´ = abs(d_act_jj´ - d_est_ij)/d_act_jj´
12.             d_acc_i[j´] =  1-d_err_ij´
13.         d_acc_ij = MaxSort(d_acc_j)
14. While there exists a node unassigned:
15.     For each unassigned node n_j ∈ N:
16.         For each node n_k ∈ N:
17.             Confidence_j[k] = 0
18.             For each base b_i ∈ B:
19.                 Confidence_j[k] = Confidence_j[k] + d_acc_ij[k]
20.         Confidence_j = MaxSort(Confidence_j)
21.         For each node n_k ∈ Confidence_i:
22.             If assigned[k] = 0:
23.                 Assign n_j as n_k
24.                 assigned[k] = j
25.                 Break
26.             Else:
27.                 n_old = assigned_j[k]
28.                 If  Confidence_i[k] > Confidence_old[k]:
29.                     assigned[k] = j
30.                     Unassign n_old
31.                     Assign n_j as n_k
32.                     Break
33.                 Else:
34.                     k++
```

Fig 3-2: Perceived Confidence w/ Contention Algorithm Pseudo-Code

contention, node 2 will originally be assigned node 3, but when node 3 contends for node 3, the two nodes will compare which one has the higher confidence. In this case, node 3's confidence (4.75) is higher than node 2's confidence (4.70). Therefore, node 3 will correctly be assigned to node 3, and node 2 will be unassigned. Nodes 4 and 5 will then contend for node 4 in a familiar fashion, but since node 4's confidence is higher (4.96) than node 5's (4.78), it will retain the assignment. Node 5 will then go to its next highest node, which is node 5. Once this is complete, the process will reiterate for all nodes left unassigned. In this case that would only be node 2. Node 2 would then assign its next lowest confidence rating, which is correctly node 2. In this basic example, we end with 100% accuracy when using contention. Without contention the accuracy would have reduced to 80%.

3.5.3 Relative Confidence without Contention(Fig. 3-3)

The relative confidence algorithm is very similar to the perceived confidence algorithm. The only real difference between the two is that instead of using the total confidence for each node, we calculate a relative confidence based on the node's confidence with the rest of the network. For example, suppose we had two confidence arrays that looked like this:

$Confidence_1$ = [4.70 4.05 2.95 3.20 2.50]

$Confidence_2$ = [4.75 4.68 4.05 3.50 2.70]

In this scenario, both nodes have contention for node 1. The perceived confidence algorithms listed above would assign these nodes incorrectly whether

Relative Confidence w/o Contention

```
1.  For each base b_i ∈ B:
2.      For each node n_j ∈ N:
3.          Transmit x packets from n_j to b_i
4.          LQI_j = 0
5.          For each packet p_k = {p_1, p_2, … , p_x}
6.              LQI_j = LQI_j + LQI(p_k)
7.          LQI_avg = LQI_j/x
8.          d_est_ij = A_LQI(LQI_avg)
9.          For each node n´_1 ∈ N:
10.             d_act_ij´ = ((c_i(0) - c_j´(0))^2+(c_i(1) – c_j´(1))^2)^½
11.             d_err_ij´ = abs(d_act_jj´ - d_est_ij)/d_act_jj´
12.             d_acc_i[j´] =  1-d_err_ij´
13.         d_acc_ij = MaxSort(d_acc_j)
14. For each node n_j ∈ N:
15.     For each node n_k ∈ N:
16.         Confidence_j[k] = 0
17.         For each base b_i ∈ B:
18.             Confidence_j[k] = Confidence_j[k] + d_acc_ij[k]
19.     TotalConfidence_j = ∑ Confidence_j
20.     For each node n_k ∈ N:
21.         Confidence_rel[k] = Confidence_j[k] / TotalConfidence_j
22.     Confidence_rel = MaxSort(Confidence_rel)
23.     For each node n_k ∈ Confidence_rel
24.         If not assigned_i[k]
25.             Assign n_j as n_k
26.             assigned[k] = true
27.             Break
```

Fig 3-3: Relative Confidence w/o Contention Algorithm Pseudo-Code

contention methods were used or not. However, if one examines the confidence arrays a little bit closer, one will notice that node 1's confidence in node 1 is very high compared to the rest of the nodes in its array; whereas, node 2's confidence in node 1 is not as much higher than the other nodes in its confidence array. If one instead calculates a relative confidence based on the confidence of each node as a percentage of the total confidence for that inquiring node, then one gets a better idea of just how confident that node is that it is the correct node as compared to the entire network. By including this extra information one should hopefully get more accurate results. The two new arrays would look like this:

$Confidence_1$ = [0.270 0.233 0.170 0.184 0.144]

$Confidence_2$ = [0.241 0.238 0.206 0.178 0.137]

In this scenario, now looking at the relative confidences, the two nodes will still contend for node 1, but node 1 will correctly be assigned to node 1, and node 2 will correctly be assigned to node 2.

3.5.4 Relative Confidence with Contention

The relative confidence algorithm can add a contention element just like the perceived confidence algorithm. The only difference is that it uses the relative confidence as described in the previous section instead of the perceived confidence. Figure 3-4 shows a detail of how the algorithm should operate.

Relative Confidence w/ Contention

```
1.  For each base b_i ∈ B:
2.      For each node n_j ∈ N:
3.          Transmit x packets from n_j to b_i
4.          LQI_j = 0
5.          For each packet p_k = {p_1, p_2, … , p_x}
6.              LQI_j = LQI_j + LQI(p_k)
7.          LQI_avg = LQI_j/x
8.          d_est_ij = A_LQI(LQI_avg)
9.          For each node n´_1 ∈ N:
10.             d_act_ij´ = ((c_i(0) - c_j´(0))^2+(c_i(1) – c_j´(1))^2)^½
11.             d_err_ij´ = abs(d_act_jj´ - d_est_ij)/d_act_jj´
12.             d_acc_i[j´] =  1-d_err_ij´
13.         d_acc_ij = MaxSort(d_acc_j)
14. While there exists a node unassigned:
15.     For each unassigned node n_j ∈ N:
16.         For each node n_k ∈ N:
17.             Confidence_j[k] = 0
18.             For each base b_i ∈ B:
19.                 Confidence_j[k] = Confidence_j[k] + d_acc_ij[k]
19.         TotalConfidence_j = ∑ Confidence_j
20.         For each node n_k ∈ N:
21.             Confidence_rel[k] = Confidence_j[k] / TotalConfidence_j
22.         Confidence_rel = MaxSort(Confidence_rel)
23.         For each node n_k ∈ Confidence_rel:
24.             If assigned[k] = 0:
25.                 Assign n_j as n_k
26.                 assigned[k] = j
27.                 Break
28.             Else:
29.                 n_old = assigned_j[k]
30.                 If  Confidence_rel[k] > Confidence_old[k]:
31.                     assigned[k] = j
32.                     Unassign n_old
33.                     Assign n_j as n_k
34.                     Break
35.                 Else:
36.                     k++
```

Fig 3-4: Relative Confidence w/ Contention Algorithm Pseudo-Code

3.5.5 Minimum Distance Error without Contention

The minimum distance error (or MDE) algorithm, which is described in Figure 3-5, is actually a simpler, slightly more intuitive approach to localization. It eliminates this notion of confidence, and instead it uses the raw distance error taking the minimum error. One key difference, though, is that this error is not taken as relative to the actual distance. It is purely an absolute error, taken as the difference in the distances. This algorithm should be very close to an actual triangulation method. By comparison, though, a triangulation method for this type of application is going to be very complex and memory-heavy. This approach, on the other hand, is about as simple of an approach as there is.

Minimum Distance Error w/o Contention

```
1.  For each base b_i ∈ B:
2.      For each node n_j ∈ N:
3.          Transmit x packets from n_j to b_i
4.          LQI_j = 0
5.          For each packet p_k = {p_1, p_2, … , p_x}
6.              LQI_j = LQI_j + LQI(p_k)
7.          LQI_avg = LQI_j/x
8.          d_est_ij = A_LQI(LQI_avg)
9.          For each node n´_1 ∈ N:
10.             d_act_ij´ = ((c_i(0) - c_j´(0))^2+(c_i(1) – c_j´(1))^2)^½
11.             d_err_ij´ = abs(d_act_jj´ - d_est_ij)
13.         d_err_ij = MinSort(d_err_ij)
14. For each node n_j ∈ N:
15.     For each node n_k ∈ N:
16.         Error_j[k] = 0
17.         For each base b_i ∈ B:
18.             Error_j[k] = Error_j[k] + d_err_ij[k]
19.     Error_j = MaxSort(Error_j)
20.     For each node n_k ∈ Error_i
21.         If not assigned_i[k]
22.             Assign n_j as n_k
23.             assigned[k] = true
24.             Break
```

Fig 3-5: Minimum Distance Error Algorithm Pseudo-Code

3.5.6 Minimum Distance Error with Contention

The minimum distance error with contention algorithm is shown in Figure 3-6. The addition of node contention is the same as in the previous algorithms. This could be extended as a relative error much like the relative confidence algorithm very easily. Should this algorithm along with the other relative algorithm both show promise, it would be an easy extension for future work.

3.5.7 Radial Triangulation without Contention

The radial triangulation algorithm, as seen in Figure 3-7, is certainly the most complex of the algorithms. Because of this, it is almost impossible to consider any contention algorithms. Because of the high error rates expected, true triangulation is not achievable. It is very unlikely that an actual (or even very close) triangular intersection can be achieved; therefore, instead of looking at line equations, this algorithm uses a probability array for the entire environment. The array is initialized as all zeros. When an estimated distance is calculated from a base point, a circle with a radius equal to the estimated distance is drawn onto the array adding a normalized value of 1. Then echo circles are drawn to take error into account. The echo circles increase in radius and decrement the added value exponentially until the value reaches zero. It also creates echoes by decreasing the radius and decrementing the value accordingly until either the value is zero or the radius is zero. Once this is completed, the algorithm moves onto the next base point and add its circle on top of the previous circle. This process is repeated until all base points are covered. At this point the array is a

Minimum Distance Error w/ Contention

```
1.  For each base b_i ∈ B:
2.      For each node n_j ∈ N:
3.          Transmit x packets from n_j to b_i
4.          LQI_j = 0
5.          For each packet p_k = {p_1, p_2, … , p_x}
6.              LQI_j = LQI_j + LQI(p_k)
7.          LQI_avg = LQI_j/x
8.          d_est_ij = A_LQI(LQI_avg)
9.          For each node n´_1 ∈ N:
10.             d_act_ij´ = ((c_i(0) - c_j´(0))^2+(c_i(1) – c_j´(1))^2)^½
11.             d_err_ij´ = abs(d_act_jj´ - d_est_ij)
13.         d_err_ij = MinSort(d_err_j)
14. While there exists a node unassigned:
15.     For each unassigned node n_j ∈ N:
16.         For each node n_k ∈ N:
17.             Error_j[k] = 0
18.             For each base b_i ∈ B:
19.                 Error_j[k] = Error_j[k] + d_err_ij[k]
20.         Error_j = MinSort(Error_j)
21.         For each node n_k ∈ Error_i:
22.             If assigned[k] = 0:
23.                 Assign n_j as n_k
24.                 assigned[k] = j
25.                 Break
26.             Else:
27.                 n_old = assigned_j[k]
28.                 If  Error_i[k] < Error_old[k]:
29.                     assigned[k] = j
30.                     Unassign n_old
31.                     Assign n_j as n_k
32.                     Break
33.                 Else:
34.                     k++
```

Fig 3-6: Minimum Distance Error w/ Contention Algorithm Pseudo-Code

Radial Triangulation w/o Contention

```
1.  For each node n_m ∈ N:
2.      2-D Array A = [[0…0]…[0…0]]
3.      For each base b_i ∈ B:
4.          For each node n_j ∈ N:
5.              Transmit x packets from n_j to b_i
6.              LQI_j = 0
7.              For each packet p_k = {p_1, p_2, … , p_x}
8.                  LQI_j = LQI_j + LQI(p_k)
9.              LQI_avg = LQI_j/x
10.             d_est_ij = A_LQI(LQI_avg)
11.             A = A + 1*Circle(d_est_ij)
12.             Factor = 0.99
13.             Iter = 1
14.             While Factor >= 0
15.                 A = A + Factor*Circle(d_est_ij + Iter*α)
16.                 Factor = Factor – Iter^2
17.                 Iter++
18.             Iter = 1
19.             While Factor >= 0 and (d_est_ij – Iter* α) >= 0
20.                 A = A + Factor*Circle(dest_ij – Iter*α)
21.                 Factor = Factor – Iter^2
22.                 Iter++
23.     Max = 0
24.     For each point p_ij ∈ A:
25.         If A[i][j] > Max
26.             Max = A[i][j]
27.             EstLocation = (i,j)
28.     For each node n_k ∈ N:
29.         Distance[k] = CalculateDistance(EstLocation,n_k)
30.     Distance = MinSort(Distance)
31.     k = 0
32.     While not Assigned[k]
33.         k++
34.     Assign n_m as n_k
```

Fig 3-7: Radial Triangulation w/o Contention Algorithm Pseudo-Code

summation of all of the radial circles of the base points. If one takes the highest element in the array, that point would be the closest estimation of an intersection. The primary difficulty in this algorithm lies in if there are multiple points with equally high probability factors. How to decide which is the most appropriate is a difficult task. Often times the highest rated points could be the original circles drawn as the circles are so far apart that no summed element goes over 1. If we assume we can ascertain a single point, potentially just having to take the first point found at the highest level or taking the average of all of the coordinates, from that point on the algorithm looks like a simple minimum distance algorithm. This point is used as the estimated location of the node. The error calculation, now, though, is the distance between the estimated location and the actual locations of the nodes. The node with the minimum distance to the estimated location is then identified as that node. If a successful implementation of this algorithm is possible, then contention can easily be added.

3.6 Expectations and Hypothesis

Each of the algorithms presented have some advantages and disadvantages over one another. The expectation is that the relative confidence should give a slightly higher result than the perceived confidence. The minimum distance and triangulation methods have potential to be extremely accurate. The triangulation method seems to have the most potential, but the complexity of the algorithm really detracts from its capability, especially for a real-world application.

3.7 Chapter Summary

This chapter has investigated the design approach that is used and implemented in the experimentation and simulation stages. Several potential algorithms have been identified, and they are built from scratch. All of the algorithms were created without any adaptation from another method. The experiments are designed to show if these algorithms are adequate or not.

Chapter 4

Experiments

4.1 Equipment and Resources

In total, there are five separate wireless nodes available for use in the experiments. There are two Chipcon CC2420DBK development boards [27], one of which is seen in Figure 4-1. These are the primary boards used for experimentations. In addition there are two MaxStream XBee development boards [25], seen in Figure 4-2, and there is also one Eaton Red Board. The primary boards used for these experiments is the Chipcon boards. The XBee radios are significantly cheaper and offer a very nice range test program that

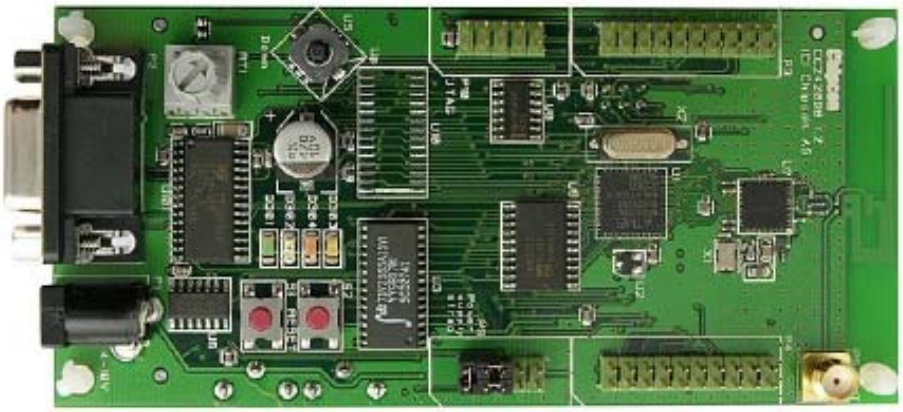

Fig 4-1: Chipcon's CC2420DBK development board [27]



Fig 4-2: MaxStream XBee Development Kit [25]

displays the RSSI value for every packet received as well as show the packet error rate. This range test program also allows one to specify how often one wants a message to be repeated, when one wants the radios to stop transmitting, what amount of data to be sent, etc. The XBee radios also come with a programming tool that allows one to change the device modem properties such as channel and address information. However, if one wants to program the boards with a custom stack implementation, it is not very straightforward. The Chipcon CC2420DB's, though, come with a version of Chipcon's Z-Stack [27]. The XBee modules claim to be compatible with Z-Stack. The Eaton Red Board is a research prototype that was built in house, and it basically serves as a reduced functionality CC2420DB running the same stack protocol.

Though the IEEE 802.15.4 standard allows for signal strength information to be

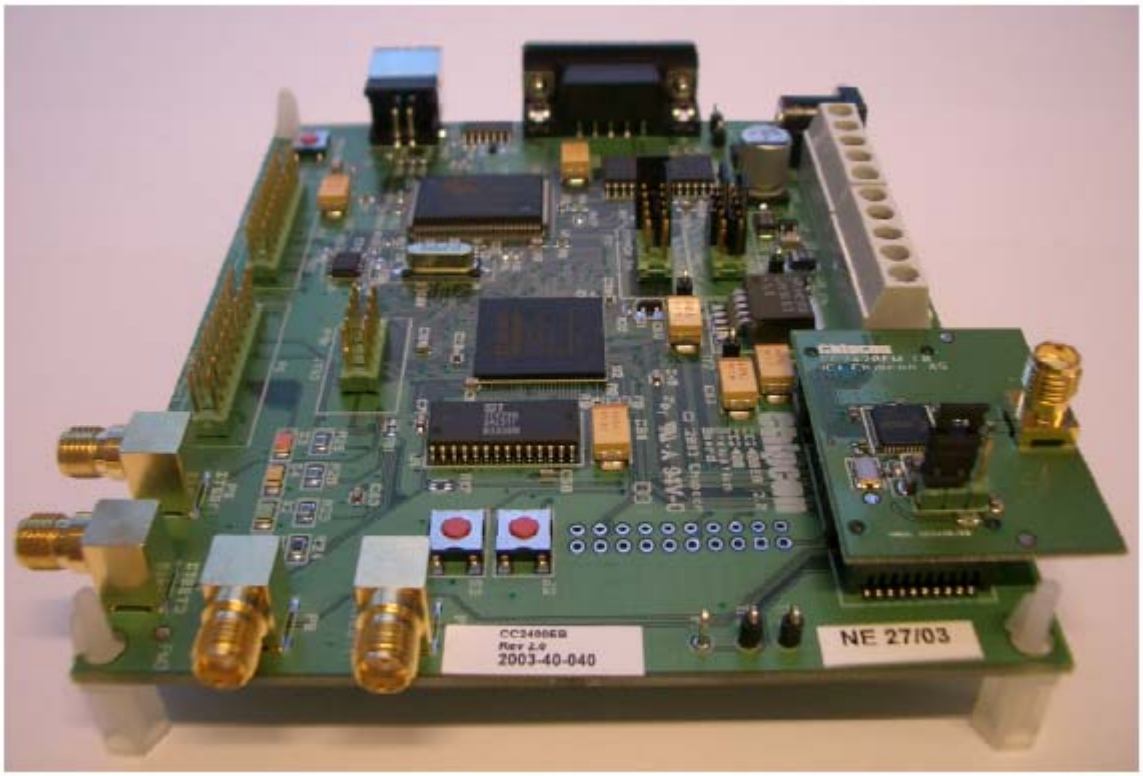

Fig 4-3: Chipcon CC2420DK Packet Sniffer

sent all the way up to the application layer [28], for these experiments the Chipcon CC2420DK Packet Sniffer, as seen in Figure 4-3, is used for most signal strength measurements done in the experiments. The packet sniffer comes with a very convenient software package that shows each packet separated by layers and data types. Figure 4-4 shows a screen shot of the software package. The sequence of messages seen is a single device emitting a series of beacon messages. The second column from the end displays the Link Quality Indicator, which allows for easy data retrieval for experiments. One can also choose to display this value in RSSI.

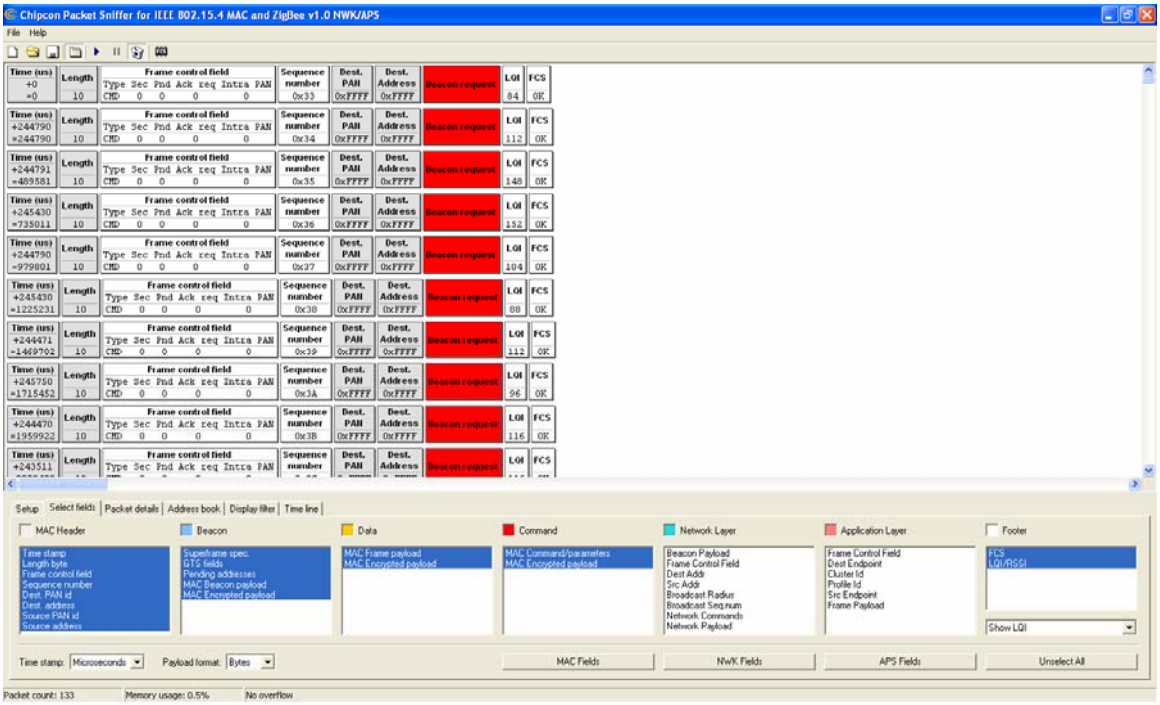

Fig 4-4: Screenshot of Chipcon Packet Sniffer Software

In addition to these ZigBee boards and software packages, MatLab is utilized for the purposes of simulations, graphing, and visualizing the spatial node density, etc.

4.2 Approach

4.2.1 Data Collection and Modeling

As mentioned in the previous section, the Chipcon boards are the primary boards used along with the sniffer for most of the data collection. In most cases, a single board is setup to repeatedly transmit a broadcast beacon. The data size being transmitted is minimal. Generally, the sniffer is statically placed and simulates a base point or reference point for the algorithms. The board itself represents a node in the network. Since the algorithms require that only one node be

transmitting at a time, the board can simply be moved around to simulate multiple nodes in a network.

Before any network simulations can be done, though, we first must model the environments. In order to model distance with LQI, two basic approaches were taken. First the node is held directly next to the base point (the sniffer), and once it starts transmitting, the node is carried away from the base point at a steady state in a straight line over a known distance. If an even interpolation of the data is assumed, a direct model of the LQI and distance can be determined from this data. A second method would be to mark off specific distances in a straight line from the base point, and let the node sit at each point transmitting data over time so that an average can be determined at each distance. This method will provide less data, but it should be more precise.

4.2.2 Interference Considerations

One of the most important considerations for this application is going to be interference [12] [17] [18]. Since building automation environments range widely, it is important to consider interference factors. Sources of interference could range from high Wi-Fi activity, large metals, microwaves or other devices that fall on the 2.4 GHz band. In addition, physical obstructions must be considered. For example, in a lighting control application, if all of the nodes are in the fixtures, then the obstructions are uniform. However, if one starts to put nodes in a cubicle in an office environment, then things such as metal cabinets or a wireless

laptop augment the environmental interference. Since the experiments in this thesis primarily revolve around an office environment, these type scenarios need to be considered.

4.2.3 Node Count and Node Density Considerations

The next factor that is a critical consideration for experiments is the network node count and node density [14]. It is important to discover how the network and the discovery algorithms will behave as node count changes and as node density changes. Node density is especially important for the applications that I am trying to emulate. Algorithms exist already that can refine localization to a 5 meter range, but the hopes of these experiments are to find an algorithm that can get down to about a 2 meter range. Changing the node count in the experiments also helps one determine the scalable complexity of the algorithms.

4.3 Environmental Scenarios

There are three main environmental scenarios that are available to explore and that are explored in these experiments. They are: a straight line-of-sight (LOS) path with limited interference, a straight LOS with typical office interference, and then multiple LOS's in the same typical office environment.

4.3.1 Line of Sight with Limited Interference

In the office where the experiments were done, there are two areas of distinct difference in interference. The back warehouse is designated as an area that

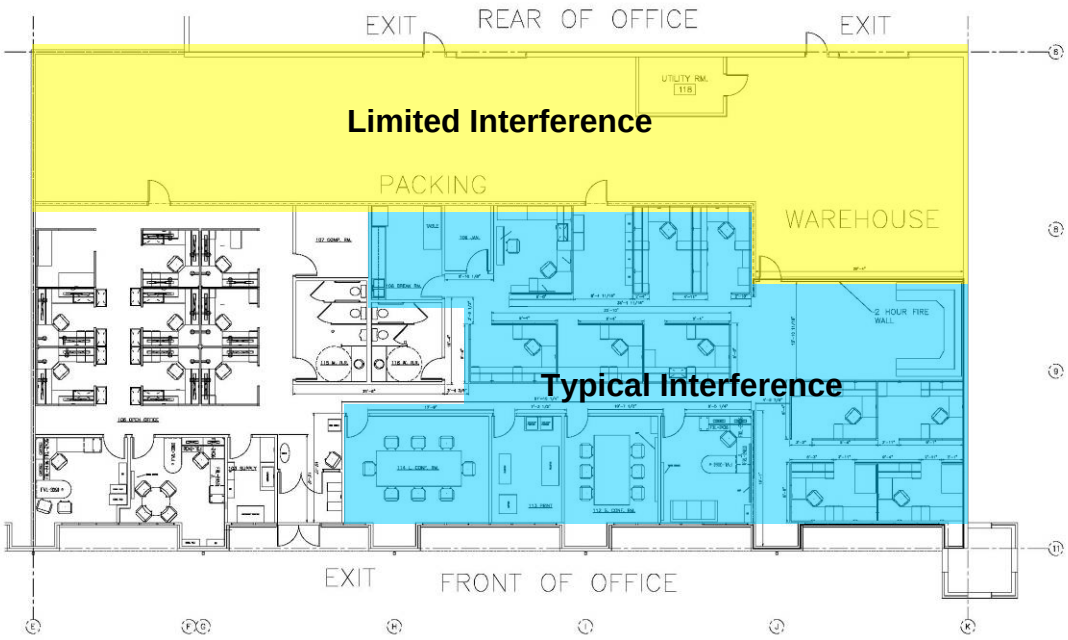

Fig 4-5: Office Layout where experiments take place

has limited interference, which is the yellow region in Figure 4-5. Of course being in an office building, there could still be some Wi-Fi interference in the warehouse coming from the office, but this should be considerably limited since it is behind a firewall. The other potential source of interference is the fact that there are a lot of large metal cabinets and machinery in the back along with quite a bit of electrical equipment that could potentially augment the interference. Overall, though, it can be considered a limited interference environment.

There are three primary experiments in this scenario. All three are intended to show the relationship between distance and LQI. The first experiment takes a straight line-of-sight path from 0 feet to 45 feet. The node are placed directly next to the sniffer, begin broadcasting its beacons, and then it is carried at a

steady pace to 45 feet away. The second experiment does the same thing except it is extended to the entire length of the warehouse, which is about 127 feet (or 38.7 m). The maximum range for 802.15.4 is about 30-40 meters (or 98-131 ft), so this should test the limits of the radio's range. Finally, to ensure that we are receiving a consistent signal strength (in other words, we are not receiving a peak value or a sink value because of peak interference), an average of packets sent over approximately 10 seconds (which would be about 45 packets) at incremental stages spaced 5 feet apart. This test is done only up to 45 feet, as it is merely a confirmation test. Therefore there are 10 discrete tests performed at 0, 5, 10, 15, 20, 25, 30, 35, 40, and 45 feet respectively. These values were modeled and used as the ideal case for sake of comparison with the more typical interferences that are seen in the following experiments. Figure 4-6 shows each of the three scenarios.

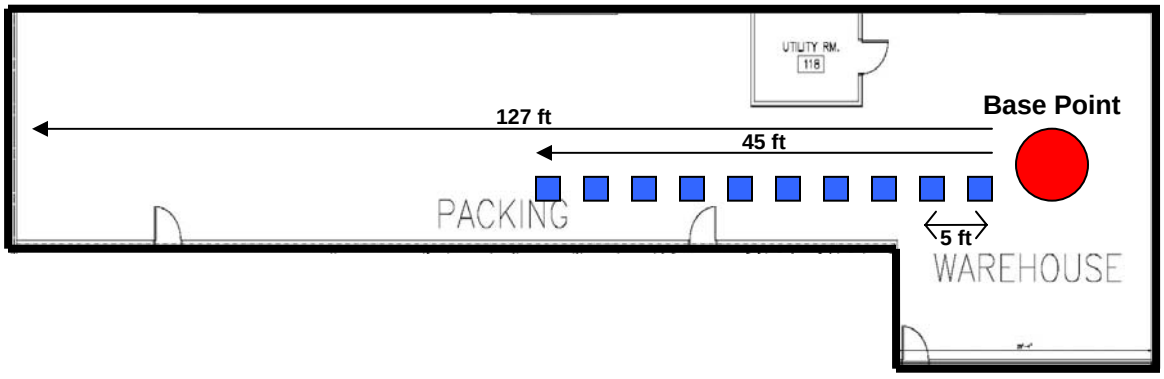

Fig 4-6: Overview of Warehouse Experiments

4.3.2 Line of Sight with Typical Interference

The next scenario involves a little bit more interference. Inside the blue region in Figure 4-7 is the office space which involves several typical interferences. The physical layout involves much more obstacles, there are various types of rooms that could involve different levels of interference such as a kitchen, meeting room, enclosed office, cubicle, lab area, etc. All of these different environments must coexist for the discovery algorithms to be successful.

For this experiment, we calculate a moving average of the LQI just like what was done in the warehouse. The node starts at 0 feet from the base point and then is carried in a straight line-of-sight away from the base point at a steady pace. The length of the line-of-sight in this scenario is 69 feet. The node starts in the lab

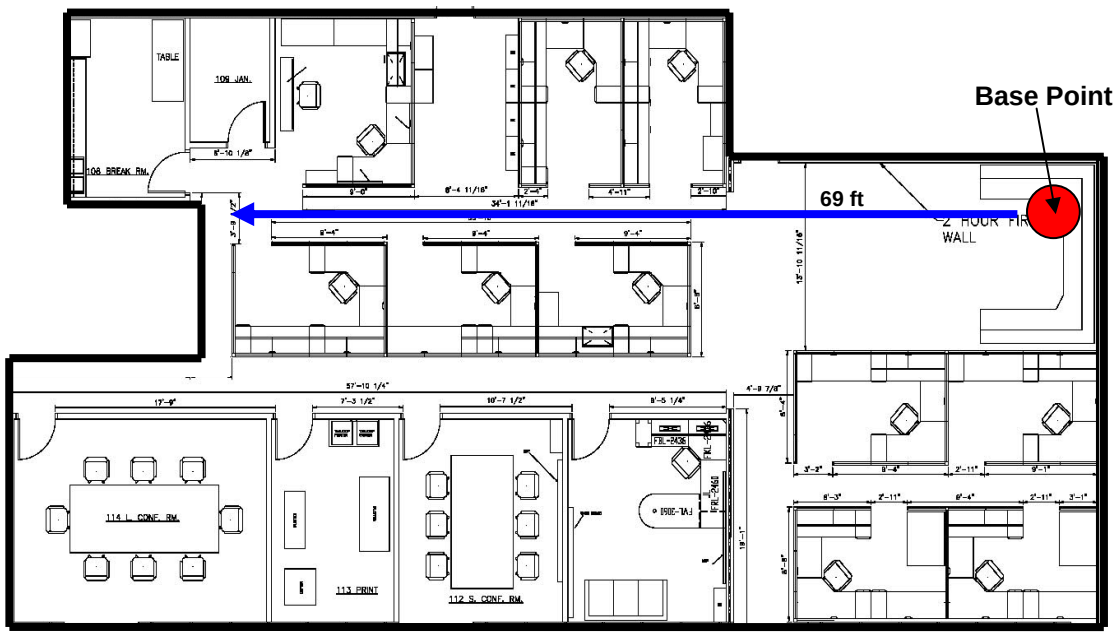

Fig 4-7: Office Space Single LOS Experiments

area, which is an open space with limited interference, then passes between several cubicles which are using Wi-Fi, and then eventually reaches a wall next to a kitchen which could provide potential interference. This experiment is performed a couple times with the Chipcon boards and then once with the MaxStream boards for sake of comparison and average. Figure 4-7 shows the positioning of the base point and the path taken to calculate the moving average.

4.3.3 Multiple LOS with Typical Interference

Once the LQI is modeled in both the limited interference and higher interference regions, we can use that to model the signal attenuation and the signal error which would allow us to simulate our algorithms. However, we would obviously like to include some real data if possible. In order to accomplish this, our node and base point were moved around the office environment to create a test case scenario. In the first experiments, the base point is placed in the lab area, and nodes are placed sporadically around the office as seen in Figure 4-8, primarily with one in each cube or room. The node is placed at each location for a few seconds to obtain enough data to get a good average of the LQI. Then the base point is moved to position B as seen on Figure 4-8. Then the node is placed at the exact same locations with the exact same orientation of the antenna (the antenna is always pointed straight up with that side of the board pointing towards the right side of the building from the overhead view). Again, several packets are sent and an average LQI is obtained from that data. Finally, a third base point is placed at position C as seen in Figure 4-8. The process is then repeated for a

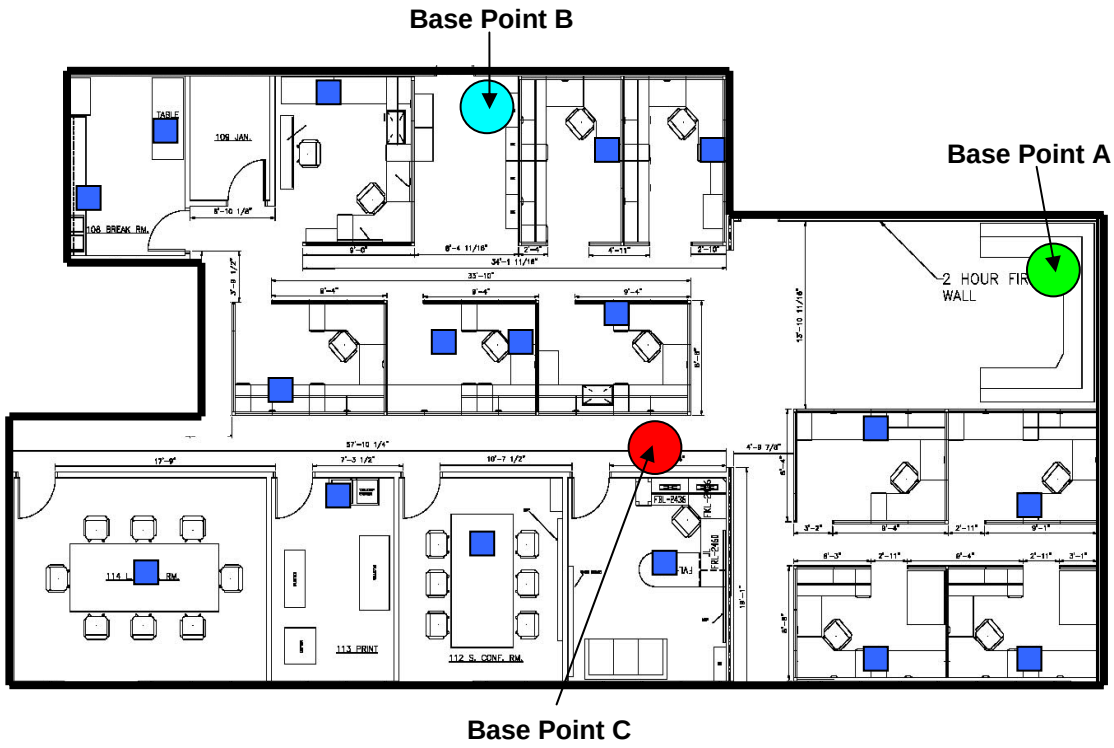

Fig 4-8: Office Space Multiple LOS Experiments

third and final time to calculate an average LQI. Once this is completed, real-time LQI measurements from a real office environment with interference are available to use in simulations.

4.4 MatLab Simulations

4.4.1 GUI Overview

The algorithms mentioned in Chapter 3 are all simulated using MatLab. MatLab provides an easy to use interface to build simulations and view the results graphically. A simple graphical user interface (GUI) is built so that the user can adjust parameters such as node count, node location, base point count, base point location, which algorithm to implement, to use real data or simulated data,

etc. This allows the experiment to be easily repeatable without having to hard-code multiple scenarios. Figure 4-9 shows a screen shot of what the GUI looks like.

4.4.2 Error Modeling

Once the actual results are modeled, a model for signal attenuation error can be inferred. The real data is investigated and compared with the eventual model to come up with some error value (+/- dBm or LQI) that can be considered a Gaussian distribution. Gaussian distributions, of course, are a natural model for any data collection and is especially natural for modeling signal noise. Gaussian distributions are characterized by a probability distribution function of the following with mean, μ, and standard deviation, σ:

$$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Eq 4-1: Gaussian Distribution

Some typical Gaussian distributions can be seen in Figure 4-10.

4.4.3 Algorithm Implementations

The code for the algorithm implementations can be found in Appendix A. Each algorithm was created and tested individually. Simulations were created on multiple levels. The first simulations look at the actual data measured in the

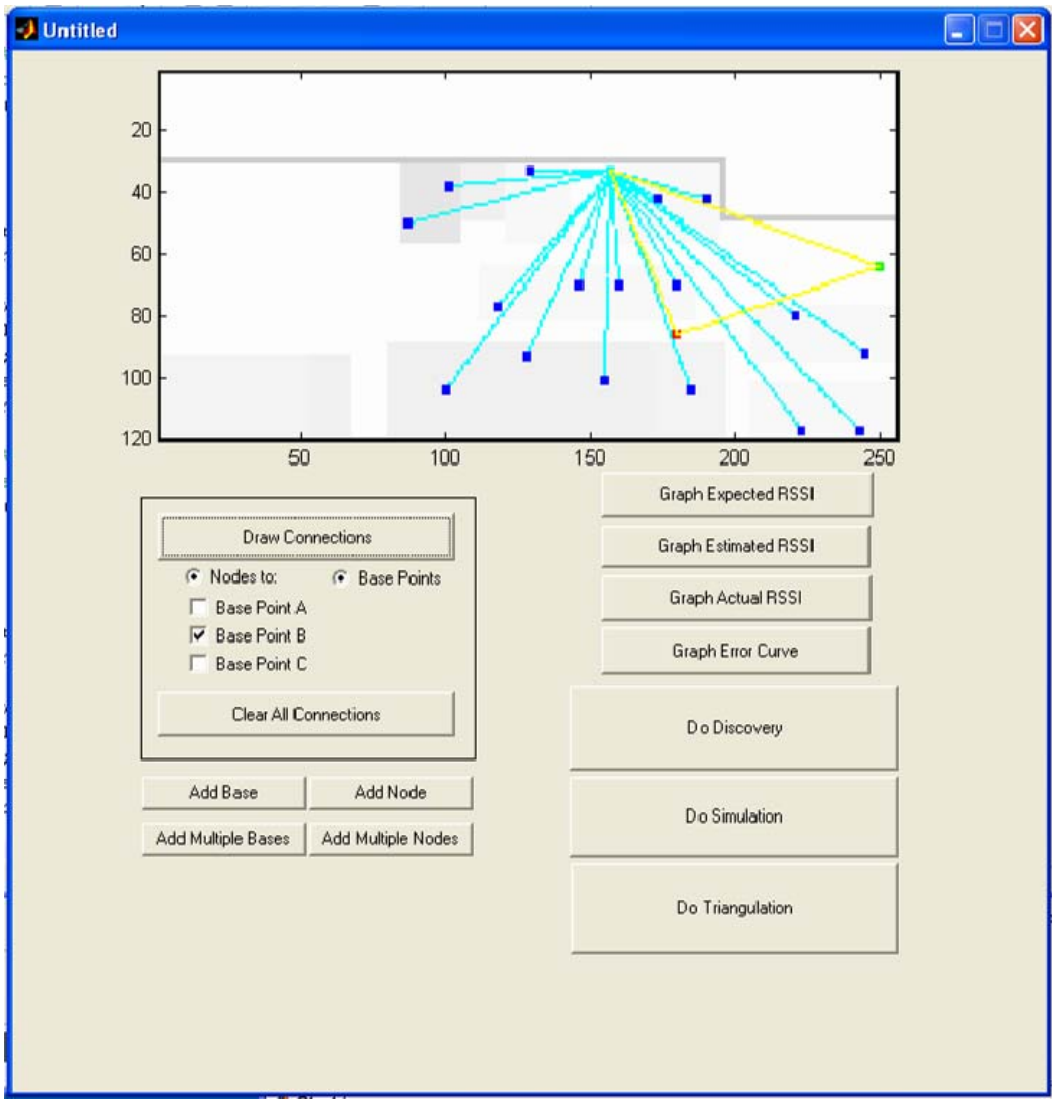

Fig 4-9: Example Screen Shot of MatLab GUI

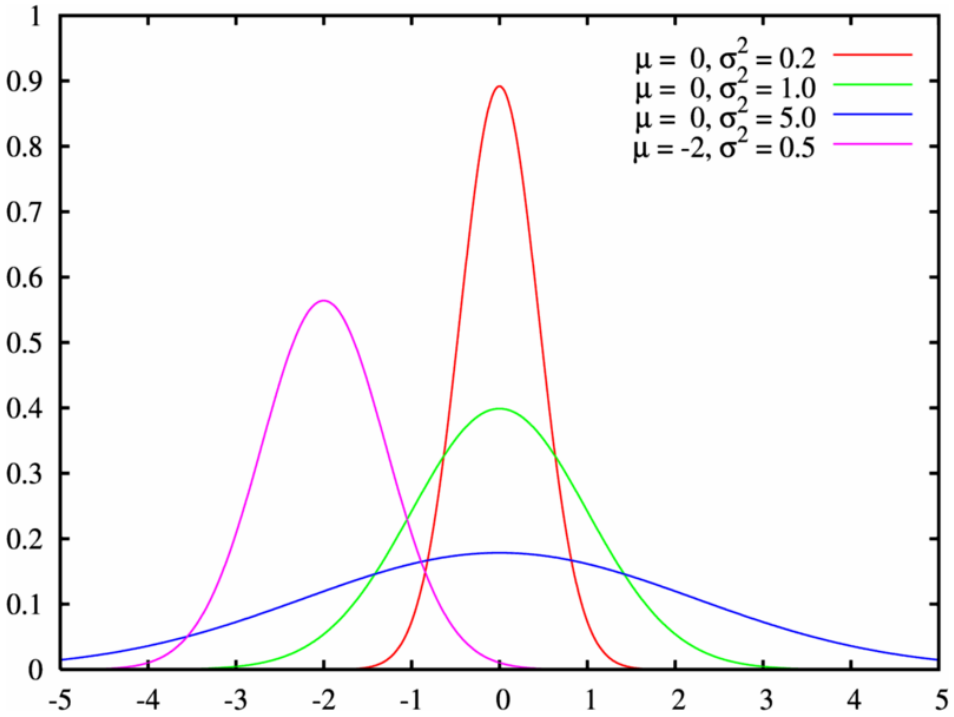

Fig 4-10: Sample Gaussian Distributions

scenarios mentioned above for each of the algorithms. The next set of simulations are performed using simulated data modeled by the perceived error obtained from the real results. Within this set of simulations, the user can modify the network layout (node count, base count, node locations, etc.) to experiment with how the network layout affects each algorithm. The next approach to simulation is to simulate each algorithm over a variety of errors. The purpose of these simulations would be to determine what error values produce viable results and which values produce unusable results. This allows us to model what signal-to-noise ratio would be necessary for the desirable accuracy. The accuracy of each algorithm is very simply measured as the number of nodes correctly identified over the total number of nodes identified over the course of the

simulation. The results of each simulation are averaged over several epochs.

4.5 Chapter Summary

In this chapter, we have explored what experiments have been done to either validate or invalidate the proposed design and algorithms. These algorithms were first tested on real data obtained from the experiments, and then were simulated over many epochs based on the error modeled in the real data. Chapter 5 covers the actual results of these experiments and simulations.

Chapter 5

Results

5.1 Experimental Results

5.1.1 Line of Sight with Limited Interference

In our first experiment, the goal is to examine an environment that has as little interference as possible. Using a warehouse area in the back of the office was the closest thing to a lossless environment available. Of course, some level of error is still expected, but the hope is that it is consistent enough to give a lossless data model.

In the first experiment, we started with the node and sniffer right next to each other (a theoretical zero distance) and then move the node away from the sniffer at a gradual pace to model the LQI over distance. This experiment was done three different times to generate an average of the data. Figure 5-1 is a graph of all three experiments and the logarithmic trend lines generated. The graph shows the received LQI versus the interpolated distance. The results are fairly consistent and follow the trend lines pretty well. All three trend lines produced are almost identical. The trend lines level off at the end, but the expectation is

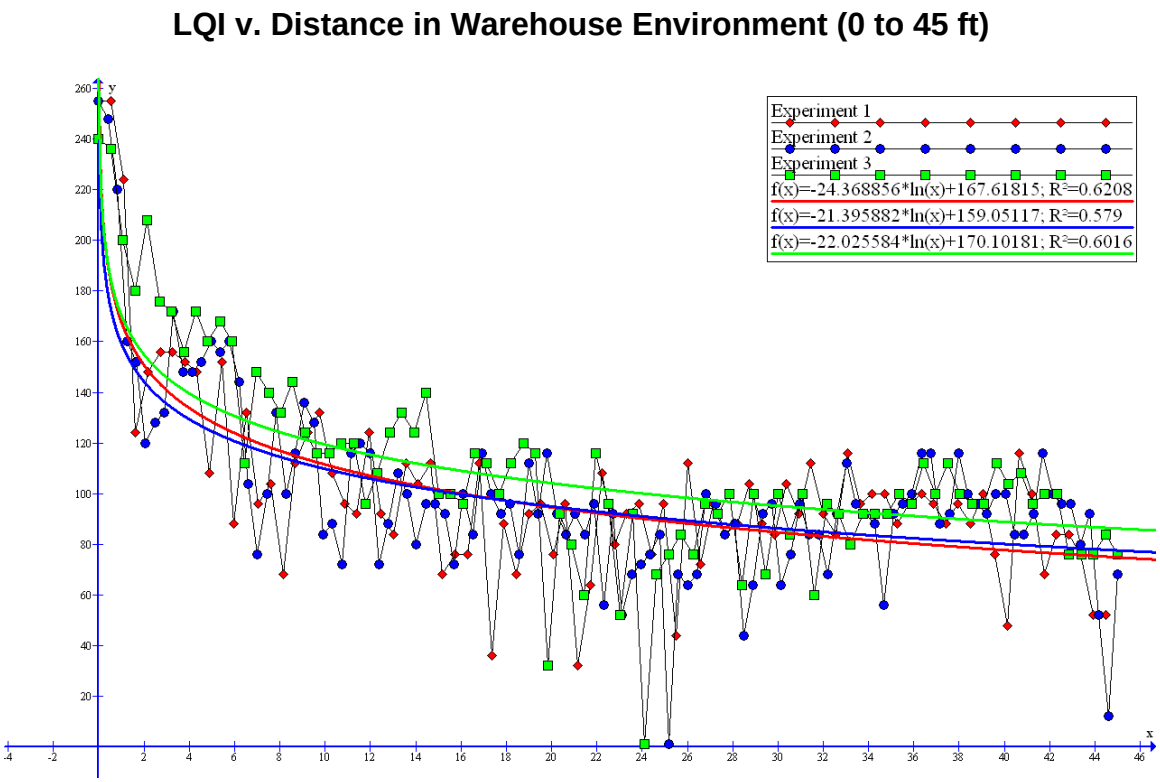

Fig 5.1: Experimental Results from Warehouse "walking" experiments ranging from 0 to 45 ft

that if the experiment were extended to longer distances, the trend line would trail off a little bit more. Equation 5-1 describes the average of the three trend lines:

$$LQI(d) = -22.5*\ln(d)+165$$

Eq 5-1: Average of the three trend lines

The next experiment involved putting the nodes at stationary positions at predefined distances while remaining in the line of sight. Figure 5-2 shows the results of this experiment. There are two trend lines, one is logarithmic and one is a power equation. In most research, the LQI versus distance models have been logarithmic, but there are occasions where other trend lines generate a

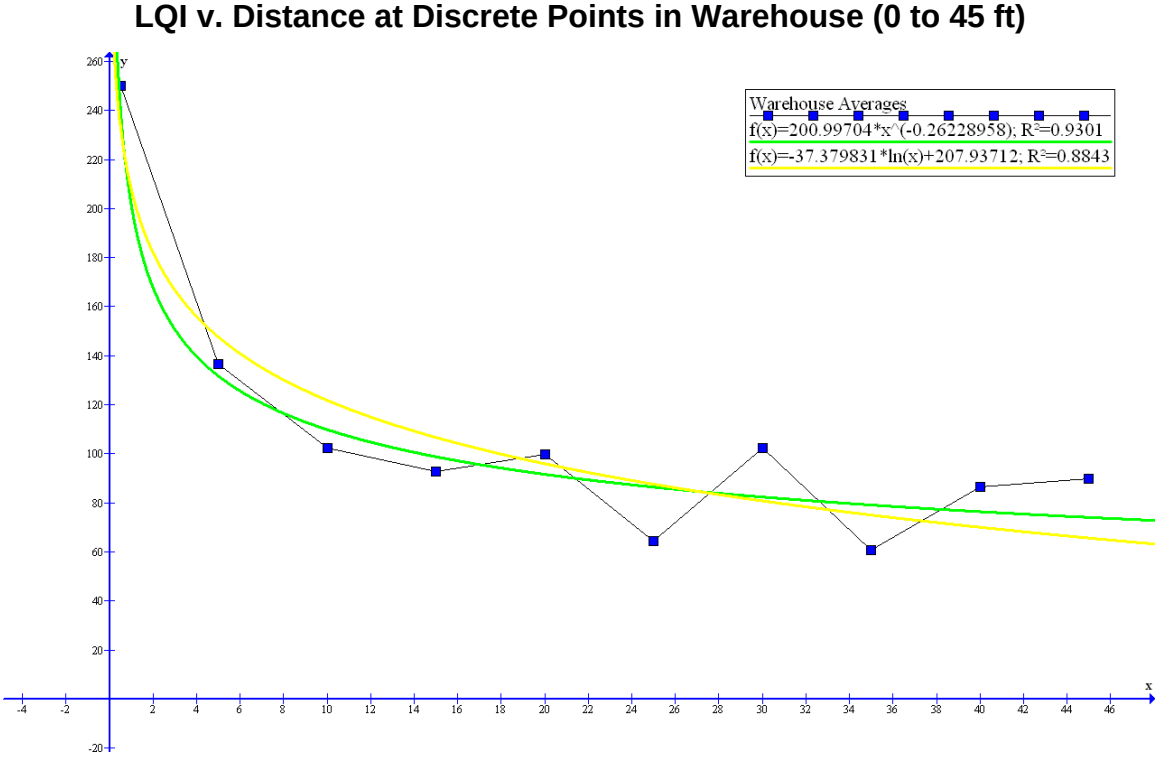

Fig 5-2: Discrete Averaging of Packet LQI

higher accuracy. This logarithmic trend line, given in equation 5-2, is slightly higher in its parameters than the previous estimation. This equation seems to make more logical sense just because it tails off more at the end. Once again, the expectation is that by adding more distance to the experiment, this would be confirmed, which is the case in later experiments.

$$LQI(d) = -37.4*\ln(d)+207.9$$

Eq 5-2: LQI as a function of distance according to the data in Figure 5-2

Figure 5-3 shows an overlapping graph of the two experiments to show that the data is truly consistent. If one looks closely, there is an evident dip in two

LQI v. Distance for All Warehouse Data (0 to 45 ft)

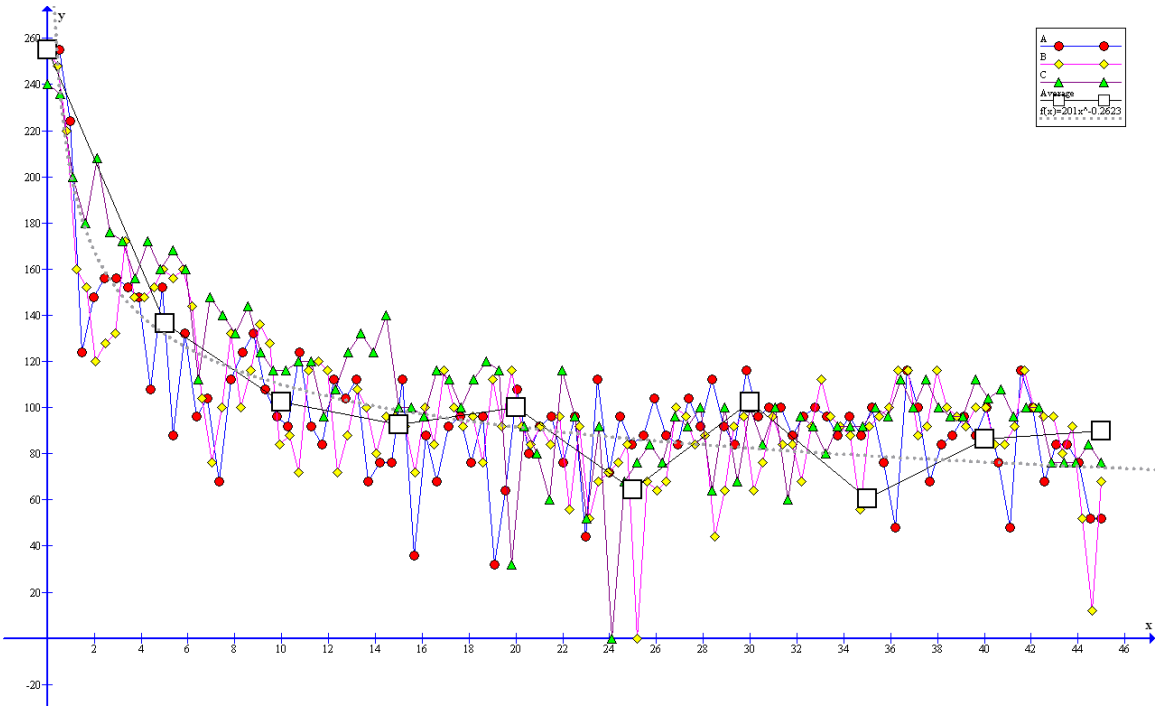

Fig 5-3: A combined graph of each of the warehouse experiments for 45 feet

locations of this graph at 25 feet and 35 feet. Upon further investigation of the environment, it was found that at almost exactly 25 feet from the sniffer, there is a set of large shelves holding large aluminum boxes. Then at almost exactly 35 feet, there is a large metal work desk right next to where the node was placed. This seems to suggest that even warehouse materials can cause a decent fluctuation in signal strength.

The final experiment in this set of lossless line-of-sight experiments in the warehouse area extends the previous experiment to 116 feet (we were unable to go the full 127 feet because of obstructions in the warehouse). Figure 5-4 below

shows the LQI over the full distance providing a much more complete model. The resulting trend line from the data (seen in Equation 5-3) seems to fall right in line with the rest of the experiments, indicating that there is some consistency in these experiments. This equation is used as the basis for a "lossless" signal. While it is not truly lossless, it contains what would seem the least possibility for interference, making it a good trend to compare against.

$$LQI(d) = -29.4*\ln(d)+154.8$$

Eq 5-3: Trend Line for data in Figure 5-4

LQI v. Distance Over Full Length of Warehouse (0 to 116 ft)

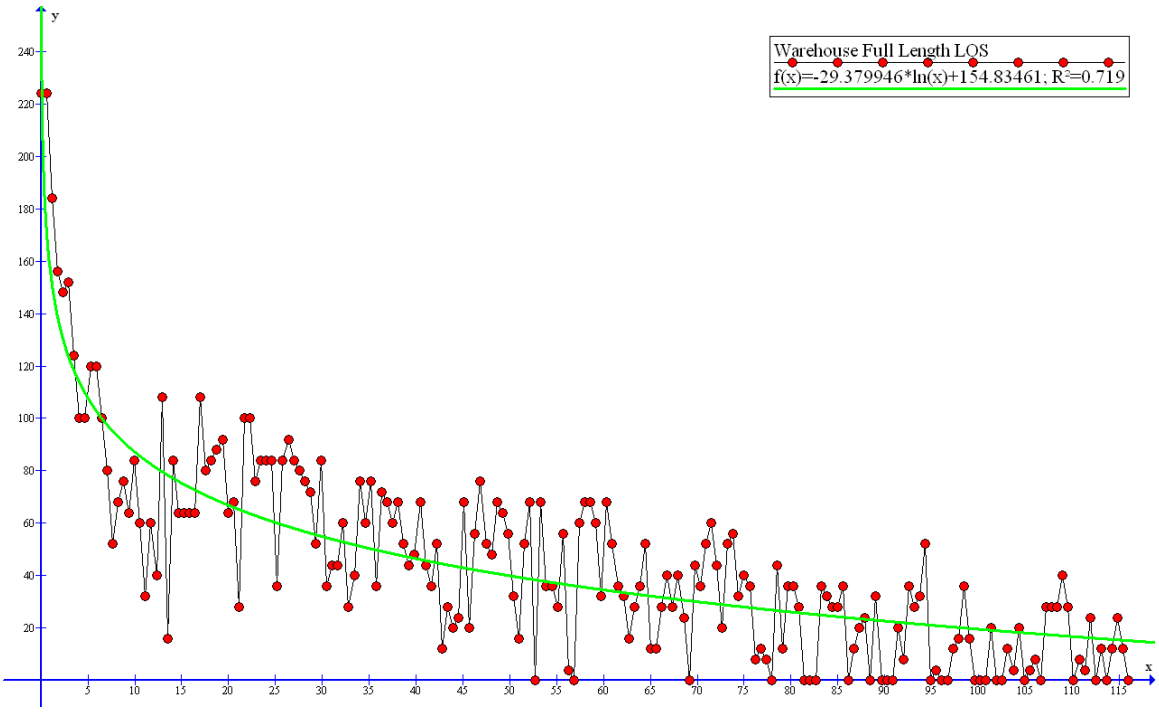

Fig 5-4: Fully modeled data in "lossless" scenario

5.1.2 Line of Sight with Typical Interference

The next set of experiments are moved out of this “lossless” environment and into a typical office environment that contains potential interference from Wi-Fi, Bluetooth, microwaves, obstructions, etc., all of which are common in the average office environment. The two line-of-sight experiments are graphed in Figure 5-5. The trending curve is similar to the “lossless” scenario except that everything is translated a little lower. This data is taken in a line-of-sight path that stretches approximately 69 feet. The average trend line between the two experiments is given in Equation 5-4.

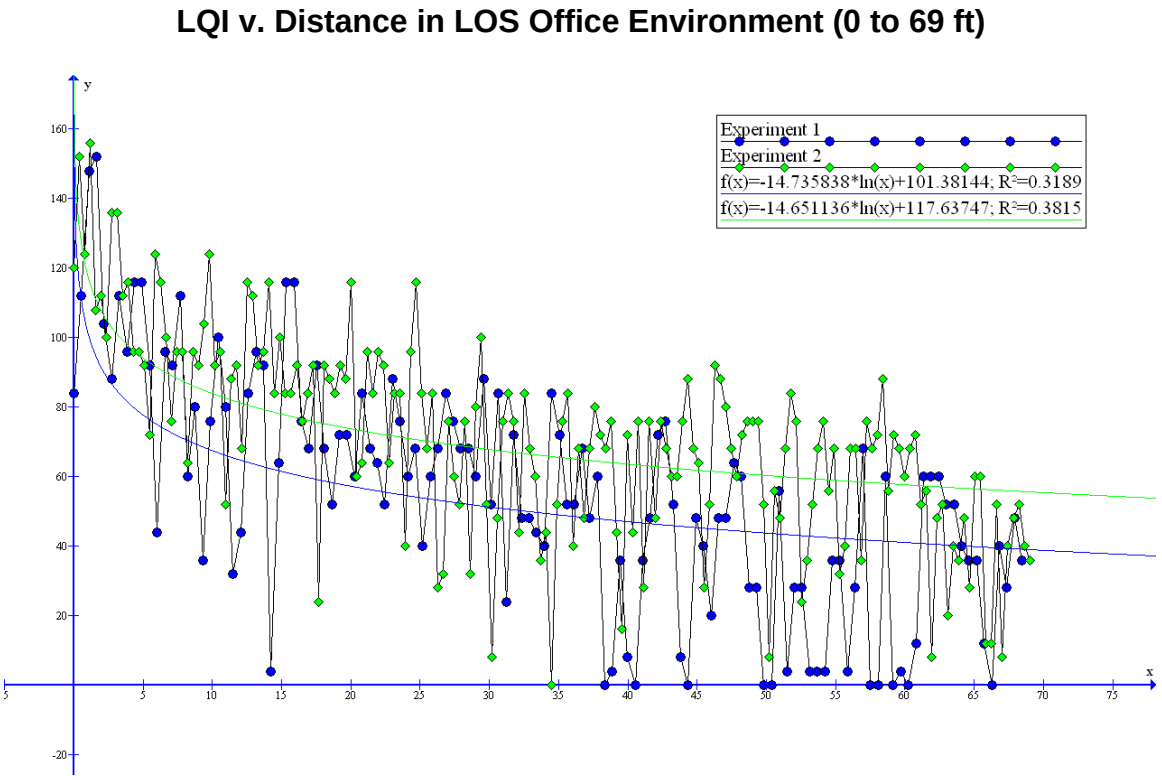

Fig 5-5: Experimental results for LOS office experiments

$$LQI(d) = -14.7*\ln(d)+109$$

Eq 5-4: Combined Trend Line for 2 LOS office experiments

The next question that might arise involves the channel selection. So, to check if the channel selection can make a difference, the 2.4GHz spectrum was analyzed to see if one channel had significantly less interference. Figure 5-6 shows a quick graph of how the spectrum was analyzed. A node was placed at a fix point in the office, and the sniffer was placed at a fixed point in the lab area. The node was programmed to various nodes and an average RSSI (in dBm) was computed using the MaxStream Range Test program [25]. The channel that is being used in all of the other experiments is channel 0x12 (or decimal channel 18). Channel 0x12 appears to be about average on the spectrum, and there doesn't appear to

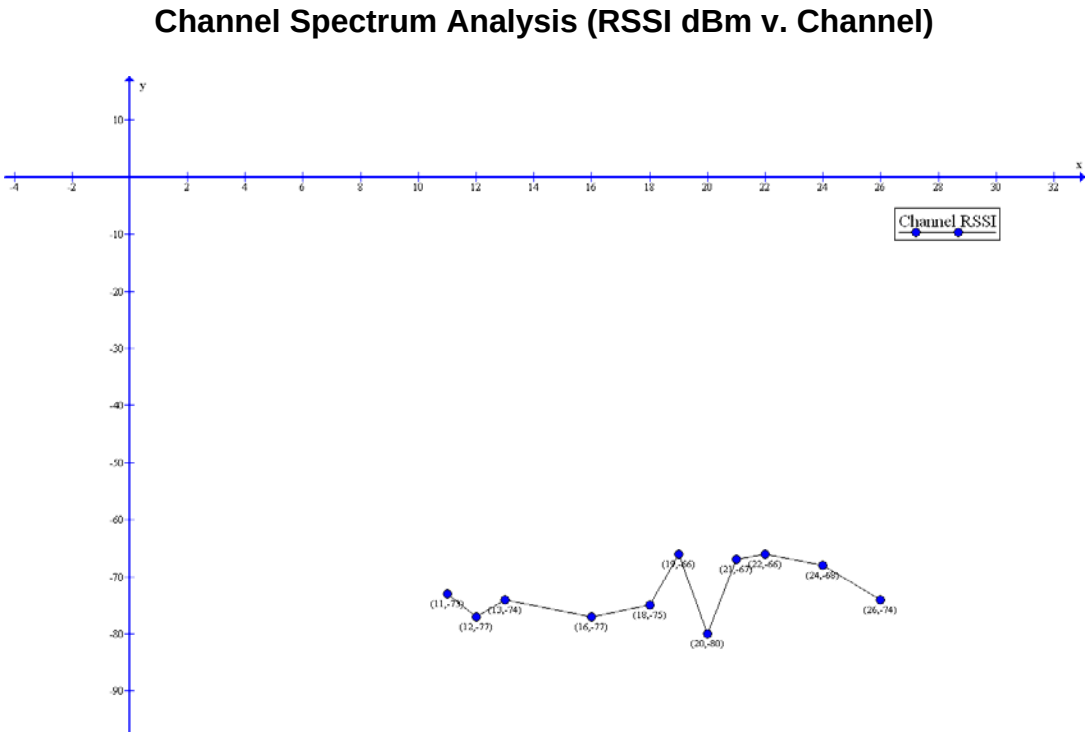

Fig 5-6: Spectrum analysis of various channels

LQI v. Distance in LOS Experiment with Channels 0x12 and 0x16

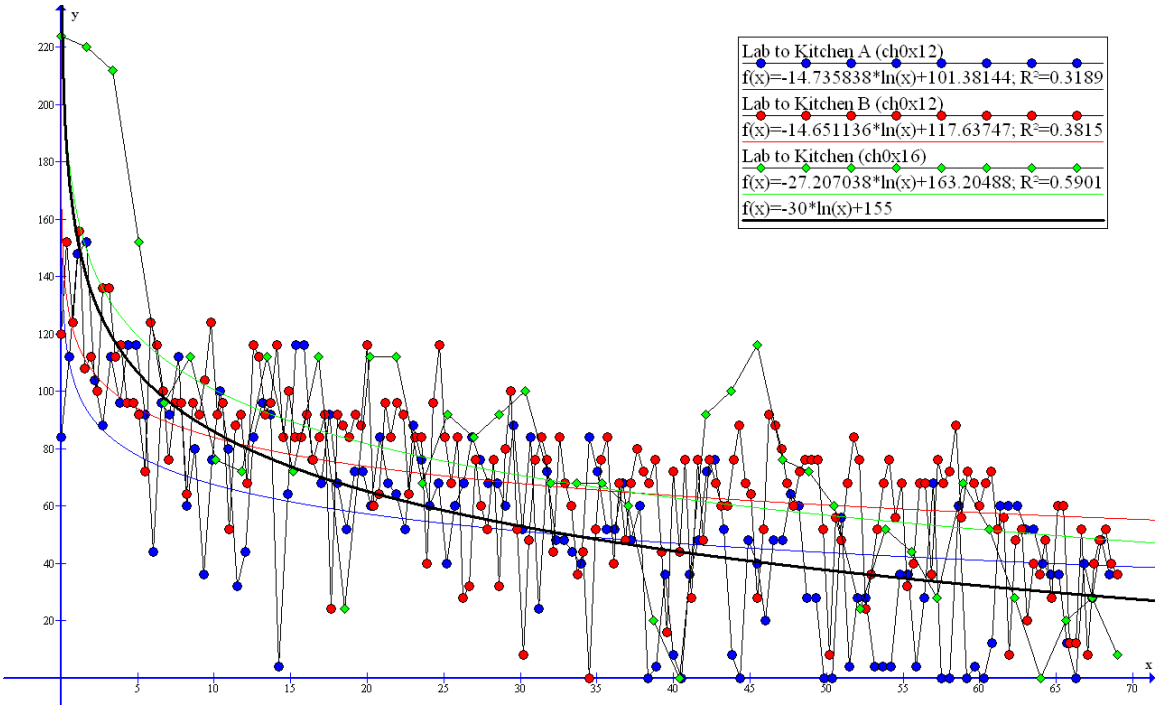

Fig 5-7: Comparing channels 0x12 and 0x16 in office LOS experiment

be too much fluctuation. Channel 0x16 (or decimal channel 22) was recorded with the smallest amount of interference. So the line-of-sight experiment was conducted on channel 0x16 for comparison and can be seen in Figure 5-7. Also included in Figure 5-7 is the "lossless" estimation from the warehouse experiment. This shows that this equation is pretty good estimation over time. It appears that the actual signal strength versus distance curve isn't quite logarithmic. It appears to decay at the beginning like a logarithmic, but then in the middle it seems to level out a little bit before beginning to decay again. It almost appears to have a third-order polynomial behavior, but for the sake of this research, a logarithmic equation is still assumed.

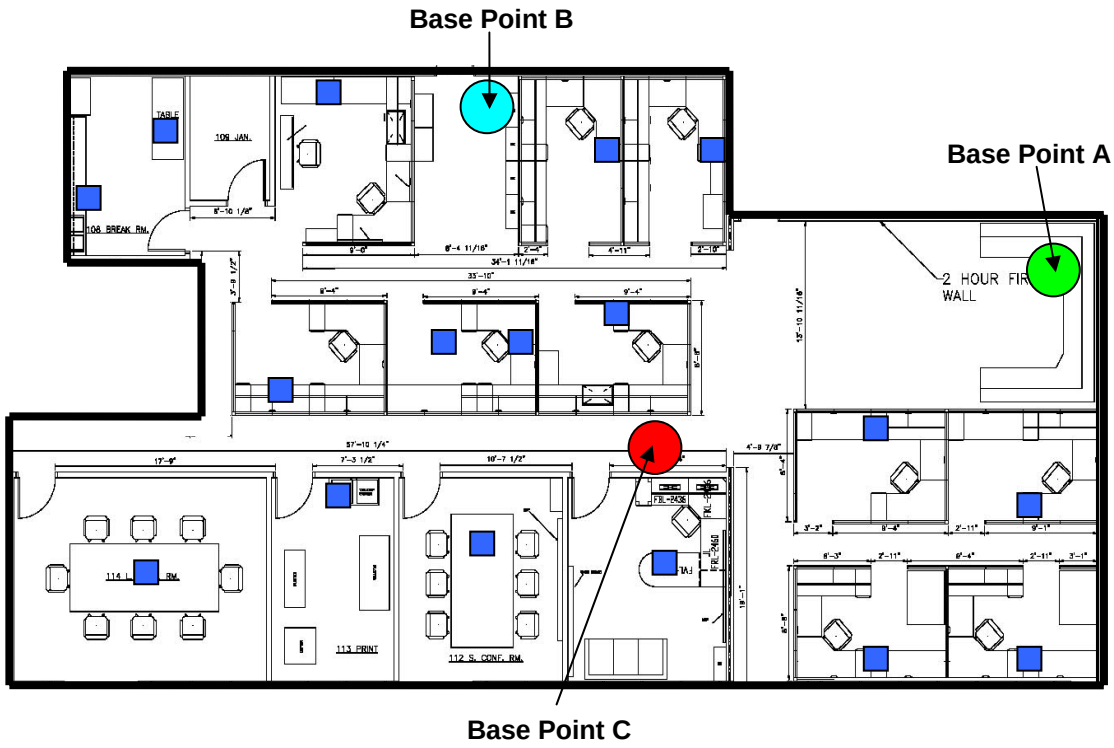

Fig 5-8: Office Layout for Multiple LOS Experiments

5.1.3 Multiple LOS with Typical Interference

The next step in the experiment phase is to begin modeling and gathering actual data that can be used in a simulation. Figure 5-8 shows the layout of the office that is used for these experiments as first discussed in chapter 4. At each location, the node sends out a series of packets that are averaged to determine the estimated LQI from that node to each base point. Figure 5-9 shows a summary of these results. The nodes are ordered in terms of distance. This graph shows that the data in the office from varying line-of-sights follows the same trends from previous experiments. The trend lines from the three base points are shown in the graph as well as the expected trend line (in black)

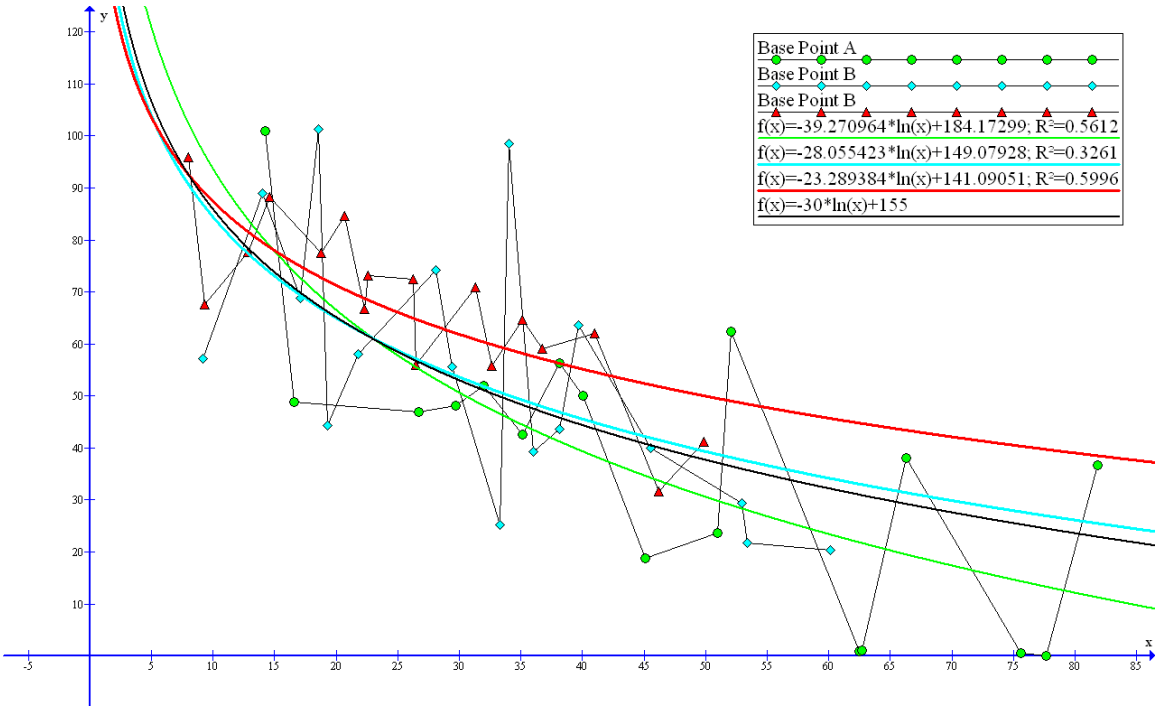

Fig 5-9: Actual data results from nodes placed throughout office

determined from the earlier experiments. This expected trend line is almost an exact average of the three lines. In this experiment, base point B seems to be the best location to get accurate results based on the estimated model. The three data sets found in Figure 5-9 is used as the "real" or "actual" data for all algorithm simulations.

5.2 Simulation Results

5.2.1 Error Modeling

Before any simulations can be done, some error model must be generated. From examining the actual data and comparing with our expected LQI equation (Equation 5-5), the contained error appears to be around +/-25 dBm with a few

$$LQI_{exp}(d) = -30*\ln(d) + 155$$

Eq 5-5: Expected LQI function of distance based on previous experiments

outliers beyond that range, with the average error being around +/-15 dBm. Figure 5-10 shows the Gaussian curve used to model that error using a standard deviation of 0.35. As the error rate is moved up or down, the accuracy changes inversely.

Figure 5-11 shows an example 100 random samples that predominantly fall in the range of +/-25 dBm.

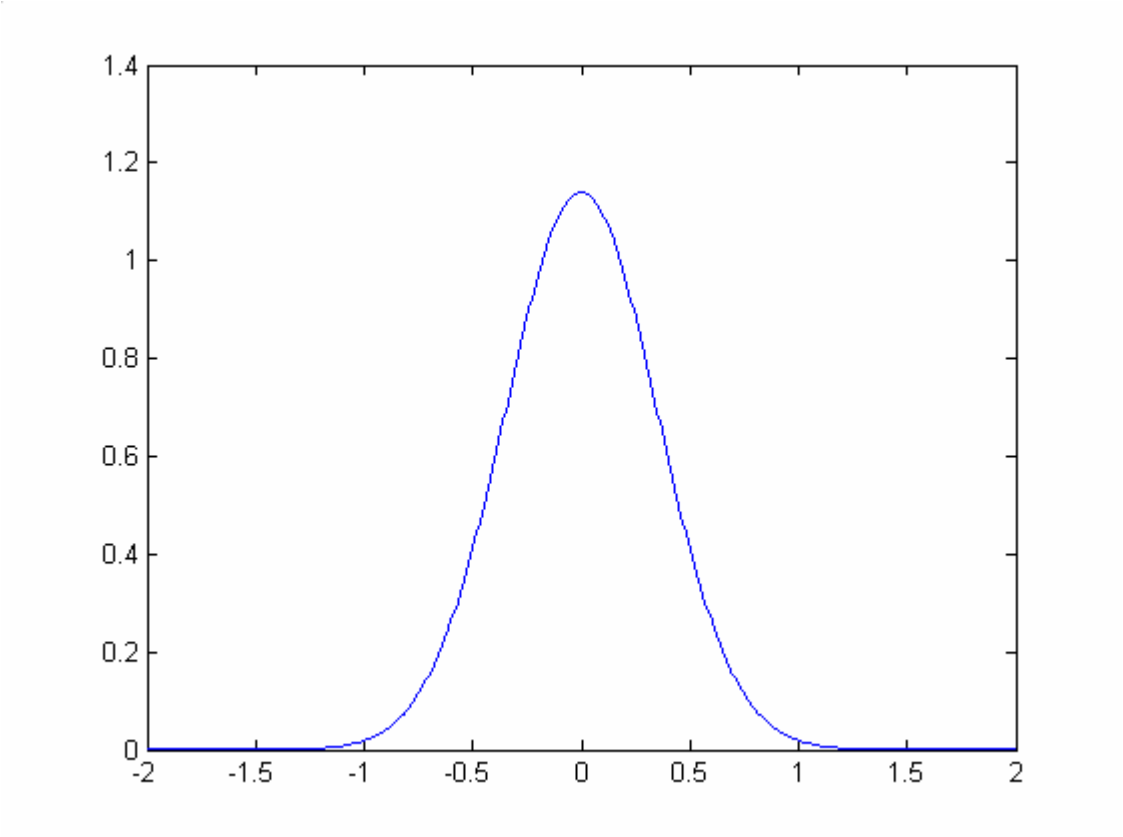

Fig 5-10: Gaussian error model for +/-25dBm

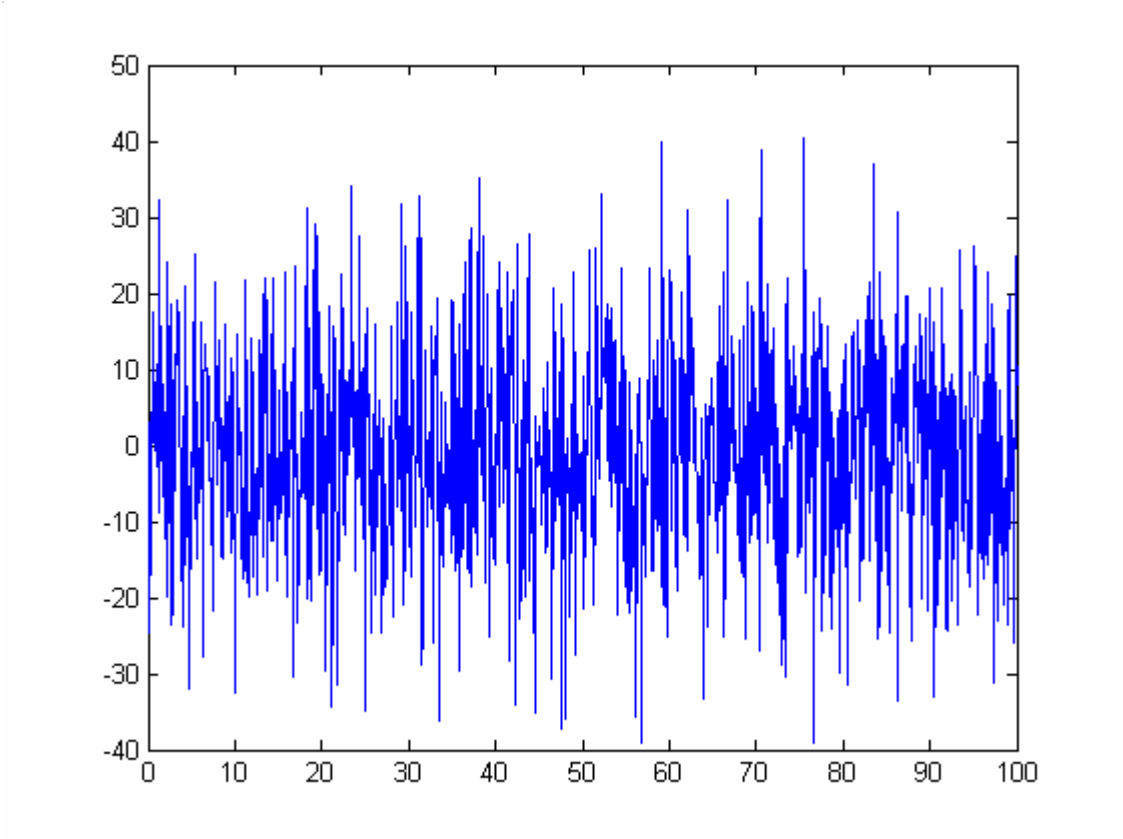

Fig 5-11: 1000 Random Samples of Gaussian distribution in +/-25 dBm range

5.2.2 Algorithm Results

Once the models were created from the gathered data, the algorithms were simulated. The only algorithm that was unable to be simulated was the Radial Triangulation method. There were multiple reasons why this simulation was never achieved. The main reason was handling contention. In the first phase of selecting a location based on the highest value in the array became difficult because there often way too many contending elements to get an accurate result.

Table 5-1: Accuracy table for algorithms using real data

Algorithm	Accuracy
Perceived Confidence w/o Contention	29.41%
Perceived Confidence w/ Contention	23.53%
Relative Confidence w/o Contention	29.41%
Relative Confidence w/ Contention	23.53%
Minimum Distance Error w/o Contention	23.53%
Minimum Distance Error w/ Contention	41.18%

The first test run of the algorithms was run on the original data set. This obviously is the most important since it is our real-world scenario that was used the generate the model. The results of the discovery algorithms using the actual data are summarized in Table 5-1. The first thing to notice is that the accuracy of the two confidence algorithms are exactly the same. As it turns out, in all of the simulations this is the case, so from this point forward, these are combined into one result, simply referred to as the Combined Confidence algorithm. It appears that the MDE with Contention method has a significant edge in accuracy.

The next step is to apply the algorithms with simulated data. The first scenario involves the default setup used to gather the actual data. The test is performed over a range of simulated errors from +/-1 dBm to +/-50 dBm. The results are seen in Figure 5-12, and it is confirmed that the two confidence algorithms behave identically once again. In the simulations, the MDE method still seems to be the highest accuracy but it is not nearly as prevalent as the actual data suggests.

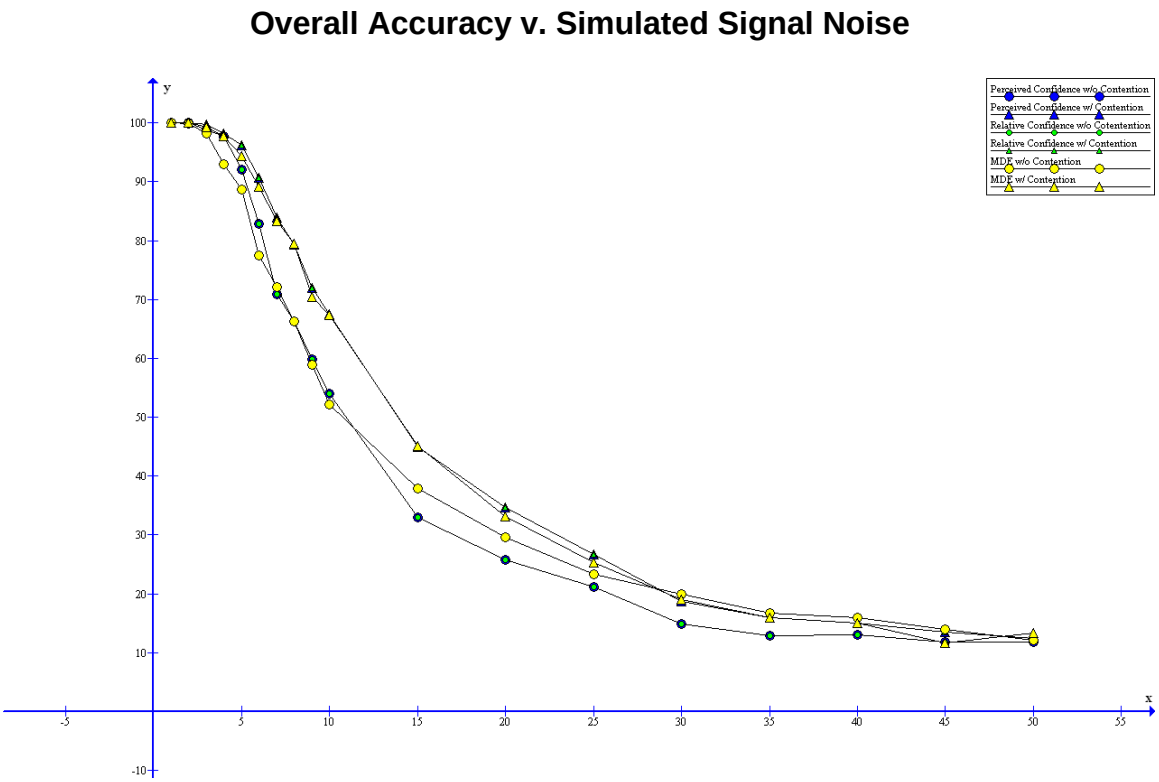

Fig 5-12: Results of the actual data run through the simulations

The next approach in the experiments is to simulate over an increasingly large and dense network to see how the algorithms respond. Each algorithm was tested in a basic network layout with nodes increasing from 5 to 30 and their density consequently increasing as a result. It starts out with just five nodes sparsely spread out, but then continues to add nodes in a fashion such that the density is increasing as the nodes increase. Figure 5-13 shows the results using the Combined Confidence Algorithms without contention, and Figure 5-14 shows the results using the Minimum Distance Error Algorithm with contention. One can see an obvious decline in the algorithm efficiency as the density increases.

Confidence Accuracy v. Error as Node Count Increases

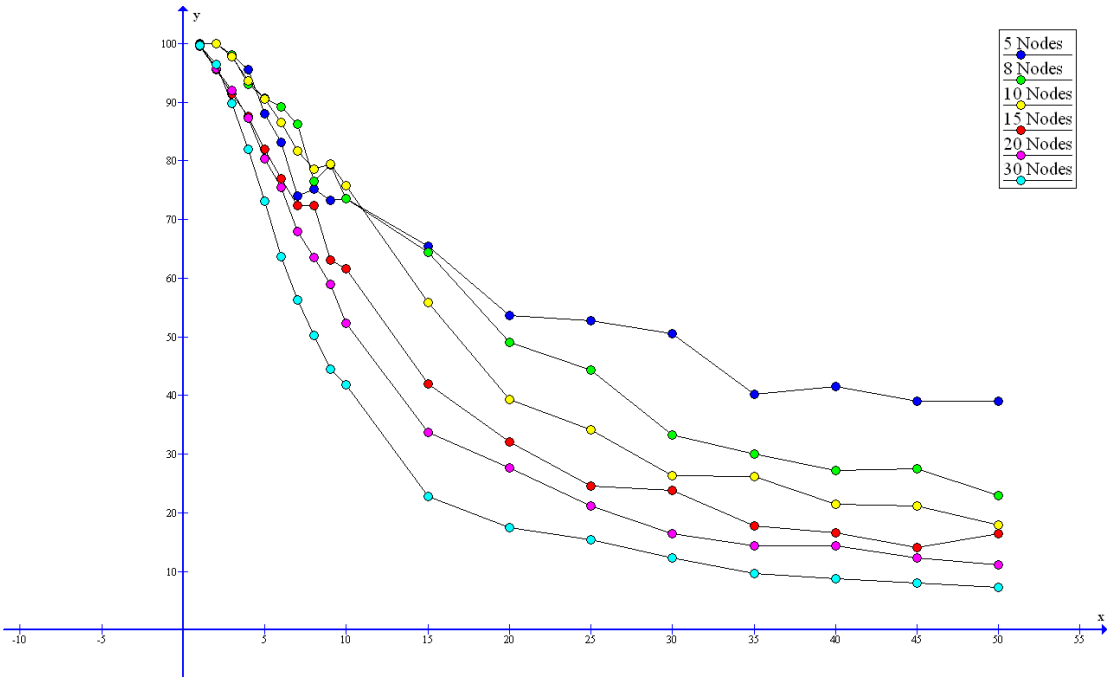

Fig 5-13: Confidence without contention algorithms as node density and error increase

MDE Accuracy v. Error as Node Count Increases

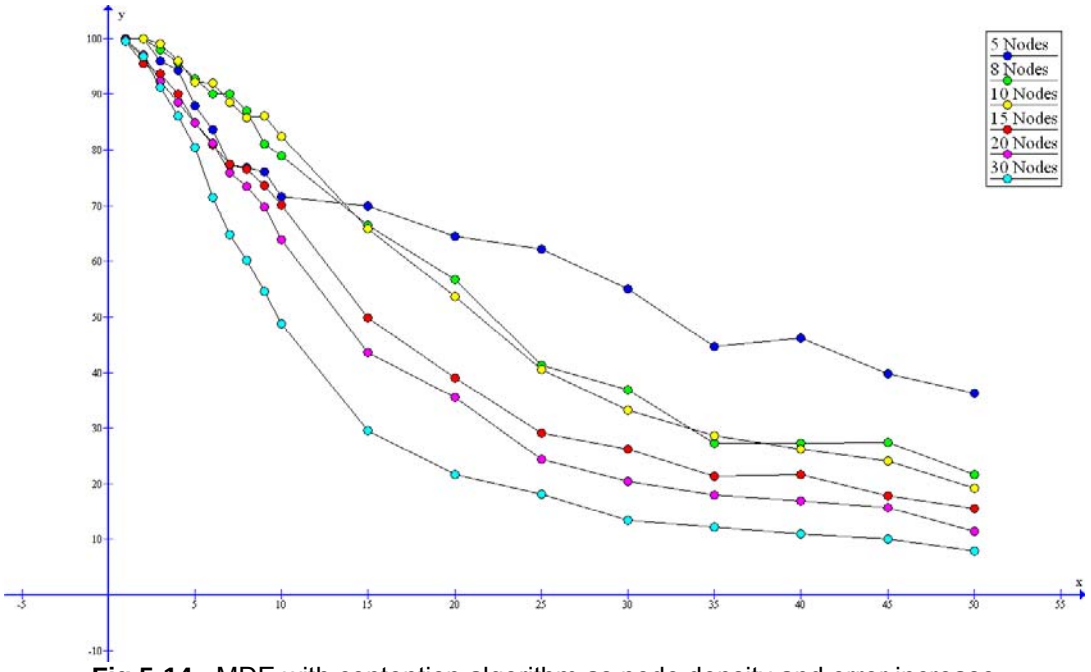

Fig 5-14: MDE with contention algorithm as node density and error increase

At this point, all of the above simulations have been done with three base points as reference nodes. The next step is to see if we can improve the results by adding extra base points. For these simulations, the node count and density are held constant at 15 nodes sparsely populated in the designated office area (which would be about 2 meters apart vertically and 4 meters apart horizontally). In the first simulation, we start with one base point and then incrementally add one more base point for each iteration while trying to maintain a common-sense balance of base point layouts. The simulation is run for up to 12 base points. This obviously doesn't make much sense to have as many or more base points as nodes, because the amount of work required is almost as much as manually identifying each node. Figure 5-15 shows the results of the simulations done with an error factor of +/-25 dBm – the peak error, and Figure 5-16 shows the results of the simulations done with an error factor of +/-15 dBm – the average error. In the second simulation, we extend the base point count to 20 to inspect just how high the accuracy reaches. It doesn't appear to have reached a saturation point, so in theory the number of points could be increased until the accuracy is as close to 100% as needed. After adding 12 base points, our accuracy with the +/-25 dBm error factor is around 50%. This obviously is not good enough. If we assume an error around +/-15 dBm, then with 12 base points we can achieve approximately 75% accuracy. If the number of base points is extended all the way out to 20, the accuracy reaches 87%. These values are starting to get into the range of acceptable values. Also, one can note from these results that the Confidence algorithm (indicated in light blue on the graphs) is

Accuracy v. Base Point Count (+/- 25 dBm)

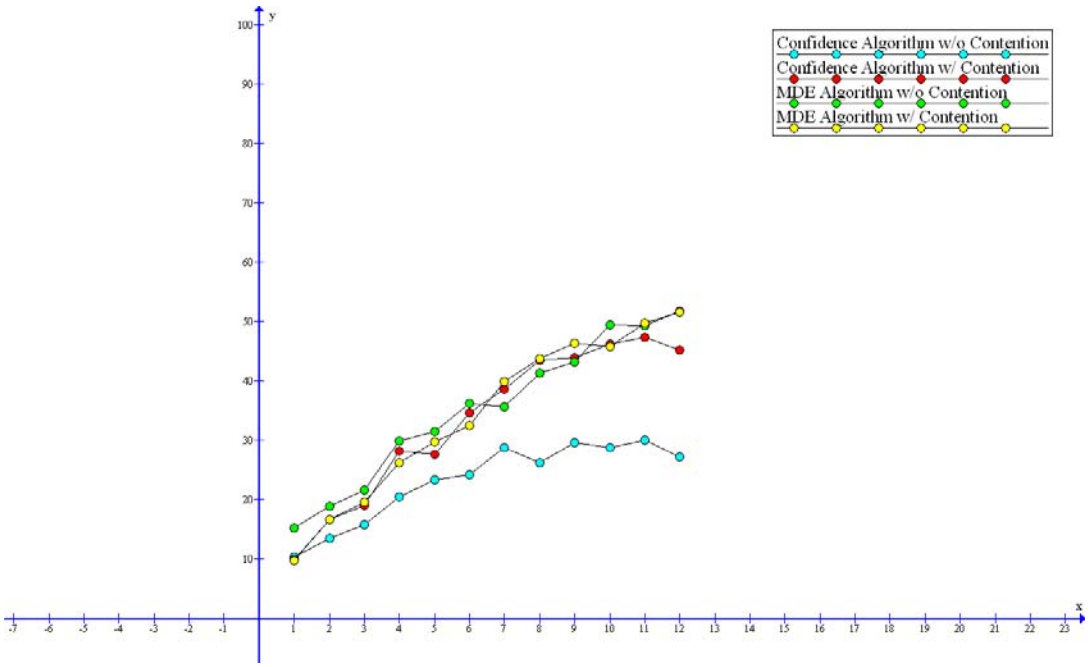

Fig 5-15: Accuracy as base point count increases (1 to 12)

Accuracy v. Base Point Count (+/- 15 dBm)

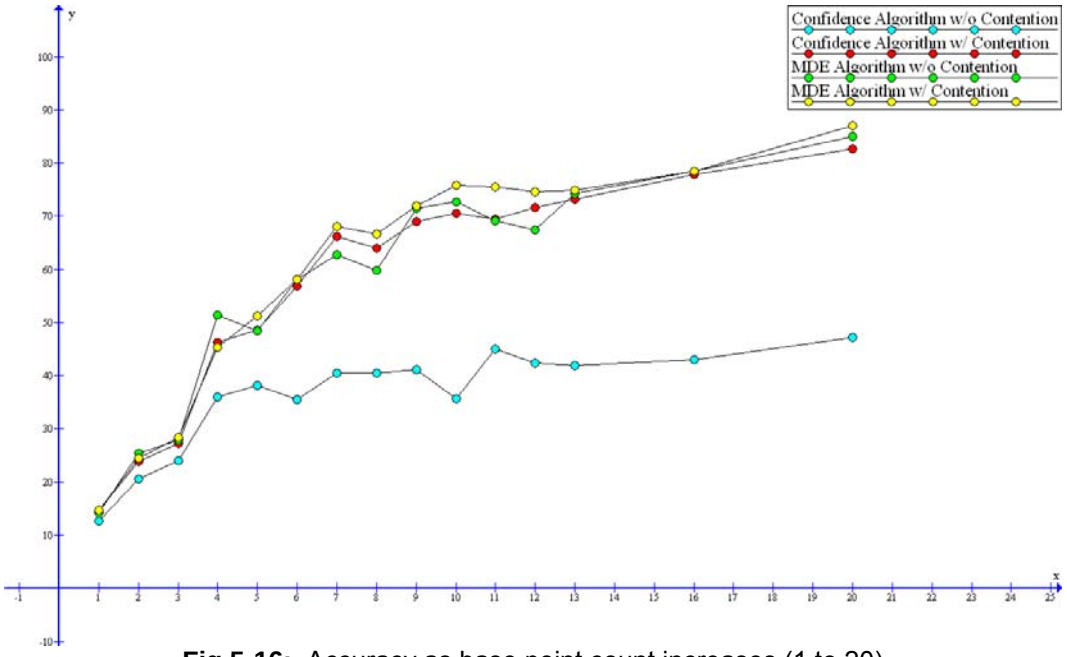

Fig 5-16: Accuracy as base point count increases (1 to 20)

insufficient; whereas the other three algorithms are relatively close with the MDE Algorithm with Contention (indicated in yellow) holding a slight edge still.

Now that the results are starting to become a little bit more feasible, the problem is that we are requiring significantly more time, equipment, and effort, making the entire process somewhat inefficient. So, the next step is, instead of having 20 unique base points, determining if we still achieve the same amount of accuracy by instead using just 5 base points and running the algorithm 4 times on 4 sets of data. Essentially this is no different than having 20 base points with 4 bases stacked one upon the other in 5 unique places. This simulation uses the same 15 node layout with the same nodal density and base point balance. Figure 5-17 shows the results of this simulation. In the graph, the first point represents a

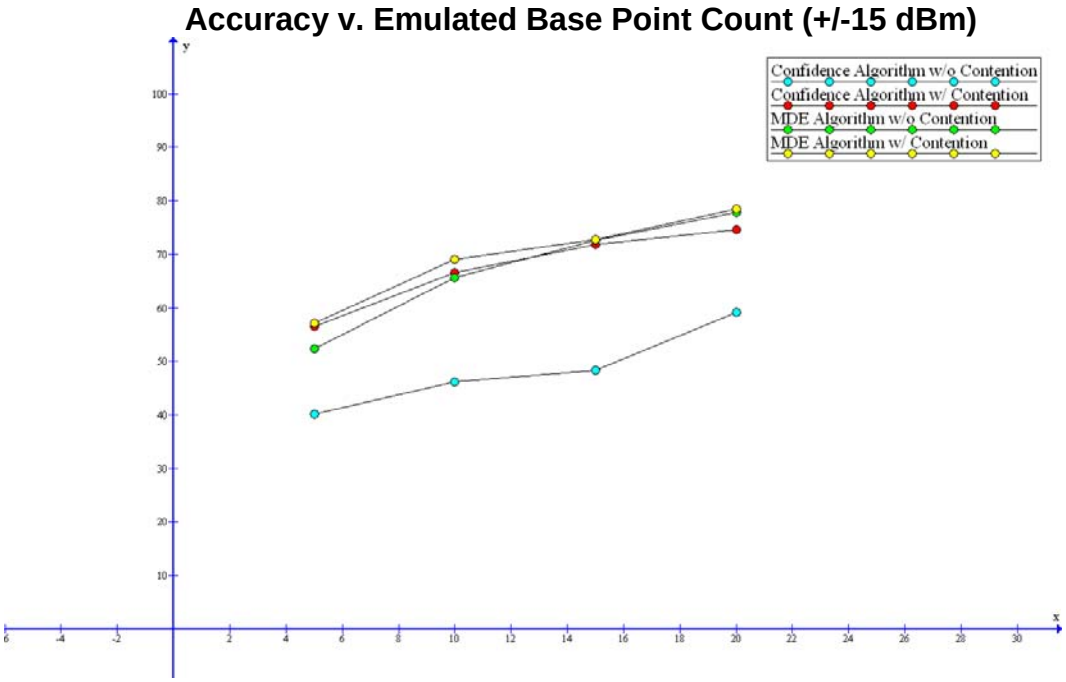

Fig 5-17: Accuracy as emulated base point count increases (5x1 to 5x4)

base 5-point layout (5x1). The next point represents that same 5-point layout executed twice (5x2). This is done up to a 5x4 simulation of 20 emulated points. From this simulation we were able to gain an accuracy of 80% which is comparable to the 87% gained in the non-emulated simulation. However, the results of this simulation might not be realistic, because with the simulation we are assuming a new random signal received during each execution cycle. If there is enough time between each execution cycle, this is feasible because the interference might be completely different, but if the amount of time between is too short or there is some obstruction that is creating a consistent signal loss, then the improvement won't be much more than the simple 5-point simulation. Only a real-data experiment could prove or disprove these hypothesis either way.

The rest of the simulation graphs and results can be seen in Appendix B. Overall there is not a great difference between each of the algorithms. It can be concluded, though, that contention does improve the algorithm significantly, which would be expected. Also, the MDE algorithms seem to have set themselves apart as just above the confidence algorithms. However, they seem to be a bit inconsistent and fluctuate quite a bit more than the confidence algorithms. If a triangulation method can be implemented, it should improve on the MDE algorithm because it will remove the mirrored information that the MDE cannot. The MDE is limited to a single distance factor whereas triangulation has two distance factors as it utilizes actual coordinate information rather than non-directional distance. Finally, it appears the only real way to improve these

algorithms as they are (without modifying the algorithm itself) is to increase the number of base points. Again, though, it is important to understand that a base point does not necessarily have to be a unique node in a commissioning environment. It obviously would not make sense for someone to deploy a base point for every node in the building in order to discover the network. It might simply require that the user carry a single node (or a small quantity of nodes) to multiple locations for short amounts of time to gather data. The line still gets blurred though as one considers whether this would improve on a basic commissioning of walking to each node and trying to identify it (by, for example in the lighting control application, turning the node, which is a light ballast in this case, on and off). These are things that will have to be experimented with once an actual implementation of these algorithms is possible.

5.3 Chapter Summary

The conclusion that can be drawn from these results and experiments are that the signal strength indicators present in ZigBee provide an error of around +/-25 dBm peak-to-peak with an average around +/-15 dBm, but these algorithms proposed, when using a basic 3-base layout, need an error less than +/-5 to achieve near perfect results or at least less than +/-10 to achieve desirable results. However, as the base point count is increased, we slowly begin to see results that approach acceptability (maxing at around 75-85% when using almost as many base points as nodes). The basic question though is what range of these parameters are feasible to actually improve on a manual identification.

Between each of the algorithms, the MDE approach seems to be the best and it is the simplest, which helps in device complexity. Also, the contention methods do add significant improvement, but they are simple in nature. A more complex contention method could improve the results considerably. It is also hypothesized that if a working triangulation method could be achieved, it should improve on the results given by MDE since it includes coordinate and directional information.

Chapter 6

Conclusions

6.1 Conclusions

The goals of this thesis were two-fold. The first was to investigate the reliability of ZigBee signal strength in a variety of environments. The second was to try to come up with an algorithm or approach that translates ZigBee's signal strength indicators into distance indicators such it could be used for discovery, identification, and localization of a high-dense network as it relates to real-world applications. There were seven algorithms envisioned, six of which were actually implemented and simulated.

After implementing several experiments, quite a bit of data was collected on ZigBee's wireless signal strength indicators. Once the data was plotted and trended, we were able to see just how much fluctuation and error is involved in trying to relate the embedded LQI factor to distance. Once a model was created and the realistic error determined, these models were applied to the algorithms. The results of the algorithm show that in at least an office environment, the reliability of the signal just isn't there for a simple algorithm unless a large

number of base points are used. However, in order to keep the process automated, one would then need to add more nodes, or if it isn't automated, then the time to complete becomes an issue. From the experiments, it was determined that the actual signal fluctuation is approximately +/-25 dBm peak-to-peak and +/-15 dBm on average, but to get any reasonably acceptable results, the error must be reduced to less than +/-10 dBm and preferably +/-5 dBm. If commissioning an office environment before the actual office were in use, then these errors are possible but still not likely.

However, from these results, it seems that there are some elements which actually show promise. There are several directions one could go to augment and improve the algorithms in place. The tradeoff, though, that must be considered is that the complexity of these algorithms cannot come at the cost of the network. Overall, it turned out that all of the algorithms pretty much performed very similarly. The simplest of the algorithms, the MDE algorithm, actually seemed to perform the best; although, it also fluctuated the most. The triangulation algorithm proposed was unable to be implemented, but it does have some potential. Contention is the primary issue that will arise with this algorithm, and it will really grow the complexity quickly. The other algorithms, though, showed that handling contention is an important factor, which of course was expected. Contention was handled in a very basic way in these algorithms, so a potential improvement in the algorithms could rest heavily on the way contention is handled.

6.2 Challenges

Probably the biggest challenge that arose in this research effort was the lack of available nodes. Because of my limitations in equipment, I was unable to explore any cooperative algorithms. A cooperative algorithm, while it would increase the complexity tremendously, seems to be a key in refining the commissioning in a real-world scenario. The next most challenging aspect was finding actual working algorithms that were related to the issues this thesis intended to cover. There were several algorithms out there that claimed to have certain accuracies, but those algorithms were either protected by IP or non-applicable to the applications examined in this thesis. Thirdly is the fact that ZigBee on the whole is a relatively new technology. While the algorithms, techniques, and fundamentals of this research were not new, the fact that it was operating on a new standard and platform provided some definite hardships. It is essential to understand the technology and how it works before any application is developed on top of it, especially considering the fact that one of the primary focus' of this thesis was to keep real-world applications in mind.

6.3 Future Work

While the reliability of ZigBee's LQI information may not have been sufficient for the algorithms I proposed, ZigBee definitely has a place in the future market. Therefore, it is essential that problems such as discovery and commissioning are addressed either at the stack level or application level. The key to localization using signal strength lies in some element of cooperative learning, including

potentially involving the user [5]. Some ideas that came to mind as this research was being done include using some level of artificial intelligence (AI) or learning techniques. This could be done over several sets of test data or it could actually be done live. In other words, during commissioning allow the system to make an attempt at discovering the nodes within the network, then ask the user node by node if it is right. As soon as an incorrect node is identified, it uses that information to refine its localization on the whole network (not just that node). One could also look into reducing the neighborhood size and only accept LQI's that meet a minimal criteria. Then use those discovered nodes to expand the circle like a ripple affect. This could especially be appealing since the LQI seems to remain much more consistent in the higher ranges. Another cooperative method that might prove useful is to actually use the nodes that exist as the actual base points. This would require a lot of recursive steps in the algorithms and crashing the network would be a big concern. Finally, trying to get a triangulation method that handles contention appropriately and effectively could present very promising results. If none of these methods work or are feasible, it may then be necessary to investigate the use of acoustic wave transmission such as the Cricket method mentioned previously so long as the cost is not an issue.

All in all, there are an ample number of directions that could be taken to continue the efforts of solving this issue. The implementation presented in this thesis was a rather simple one, but it can become complex quickly. The keys to the success

of the technology, though, are keeping the real-world applications – like commissioning – in mind. Therefore, a practical solution that combines both accuracy and reduced cost and complexity is needed. This thesis shows that a simple solution might not be sufficient but that there is definite promise that a sufficient solution exists.

BIBLIOGRAPHY

Bibliography

[1] Akyildiz, Ian F., Xudong Wang, and Weilin Wang. "Wireless mesh networks: a survey." Broadband and Wireless Networking Lab, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA. Computer Networks. Science Direct. December 20, 2004. Elsevier 2004.

[2] Bahl, Paramvir and Venkata N. Padmanabhan. "RADAR: An In-Building RF-based User Location and Tracking System." Microsoft Research. *Proceedings of IEEE Infocom 2000*. Tel-Aviv, Israel. March 2000.

[3] "Bluetooth.com." http://www.bluetooth.com/. Bluetooth SIG, Inc. 2007.

[4] Cao, Qing, Tian He, Lei Fang, Tarek Abdelzaher, John Stankovic, and Sang Son. "Efficiency Centric Communication Model for Wireless Sensor Networks." Department of Computer Science, University of Virginia, Charlottesville, VA.

[5] Chen, Wenping and Xiaofeng Meng. "A Cooperative Localization Scheme for ZigBee-based Wireless Sensor Networks." College of Information, Renmin University of China, Beijing, China. 0-7803-9746-0/06. IEEE, 2006.

[6] Ergen, Sinem Coleri. "ZigBee/IEEE 802.15.4 Summary." University of California, Berkeley, EECS. September 10, 2004.

[7] Ferris, Brian, Dirk Hahnel, and Dieter Fox. "Gaussian Processes for Signal Strength-Based Location Estimation." University of Washington, Department of Computer Science and Engineering, Seatle, WA. Intel Research Seatle, WA. *Proc. Of Robotics: Science and Systems*, 2006.

[8] Gungor, V Cagri, and Frank Lambert. "A Survey on Communication Networks for Electric System Automation." School of Electrical and Computer Engineering, Georgia Institute of Technology. National Electric Energy Testing, Research, and Applications Center. Atlanta, GA.

[9] Hansen, Mats Skogholt, and Stig Stoa. "Pracitcal Evaluation of IEEE 802.15.4/ZigBee Medical Sensor Networks." Norwegian University of Science and Technology, Department of Electronics and Telecommunications.

[10] Hightower, Jeffrey, Gaetano Borriello, and Roy Want. "SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength." University of Washington, Computer Science and Engineering, Seatle, WA. Xerox Palo Alto Research Center, Palo Alto, CA. UW CSE Technical Report #2000-02-02. February 18, 2000.

[11] "Home Hearbeat – Home Awareness, Monitoring, Automation & Safety that Security Systems do not Offer." http://www.homeheartbeat.com. Eaton Electrical, 2007.

[12] Jain, Kamal, Jitendra Padhye, Venkat Padmanabhan, and Lili Qiu. "Impact of Interference on Multi-hop Wireless Network Performance." Microsoft Research. *ACM Mobicom 2003.* San Diego, CA. September 2003.

[13] Kiger, Chad. “Physical Layer Simulation Study for the Co-existence of WLAN Standards.” December 2005. University of Tennessee, Knoxville, Masters Thesis, Department of Electrical Engineering.

[14] Kohvakka, Mikko, Mauri Kuorilehto, Marko Hannikainen, and Timo D. Hamalainen. “Performance Analysis of IEEE 802.15.4 and ZigBee for Large-Scale Wireless Sensor Network Applications.” Tampere University of Technology, Institute of Digital and Computer Systems, Tampere, Finland. PE-WASUN ’06, October 6, 2006. Torremolinos, Malaga, Spain. ACM 2006.

[15] Krishnamachari, Bhaskar, and John Heidemann. “Application-Specific Modelling of Information Routing in Wireless Sensor Networks. Department of Electrical Engineering, Information Sciences Institute, University of Southern California. ISI-TR-576 August 2003.

[16] Lorincz, Konrad, and Matt Welsh. “MoteTrack: A robust, decentralized approach to RF-based location tracking.” Pers Ubiquit Comput. November 9, 2005. Springer-Verlag London Limited 2006.

[17] Lymberopoulos, Dimitrios, Quentin Lindsey, and Andreas Savvides. “An Emperical Characterization of Radio Signal Strength Variability in 3-D IEEE 802.15.4 Networks Using Monopole Antennas.” Embedded Networks and Applications Lab, ENALAB. Yale University, New Haven, CT. EWSN 2006, LNCS 3868, pp. 326-341. Springer-Verlag Berlin Heidelberg 2006.

[18] Padhye, Jitendra, Sharad Agarwal, Venkata N. Padmanabhan, Lili Qiu, Ananth Rao, and Brian Zill. “Estimation of Link Interference in Static Multi-hop Wireless Networks.” Microsoft Research, University of Texas, and

University of California. *ACM/USENIX Internet Measurement Conference.* Berkeley, CA. October 2005.

[19] Pereira, Luis R. "The IEEE 802.15.4 Standard and ZigBee Specifications." Embedded Systems and Communications Group, Eaton Innovation Center. Eaton Corporation, 2006.

[20] Priyantha, Nissanka Bodhi. "The Cricket Indoor Location System." Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. June 2005.

[21] Priyantha, Nissanka Bodhi. "Providing Precise Indoor Location Information to Movile Devices." Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. January 2001.

[22] Priyantha, Nissanka B., Anit Chakraborty, and Hari Balakrishnan. "The Cricket Location-Support System." MIT Laboratory for Computer Science. 6th ACM International Conference on Mobile Computing and Networking (ACM MOBICOM). Boston, MA. August 2000.

[23] Sahinoglu, Zafer, Philip Orlik, Jinyun Zhang, Bharat Bhargava, and Gang Ding. "Reliable Broadcasting in ZigBee Networks." Mitsubishi Electric Research Laboratories. TR2005-029. Cambridge, Massachusetts. December 2005.

[24] Small, Jason, Asim Smailagic, Daniel P. Siewiorek. "Determining User Location for Context Aware Computing Through the Use of a Wireless LAN Infrastructure." Institute for Complex Engineered Systems. Carnegie Mellon University, Pittsburgh, PA.

[25] “XBee ZigBee Development Kit – MaxStream.net.” http://www.maxstream.net/products/xbee/dev-kit-zigbee.php. MaxStream, Inc., 2007.

[26] “ZigBee Alliance.” http://www.zigbee.org/. ZigBee Alliance, 2007.

[27] “ZigBee / IEEE 802.15.4 from Texas Instruments.” http://focus.ti.com/analog/docs/gencontent.tsp?familyId=367&genContentId=24190. Texas Instruments Incorporated, 1995-2007.

[28] ZigBee Specification. ZigBee Document 053474r13. December 1, 2006. ZigBee Alliance, Inc. San Ramon, CA. ZigBee Standards Organization, 2006.

APPENDIX

Appendix A

MatLab Code

Thesis.m:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% thesis.m
% Author: Brandon Rogers
% Date:  4/20/07
%
% Description:
% This file contains the GUI construction
% and all of the function definitions for
% for each of the algorithms implemented
% for my thesis entitled "The Reliability
% of ZigBee Signal Strength for Discovery,
% Identification, and Localization, of
% High-Density Wireless Sensor Networks."
%
%
% Last Modified:  4/20/07
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = thesis(varargin)
% THESIS M-file for thesis.fig
%      THESIS, by itself, creates a new THESIS or raises the existing
%      singleton*.
%
%      H = THESIS returns the handle to a new THESIS or the handle to
%      the existing singleton*.
%
%      THESIS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in THESIS.M with the given input arguments.
%
%      THESIS('Property','Value',...) creates a new THESIS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before thesis_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to thesis_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help thesis

% Last Modified by GUIDE v2.5 18-Apr-2007 17:31:47

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                   'gui_OpeningFcn', @thesis_OpeningFcn, ...
                   'gui_OutputFcn',  @thesis_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before thesis is made visible.
function thesis_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to thesis (see VARARGIN)

% Choose default command line output for thesis
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes thesis wait for user response (see UIRESUME)
% uiwait(handles.figure1);

global g_img_w;
global g_img_h;
global g_A;
global g_F;
global g_FwN;
global g_factor;
global f_open;
global f_cube;
global f_office;
global f_drywall;
global f_firewall;
global f_kitchen;
global f_generic_rm;
global f_interference;
global sig_max;
global sig_min;
global g_nodes;
global g_nodecoords;
global g_nodecount;
global g_nodedescripts;
global g_bases;
global g_basecoords;
global g_basecount;
global g_basedescripts;
global MAX_NODES;
global MAX_BASES;
global ERROR_RATE;
global NUM_EPOCHS;
global bpList;
global nodesto;
global basepoints;

cla;

MAX_NODES = 256;
MAX_BASES = 16;
ERROR_RATE = 25;
NUM_EPOCHS = 500;

nodesto = 1;
basepoints = 1;

bpList = zeros(1,MAX_BASES);

g_img_w = 256;
g_img_h = 120;
```

```
g_factor = 0.5;

f_open = 0.99;
f_cube = 0.97;
f_office = 0.95;
f_kitchen = 0.90;
f_generic_rm = 0.94;
f_drywall = 0.90;
f_firewall = 0.80;
f_outerwall = 0.10;
f_bathroom = 0.94;

f_interference;

sig_max = 1.0;
sig_min = 0.05;

g_A = zeros(g_img_h,g_img_w);
g_nodes = zeros(g_img_h,g_img_w);
g_nodecount = 0;
g_nodecoords = zeros(MAX_NODES,2);

g_bases = zeros(g_img_h,g_img_w);
g_basecount = 0;
g_basecoords = zeros(MAX_BASES,2);

g_A(2:g_img_h-1,2:g_img_w-1) = f_open*100;
g_A(29:30,2:196) = f_firewall*100;
g_A(30:49,195:196) = f_firewall*100;
g_A(48:49,196:g_img_w-1) = f_firewall*100;

g_A(g_img_h-1-17:g_img_h-1,g_img_w-1-49:g_img_w-1) = f_cube*100; % Adam and Matt D's
Cubes
g_A(g_img_h-1-17-8-17:g_img_h-1-17-8,g_img_w-1-49:g_img_w-1) = f_cube*100; % Jesse and
Robert's Cubes
g_A(31:56,g_img_w-1-134:g_img_w-1-112) = f_cube*100; % Barb's Cube
g_A(31:56,g_img_w-1-95:g_img_w-1-78) = f_cube*100; % Matt P's Cube
g_A(31:56,g_img_w-1-77:g_img_w-1-61) = f_cube*100; % Justin's Cube
g_A(64:81,g_img_w-1-143:g_img_w-1-69) = f_cube*100; % Danielle, Brandon, and Manson's
Cubes

g_A(31:56,g_img_w-1-170:g_img_w-1-150) = f_kitchen*100; % Kitchen
g_A(g_img_h-1-30:g_img_h-1,g_img_w-1-49-10-26:g_img_w-1-49-10) = f_office*100;
g_A(g_img_h-1-26:g_img_h-1,2:52) = f_office*100;

g_A(g_img_h-1-30:g_img_h-1,g_img_w-86-90:g_img_w-86) = f_generic_rm*100;
g_A(g_img_h-1-26:g_img_h-1,53:53+14) = f_generic_rm*100;
g_A(31:49,g_img_w-1-149:g_img_w-1-135) = f_generic_rm*100;

g_F = g_A;

% addNode(38,101,'Kitchen Table',1);
% addNode(50,87,'Kitchen Counter',2);
% addNode(33,129,'Barbs Cube',3);
% addNode(42,173,'Matt Ps Cube',4);
% addNode(42,190,'Justins Cube',5);
% addNode(77,118,'Danielles Cube',6);
% addNode(70,146,'Brandons Cube (b)',7);
% addNode(70,160,'Brandons Cube (a)',8);
% addNode(70,180,'Mansons Cube',9);
% addNode(g_img_h-16,g_img_w-156,'Lg Conf Rm',10);
% addNode(g_img_h-27,g_img_w-128,'Lg Print Rm',11);
% addNode(g_img_h-19,g_img_w-101,'Sm Conf Rm',12);
% addNode(g_img_h-16,g_img_w-71,'Rons Office',13);
% addNode(g_img_h-3,g_img_w-13,'Roberts Cube',14);
% addNode(g_img_h-3,g_img_w-33,'Jesses Cube',15);
% addNode(g_img_h-28,g_img_w-11,'Matt Ds Cube',16);
% addNode(g_img_h-40,g_img_w-35,'Adams Cube',17);
%
% addBase(64,g_img_w-6,'Base A',1);
% addBase(33,157,'Base B',2);
% addBase(g_img_h-1-33,g_img_w-1-75,'Base C',3);

g_FwN = g_A;

cmap = zeros(106,3);
for i=1:100
    for j=1:3
        cmap(i,j) = i/100;
```

```
    end
end

cmap(101,:) = [0 0 1];
cmap(102,:) = [0 1 0];
cmap(103,:) = [0 1 1];
cmap(104,:) = [1 0 0];
cmap(105,:) = [1 0 1];
cmap(106,:) = [1 1 0];

colormap(cmap);

h=image(g_A);
set(gcf,'Renderer','zbuffer');
hold off;




% --- Outputs from this function are returned to the command line.
function varargout = thesis_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%%%%%%%%%%%%%%%%%%%%%
% FUNCTION addNode %
function addNode(h, w, description, index)

global g_A;
global g_FwN;
global g_nodes;
global g_nodecoords;
global g_nodecount;

g_A(h-1:h+1,w-1:w+1) = 101;
g_FwN(h-1:h+1,w-1:w+1) = 101;
g_nodes(h,w) = 1;
g_nodecoords(index,:) = [h w];
g_nodecount = g_nodecount + 1;

%%%%%%%%%%%%%%%%%%%%%
% FUNCTION addBase %
function addBase(h, w, description, index)

global g_A;
global g_FwN;
global g_bases;
global g_basecoords;
global g_basecount;

g_A(h-1:h+1,w-1:w+1) = 101+index;
g_FwN(h-1:h+1,w-1:w+1) = 101+index;
g_bases(h,w) = 1;
g_basecoords(index,:) = [h w];
g_basecount = g_basecount + 1;


%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION drawConnection %
function drawConnection(from, to, color)

global g_A;

[x,y] = getFunction(from,to,2);

for i=1:length(y)
    if g_A(round(y(i)),round(x(i))) ~= 101
        g_A(round(y(i)),round(x(i))) = color;
    end;
    if mod(i,5)==1
        drawnow
```

```
        image(g_A);
    end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION getFunction %
function [x,y]=getFunction(from, to, ppp)

n = getDistance(from,to)*ppp;

if from(2)-to(2)==0
    if from(1) < to(1)
        y = linspace(from(1),to(1),n);
    else
        y = linspace(to(1),from(1),n);
    end;
    x = ones(1,length(y))*to(2);
else
    slope = (from(1)-to(1))/(from(2)-to(2));
    b = from(1)-slope*from(2);
    if to(2) > from(2)
        x = linspace(from(2),to(2),n);
    else
        x = linspace(to(2),from(2),n);
    end;
    y = slope*x+b;
end;

%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION getDistance %
function d=getDistance(from, to)

d = sqrt((from(1)-to(1))^2+(from(2)-to(2))^2)*0.5;

%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION estRSSI %
function rssi=estRSSI(from, to)

global g_F;

d=getDistance(from,to);
cRSSI=255;
[x,y]=getFunction(from,to,2);
%g_F
for i=1:length(x)
    cRSSI = cRSSI*(g_F(round(y(i)),round(x(i)))/100);
end;

rssi = cRSSI;

%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION graphRSSI %
function [x,y]=graphRSSI(bp,mt)

global g_nodecoords;
global g_nodecount;
global g_basecoords;

rssi = zeros(1,g_nodecount);
dist = zeros(1,g_nodecount);
for i=1:g_nodecount
    rssi(i) = estRSSI(g_basecoords(bp,:),g_nodecoords(i,:));
    dist(i) = getDistance(g_basecoords(bp,:),g_nodecoords(i,:));
end;

x = dist;
y = rssi;

figure(1);
scatter(x,y,mt);

% --- Executes on button press in drawConnections.
function drawConnections_Callback(hObject, eventdata, handles)
% hObject    handle to drawConnections (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global bpList;
global g_A;
global g_FwN;
```

```
global g_basecount;
global g_nodecount;
global g_basecoords;
global g_nodecoords;
global basepoints;
global nodesto;

g_A = g_FwN;

if nodesto==1
    for j=1:g_basecount
        if bpList(j)==1
            for i=1:g_nodecount
                drawConnection(g_basecoords(j,:),g_nodecoords(i,:),101+j);
                getDistance(g_basecoords(j,:),g_nodecoords(i,:));
                drawnow
                image(g_A);
                %hold;
                %pause(0.5);
            end;
        end;
    end;
end

if basepoints==1
    for j=1:g_basecount
        for i=j:g_basecount
            drawConnection(g_basecoords(j,:),g_basecoords(i,:),106);
        end;
    end;
end;


% --- Executes on button press in check_bpA.
function check_bpA_Callback(hObject, eventdata, handles)
% hObject    handle to check_bpA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check_bpA
global bpList;
bpList(1) = get(hObject,'Value');

% --- Executes on button press in check_bpB.
function check_bpB_Callback(hObject, eventdata, handles)
% hObject    handle to check_bpB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check_bpB
global bpList;
bpList(2) = get(hObject,'Value');


% --- Executes on button press in check_bpC.
function check_bpC_Callback(hObject, eventdata, handles)
% hObject    handle to check_bpC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check_bpC
global bpList;
bpList(3) = get(hObject,'Value');


% --- Executes on button press in radio_nodesto.
function radio_nodesto_Callback(hObject, eventdata, handles)
% hObject    handle to radio_nodesto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radio_nodesto
global nodesto;
nodesto = get(hObject,'Value');


% --- Executes on button press in radio_basepoints.
function radio_basepoints_Callback(hObject, eventdata, handles)
% hObject    handle to radio_basepoints (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radio_basepoints
global basepoints;
basepoints = get(hObject,'Value');


% --- Executes on button press in clearAll.
function clearAll_Callback(hObject, eventdata, handles)
% hObject    handle to clearAll (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global g_A;
global g_FwN;
g_A = g_FwN;
image(g_A);
drawnow


% --- Executes on button press in graphRSSI.
function graphRSSI_Callback(hObject, eventdata, handles)
% hObject    handle to graphRSSI (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
graphRSSI(1,'go');
hold on;
graphRSSI(2,'cs');
graphRSSI(3,'rd');


% --- Executes on button press in graphActualRSSI.
function graphActualRSSI_Callback(hObject, eventdata, handles)
% hObject    handle to graphActualRSSI (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global g_basecount;
global g_nodecount;
global g_basecoords;
global g_nodecoords;
data = csvread('lqi_data.csv',1,1);
figure(2);
hold on;
for j=1:g_basecount
%     for i=1:g_nodecount
%         x(i) = getDistance(g_basecoords(j,:),g_nodecoords(i,:));
%         y(i) = data(j,i);
%     end;
    [x,y]=getActualRSSI(j);
    if j==1
        scatter(x,y,'go');
    elseif j==2
        scatter(x,y,'cs');
    else
        scatter(x,y,'rd');
    end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION getActualRSSI %
function [x,y]=getActualRSSI(bp)
global g_basecount;
global g_nodecount;
global g_basecoords;
global g_nodecoords;
data = csvread('lqi_data.csv',1,1);
for i=1:g_nodecount
    x(i) = getDistance(g_basecoords(bp,:),g_nodecoords(i,:));
    if bp < 4
        y(i) = data(bp,i);
    else
        y(i) = 0;
    end;
end;

%%%%%%%%%%%%%%%%%%%%%
% FUNCTION estDecay %
function rssi=estDecay(d)
%rssi = -40*log(d)+213;
```

```
%rssi = -53*log(d)+255;
rssi = -30*log(d)+155;

%%%%%%%%%%%%%%%%%%%%
% FUNCTION estDist %
function d=estDist(rssi)

d = exp((rssi-155)/(-30));

% --- Executes on button press in graphEstimatedRSSI.
function graphEstimatedRSSI_Callback(hObject, eventdata, handles)
% hObject    handle to graphEstimatedRSSI (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global g_nodecoords;
global g_basecoords;
global g_nodecount;
global g_basecount;

figure(3);
hold on;

d=0.01:0.5:124;
plot(d,estDecay(d));
axis([0 124 0 255]);

%%%%%%%%%%%%%%%%
% FUNCTION pdf %
function y=pdf(x,mu,sigma)

y = (1/(sigma.*sqrt(2.*pi))).*exp(-1.*(x-mu).*(x-mu)/(2.*sigma.*sigma));

%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION doDiscovery %
function [c,i]=doDiscovery(x,y,verbose)

global g_nodecoords;
global g_basecoords;
global g_nodecount;
global g_basecount;

% Show Measured RSSI, and estimated and actual distances
if verbose
    fprintf(1,'\n\n***  DOING DISCOVERY  ***\n\n');
end;
for i=1:g_nodecount
    for b=1:g_basecount
        d = estDist(y(b,i));
        d_error = 100*abs(d-x(b,i))/d;
        rssi = estDecay(x(b,i));
        rssi_error = (rssi-y(b,i));
        error=rssi_error;
        if verbose
            fprintf(1,'Base %d--> Node: %d \tRSSI: %4.3f \tEstimated Distance: % 4.3f
\tActual: %4.3f \tEstimated RSSI: %3.2f \tError:
%2.2f\n',b,i,y(b,i),d,x(b,i),rssi,error);
        end;
    end;
    if verbose
        fprintf(1,'\n');
    end;
end;

for i=1:g_nodecount
    for b=1:g_basecount
        dact(b,i) = getDistance(g_nodecoords(i,:),g_basecoords(b,:));
        dest(b,i) = estDist(y(b,i));
    end;
end;

d_err = zeros(g_nodecount,g_nodecount);
for n=1:g_nodecount
    for k=1:g_nodecount
        d_err(n,k)=0;
        for b=1:g_basecount
            errval = abs(dest(b,n)-dact(b,k));
            d_err(n,k) = d_err(n,k)+errval;
        end;
    end;
```

```
end;

for n=1:g_nodecount
    if verbose
        fprintf('Discovering node %d:\n',n);
    end;
    for b=1:g_basecount
        dest(b) = estDist(y(b,n));
    end;

    for b=1:g_basecount
        node_array(b,:) = minSort(dest(b),dact(b,:));
        conf_array(b,:) = getConfidence(node_array(b,:),dest(b),dact(b,:));
    end;

    total_total_conf = 0;
       for i=1:g_nodecount
        total_conf(i) = 0;
        for b=1:g_basecount
            total_conf(i) = total_conf(i) + conf_array(b,findNode(node_array(b,:),i));
        end;
        total_total_conf = total_total_conf + total_conf(i);
       end;

    for i=1:g_nodecount
        relative_conf(n,i) = total_conf(i)/total_total_conf;
    end;

       final_order(n,:) = maxSort(total_conf);
    final_conf(n,:) = total_conf;
    if verbose
        fprintf('Order of node confidence: \n');
    end;
    if verbose
        for i=1:g_nodecount
            fprintf('%-7d',final_order(n,i));
        end;
        fprintf('\n');
        for i=1:g_nodecount
            fprintf('%-1.3f  ',final_conf(n,final_order(n,i)));%1.3f
',final_order(n,i),final_conf(n,final_order(n,i)));
        end;
        fprintf('\n');
        for i=1:g_nodecount
            fprintf('%-2.3f  ',100*relative_conf(n,final_order(n,i)));
        end;
        fprintf('\n\n');
    end;
end;

%% METHOD 1 %%
if verbose
    fprintf('Discovery Method 1 (w/o contention):\n');
end;
already_used = zeros(1,g_nodecount);
confidence_used = zeros(1,g_nodecount);
used_by = zeros(1,g_nodecount);
correct = 0;
for n=1:g_nodecount
    real_max_confidence=0;
    real_max_node=0;
    max_confidence=0;
    max_node=0;
    max_node_used=0;
    max_confidence_used=0;
    for i=1:g_nodecount
        if final_conf(i,n) > max_confidence && already_used(i)==0
            max_confidence=final_conf(i,n);
            max_node=i;
            if max_confidence > real_max_confidence
                real_max_confidence = max_confidence;
                real_max_node = max_node;
            end;
        elseif final_conf(i,n) > max_confidence && already_used(i)~=0
            node_used = already_used(i);
            used_max_confidence = final_conf(node_used,i);
            max_node_used=i;
            max_confidence_used = final_conf(i,n);
            if final_conf(i,n) > real_max_confidence
```

```
                real_max_confidence = final_conf(i,n);
                real_max_node = i;
            end;
        end;
    end;
    confidence_used(max_node) = max_confidence;
    already_used(max_node) = n;
    if verbose
        fprintf('Assign Node %d as Node %d with confidence %3.2f, but contended for Node
%d with confidence %3.2f\n',n,max_node,max_confidence,max_node_used,max_confidence_used);
    end;
    if n==max_node
        correct = correct+1;
    end;
end;

if verbose
    fprintf('\nCORRECT: %d     INCORRECT: %d     ACCURACY:
%2.2f%%\n',correct,g_nodecount-correct,100*correct/g_nodecount);
end;

c(1) = correct;
i(1) = g_nodecount-correct;


%% METHOD 2  w/ Contention %%
node_assignment = zeros(1,g_nodecount);
nodes_used = zeros(1,g_nodecount);
nodes_assigned = 0;
current_node = 1;
while nodes_assigned < g_nodecount
    if node_assignment(current_node) == 0
        order = 1;
        new_node = final_order(current_node,order);
        while nodes_used(new_node) ~= 0
            if final_conf(nodes_used(new_node),new_node) <
final_conf(current_node,new_node)
                node_assignment(nodes_used(new_node)) = 0;
                nodes_used(new_node) = 0;
                nodes_assigned = nodes_assigned-1;
            else
                order = order+1;
                new_node = final_order(current_node,order);
            end;
        end;
        node_assignment(current_node) = new_node;
        nodes_used(new_node) = current_node;
        if current_node == g_nodecount
            current_node = 1;
        else
            current_node = current_node+1;
        end;
        nodes_assigned = nodes_assigned+1;
    else
        if current_node == g_nodecount
            current_node=1;
        else
            current_node = current_node+1;
        end;
    end;
end;

correct=0;
if verbose
    fprintf('\n\n');
    fprintf('Discovery Method 2:\n');
end;
for n=1:g_nodecount
    if verbose
        fprintf('Assign Node %d as Node %d\n',n,node_assignment(n));
    end;
    if n==node_assignment(n)
        correct = correct+1;
    end;
end;
if verbose
    fprintf('\nCORRECT: %d     INCORRECT: %d     ACCURACY:
%2.2f%%\n',correct,g_nodecount-correct,100*correct/g_nodecount);
end;
```

```
c(2) = correct;
i(2) = g_nodecount-correct;
%% METHOD 3 Relative Confidence w/o Contention
if verbose
    fprintf('\n\nDiscovery Method 3 (w/o contention):\n');
end;
already_used = zeros(1,g_nodecount);
confidence_used = zeros(1,g_nodecount);
used_by = zeros(1,g_nodecount);
correct = 0;
for n=1:g_nodecount
    real_max_confidence=0;
    real_max_node=0;
    max_confidence=0;
    max_node=0;
    max_node_used=0;
    max_confidence_used=0;
    for i=1:g_nodecount
        if relative_conf(i,n) > max_confidence && already_used(i)==0
            max_confidence=relative_conf(i,n);
            max_node=i;
            if max_confidence > real_max_confidence
                real_max_confidence = max_confidence;
                real_max_node = max_node;
            end;
        elseif relative_conf(i,n) > max_confidence && already_used(i)~=0
            node_used = already_used(i);
            used_max_confidence = relative_conf(node_used,i);
            max_node_used=i;
            max_confidence_used = relative_conf(i,n);
            if relative_conf(i,n) > real_max_confidence
                real_max_confidence = relative_conf(i,n);
                real_max_node = i;
            end;
        end;
    end;
    confidence_used(max_node) = max_confidence;
    already_used(max_node) = n;
    if verbose
        fprintf('Assign Node %d as Node %d with confidence %3.2f, but contended for Node
%d with confidence
%3.2f\n',n,max_node,100*max_confidence,max_node_used,100*max_confidence_used);
    end;
    if n==max_node
        correct = correct+1;
    end;
end;

if verbose
    fprintf('\nCORRECT: %d     INCORRECT: %d     ACCURACY:
%2.2f%%\n',correct,g_nodecount-correct,100*correct/g_nodecount);
end;

c(3) = correct;
i(3) = g_nodecount-correct;

%% METHOD 4  w/ Contention %%
node_assignment = zeros(1,g_nodecount);
nodes_used = zeros(1,g_nodecount);
nodes_assigned = 0;
current_node = 1;
while nodes_assigned < g_nodecount
    if node_assignment(current_node) == 0
        order = 1;
        new_node = final_order(current_node,order);
        while nodes_used(new_node) ~= 0
            if relative_conf(nodes_used(new_node),new_node) <
relative_conf(current_node,new_node)
                node_assignment(nodes_used(new_node)) = 0;
                nodes_used(new_node) = 0;
                nodes_assigned = nodes_assigned-1;
            else
                order = order+1;
                new_node = final_order(current_node,order);
            end;
        end;
        node_assignment(current_node) = new_node;
        nodes_used(new_node) = current_node;
```

```
        if current_node == g_nodecount
            current_node = 1;
        else
            current_node = current_node+1;
        end;
        nodes_assigned = nodes_assigned+1;
    else
        if current_node == g_nodecount
            current_node=1;
        else
            current_node = current_node+1;
        end;
    end;
end;

correct=0;
if verbose
    fprintf('\n\n');
    fprintf('Discovery Method 4:\n');
end;
for n=1:g_nodecount
    if verbose
        fprintf('Assign Node %d as Node %d\n',n,node_assignment(n));
    end;
    if n==node_assignment(n)
        correct = correct+1;
    end;
end;
if verbose
    fprintf('\nCORRECT: %d     INCORRECT: %d     ACCURACY:
%2.2f%%\n',correct,g_nodecount-correct,100*correct/g_nodecount);
end;

c(4) = correct;
i(4) = g_nodecount-correct;

%% METHOD 5 MDE w/o Contention %%
%% d_err(i,j) is the error between the perceived distance of i and the
%% actual distance of j

assigned = zeros(1,g_nodecount);
node_assignment = zeros(1,g_nodecount);
for i=1:g_nodecount
    min_err = 99999;
    min_node = 0;
    for j=1:g_nodecount
        if d_err(i,j) < min_err
            if assigned(j) == 0
                min_err = d_err(i,j);
                min_node = j;
            end;
        end;
    end;
    assigned(min_node) = i;
    %% Assign i as min_node
    node_assignment(i) = min_node;
end;
correct = 0;
if verbose
    fprintf('\n\nDiscovery Method 5:\n');
end;
for n=1:g_nodecount
    if verbose
        fprintf('Assign Node %d as Node %d\n',n,assigned(n));
    end;
    if n==assigned(n)
        correct = correct+1;
    end;
end;
if verbose
    fprintf('\nCORRECT: %d     INCORRECT %d     ACCURACY: %2.2f%%\n',correct,g_nodecount-
correct,100*correct/g_nodecount);
end;

c(5) = correct;
i(5) = g_nodecount-correct;

%% METHOD 6 MDE w/ Contention %
for i=1:g_nodecount
```

```
    d_minErr(i,:) = minDSort(d_err(i,:));
end;

node_assignment = zeros(1,g_nodecount);
nodes_used = zeros(1,g_nodecount);
nodes_assigned = 0;
current_node = 1;
while nodes_assigned < g_nodecount
    if node_assignment(current_node) == 0
        order = 1;
        new_node = findNode(d_err(current_node,:),d_minErr(current_node,order));
        while nodes_used(new_node) ~= 0
            if d_err(current_node,new_node) < d_err(nodes_used(new_node),new_node)
                node_assignment(nodes_used(new_node)) = 0;
                nodes_used(new_node) = 0;
                nodes_assigned = nodes_assigned-1;
            else
                order = order+1;
                new_node =
findNode(d_err(current_node,:),d_minErr(current_node,order));%d_minErr(current_node,order
);
            end;
        end;
        node_assignment(current_node) = new_node;
        nodes_used(new_node) = current_node;
        if current_node == g_nodecount
            current_node = 1;
        else
            current_node = current_node+1;
        end;
        nodes_assigned = nodes_assigned+1;
    else
        if current_node == g_nodecount
            current_node=1;
        else
            current_node = current_node+1;
        end;
    end;
end;

correct = 0;
if verbose
    fprintf('\n\nDiscovery Method 6:\n');
end;
for n=1:g_nodecount
    if verbose
        fprintf('Assign Node %d as Node %d\n',n,node_assignment(n));
    end;
    if n==node_assignment(n)
        correct = correct+1;
    end;
end;
if verbose
    fprintf('\nCORRECT: %d     INCORRECT %d     ACCURACY: %2.2f%%\n',correct,g_nodecount-
correct,100*correct/g_nodecount);
end;

c(6) = correct;
i(6) = g_nodecount-correct;


%%%%%%%%%%%%%%%%%%%%%
% FUNCTION minDSort %
function sorted=minDSort(d)

for i=1:length(d)
    for j=i:length(d)
        if d(j) < d(i)
            d=swap_nodes(d,i,j);
        end;
    end;
end;

sorted=d;

%%%%%%%%%%%%%%%%%%%%%
% FUNCTION findNode %
function index=findNode(node_array,i)
for j=1:length(node_array)
```

```
    if node_array(j)==i
        index = j;
    end;
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION getConfidence %
function conf_array=getConfidence(node_array,dest,dact)
global g_nodecount;
total_error = 0;
for i=1:g_nodecount
    total_error = total_error+abs(dest-dact(i));
end;
for i=1:g_nodecount
    conf_array(i) = 1-abs(dest-dact(node_array(i)))/total_error;
end;


%%%%%%%%%%%%%%%%%%%%
% FUNCTION minSort %
function node_array=minSort(dest,dact)

global g_nodecount

for i=1:g_nodecount
    node_array(i) = i;
end;
for i=1:g_nodecount
    min_dist = abs(dest-dact(node_array(i)));
    min_node = node_array(i);
    for j=i+1:g_nodecount
        new_dist = abs(dest-dact(node_array(j)));
        if new_dist < min_dist
            min_dist = new_dist;
            min_node = node_array(j);
            node_array = swap_nodes(node_array,i,j);
        end;
    end;
end;

%%%%%%%%%%%%%%%%%%%%
% FUNCTION maxSort %
function node_array=maxSort(array)

for i=1:length(array)
    node_array(i) = i;
end;

for i=1:length(array)
    max = array(node_array(i));
    max_node = node_array(i);
    for j=i:length(array)
        if array(node_array(j)) > max
            max = array(node_array(j));
            max_node = node_array(j);
            node_array = swap_nodes(node_array,i,j);
        end;
    end;
end;


%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION swap_nodes %
function new_array=swap_nodes(old_array,n1,n2)

new_array = old_array;
new_array(n1) = old_array(n2);
new_array(n2) = old_array(n1);


% --- Executes on button press in doDiscovery.
function doDiscovery_Callback(hObject, eventdata, handles)
% hObject    handle to doDiscovery (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global g_basecount;

for bc=1:g_basecount
    [x(bc,:),y(bc,:)]=getActualRSSI(bc);
```

```
end;

doDiscovery(x,y,1);


% --- Executes on button press in addBase.
function addBase_Callback(hObject, eventdata, handles)
% hObject    handle to addBase (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global g_basecount;
global g_A;

[x,y]=ginput(1);
%g_basecount = g_basecount+1;
addBase(round(y),round(x),'New Base',g_basecount+1);
image(g_A);

% --- Executes on button press in graphError.
function graphError_Callback(hObject, eventdata, handles)
% hObject    handle to graphError (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ERROR_RATE

x = -2:0.01:2;
y = pdf(x,0,0.35);

figure(4);
plot(x,y);

x = 0:0.1:100;
y = ERROR_RATE*sqrt(0.35)*randn(1,length(x));

figure(5);
plot(x,y);


% --- Executes on button press in doSimulation.
function doSimulation_Callback(hObject, eventdata, handles)
% hObject    handle to doSimulation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global csv_data;
global nrows;

nrows=1;
csv_data = zeros(50,3);
for i=15
    if (i<10 || mod(i,5)==0)
        doTheSimulation(i,100);
        nrows = nrows+1;
    end;
end;

csvwrite('simulation_results.csv',csv_data);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION doTheSimulation %
function doTheSimulation(err,epochs)
global g_nodecount;
global g_nodecoords;
global g_basecount;
global g_basecoords;
global ERROR_RATE;
global NUM_EPOCHS;
global csv_data;
global nrows;

if err ~= -1
    ERROR_RATE = err;
end;

if epochs ~= -1
    NUM_EPOCHS = epochs;
end;

attempts = 1;
```

```
fprintf('***  STARTING SIMULATION  ***\n');
fprintf('     ERROR = %d\n',ERROR_RATE);
csv_data(nrows,1) = ERROR_RATE;
for i=1:attempts
        for b=1:g_basecount
         for n=1:g_nodecount
             d = getDistance(g_basecoords(b,:),g_nodecoords(n,:));
             x(b*(i-1)+b,n) = d;
             rssi = estDecay(d);
             rssi = rssi+ERROR_RATE*sqrt(0.35)*randn(1);
             if rssi < 0
                 rssi = 0;
             end;
             y(b*(i-1)+b,n) = rssi;
         end;
        end;
end;

% figure(6);
% hold on;
% types = {'go' 'cs' 'rd' 'mx' 'y^' 'b+' 'k*' 'gx' 'cd' 'r^' 'm+' 'y*' 'bs' 'ko'};
% for b=1:g_basecount
%     scatter(x(b,:),y(b,:),types{mod(b-1,length(types))+1});
% end;

[c,i]=doDiscovery(x,y,0);
correct = zeros(1,length(c));
incorrect = zeros(1,length(i));
for j=1:length(c)
    correct(j) = correct(j)+c(j);
    incorrect(j) = incorrect(j)+i(j);
end;
%fprintf('\tFINISHED EPOCH 1 \n');%\t[CORRECT: %d, INCORRECT: %d, ACCURACY:
%2.2f%%]\n',);
for epoch=2:NUM_EPOCHS
    for i=1:attempts
                for b=1:g_basecount
            for n=1:g_nodecount
                d = getDistance(g_basecoords(b,:),g_nodecoords(n,:));
                x(b*(i-1)+b,n) = d;
                rssi = estDecay(d);
                rssi = rssi+ERROR_RATE*sqrt(0.35)*randn(1);
                if rssi < 0
                    rssi = 0;
                end;
                y(b*(i-1)+b,n) = rssi;
            end;
               end;
    end;
    [c,i]=doDiscovery(x,y,0);
        for j=1:length(c)
         correct(j) = correct(j)+c(j);
         incorrect(j) = incorrect(j)+i(j);
        end;
%     if epoch < 10 || (epoch < 100 && mod(epoch,10) == 0) || mod(epoch,50) == 0
%         fprintf('\tFINISHED EPOCH %d\n',epoch);
%     end;
end;

fprintf('\n');
for i=1:length(correct)
    acc = 100*correct(i)/(g_nodecount*NUM_EPOCHS);
    fprintf('METHOD %d -->\tCORRECT: %d     ACCURACY: %2.2f%%',i,correct(i),acc);
    csv_data(nrows,i+1) = acc;
    fprintf('\n\n');
end;
fprintf('***  SIMULATION COMPLETE ***\n\n');


% --- Executes on button press in addMultBase.
function addMultBase_Callback(hObject, eventdata, handles)
% hObject    handle to addMultBase (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global g_basecount;
global g_A;

[x,y]=ginput;
```

```
for i=1:length(x)
    %g_basecount = g_basecount+1;
    addBase(round(y(i)),round(x(i)),'New Base',g_basecount+1);
    image(g_A);
end;


% --- Executes on button press in addNode.
function addNode_Callback(hObject, eventdata, handles)
% hObject    handle to addNode (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in addMultNodes.
function addMultNodes_Callback(hObject, eventdata, handles)
% hObject    handle to addMultNodes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global g_nodecount;
global g_A;

[x,y]=ginput;
for i=1:length(x)
    %g_nodecount = g_nodecount+1;
    addNode(round(y(i)),round(x(i)),'New Node',g_nodecount+1);
    image(g_A);
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION doTriangulation %
function doTriangulation(base1,base2,base3,node)

global g_A;
global g_nodecoords;
global g_basecoords;
global ERROR_RATE;
global g_img_h;
global g_img_w;
global likelihood;

ncoord = g_nodecoords(node,:)
bcoord1 = g_basecoords(base1,:);
bcoord2 = g_basecoords(base2,:);
bcoord3 = g_basecoords(base3,:);

d1 = getDistance(bcoord1,ncoord);
d2 = getDistance(bcoord2,ncoord);
d3 = getDistance(bcoord3,ncoord);

% rssi1 = estDecay(d1);
% rssi2 = estDecay(d2);
% rssi3 = estDecay(d3);
% rssi1 = rssi1+ERROR_RATE*sqrt(0.35)*randn(1);
% rssi2 = rssi2+ERROR_RATE*sqrt(0.35)*randn(1);
% rssi3 = rssi3+ERROR_RATE*sqrt(0.35)*randn(1);

[x rssi1] = getActualRSSI(1);
[x rssi2] = getActualRSSI(2);
[x rssi3] = getActualRSSI(3);

dest1 = estDist(rssi1(node));
dest2 = estDist(rssi2(node));
dest3 = estDist(rssi3(node));

likelihood = zeros(g_img_h,g_img_w);

drawCircle(bcoord1,dest1*2,1);
drawCircle(bcoord2,dest2*2,1);
drawCircle(bcoord3,dest3*2,1);

%drawCircle([2 1],16);

%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION drawCircle %
function drawCircle(center,radius,factor)

%  (y-y1)^2 + (x-x1)^2 = r^2
```

```
%  (y-y1)^2 = r^2 - (x-x1)^2
%  y-y1 = +/-(sqrt(r^2 - (x-x1)^2)
%  y = +/-(sqrt(r^2 - (x-x1)^2) + y1

global likelihood;

lb = -radius+center(2);
rb = radius+center(2);
ub = radius+center(1);
db = -radius+center(1);

x1 = lb:0.01:rb;
x2 = x1;

y1 = sqrt(radius.*radius-(x1-center(2)).*(x1-center(2)))+center(1);
y2 = -sqrt(radius.*radius-(x2-center(2)).*(x2-center(2)))+center(1);

%figure(6);
hold on;
plot(x1,y1);
plot(x2,y2);

figure(6);
xa = round(lb):1:round(rb);
xb = xa;
ya = round(sqrt(radius.*radius-(xa-center(2)).*(xa-center(2)))+center(1));
yb = round(-sqrt(radius.*radius-(xb-center(2)).*(xb-center(2)))+center(1));

for i=1:length(xa)
    likelihood(ya(i),xa(i)) = likelihood(ya(i),xa(i))+factor;
end;
for i=1:length(xb)
    likelihood(yb(i),xb(i)) = likelihood(yb(i),xb(i))+factor;
end;
image(likelihood);

%axis([lb-1 rb+1 db-1 ub+1]);
%axis([-radius radius -radius radius]);


% --- Executes on button press in doTriangulation.
function doTriangulation_Callback(hObject, eventdata, handles)
% hObject    handle to doTriangulation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
doTriangulation(1,2,3,13);


% --- Executes on button press in listDistances.
function listDistances_Callback(hObject, eventdata, handles)
% hObject    handle to listDistances (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global g_nodecount;
global g_basecount;
global g_nodecoords;
global g_basecoords;

fprintf('        ');
for n=1:g_nodecount
    fprintf(' %5d ',n);
end;
fprintf('\n');
for b=1:g_basecount
    fprintf('Base %d:  ',b);
    for n=1:g_nodecount
        fprintf(' %3.2f ',getDistance(g_basecoords(b,:),g_nodecoords(n,:)));
    end;
    fprintf('\n');
end;


% --- Executes on button press in addDefaultNodes.
function addDefaultNodes_Callback(hObject, eventdata, handles)
% hObject    handle to addDefaultNodes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
global g_img_h;
global g_img_w;
global g_FwN;
global g_A;

addNode(38,101,'Kitchen Table',1);
addNode(50,87,'Kitchen Counter',2);
addNode(33,129,'Barbs Cube',3);
addNode(42,173,'Matt Ps Cube',4);
addNode(42,190,'Justins Cube',5);
addNode(77,118,'Danielles Cube',6);
addNode(70,146,'Brandons Cube (b)',7);
addNode(70,160,'Brandons Cube (a)',8);
addNode(70,180,'Mansons Cube',9);
addNode(g_img_h-16,g_img_w-156,'Lg Conf Rm',10);
addNode(g_img_h-27,g_img_w-128,'Lg Print Rm',11);
addNode(g_img_h-19,g_img_w-101,'Sm Conf Rm',12);
addNode(g_img_h-16,g_img_w-71,'Rons Office',13);
addNode(g_img_h-3,g_img_w-13,'Roberts Cube',14);
addNode(g_img_h-3,g_img_w-33,'Jesses Cube',15);
addNode(g_img_h-28,g_img_w-11,'Matt Ds Cube',16);
addNode(g_img_h-40,g_img_w-35,'Adams Cube',17);

addBase(64,g_img_w-6,'Base A',1);
addBase(33,157,'Base B',2);
addBase(g_img_h-1-33,g_img_w-1-75,'Base C',3);

g_FwN = g_A;
image(g_A);
```

Appendix B

Other Figures and Simulation Results

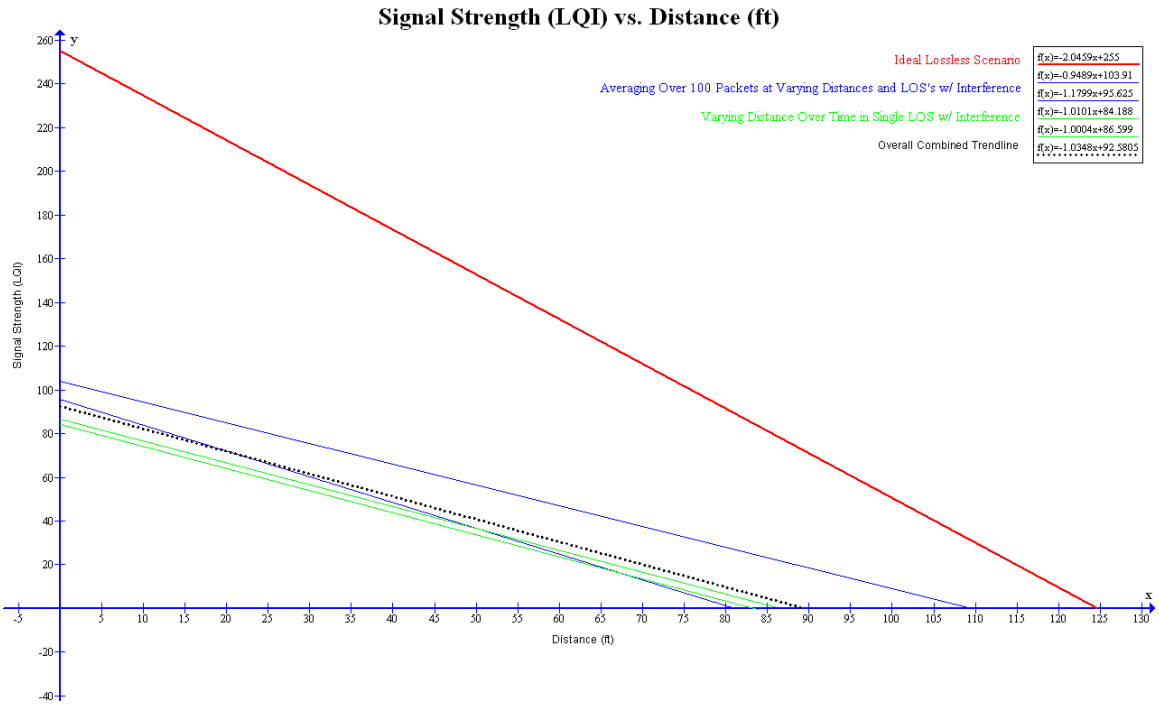

Fig B-1: Linear modeling of LQI versus Distance

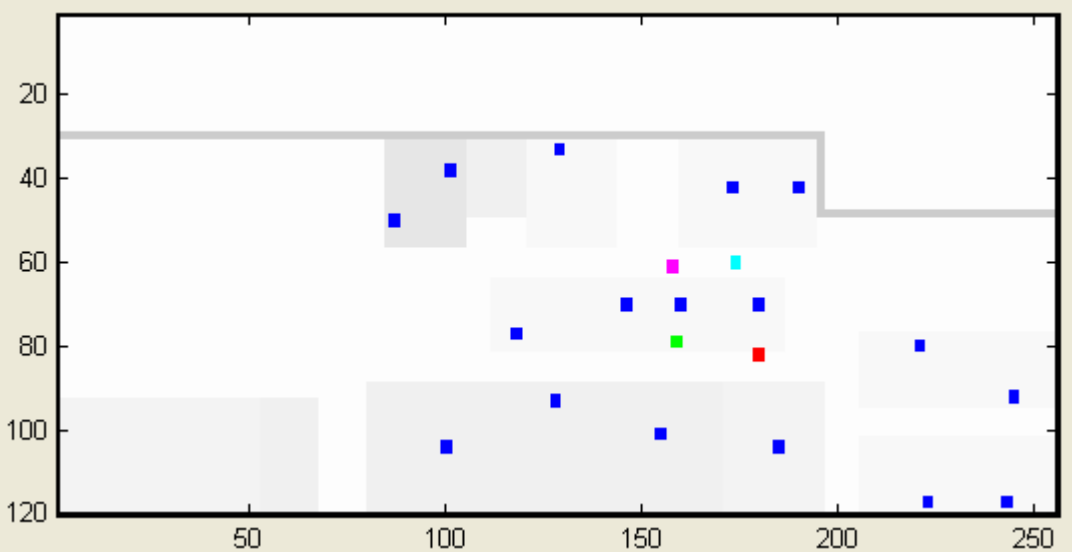

Fig B-2: Example 4-base (centrally-located) node layout in GUI

(Any customizable layout was possible for any combination of nodes and bases)

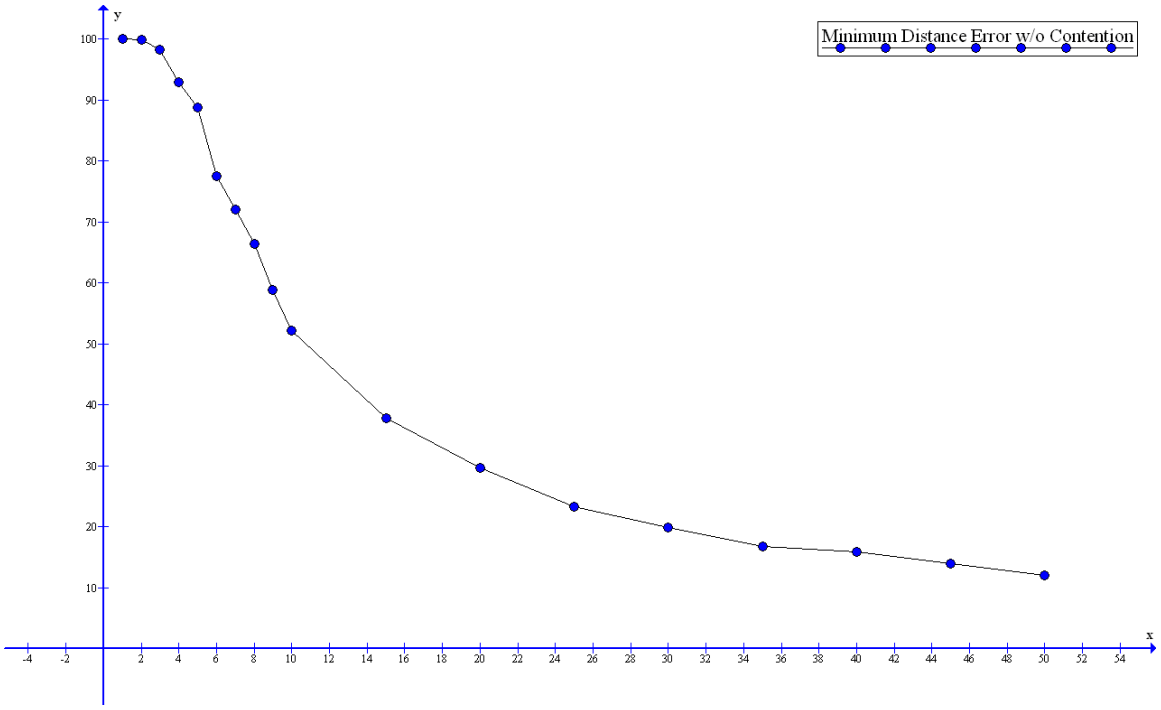

Fig B-3: Individual simulation result for MDE w/o contention

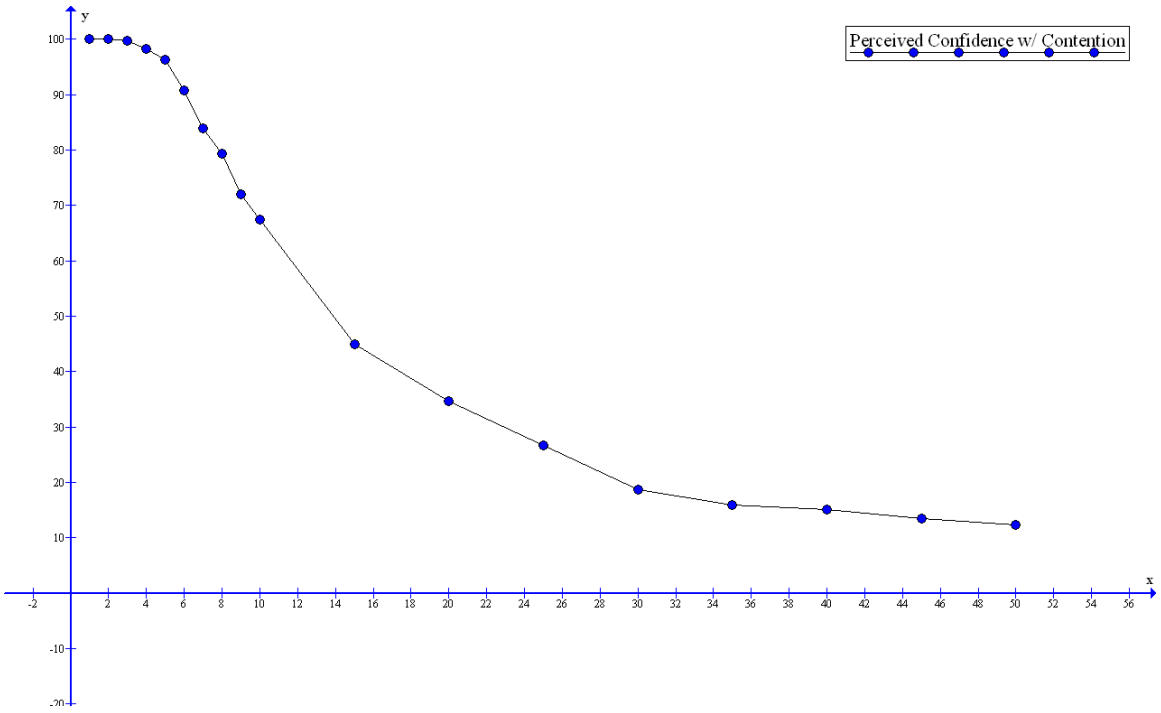

Fig B-4: Individual simulation result for perceived confidence with contention

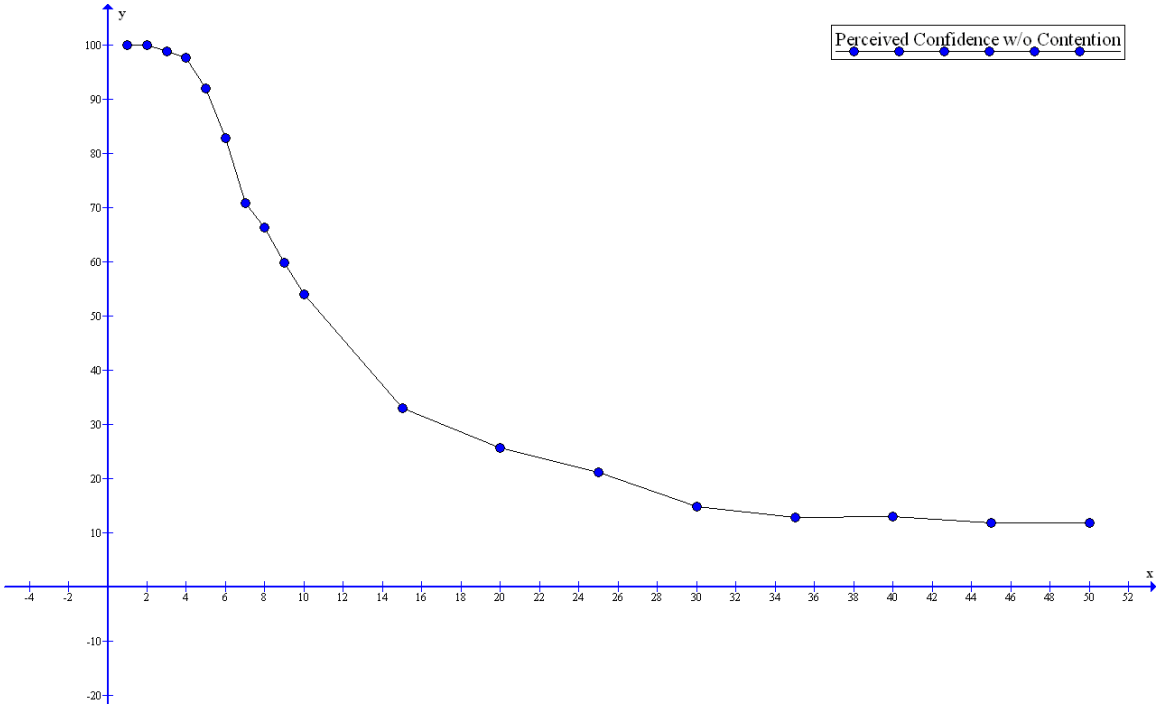

Fig B-5: Individual simulation result for perceived confidence without contention

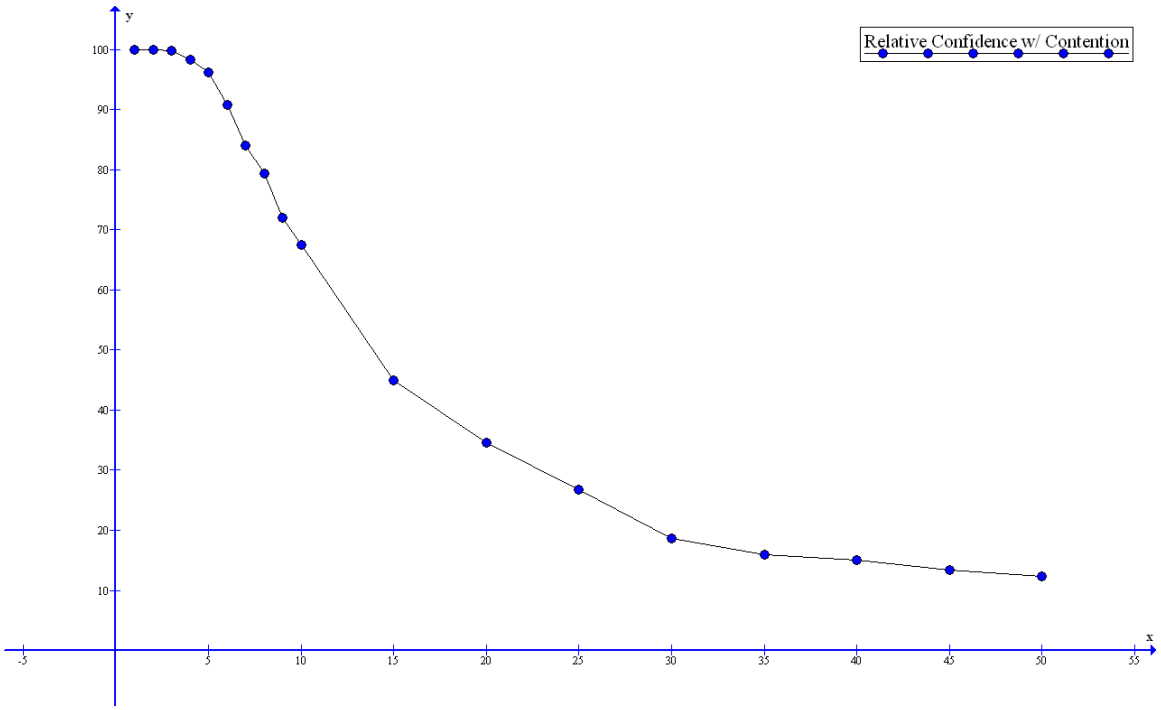

Fig B-6: Individual simulation result for relative confidence with contention

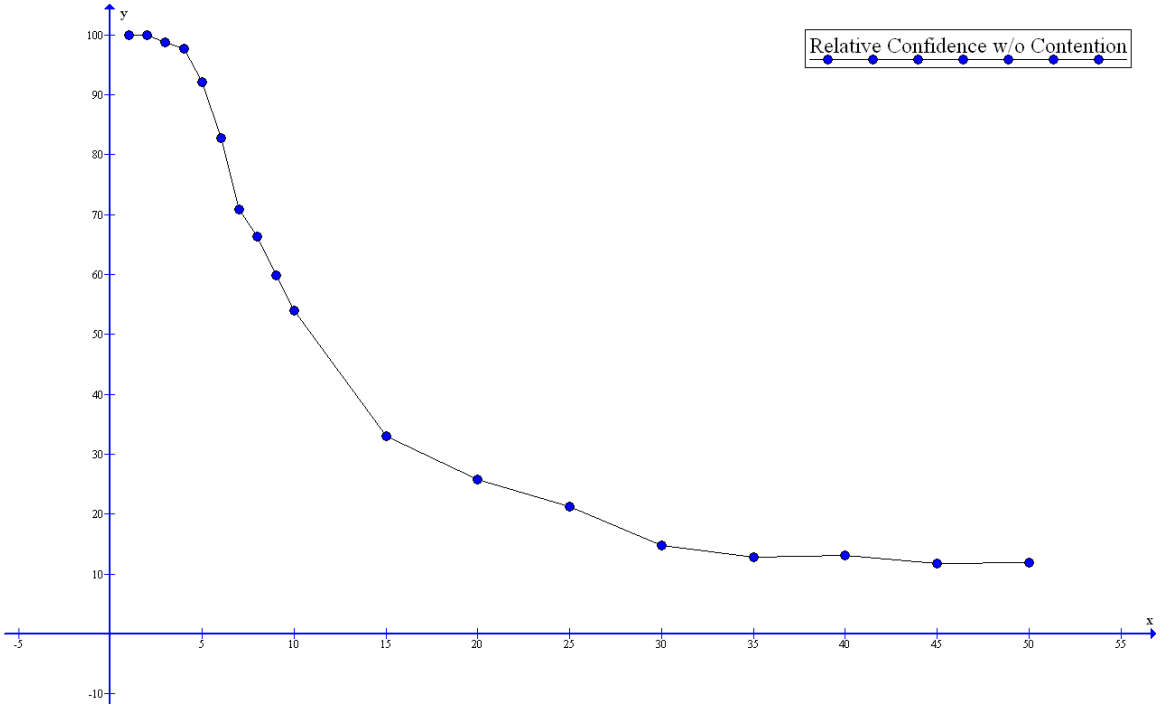

Fig B-7: Individual simulation result for relative confidence without contention

Vita

Brandon Jeremy Rogers was born in Memphis, TN on February 28, 1983 and grew up in Tupelo, MS. He is the son of Fred and Nancy Rogers. He graduated high school with honors from Collierville High School in May 2001 in Collierville, TN, where he received several scholarships for academics as well as art. He then attended the University of Tennessee in Knoxville, TN, pursuing an undergraduate degree in Computer Engineering. During this time he worked as an intern for two summers with FedEx at their world headquarters in Memphis, TN. He completed his undergraduate coursework and graduated *cum laude* with a Bachelor of Science degree in Computer Engineering in May 2005. He then decided to remain at the University of Tennessee in Knoxville, TN, to pursue his graduate degree. He has spent his time as a graduate student as the head advisor for the University of Tennessee's local chapter of Alpha Phi Omega, Iota Alpha, a national co-ed service fraternity. During his tenure as a graduate student he has also worked as an intern with Eaton Electrical and has since then accepted a full time position with Eaton's local office in Louisville, TN. He graduated in August 2007 with his Master of Science degree in Electrical Engineering with a concentration in Computer Engineering.