Project Number: EOA-0003

LOCUS: WIRELESS LAN LOCATION SENSING

A Major Qualifying Project Report

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

____
Arvinder Singh

____
Ali Taheri

Date: January 19, 2004

Approved:

1. computers
2. wireless LAN
3. location sensing

____
Professor Emmanuel O. Agu, Major Advisor

Abstract

The purpose of this Major Qualifying Project was to create a platform independent, software-only, indoor geolocation solution. Locus utilizes the existing 802.11b Wireless Local Area Network infrastructure without using the Global Positioning System (GPS) or proprietary tags. Locus was designed using object-oriented techniques and implemented in Java with platform dependent modules clearly abstracted. The proximity-based location sensing algorithm compares runtime signal strength values to pre-recorded calibration values. Location is displayed on an actual floor plan using Scalable Vector Graphics (SVG).

Acknowledgments

We would like to give special thanks to our advisor Professor Emmanuel Agu, of the WPI Computer Science department, for his guidance, encouragement, and patience. His dedication and enthusiasm for Locus was instrumental to its success.

The following individuals have generously contributed their time and resources in support of Locus. Dr. Xinrong Lee of the WPI Electrical and Computer Engineering department first introduced us to the WRAPI project and on multiple occasions discussed different geolocation approaches with us. Dr. Anand Balachandran, of Intel Research and the University California San Diego Wireless Research API Project, provided crucial support in the development of the WRAPI portions of the project. Kutty Banerjee, of the WPI Computer Science department provided us with guidance in the area of Windows systems programming and debugging. Sean O'Connor, Manager of Network Operations at WPI, and Joseph Krzeszewski, also of Network Operations, provided guidance with understanding the intricacies of the WPI Wireless Network. John Miller, Director of WPI Plant Services, supplied us with electronic copies of WPI floor plans which were incorporated into the graphics subsystem.

In addition, Ali would like to give thanks to his parents Dr. Nasrollah Taheri and Dr. Simin Shafii for their support of his interests in this project, computer science and WPI. Arvinder would like to express his appreciation to his parents Brijpal Singh and Perminder Kaur for their support of his endeavors at WPI.

Table of Contents

List of Figures

List of Tables

1 Introduction

The twentieth century brought us the computer revolution and forever transformed our lives. Through the rapid advancement of technology computers became faster, more versatile, and much smaller. At the end of the last century we saw the birth of the information super highway, where computers became increasingly interconnected through the global network that we know as the Internet. At the dawn of the twenty first century we are embarking on a whole new revolution made possible by the computer revolution of the twentieth century. This revolution will result in a truly ubiquitous and wireless network where computers, every day devices, and humans all work seamlessly with out the physical limitations of the early wired networks. This Major Qualifying Project (MQP) aspires to make its contribution to this new wireless world.

1.1 Project Goals

This MQP proposes to create a software only solution for determining the physical location of a mobile device on a Wireless Local Area Network (WLAN) by using only the existing WLAN infrastructure and no additional hardware. In other words if you have a Notebook PC or a Personal Digital Assistant (PDA) your physical location can be determined without the need to install any additional equipment. This approach is different from other proposed solutions which require attaching a tag to the mobile devices. The main objectives of the project are to demystify the technology behind location sensing and to develop modular software that will encompass two general modules: a Location Sensing Module (LSM) and a Graphical Display Module (GDM). By completing this project and developing the software we hope to encourage further research and commercial development in this area. In the next section, “Vision and Usage Scenarios”, we discuss some of the envisaged possibilities.

1.2 Vision and Scenario

A WLAN provides the ability to extend the services of a Local Area Network without the need to run wires. In the past few years there has been a rapid growth of WLANs. Worcester Polytechnic Institute (WPI) is just one of the many educational, commercial, governmental, and even personal groups that have created such networks. The most significant advantage of wireless networks over traditional networks is that the network is brought to the user rather than the user going to the network and having to "plug in". The WLAN gives users the ability to access the network resources anywhere within their site. Using the WLAN infrastructure we can create some new technologies that were previously not possible. Consider the following three usage scenarios.

Scenario 1:

John Smith is a new freshman a WPI. Before coming to school he learns about the extensive coverage of the wireless network on campus. He decides to purchase a PDA equipped with a Wireless network card. This year, WPI has decided to adopt the WLAN location sensing technology called Locus, created by former students Ali Taheri and Arvinder Singh. John's first class assigns a chapter to be read from a book in the WPI Gordon Library. John does not know his way around WPI yet, so he decides to download the Locus software from the web. After loading the software on his PDA, John is presented with a map of the WPI campus with a blinking dot. This dot indicates John's current location on campus. From a list of locations he selects the library and the software displays the directions on how to get to the library. At the library he uses the wireless web browser on his PDA to look up the book that he is looking for. The library has recently integrated its database with an implementation of the Locus. When John finds the book in the library database he is also able to see on his map where in the library the book is located and how to get there.

Scenario 2:

Nearby at the newly built Goddard Convention Center in downtown Worcester Jane Johnson arrives for the annual International Conference on Mobile Computing and Networking (MobiCom). The convention center uses state of the art technology including wireless access through out the convention halls and the conference rooms.

When Jane arrives at the convention center she uses her PDA to download a map of the center for use with the Locus software. This convention center uses centralized software for managing the location of attendees. Jane, like many of the attendees, has previously chosen to register with the service. While at the convention center she selects the name of another attendee from a list and the software on her PDA displays the current location of Joe Jackson, and how Jane can get from the reception hall to the 3rd floor conference room where Joe is located.

Scenario 3:

Back at WPI a group of students are working on their Major Qualifying Project and have recently completed a project proposal for Professor Emmanuel Agu in the Computer Science Department. They are proposing to create a new software module that will integrate Locus with various Instant Messaging (IM) programs. An IM user will download and install the module and be able to configure it as part of his or her profile. A typical IM profile currently provides other IM users information such as telephone numbers and personal interests; with this new module the users will be able to advertise their physical location and proximity to landmarks, to people on their contact list.

2 Literature Review

In this section we review the current literature and related work in the areas of networks, location sensing or geolocation, and computer graphics. Through examination of various online resources, journals, and texts, we have attained the necessary background information for the rest of this MQP.

2.1 Introduction to Wireless networks

Wireless networking is currently the fastest growing technology in communications and computing. The past decade has seen new technologies develop, ranging from digital cellular phones to WLANs. New protocols and standards are constantly being developed, making the network more efficient and secure. Even the devices used on such networks are getting increasingly 'smart'. Newer mobile devices are smaller but more powerful and offer a range of flexible features with some technologies often overlapping. Some new cellular phones, for instance, are flexible enough to be used as mobile computers, PDAs and GPS Receivers with applications ranging from telecommunications and wireless internet access to location sensing. The following sections provide a brief overview of some of these wireless technologies.

2.1.1 Cellular Networks

Cellular networks have fast developed into an extensive wireless communication infrastructure providing wireless voice and data communications with almost world-wide coverage. Use of cellular phones is greatly increasing worldwide, and the number of cell phone subscribers has quadrupled to over half a billion in the past five years [STO02].

A cellular network is a wireless communication service area subdivided into hexagonal areas termed as cells. These cells vary in size from a few kilometers in diameter, in modern digital networks, to around a hundred kilometers in older analog networks. Each of these cells has a base station (cellular tower) associated with it. Figure 1 shows a simplified cellular architecture.

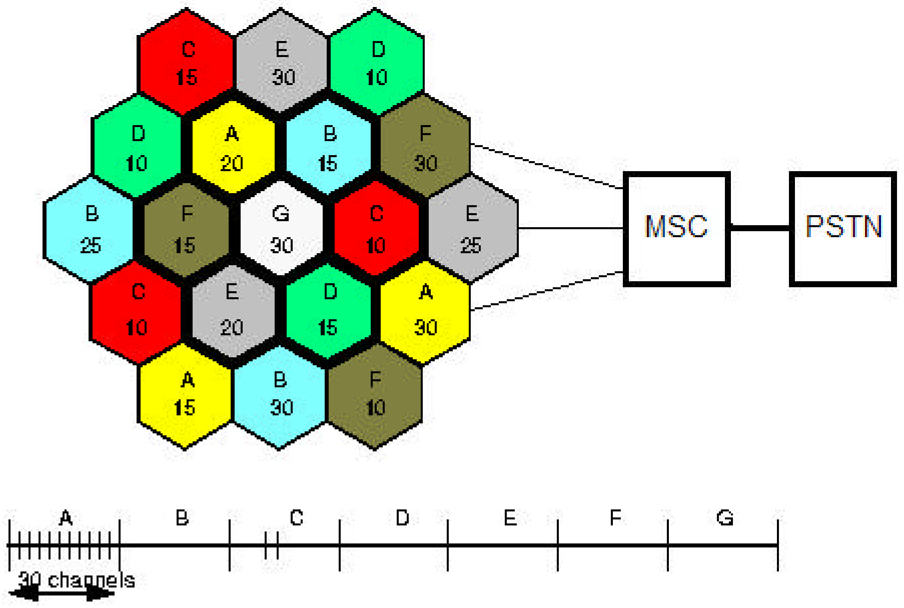

Figure 1: Cellular Network

Each base station has a certain range of radio frequency channels associated with it. To avoid overlap and radio interference, these channels are different from channels associated with all of its neighboring cells. Channels can be reused in cells that are far enough so that no interference occurs. These cells are grouped together as clusters for a required coverage area.

All the Base Stations (BS) are connected to a Mobile Switching Center (MSC) using fixed links. Each MSC of a cluster is then connected to the MSCs of other clusters and a Public Switched Telephone Network (PSTN) switching centre. The MSC stores information about the subscribers located within the cluster and is responsible for directing calls to them.

One of the most important issues in cellular networks is tracking of a mobile client when it is moving through a network. To counter this, cellular networks use Location Management techniques. Location Management essentially involves two processes, location update and paging. Location update is the information provided by the mobile device to the network about its current location. Paging, on the other hand, is done by the network where it actively queries the mobile device to find out what cell it is located in, so that the incoming call can be routed correctly to the appropriate BS.

The performance of a network greatly depends on the Multiple Access Control (MAC) techniques used for sharing the available frequencies among multiple subscribers. The three major MAC techniques in use currently are Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA) [PAN99].

FDMA divides the available channel bandwidth into equal sub-channels among the mobile users. Each user is assigned a sub-channel for transmission. Interference between these channels is reduced by inserting guard bands between adjacent sub-channels. This technology is used by most analog cellular services.

TDMA works on a similar logic to that of FDMA, but it also divides each sub-channel into a number of time slots in order to increase the amount of data that can be carried. A mobile user is assigned the entire channel for a period of time. In this case guard bands are required in between frequency channels and time slots as well. GSM (Global System for Mobile Communications) and the North American Digital Standard IS-136 both work on the principles of TDMA.

CDMA uses a spread spectrum technique to scatter a radio signal across a wide range of frequencies. The frequency channel (1.23 MHz in IS-95) is shared between all the users of the system. A signal to be broadcast on the channel is first spread-out over the entire bandwidth using a unique PN code. Since each signal on the channel is then unique, it can be distinguished from the other signals and can be recovered at the receiving end using the unique PN code [PAN99].

2.1.2 Global Positioning Satellite Network (GPS)

Global Positioning System (GPS) is a satellite based radio navigation system that provides accurate three-dimensional positioning information and velocity readings. It was developed by the United Stated Department of Defense for military use, but it is now widely adopted for civilians use.

The GPS location sensing technique is based on a constellation of 24 satellites in six orbital planes, 21 of which are operational and 3 are backups. These satellites orbit the earth every 12 hours at altitudes of approximately 10,900 nautical miles. The configuration of these satellites ensures that at any given time there are at least four satellites above the horizon from any given point on the Earth. With more satellites a GPS receiver can more accurately calculate its location. On average, there are eight satellites visible from a point on the earth at a given time which leads to fairly accurate calculations.

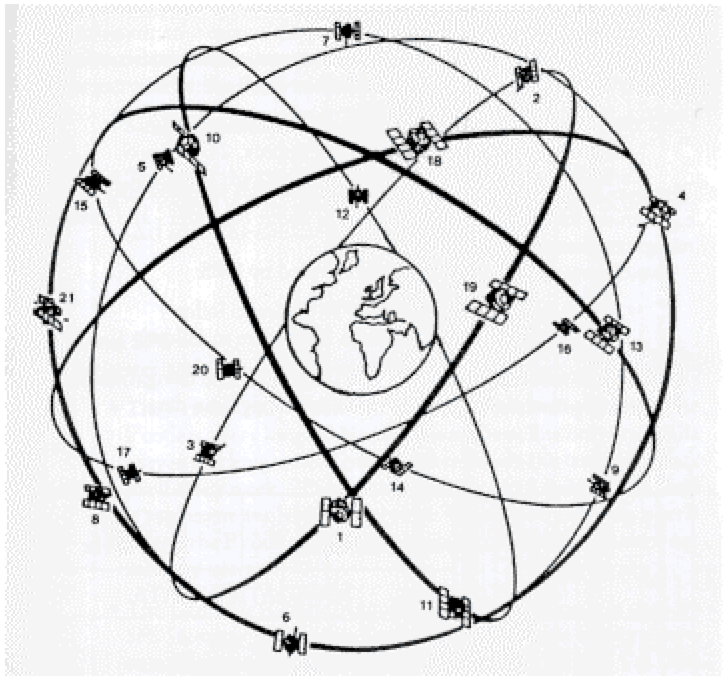

Figure 2: GPS Constellation [ESOweb]

The satellites transmit high-frequency radio signals containing precise time and distance data that can be picked up by any GPS receiver. The receiver's exact location can be identified using a technique called triangulation, which is further explained in Section 2.4.4. Thus, the three main parts of a GPS system can be identified as the satellites that emit the signal, the GPS receiver that detects the signal, and the software that decodes the signals from the satellites and displays it on a GPS device. A good GPS receiver can calculate location with an accuracy of 10 feet.

GPS devices have a variety of applications. The devices can be divided into four categories: handheld, automobile, marine and aircraft. Marine and aircraft based GPS devices are used for accurate navigation by commercial, military and recreational aircrafts and sea vessels. Such devices are generally larger in size and more advanced than smaller and cheaper handheld devices. Land-based applications are more diverse, ranging from small handhelds to fixed automobile based systems. GPS devices are increasingly common feature in today's luxury automobiles and are capable of providing services such as real time driving directions and better emergency roadside assistance. Handheld devices are the most versatile, providing excellent portability and decent accuracy at an affordable price. Apart from standalone devices that are specifically designed for GPS add-on GPS kits allow GPS capabilities to be added to PDA's and mobile computers.

2.1.3 Wireless LANs

After Cellular networks and GPS, Local Area Networks (LAN) are the next common type of networks that we will focus on. When two computers are connected together they form a network of computers, similar to any other network such as GPS satellites or cellular phones. A LAN can be composed of anywhere from just a few to several hundred computers connected together by physical Ethernet cables. The motivation for the growth of LANs started in the 1970s to make it possible to share expensive resources such as printers [PAL02]. By connecting all the computer terminals in an office to the LAN, and then connecting one printer to the LAN, all the terminals could share one printer. Eventually LANs were connected to other LANs; such as a home office and a remote office. Networks of geographically distributed LANs are also known as Wide Area Networks (WAN). The most familiar WAN is the Internet which interconnects computers and LANs world wide.

A Wireless LAN (WLAN) is a network in which the medium for connecting nodes or computers is wireless. A WLAN is conceptually very similar to both a cellular network and a LAN. The major differences between cellular and WLAN are in how they are implemented and include "the method of delivery of data to users, data rate limitations and frequency band regulations" [PAL02]. While a cellular network was designed to serve similar functions to that of traditional telephone networks, a WLAN was designed to serve the functions of a Wired LAN. The breakthrough in both of these networks is that devices on the network do not need to be connected together by a physical wire in order to communicate and instead use radio waves as the communication medium. Figure 3: Typical LAN and WLAN Configuration relates WLANs, LANs, and clients. One possible WAN configuration might connect two or more of these LAN/WLANs that are separated by a long distance.

WLANs have matured significantly since they were first introduced at the IBM Rüschlikon Laboratories in Switzerland the late 1970s [PAL02]. The original motivation behind these networks was to reduce the cost and complexity involved in laying down

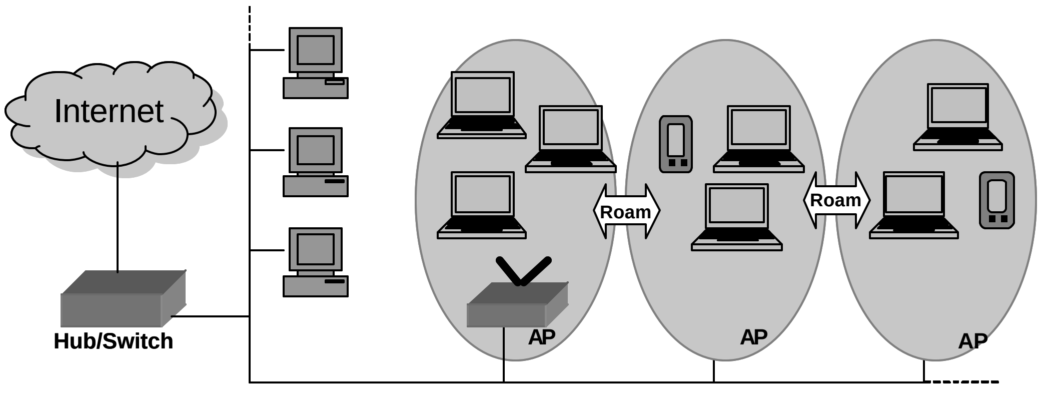

Figure 3: Typical LAN and WLAN Configuration

wires. Though that is still a major part of choosing to deploy a wireless network, more and more it is about the convenience it that provides to the user and the new applications that have emerged through the availability of WLANs. One particular benefit can be seen when setting up a network in a historical building. In such building the physical impact of creating a network is minimized by not having to lay down LAN cables. The benefits also lie in the multitude of devices can be connected to WLANs which include traditional PCs, notebook computers, PDAs, and appliances such as televisions and stereo systems. There are two main categories of WLANs: infrastructure networks in which there is a backbone and ad hoc networks with no backbone. These different networks could still be further interconnected and provide a truly ubiquitous network.

This chapter gave a general overview of how WLANs work and their relationship to some of the other networking technologies. In the next chapter we take a closer look at the underlying components of WLANs to understand how some of the new applications have emerged and how new applications such as location sensing can be developed.

2.1.4 Other Wireless Technologies

"Wireless means more than just 'wire-less'" [WEA02]. This seemingly strange statement points to the fact that wireless technologies are not merely the result of replacing the wires with radio waves; they are really about changes in the way the network is used, and its impact. What is significant is the "ability to communicate information between two devices, without requiring supporting infrastructure" [WEA02]. Such ability is provided in networks such as Bluetooth, which was designed to connect personal devices within a short range of each other. The universal remote control and the various appliances it controls such as the television, cable box, and stereo, also form a wireless network. However, such networks are limited because in order to establish a connection using infrared, a line of sight must be established between the two devices. One other interesting application of wireless networks is the Prototype Embedded Network under development at Cambridge University. In this type of a network tiny low power and low cost networking devices are attached to everything from light switches to temperature sensors. This new miniature network allows such applications as home automations and laboratory monitoring.

2.2 Overview of Wireless LANs

In the previous chapter we discussed various types of wireless networks and provided an overview of how they operate and the applications that they make possible. In this section we will take a more in depth look at WLANs in preparation for discussing location sensing methodologies in the following section as well as the system design presented in the next chapter.

2.2.1 Networking Layers

WLANs are a specific type of a computer network and follow the same networking principles as do LANs and WANs. The standard networking model used is the TCP/IP Reference Model. This model establishes a framework consisting of five layers. For data to go from one application on one machine to another application on another machine, it must travel down and then up the five layers. The full extent of how the TCP/IP model is organized is a topic for textbooks and networking courses. In this section we briefly discuss each of the five network layers drawing upon general knowledge gained in various WPI Computer Science courses, Networks text books and some World Wide Web resources.

These layers can be thought of as a hierarchical system where for a service to be offered at a given level, a service has to be provided at the level below it.

Layer 5: Application

This layer as the name suggests is application specific. In this layer the end users of a particular application such as email or web browsing are identified. Also other abstract notions (when compared to concrete ideas such as physical cables) of the application are considered at this layer. For email this includes authentication and privacy; the user might use a username and password for identification to limit access to an email account and keep contents private.

Layer 4: Transport

Once a session is established between two applications some data will presumably be transferred between two end systems. The data transfer is handled by the Transport Layer, where end-to-end error checking and error recovery is conducted to ensure a reliable transmission. After establishing the session our email program downloads new emails through the transport layer by retrieving them from the server and storing them locally in the email client.

Layer 3: Network Layer

This layer handles data routing and the path the data takes when traveling through the network. A common method for transferring data through a network is to break up the data into small chunks called packets. These packets are individually sent through the network and when they arrive at the destination, they are reassembled into the original email or picture. To go from the source to the destination the data must travel through several other machines and devices connected to the network. This layer is responsible for the proper routing of this data over the network or the Internet and has to take care of correct ordering of the routed packets. The layer uses IP (Internet Protocol) for its official packet format and protocol.

Layer 2: Data Link Layer

The data link layer is very closely tied to next layer, the physical layer. Before the data packets can physically travel over the airwaves or the network cables they must be encoded and decoded into zeros and ones (bits). A protocol establishes the encoding scheme and how the bits are to be managed before being sent over the physical layer. The data link layer also decides how multiple nodes on the same local area network share the medium.

Layer 1: Physical Layer

At the bottom of the TCP/IP model is the Physical Layer. In this layer the data is physically moved (as electrical signals) through cables or airwaves. To send the bits through a cable the zeros and ones have a corresponding relationship with high and low levels of voltage. By controlling voltage levels through the cable we send a zero or a one bit.

From the application developer's point of view there are no differences between a wired and wireless network. For the purpose location sensing the difference in the Physical Layer is very important. In the following sections we will look at the various components of a wireless network and issues with the Physical Layer.

2.2.2 Wireless Local Area Network (WLAN) Architecture

The basic components of the WLAN are access points (AP) and the mobile clients (MC), typically a laptop or a PDA with a WLAN card. To create a wired network infrastructure, Ethernet cables are placed through out the building and then buildings are connected together using fiber optic cables. With a Wireless LAN, in order to create the network infrastructure APs are placed in various locations throughout a building and even outdoors. Various mobile clients then communicate with each other by first communicating with these access points.

In the simplest configuration there is one AP at the center and one or more MCs spread out around the AP. When additional APs are added the coverage area of the network increases and the MC selects to closest AP to communicate with. The entire WLAN could consist solely of APs and MCs but it is common to find APs connected to other APs by Ethernet cable, and the network of APs then connected to a LAN or the internet through other networking devices. Such an arrangement is especially beneficial if a wired network is already deployed at a site. APs can be placed at locations far from each other where they provide local coverage, but the MCs in each local coverage area can still communicate with each other since the APs are connected to the wired network. WPI has taken this approach as it expands its network from local coverage areas in the campus center and the library to the entire campus.

2.2.3 Wireless LAN Standards: IEEE 802.11 and HIPERLAN

The Physical and Data Link Layers of a wireless network must be standardized in order for devices to be able to communicate with each other. When a mobile client talks to an access point, both devices must operate on the same frequency so that they can receive each other's transmissions and both must also use a previously agreed upon bit encoding scheme so that they can "understand" each other. WLAN standardization centers around two sets of protocols called the IEEE (Institute of Electrical and Electronics Engineers) 802.11 and the ETSI HIPERLAN-2. The differences lie in the implementation of the sub layer of the Data Link layers called the MAC layer [PAL02].

The IEEE 802.11 WLAN standard evolved from data-oriented computer communications and is considered connectionless, meaning the services that can take advantage of it are not defined as part of the standard. Everyday services such as the web and email were not defined when 802.11 was standardized; however, they and other services use 802.11. This is in contrast to the HIPERLAN-2 standard which is a connection-based WLAN protocol that is closer in nature to cellular telephone networks. In a connection-based protocol the services that are available are defined as part of the protocol [PAL02]. The cellular telephone network defines frequencies and device types but it also states that a voice connection is made when a number is dialed allowing a two way conversation.

There are several specifications in the 802.11 family: 802.11, 802.11a, 802.11b, and 802.11g [WPEDweb]. The original 802.11 specification provides 1 or 2 Mbps data transfer rates and operates in the 2.4GHz band. 802.11a is one of the three extensions to 802.11; it provides data transfer rates of up to 54 Mbps and operates in the 5GHz band. The 802.11b extension, also referred to as Wi-Fi, is currently the most widely adopted WLAN standard; it provides data rates of 1, 2, 5.5, and 11 Mbps and operates in the 2.4GHz band. The WPI WLAN is currently based on this standard. The newest of the extensions is the 802.11g which provides data rates of 20+ Mbps and also operates in the 2.4GHz band. This standard is also backwards compatible with the 802.11b standard.

HIgh PErformance Radio LAN (HIPERLAN) is a European standard for high-speed wireless local networks [PAL02]. The first incarnation of this standard is the HIPERLAN-1 which operates in the 5 GHz band and supports data transfer rates of up to 23 Mbps. The HIPERLAN-2 also operates in the 5GHz band and supports data transfer rates of upto 25Mbps. Unlike the first HIPERLAN, the work on transmission techniques for HIPERLAN-2 was coordinated with the work on the IEEE 802.11. There is also work being done on HIPERLAN-2 that will integrate WLANs into next-generation cellular systems.

2.2.4 Mobile Devices

The work done on standardizing the 802.11 and HIPERLAN protocols enables us to use a wide range of devices and applications to communicate with the network; as long the particular device understands the protocol(s) used by the network. Simply put the WLAN is a hardware and software independent communications medium.

The two common hardware platforms for connecting to a WLAN are notebook computers and PDAs. However, we are not limited to these traditional devices and devices such as television recorders and digital picture frames are just two emerging technologies that can take advantage of WLANs. Notebook computers and PDAs are controlled by an operating system (OS) that is typically specific to that family of device. As long as both the device and the OS conform to the protocol, then any application that runs on top of the OS can communicate with the WLAN.

2.3 WPI Network

Worcester Polytechnic Institute (WPI) has an extensive data network and has been consistently ranked among America's most wired colleges. The University's network consists of various computers in labs, offices, dormitories and fraternities, and also other devices such as file and print servers and network printers. Recent additions to the network are devices that make us of the expanding WLAN. The major applications that use the network include file transfers, e-mail delivery, Internet access, software accessibility, and remote access to network resources.

The WPI network covers the main campus in Worcester and the satellite campuses at 60 Prescott St., 85 Prescott St. (Mass Academy), Waltham and Southborough. The network on the main campus comprises of a Gigabit Ethernet backbone connecting 27 academic buildings, 4 satellite campuses, and 36 dorms and fraternities. The WPI network's Internet connection, is via a 45 Mbps T3 line. The network also has a 155 Mbps OC3 link to the Internet2 Abilene network. A detailed explanation of the networking technology used is available on the WPI NetOps webpage (http://www.wpi.edu/+netops). In total, WPI Network Operations manages approximately a total of 8000 hosts and maintains approximately 2.75 million feet of copper cable and 700,000 feet of fiber [WPINOweb].

2.3.1 WPI WLAN Overview

The past few years WPI has greatly expanded its wireless network. There are currently approximately 60 Access Points throughout campus, with the number soon expected to reach about 160. There are two types of APs presently used: Symbol 4121 and Symbol 4131. WPI is introducing a new AP, the Symbol Mobius Axon which will primarily serve the dormitories. The WPI Wireless LAN uses the 802.11b standard [OCO03].

Examining the APs from a software development point of view, the 4121 and 4131 provide a consistent interface for extracting information from them. The software interface for the Mobius APs is still under development, since it is a relatively new product. Also, the design of the Mobius Axon is such that it uses multiple remote antennas connected to a central access point. Since these antennas could be located at a considerable distance from each other, it is difficult to identify which antenna a mobile client is communicating with.

2.3.2 WPI WLAN Availability

Most buildings on WPI's main campus are covered by the WPI Wireless LAN. Figure 4 highlights the areas on campus with wireless coverage. All buildings with wireless access are shown in red. The following list [WPINOweb] associates the areas shown on the map with the numbers in parentheses (corresponding to the numbers on the map).

- Gordon Library - Basement, 1st, 2nd, and 3rd Floors. (13)
- Campus Center - All floors as well as the front and back patios. (6)
- Freeman Plaza / Reunion Plaza (Fountain) area of West Street.
- Olin Hall 107 Classroom. (22)
- Higgins Labs - 1st, 2nd, and 3rd Floors including the Discovery Classroom. (17)
- Fuller Labs - Basement, 1st, 2nd, and 3rd Floors including Perreault Hall. (11)
- Atwater Kent - 1st, 2nd, and 3rd Floors including Newell Hall. (4)
- Alden Hall - Great Hall. (2)
- Harrington Auditorium – Basketball court and stands. (14)
- Morgan Hall - Dining Hall, 1st, 2nd, 3rd, and 4th Floors. (H)
- Founders Hall - Dining Hall, 1st, 2nd, 3rd, and 4th Floors. (D)
- Daniels Hall - 1st, 2nd, 3rd, and 4th Floors. (A)
- Riley Hall - 1st, 2nd, 3rd, and 4th Floors. (I)
- Washburn Labs - Machine Shops & Robotics Lab. (30)
- Stratton Hall - 1st and 2nd Floors. (28)

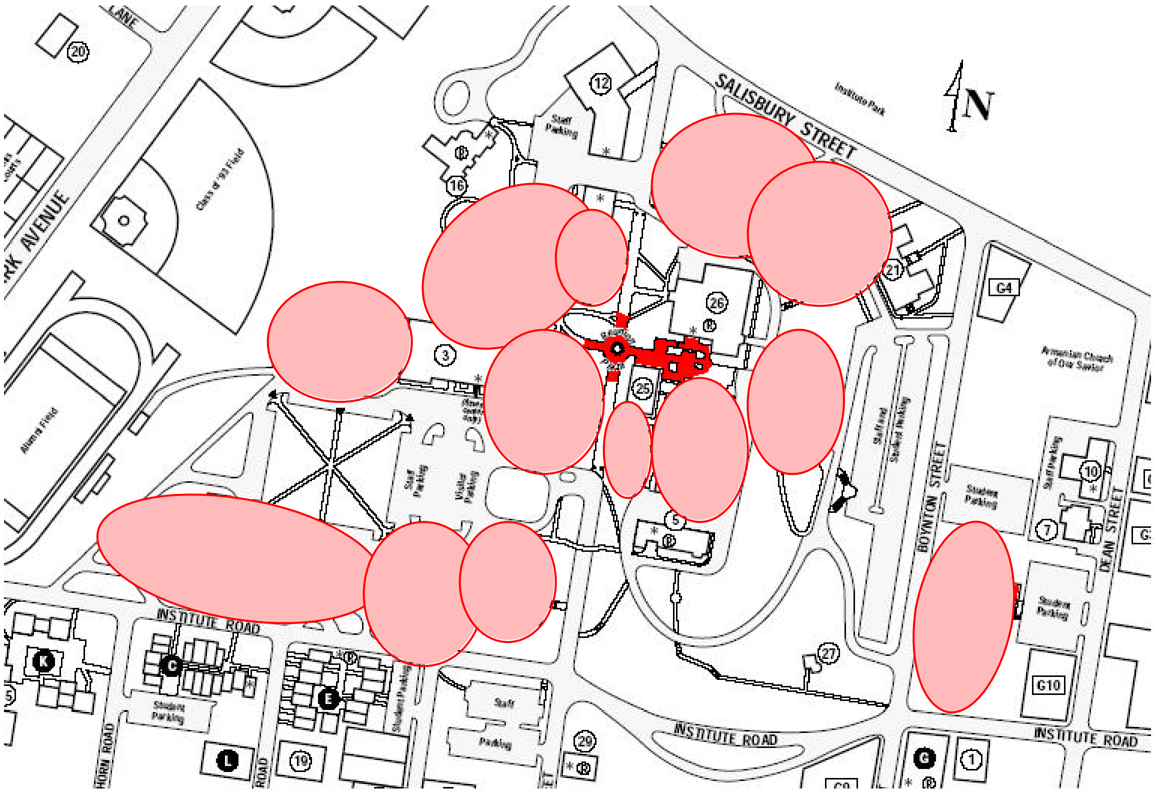

Figure 4: WPI Wireless LAN Map

Although the wireless network officially exists within each of the listed buildings, in some cases it is possible to get wireless connectivity outside these buildings. An approximation of this area is shown by the ellipses on the map.

The range and quality of wireless connectivity largely depends on the distance between a client and an AP, as well as on the properties of the physical medium for wave propagation. Each building has its own characteristic behavior when it comes to signal strength and quality of connection. For example, the Gordon Library has seven APs, two on each floor and one on the lower level. The construction of the library is uniform across the floors with fewer closed rooms or thick walls. Also, the construction material used is relatively more permeable to the wireless signals when compared to other buildings on campus. This uniform construction allows for fairly uniform wave propagation. Fuller Labs, on the other hand, has a higher concentration of Access Points, but the Wireless signal quality is not consistent because of the construction of the building.

2.4 The Wireless Medium and Location Sensing

In a LAN the medium used to transmit data are the physical cables, and in a WLAN the medium used is the air. The wireless medium poses several challenges that must be considered with respect to determining the location of clients on a WLAN. In this section we lay out some the considerations outlined in [PAH02], which the reader should refer to for an in-depth analysis.

2.4.1 Wired and Wireless Transmission

In a wired network a cable connects two terminals and the signals travel along a fixed path. There is a limited amount of radio interference caused by transmission through cable. However this amount is significantly small when compared to interference caused by transmissions through the air. The wired interference can depend, among other factors, on the type of cable used and the coating which acts as a shield. With regards to expansion of a wired network, doubling the number of cables doubles the amount of bandwidth available.

In a wireless network all the signals share the same medium, the air. With a wired network different cables are used to separate networks and applications; in the wireless medium different frequencies are used to distinguish among applications. Because there are only a limited number of frequency bands available and they are all shared by everyone, strict rules must exist governing their use. Three common bands in use are: the 1GHz range in use by the cellular telephone networks, the 2GHz and the 5GHz bands used by the WLANs, and the 28-60GHz band used by high speed point-to-point networks. The important relationship to consider when looking at creating or expanding a wireless network is that by increasing the frequency, we increase the possible data rate, which leads to an increased deployment cost and a decreased ability to penetrate walls [PAH02]. These considerations impacts the type of methodology used in creating a location sensing system, depending on whether it is a low bandwidth cellular network used in rural areas or a high bandwidth WLAN used in urban and indoor areas.

2.4.2 Radio Propagation

"Radio wave propagation is defined as the transfer of energy by electromagnetic radiation at radio frequencies" [SMI98]. Radio Propagation studies look at how radio waves travel through a given medium such as air. Radio waves encounter various outdoor objects ranging from buildings and plants, to indoor objects such as walls and furniture. These objects obstruct radio wave propagation and affect the amount of time it takes for the wave to reach the receiver from the transmitter. In addition to the interference caused by obstructions, other factors such as terrain, wave frequency and velocity of the transmitter and receiver, all impact radio propagation. To study propagation, mathematical models are developed which help predict coverage areas and interference [PAH02]. Particularly relevant to location sensing are the radio propagation behaviors that cause the signal transmission to take a longer path or be delayed. One approach involves measuring the time it takes for a signal to reach its destination and based on that measurement calculating the distance traveled. Any radio propagation delays can ultimately introduce errors into the distance calculations. In this section we focus on some common sources of delay, then in the following sections we look at the mathematical models used to describe location sensing, followed with taxonomy of location sensing methodologies.

The wavelength of the radio waves used by a WLAN is significantly smaller than the obstructions that the radio waves encounter; therefore, we can simplify the study of these waves by treating them as rays traveling in straight lines. The shortest path that a wave can take is the unobstructed path or the Line-Of-Sight (LOS). When obstructions are encountered, the signal has to take multiple paths to travel from the transmitter to the receiver. This behavior, called Multipath Delay Spread, introduces a delay in the transmission time when compared to LOS. Another property of the radio propagation that can cause delays is the Doppler Spread which quantifies the fluctuations caused by the movement of the transmitter, receiver, or the objects in between them. The Doppler spread is especially relevant when considering moving vehicles and airplanes [PAH02].

Two concepts that are prerequisite to the mathematical modeling of radio propagation are transmission power and signal strength. Radio propagation is the transfer of energy and is measured in terms of units of power or Watts. This power is measured at the transmitter (transmission power) and also measured at the receiver; the signal strength is the total amount of power measured at the receiver. Due to the nature of radio wave propagation the latter measurement is less than the former because the signal looses power as it moves through the air in the form of radio waves.

2.4.3 Path-Loss Modeling

Path-loss models measure received signal strength as a function of distance. Signal strength measurements are affected by various radio wave propagation mechanisms [PAH02]. The models become more complex as each of these mechanisms and the previously discussed sources of delay are taken into account. In this section we look at three ways that signal strength can be affected: reflection, diffraction, and scattering. We then present a simplified path loss model and discuss how it could be used in location sensing.

Radio waves typically encounter obstructions larger than the wavelength and depending on the wave frequency and the angle at which they hit the obstruction, the rays are reflected away from the obstruction. Reflection is an important consideration in indoor applications [PAH02]. The radio waves are diffracted when they encounter edges such as when they come into contact with the edge of a building or a wall. As a result of diffraction the waves are able to propagate away from the edge, and reach places that are not directly within LOS. The third mechanism that needs to be considered is called scattering. Irregular objects such as walls, furniture and leaves on a tree can cause the rays to scatter in all directions. Scattering is observed when the object dimensions are close to the radio wavelength, and becomes a significant issue if the transmitter and/or receiver are located in a highly cluttered area [PAH02].

In a simple model we can assume that the radio waves do not encounter any obstructions, meaning they take the LOS path. We ignore the impact of Multipath Delay Spread, refraction, diffraction, and scattering. We also assume that the transmitter and the receiver are both stationary so we can ignore the Doppler Spread. Given these assumptions the signal loss can be stated as such [SMI98]:

$$L = \frac{P_r}{P_t} = \left[\frac{4pr}{l}\right]^2$$

where

P_r	Power measured at the receiver	[Watts (W)]
P_t	Power measured at the transmitter	[Watts (W)]
l	Wavelength	[Meters (m)]
r	The distance of the receiver from the transmitter	[Meters (m)]

The significance of this model with respect to location sensing is that we can measure all the components with the exception of the distance. By manipulating the equation we can solve for the distance and determine where the mobile client that we are interested in, is located. The above model makes several assumptions which in the real world can not be ignored. A WLAN is typically deployed indoors where it encounters obstacles such as furniture, walls, and ceilings. Researchers have further refined propagation models using empirical methods; the transmitter location is fixed and measurements are made for determining distance-power relationships when the transmitter and receiver are separated by different types of materials and also by one or more floors in a building [PAH02]. In the final design of a location sensing system based on radio wave propagation, the complexity of the Path-Loss model will determine the accuracy.

2.4.4 Location Sensing Taxonomy

There are several different approaches for determining the location of an object, typically a laptop or PDA, in a wireless network. Outdoors, this typically involves determining the latitude and longitude of an object, whereas indoors location is determined with respect to a local 2-D or 3-D coordinate system. The process of determining location is called location sensing, geolocation, position location or radiolocation [PAH02]. Several approaches exist for determining location and they range from visual inspection to calculation of signal strength loss. This section provides an overview of the location sensing taxonomy as described in [PAH02] and [HIG01] with a summary provided in Figure 5.

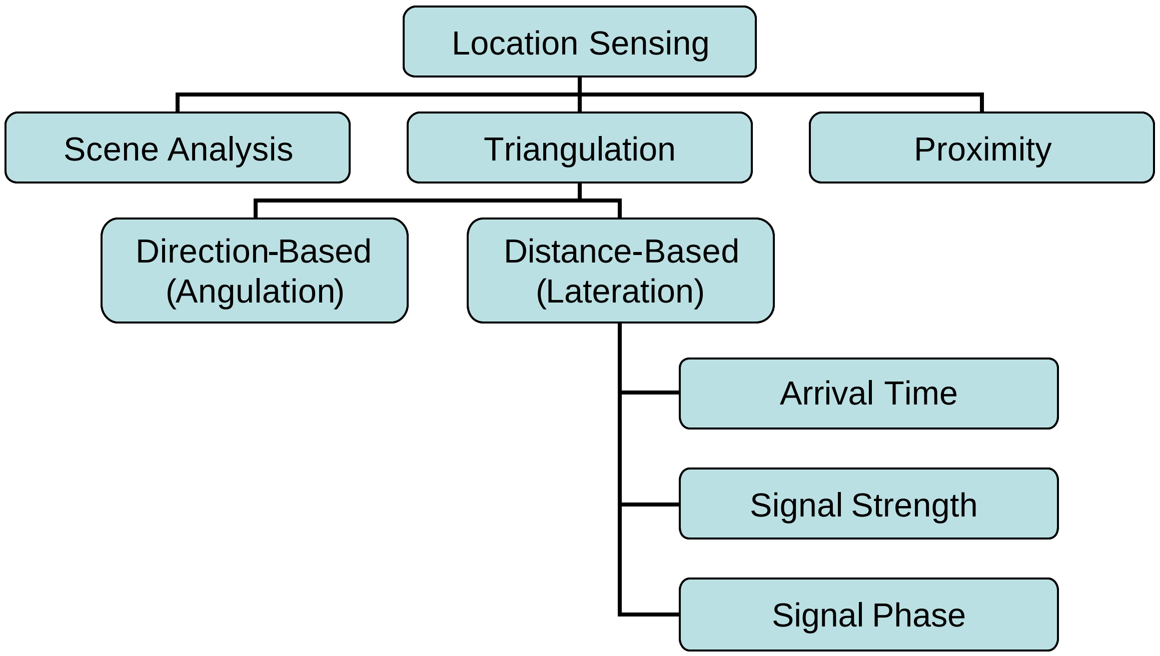

Figure 5: Location Sensing Taxonomy

Location sensing techniques can be divided into three general categories: scene analysis, triangulation and proximity. These various techniques are typically discussed in terms of being physical or symbolic and relative or absolute [HIG01]. A physical technique generally results in a set of coordinates such as the GPS longitude and latitude coordinates, whereas symbolic techniques provides more of an abstract description such as location in terms of which building an object is in. The difference between absolute and relative systems is the frame of reference. In an absolute system, one global frame of reference is used; for example GPS uses the same coordinate system to describe an object anywhere in the world. A relative system however uses local references where physical and symbolic techniques may be combined. We might use GPS to determine the coordinates of the WPI library, and a relative coordinate system to determine in what room or floor an object is located in the library.

The proximity technique is used by ordinary people everyday when asking or giving directions. For example, the location of the grocery store might be specified as one mile east from the town center which is a known location. In WLAN location sensing we can measure the signal strength at a MC (receiver) and the signal strength at the transmitter. We could then use a radio wave propagation model to calculate the distance the signal has traveled. This approach provides the location of a MC relative to the known location of an AP. Using only one measurement would allow us to place a MC in part of a building as opposed to another part and not be able to easily provide us with a coordinate for the location. This approach depends on the propagation model used and is hampered by interference caused by walls and furniture. An alternative proximity approach involves creating a table of measured signal strengths and then comparing the signal strength of a MC to the values in the table. Sampling and recording more values in the table would increase the resolution of the system. This approach is not as susceptible to error due to interference; however, it involves more work to set up the location system. The Ekahau and Princeton ISP case studies (discussed in section 2.5) are examples of this approach.

The scene analysis technique examines a scene such as a room from a fixed vantage point. One such approach is adopted by the Microsoft Research group in the RADAR implementation. RADAR measures signal strengths of mobile devices from the vantage point of a fixed base station. These measurements are then used to calculate the position of the devices on a 2D coordinate system local to the building [HIG01].

The last major technique that we will consider is triangulation. This method derives its name from trigonometric calculations and "can be done via lateration, which uses multiple distance measurements between known points, or via angulation which measures an angle or bearing relative to points with known separation" [HIG01]. These two techniques are also referred to as direction-based and distance based techniques in [PAH02].

Direction-based techniques measure the angle of arrival (AOA). Using directional antennas the receiver must measure the direction of the signal from the transmitter with respect to a fixed direction such as east or west [PAH02]. Two or more AOA measurements can tell us where paths between the transmitters and the receiver intersect, and using trigonometry we can calculate the location. Because this particular triangulation technique requires the use of special antennas it would not be suitable for a WLAN location sensing application that mandates the use of standard components.

Distance-based techniques involve measurement and calculation of the distance between a receiver and one or more transmitters whose locations are known. These techniques involve using one or a more of the following signal attributes: signal arrival time, signal strength, and signal phase. Information on the signal phase method is available in [PAH01] and will not be discussed here; further exploration is suggested for future work.

Signals travel through the air at the speed of light or $3x10^8$ meters/second, and radiate outwards in all direction in the form of a sphere. If we measure the precise time a signal leaves a transmitter and the precise time the signal arrives at a receiver we can determine the time of arrival (TOA); the time it takes for the signal to arrive at the receiver. Since we know how fast the signal was traveling and for how long the signal was traveling then we can determine the distance traveled according to the following relationship:

$$\text{distance} = \text{rate x time}$$

In two dimensions, one such calculation tells us that the object is located anywhere on a circle with a radius d (distance), centered at the transmitter. In three dimensions the object is located anywhere on a sphere with a dimension of d, centered at the transmitter. Whether a circle or a sphere is used for the location calculation depends on whether we are trying to locate the object in two or three dimensions. Distance can be calculated with reference to more receivers which result in additional circles on which the object is located. The only way that the object can be located somewhere on two or more circles is that if the circles intersect. The point of intersection of the circles is where the object is actually located. Because circles can intersect at more than one point three distance measurements are needed and similarly four different measurements are need for a 3-D system.

The signal strength method works similar to the TOA method with the difference being in how the distance is calculated. Instead of measuring the departure and arrival times, the signal power is measured at the transmitter and the receiver. As explained in the radio propagation section this information can be used in a mathematical model to determine the distance that a signal has traveled. Multiple distance measurements are used similarly to TOA to calculate the location. Both TOA and signal strength are promising approaches for WLAN location sensing; however any application will be affected by the issues discussed in the radio propagation section.

2.5 Commercial WLAN Location Sensing Technologies

In this section, we look at two existing commercial technologies that provide location sensing on a Wireless LAN: Ekahau and Newbury Networks.

2.5.1 Ekahau

Ekahau is a Finnish company that provides software solutions for location sensing on a Wireless LAN. The Ekahau system has been specifically developed to provide GPS-like positioning inside buildings and to a certain extent outside buildings over wireless networks. The system works over different WLAN standards such as 802.11 and HiperLAN.

Ekahau's positioning software suite (named Ekahau Positioning Engine 2.0) includes three sub-components, the Ekahau Client, Ekahau Positioning Engine and Ekahau Manager. Figure 6 demonstrates the individual roles of the components and the relationships between them.

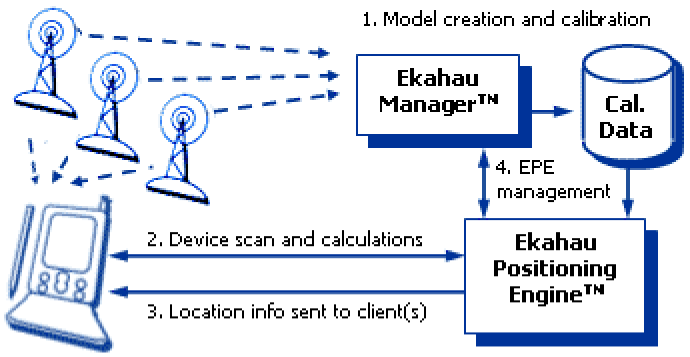
1. Model creation and calibration
Ekahau Manager™
Cal. Data
4. EPE management
2. Device scan and calculations
Ekahau Positioning Engine™
3. Location info sent to client(s)

Figure 6: Ekahau Architecture [EKAweb]

To be 'trackable' each mobile client (Laptop or PDA) must be running the client software. The Ekahau Positioning Engine is a Java-based server software that is used for the calculation of the mobile client locations. The Ekahau Manager displays maps used for positioning, allowing visual tracking of devices, and analyzing accuracy and performance of the software. Currently the supported Operating Systems are Windows 2000/XP and Pocket PC 200X [EKAweb].

Before the Positioning Engine can be used for tracking devices over the WLAN, it needs to be calibrated manually. A floor map is first uploaded on which the Positioning Engine draws tracking rails to create a positioning model. The calibration of the system requires the user to physically move around the area with a mobile client equipped with the Ekahau Client. Approximately every ten feet the current position has to be indicated by clicking on the uploaded map on the client to record sample points containing received signal strength samples. The software does not need the location co-ordinates of the access points [EKAweb].

Tracking of mobile clients on the WLAN can be initiated after calibrating the system though the Ekahau Manager. Real time location of users is shown on the provided floor plan. To locate clients, the software calculates the signal strengths from the access points and compares them to calibrated signal strength samples and to the map of the location. Increasing the number of access points increases the accuracy of the system. Ekahau claims to have up to 1 meter (3.5 ft) average accuracy of locating a mobile client [EKAweb2].

Since Ekahau's system requires the Ekahau Client to be installed on all mobile users, it can only be used over a private network where the company has some administrative rights over the wireless users.

2.5.2 Newbury Networks

Newbury Networks is a Boston based company, providing solutions for Wireless LAN security. Their latest offering "WiFi Watchdog" is an advanced location-based software solution that secures wireless networks against intruders and rogue users. The system works with a standard WLAN setup to locate and monitor authorized and unauthorized traffic on the 802.11 network as well as get information on rogue access points [NEWweb].

Newbury Networks has another location-based product called Digital Docent, which provides location based content on a Mobile Client. Such information can be useful in museums, exhibitions and education campuses by providing dynamic information based on a user's current location.

Both these applications are based on a core technology called LocaleServer. One of the key capabilities of LocaleServer is the ability to detect locations of wireless devices within range of the WLAN. The advantage that LocaleServer has above traditional networking tools is that it can detect all 802.11 traffic within range of the network, not just the traffic on the network. LocaleServer can also be used for tracking and monitoring all 802.11 devices and equipment [NEWweb].

Since this system does not require a "calibration" step unlike Ekahau's Positioning Engine, the technology used for sensing locations is also different. Detailed technical information for the LocaleServer was unavailable. A probable solution this might use is triangulation, using wireless signal strengths and signal phase.

2.6 WLAN Location Sensing Research

Wireless LAN location sensing has become a popular research subject in institutions throughout the country. There has been research done at the doctoral, graduate and undergraduate levels at different colleges and universities. Most of the location sensing technologies that were derived from these research projects, have a similar structure that requires an offline training or calibration phase and an online location sensing phase. In this section we focus on some of the research done in the field and look at the details of indoor location sensing techniques developed by the research.

2.6.1 Indoor Positioning – Independent Study Project, Princeton

An independent study project by Kalid Azad, a student in Princeton University, dealt with the problem of determining location over a Wireless LAN. The logic behind the location algorithm is similar to that used by Ekahau [ISPweb].

The Indoor Positioning system requires a user to record Signal Strengths of all Access Points in range from a point in a table. This is done for various points in the WLAN area. To find out location of a mobile client, the signal strengths from all AP's are measured and then the values are compared to the entries in the table. The closest entry in the table is the probable location of the user. The more points a user calibrates, the better the resolution of the software.

To measure signal strengths, this project used a freely available tool called NetStumbler. The software logs signal strengths to a file over a period of time. A Perl script then goes through the logs and parses the signal strengths and calculates the average. To improve the accuracy of the measurements, two home-made directional antennas were created by using a Pringles can and a metallic can. This provided more stable signal readings. As claimed by the author, the system can resolve to about 10-15 feet with extremely high confidence, with the average case on the order of a meter [ISPweb].

2.6.2 WLAN Location Determination via Clustering and Probability Distributions

A research study was conducted in the University of Maryland to determine a WLAN location sensing technique that would use signal strength probability distributions and clustering to counter the noisy characteristics of signal propagation. This study was conducted by Moustafa A. Youssef, Ashok Agrawala, and Uday Shankar of University of Maryland.

This research concentrated on studying the noisy characteristics of signal in an 802.11 LAN. Since 802.11 works over the 2.4 GHz frequency there is interference from microwaves, bluetooth devices, cordless phones and other similar devices. Multi-path fading, where a signal reaches the receiver through different paths, each having its own phase and amplitude is another common problem faced by radio waves. Even environmental changes such as humidity and temperature affect the signal strength. At a fixed location, the signal strength received from an access point varies with time and its physical surroundings.

The location sensing technology discussed in this research paper uses a joint clustering technique and joint probability distributions and involves an offline and an online phase. In the offline phase signal measurements are taken at several points or training locations. At each training location, a model for the joint probability distribution of the visible access points at that location is stored. The training phase results in a collection of probability distribution models for the signals at each point. Maximum likelihood estimation was used to calculate these joint probabilities. Also, the tuples recorded at the training locations were clustered according to APs that were visible at that location. Since the order of the APs seen could vary, the tuples recorded at these locations were not ordered tuples.

The online phase consists of collecting samples from some access points at an unknown location. The strongest access points are used to determine the cluster to search within for the most probable location. Then Baye's theorem is used to estimate the probability of each location within the cluster using the observed tuples and the collected data during the offline phase [MOUweb]

2.7 Computer Graphics

Computer Graphics (CG) is a very broad term defining almost everything you see on a computer screen. Over time CG has developed from creating simple line art to creating complex and lifelike 3D graphics. Some important applications of CG include GUIs, ray tracers, rendering engines, graphics libraries and CAD tools. The following sections discuss some relevant graphics technologies to this MQP as well as their applications.

2.7.1 Graphical User Interfaces (GUIs)

A Graphical User Interface (GUI) was developed as an easier alternative to plain text interfaces. Early plain text interfaces required the user to remember complex command line syntax and provided simple non-interactive output. GUIs were based on the principle that human beings understand better through visualization. GUIs are both a part of the operating system interface and applications that run on it. For example, in a location sensing application, a GUI can present the user with a detailed map of an area and pin point the user location on that map. Previous to GUIs this information might have been relayed to the user via numerical co-ordinates and a textual description. GUIs also provide interactive features such as zoom and multiple viewpoints which are not possible in a text based system. Overall a GUI enhances the functionality, usefulness, user-friendliness and interactivity of and application.

2.7.2 Categories of Graphics Technologies

Graphics technologies can be broadly defined into two categories based on the type of platform: Web-based and Standalone. As the name implies, web-based graphics technologies require the use of the internet to show the users the required graphical images. Web-based graphics require a server to be maintained and potentially have the ability to display real time graphics. Stand alone graphics technologies employ user side graphic applications that do all the processing on the client to display the required image to the user. The line between these two categories is gradually diminishing as more and more applications have integrated both of these technologies. Programs like Macromedia

Flash allow real-time web based graphics to be displayed on the browser, but also have the capability of locally available flash files.

2.7.3 Graphics Technologies

Some key graphics technologies are SVG, Java 2D/3D, Flash, OpenGL and CAD. In this section we will discuss these technologies as they relate to web-based and stand-alone types of graphics.

Web-based graphics can be further divided into conventional web graphics and vector graphics. Most static images seen on the web are conventional graphics and are also known as raster graphics. They include common image formats such as JPEG, Bitmaps, and static and animated GIFs. Raster graphics create images pixel by pixel and the quality depends on how compact the pixels are. A vector graphics system transmits commands to perform basic graphics operations such as drawing points and straight lines. These operations are implemented using mathematical functions to specify fills for enclosed areas instead of specifying colors for each pixel.

Both Flash and SVG are vector graphics utilities. Flash is the more popular and commercial program while SVG is the upcoming technology. Most tools for developing Flash are GUI based and provide a WYSIWYG (What You See Is What You Get) interface. Most standard image editing programs support the exporting of images to the Flash format thus making it popular even among non-programmers. SVG on the other hand is based on XML and is more oriented towards a computer programmer. Any SVG graphic can be created by writing code in XML format using a text editor. Programs, such as Adobe Illustrator 10, are now emerging with the ability to export files into SVG format. There are several examples of SVG being used in maps and web based architectural designs. CAD (computer aided design) is a popular standard in designing maps and architectural drawings. Available tools are make it easy to convert CAD drawings directly to SVG, thereby allowing them be integrated into various applications.

Java with its extensive array of graphic developing tools can bridge the gap between web-based and standalone graphic standards. Java 2D is entirely based on standard Java and uses provided graphics classes. Java 2D is used for tasks such as drawing lines and shapes, filling shapes with gradients and textures, moving and scaling text and graphics and manipulating image data. A complex bar-graph is a common example where Java 2D is used. Java also has 3D capabilities although it requires a special programming API (Application Programming Interface) which is not a part of the core APIs. Java3D is a low level 3D scene-graph based graphics programming API for the java language (j3d.org). The Java 3D API is essentially a high level wrapper and may use technologies such as OpenGL or Direct 3D to create an interface with the hardware.

3 System Design

In chapter one, we presented the motivation for creating a software only solution for indoor location sensing. In chapter two, we presented some necessary background information for understanding the design and implementation of the software solution that we are proposing. In this chapter we present the design for a modular indoor geolocation system.

3.1 Design Goals

In designing our location sensing system, we laid out several design goals which are presented in the following sections.

3.1.1 Software-only Solution

One of the primary goals of this MQP is to develop an indoor geolocation solution with minimum cost and maximum flexibility. The WLAN infrastructure provides an excellent base to build upon. Money and time has already been spent on deployment of access points, and many mobile clients already have wireless network adaptors. A software-only solution will utilize this infrastructure and not impose any additional costs in terms of time or money other, than what is required to configure the software.

3.1.2 Platform Independence

Another benefit of a software-only solution will be flexibility, easier configuration and portability. If the system has a no dependence on a particular hardware deployment then the software system could be used in any setting where a minimum set of requirements are met. The solution will be truly platform independent if it can run on any operating system and on any standard WLAN. By meeting these conditions the software can be easily deployed across a variety of systems.

3.1.3 Modular Design

Modular design is crucial to the success of any large development project. Design and development of modular components has many advantages, among which are: concurrent development, design flexibility and expandability, and easier testing. In this MQP the graphics modules and location sensing modules should be developed and tested concurrently but also independently. If designed properly, when the modules are completed they should simply "plug-into" each other. By developing the software as independent modules different graphical interfaces could be developed and the software could be expanded in ways not apparent during the initial design phase.

3.1.4 Intuitive GUI

Visualization of information is a very important component of any system. After determining location, that information should be relayed to the user of the system in an intuitive and concise manner. Building name, floor name and (x,y) coordinates will tell the user of the system where a target is located; however, displaying that information on a map will make it easier to analyze.

3.1.5 Simple Setup and Generation of Maps

It is expected that the system will require a certain amount of configuration before it is ready for run time. The setup process should be simplified so as to not detract attention from the main features of the program. The graphical display portion will use floor maps; the process from map acquisitions to integration, into the software, should be relatively smooth with as few steps as possible.

3.2 Design Considerations

The initial design idea was to create a system that measured location using triangulation similar to how GPS works. During the initial background research and literature review several design choices came to light including how and where to measure signal strengths, whether to have a client-server model or a stand alone model, and how to best incorporate maps into the program. These issues and design choices are now discussed further.

3.2.1 Signal Strength Measurements

Signal strength values are measured by both access points and by the wireless network adaptors. If signal strengths of clients are measured by an AP then the AP could theoretically be queried for the information. This would allow an administrator, for example, to monitor changing signal strengths of various clients, and from that calculate the changing locations of various clients. By querying the APs there would be no need to install any software on clients. This AP-centric approach was discussed with the WPI Networks Operations [WPINOint] and two main problems were pointed out. First, the programmatic interface to the access points is not uniform and different pieces of software would have to be written to query different AP models. Second, there would be issues with stale information. An AP maintains state about clients which are associated with it. Once a client roams away from that AP then its state is marked as away, which would present stale signal strength information if that AP was queried for client that has roamed away.

The alternative to AP-centric measurements would be client-centric, where signal strengths of APs are measured from the point of view of the AP. If the application is designed with modularity in mind, then the portion that is responsible for signal strength measurements should be completely separate from the rest of the application. Since there are far fewer common operating systems than there are APs, writing modules to measure signal strengths on different operating systems would be simpler than writing modules to communicate with the different APs.

Once signal strengths of visible APs are measured by the client, the location can be determined using two different approaches. The first approach would involve triangulation, similar to GPS. The problem discovered during the research phase of the project was that there is significant interference indoors which will cause inaccuracy with the triangulation method. The alternative approach is a proximity-based one, which involves recording the signal strengths of several visible APs for a given number of known coordinates. Then during run time, the observed signal strengths of APs are compared against the stored values and the closest match results in a location approximation. With this approach the sources of interference become part of the recorded and observed measurements. The downsides of this approach are that the resolution of the system depends on the number of recordings made and significant changes in the surroundings could render the recorded measurements invalid. After taking the various factors under consideration, the approach of comparing observed measurements against recorded measurements was selected over triangulation. It should be noted that both approaches could be improved on by using radio propagation models, but that would add another level of complexity to the design and configuration of the system.

3.2.2 Where Am I, Where Are You, Where Are They

"Where Am I?" or WAI, "Where Are You?" or WAY, and "Where Are They?" or WAT, are three questions to which a physical location is the answer. These three questions also form the basis for the design of the Locus location sensing solution. With WAI, a user or client is interested in finding out where he or she is located. People who use a handheld or in-car GPS unit are essentially using a WAI type of a system. WAY is the connecting of two different WAI system through a networking interface. Once a client is able to determine its own location, other clients can request that information. In WAY the layer that sits on top of WAI will involve the sending of request messages for a client's location and responses that contain the location.

A WAT system is envisioned as an administrative tool for monitoring the location of various authorized clients and potentially the detection of unauthorized clients on the WLAN. WAT would work very similar to WAY; however, the administrative software only sends request for clients' location and the clients respond with their location. This project will primarily focus on the development of a WAI system which is the foundation for the other systems. WAY and WAT development is envisaged as future work.

3.2.3 Local, Client-server, Peer-to-Peer

Local, client-server and peer-to-peer are three models used in designing applications that do and do not require intercommunication. In this section we briefly discuss how these application development models relate to location sensing.

An application that does not need to communicate with other remote applications is considered to run locally. A word processor or a WAI implementation of a location sensing solution would be fall into this category. Most web applications are composed of two parts. The client runs one of the parts locally, and communicates with the remote application, the server which runs the other part. Typically the client is more "light-weight" than the server in terms of it processing abilities. The "brains" of the application is typically part of the server. In a WAT implementation most of the clients would be fairly light-weight and potentially would only have to measure signal strength values. The server would then calculate the location and display it to an administrator. Using a light-weight design for the client applications would allow the application to silently run as a background process and create the possibility of a client being monitored without user intervention. The third application development model, peer-to-peer, provides a client to client connection where the lines between a client and server are blurred. With this model the client and server are a single application with all the connected clients having the same set of capabilities. A WAY implementation would be a type of peer-to-peer application.

3.2.4 Graphical Displays and Maps

Among the design goals for this project is the creation of a graphical user interface that is both intuitive and easy to configure. Once the location of a client is determined the most obvious and clear way to relay that information to a user is to display it on a map of the area where the client is located. For buildings, maps are usually readily available as building floor plans. Acquiring these floor plans in electronic format such as

CAD (Computer Aided Drafting) drawings will significantly speed up their integration into whatever graphical display module is developed. A popular application used to create and view CAD files is AutoCAD made by Autodesk. AutoCAD is a large and expensive application and not meant to be incorporated into other applications. Using such an application would also be overkill since the final location sensing solution would only need to display and not edit the maps.

The basic units of data in maps and CAD files are simple geometric shapes and line segments. These units are combined in various ways to form a map. Scalable Vector Graphics (SVG), used for many different purposes including animations and maps, also uses these same basic units of data. SVG is an open standard and there is open source software for viewing of SVG files. Because of the close relationship between SVG and CAD, the availability of software to convert between the two, and open source software to display SVG, SVG is recommended as a basis for the graphics display module. Any alternative solutions would have to support the same basic units discussed above.

3.3 Object-Oriented Analysis

In the previous section we laid out the design goals. In this section we go through the various stages of the object-orient analysis process. We start with a detailed statement of the system requirements followed by usage scenarios. Based on the scenarios, we develop subsystems and components that together will make up Locus. Finally we break down the components in the detailed class design section to make it ready for implementation.

3.3.1 System Requirements

In this section we break down the system requirements into three phases. The goal of this MQP is to implement Phase 1, and in doing so set the groundwork for implementation of phases 2 and 3. Implementation of Phases 2 and 3 provides an opportunity for future work. Note that Phase 2 and 3 in Table 1 reflect changes to the corresponding entry in Phase 1. A blank entry indicates that the implementation in Phase 2 or 3 is similar to that in Phase 1.

Phase 1	Phase 2	Phase 3
Where Am I? (WAI) This approach allows the client to determine its own location.	Where Are You? (WAY) This approach allows two clients that implement WAI to query each other for their location.	Where Are They? (WAT) This approach allows an administrator to monitor the location of multiple clients.
The System is calibrated by a user walking around and recording measurements. These measurements will be stored in database. Meas-urements made during run-time are compared to database value to calculate location.	Measurements made in Phase 1 are used to derive a mathematical model. This model describes the change in signal strength across a floor.	
Reporting of location will be done in pseudo real-time. The client's location is updated either after a fixed time interval or on request.	User can select different time intervals at which the location will update.	In WAY or WAT the user can determine different update intervals for dif-ferent clients.

Phase 1	Phase 2	Phase 3
The software will make measurements of signal strengths calculate location. A delay is associated with the calculation.		
After the location is calculated a map is updated with the client's location.		
The client's current location is represented by a dot on the map.	Multiple clients are represented by different colored dots.	Option to interpolate the path a client has taken based on the set of measurements.
Floor maps are stored locally for each application. If the user is in an area that a map is not available for then no map is displayed.		A central server stores building maps. The client determines the floor it is located on and downloads the appropriate map. Requests for non-existing maps could be monitored and new maps created as needed.
Calibration data is stored in a database implemented as a flat file or an object dump.	Calibration data is stored inside of a local database.	A central database stores all calibration data and maps. Clients download the DB specific to the floor they are on.
A floor name is associated with map files. Once the floor is determined then the correct map can be loaded.		
Installation, Configuration, Calibration, and the main run time application will be integrated into one application.	User is given the option or impression of switching between different applications.	
Demonstrate where the user is located on a given floor of the WPI library.	Demonstrate where two users are located on the same floor of the library.	Show where two or more user are located across different floors and or buildings.

Table 1: Summary of Locus implementation phases

3.3.2 Scenarios

This section contains more detail on several different usage scenarios that further refine the design of the Locus location sensing system. These scenarios are guidelines, although the final implementation should resemble the behavior described here fairly closely. The scenarios fall into the three categories of configuration, calibration, and runtime. Figure 7 summarizes the flow of events between each of these applications or sub-applications (depending on the particular implementation).

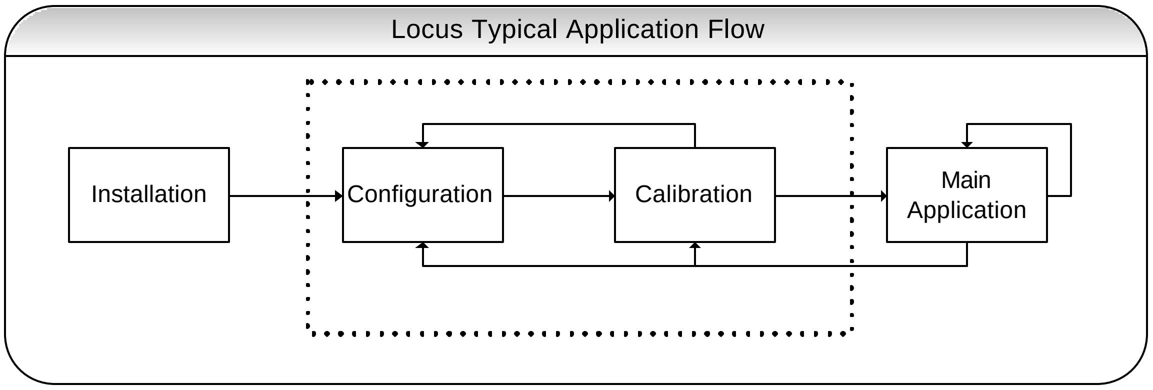

Figure 7: Locus Typical Application Flow

Runtime Scenarios

Scenario Name	WAI Basic Runtime
Actors	User who is interested in determining his or her own location.
Entry Condition	1. User starts main application.
	2. Software presents display where the map will appear, and also several buttons. 3. User presses "Start Location Sensing" button. 4. Map displays a dot representing location 5. Map updates at fixed interval 6. User presses "Stop" button 7. Map stops updating and dot stops blinking 8. User presses exit button
Exit Condition	Software exits

Table 2: WAI Basic Runtime

Scenario Name	WAI-WAY Basic Runtime
Actors	User A and User B. Each user is interested in determining his own location and the location of one or more other users. The sequence of events is the same for all users. Different colors will be used to represent each user.
Entry Condition	1. User A and B are both on the same floor.
	2. User A Starts software. 3. User B Starts software. 4. Software presents display where the map will appear, several buttons, and some fields to allow entry of IP Addresses or machine names. 5. User presses “Start Location Sensing” button. 6. The software loads the map for the floor the user is currently located on. If a map of the floor is not available the user is given a text description of the campus/building/floor he is located on. 7. Map displays a blinking dot representing the user’s own location 8. User adds one or more IP addresses of users to monitor to the list of other users. 9. User presses “Monitor Other Users” button. 10. For each user a dot is displayed similar to the user running the application. Different users are represented with different colors. 11. Map updates at fixed interval. 12. User presses “Stop” button. 13. Map stops updating and dots stop blinking. 14. User presses exit button.
Exit Condition	15. Software exits

Table 3: WAI-WAY Basic Runtime

Configuration Scenarios for WAI

Scenario Name	Create Floor Names Scenario
Actors	User responsible for configuration of application.
Entry Condition	1. User prepares list of building names and floor numbers.
	2. A floor name is composed of the network name, building name, and floor number. 3. A floor name might have the following example naming convention (without any spaces): [network_name]- [building_name]-floor-[floor_number]
Exit Condition	4. The chosen naming convention will result in unique names for every floor.

Table 4: Create Floor Names Scenario

Scenario Name	Prepare Map Scenario
Actors	User with access to building maps and conversion software.
Entry Condition	1. User obtains a map of the floor.
	1. User creates a CAD file for the map or obtains an existing CAD file for the floor. 2. User converts CAD file to an SVG file using Adobe Illustrator or another conversion utility. 3. Map file is named according to the floor naming convention. 4. User copies the SVG file to appropriate application directory.
Exit Condition	5. Application map directory contains new map.

Table 5: Prepare Map Scenario

Calibration Scenarios for WAI

Scenario Name	Calibrate Map Scenario
Actors	User responsible for calibration of system.
Entry Condition	1. User starts calibration application.
	2. User enters floor name to be calibrated. 3. User selects “Start Calibration”. 4. The floor map is displayed. 5. The software instructs the user to go to a point on the floor. 6. User clicks on the corresponding point on the map and signal strength measurements for that point are recorded. 7. The software repeats steps 5 and 6 as necessary. When all the points necessary are recorded the software indicates that calibration is complete.
Exit Condition	8. Floor level list is displayed

Table 6: Calibrate Map Scenario

3.3.3 Locus Subsystems and Components

Two of our design goals were modularity and flexibility. By designing Locus as different components or subsystems that interact with each other, the subsystems can be implemented, tested, and changed independently of each other. The internals of a component can be completely changed with no impact on the rest of the system as long as the subsystem interfaces do not change. Four platform independent and one platform dependent subsystem will make up the Locus software. Figure 8 shows these subsystems. A functional description of each subsystem is provided in Table 7.

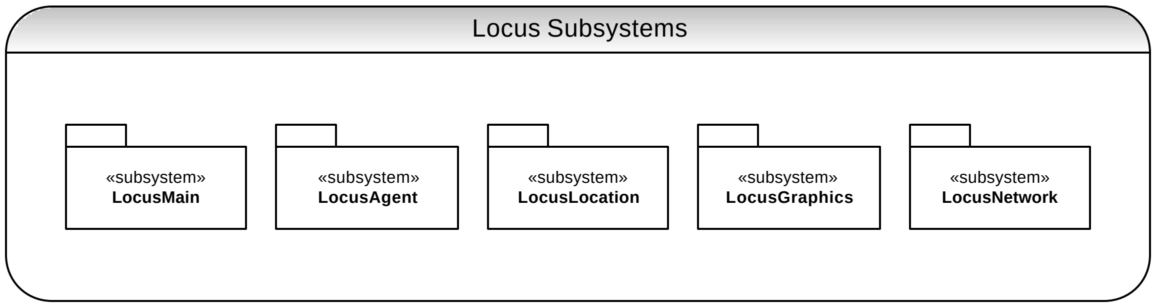

Figure 8: Locus Subsystems

Subsystem	Function
Main	Application entry point and driver.
Agent	Platform dependent communications with wireless network adaptor for retrieval of AP signal strengths.
Location	Algorithms for calculating location based on recorded and observed AP signal strengths.
Graphics	Recording of coordinate during calibration and display of coordinate during runtime.
Network	Communication between different clients. This subsystem left for future development

Table 7: Functional Description of Locus Subsystems

.

3.3.4 Detailed Class Design

In this section we present the individual classes that make up Locus. The first sets of classes discussed are the controllers which directly correspond to the subsystems. This is followed up with the Locus data containers and the individual classes that make up each subsystem.

3.3.4.1 Subsystem Controllers

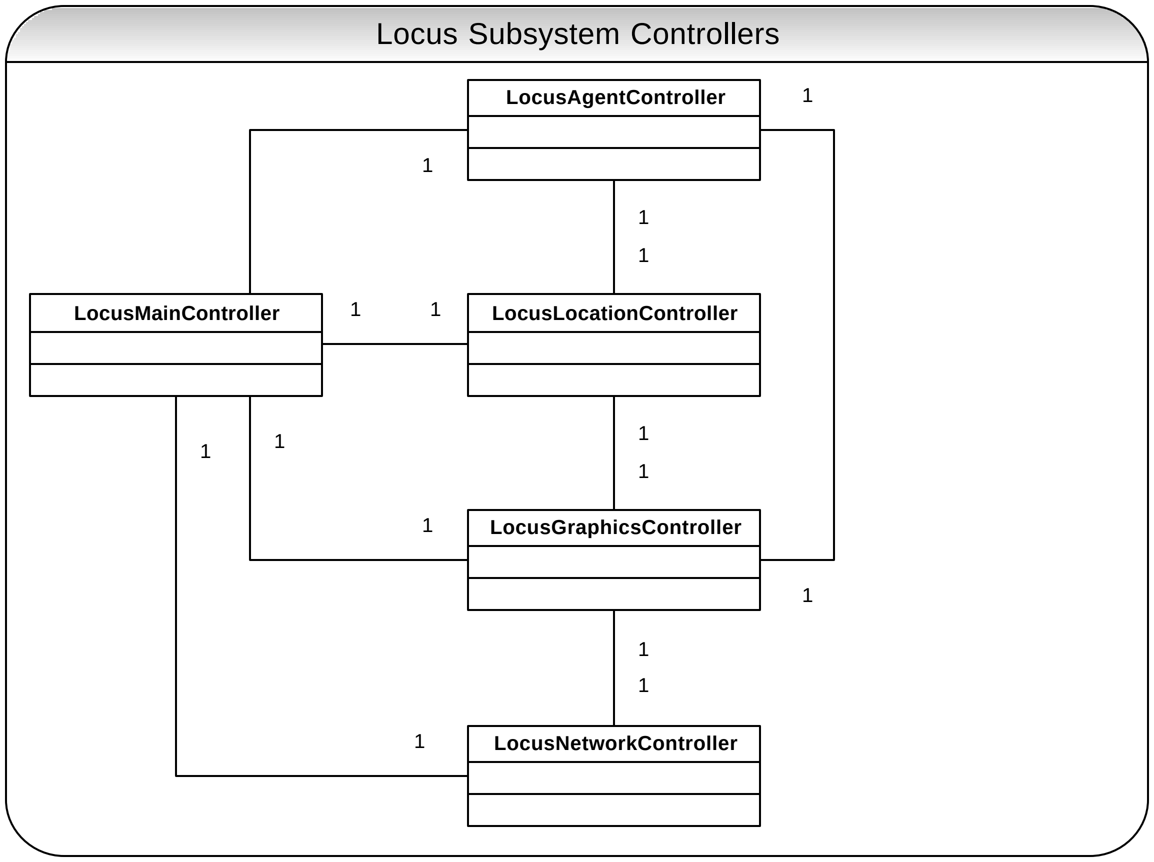

Figure 9: Locus Subsystem Controllers

Each of the Locus subsystems has a controller class which is responsible for driving all the functionality in that subsystem. The main controller is the entry point for the application and starts each of the subsystems. During start up the order of execution of controllers is read from top to bottom of Figure 9. This figure also depicts the relationships between the controllers. The network controller is presented here for completeness but its detailed design and implementation is outside the scope of this MQP.

3.3.4.2 Data Containers

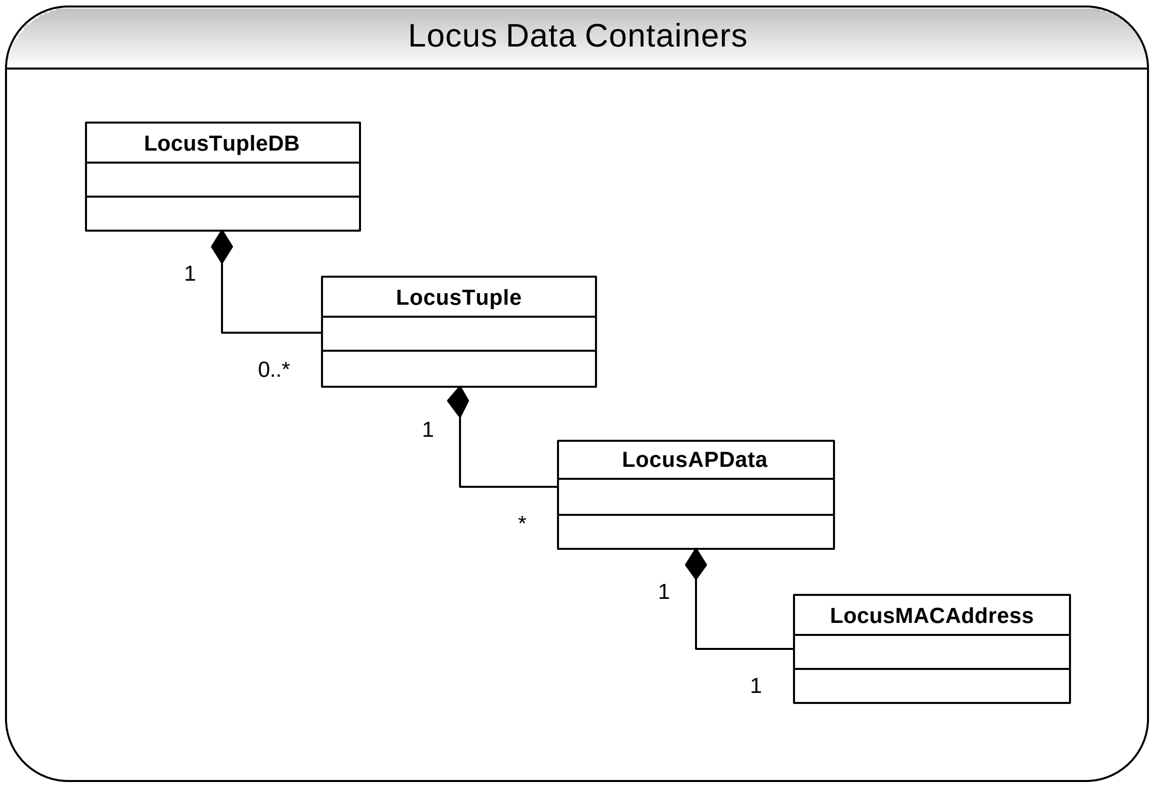

Figure 10: Locus Data Containers

Data Containers are objects that store and manipulate the basic data objects. The most basic Locus data container is the LocusMACAddresss class. APs are uniquely identified by their MAC addresses. Since there is no basic data type to represent a MAC address a container class is needed to represent different addresses. This class contains a physical MAC address value along with methods to compare two MAC addresses for equality and accessing values. Figure 10 shows the hierarchy of data containers.

For every visible AP that the agent observes we are interested in the AP MAC address and the signal strength. The LocusAPData container stores these two values for a given MAC address and provides methods for accessing the values. The next two data containers are the LocusTuple and LocusTupleDB. If the LocusMACAddress class is the most basic unit of the agent subsystem then the LocusTuple is the most basic unit of the location subsystem. An instance of LocusTuple will contain up to three LocusAPData instances and an (x,y) coordinate. A LocusTuple object should be unique for a given point on a given floor. A collection of these objects is stored in the database LocusTupleDB.

3.3.4.3 Agent Subsystem

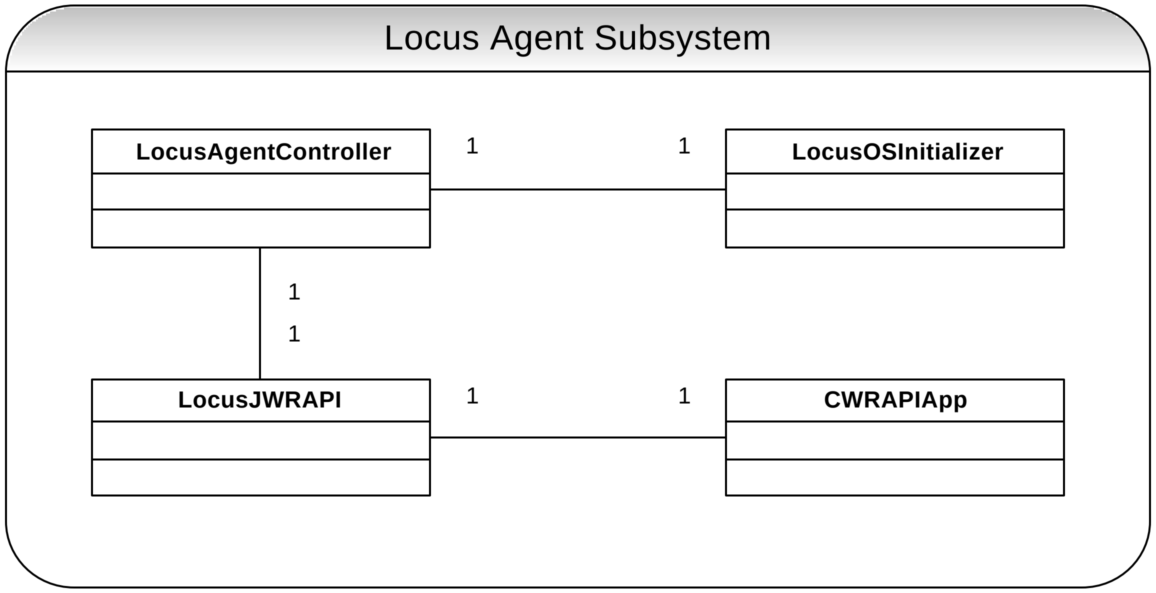

Figure 11: Locus Agent Subsystem

The agent subsystem is responsible for communications with the wireless network adaptor on the client. This subsystem is the only portion of the Locus design that is allowed to be system dependent, and in fact all current research indicates that it will have to be system dependent due to the wide range of target platforms and operating systems. Figure 11 shows the components of the agent subsystem and the relations between them. The LocusAgentController is the driver for the subsystem and provides the interfaces to the rest of the system to take advantage of the functionality provided by the agent subsystem. The most important method of LocusAgentController is the getTuple() method which returns the observed AP MAC addresses and signal strengths in the form of a LocusTuple.

The information returned by the agent is ultimately retrieved from the wireless network adaptor. If this information were to be directly retrieved from the wireless adaptor the agent would be hardware dependent. Since different operating systems provide interfaces for accessing hardware information it make sense to develop an operating system dependent agent which retrieves the signal strength data from the operating system networking API or NIC device driver.

The wireless research API team at the University of California San Diego has developed WRAPI (Wireless Research API) which provides access to signal strengths as observed by a client through the CWRAPIApp class. The LocusJWRAPI wrapper class is a system independent class containing methods similar to CWRAPIApp. In order to make Locus available on other platforms the equivalent of the CWRAPIApp class will need to be re-implemented for that system. The wrapper class exposes an interface to WRAPI. As long as that interface remains unchanged, WRAPI could be replaced or a similar API developed for different systems, without any impact on other portions of the application.

The LocusOSInitializer class is meant to handle any operating system specific initializations. Depending on the required functionality for a particular operating system implementation of this class may not be required.

3.3.4.4 Location Subsystem

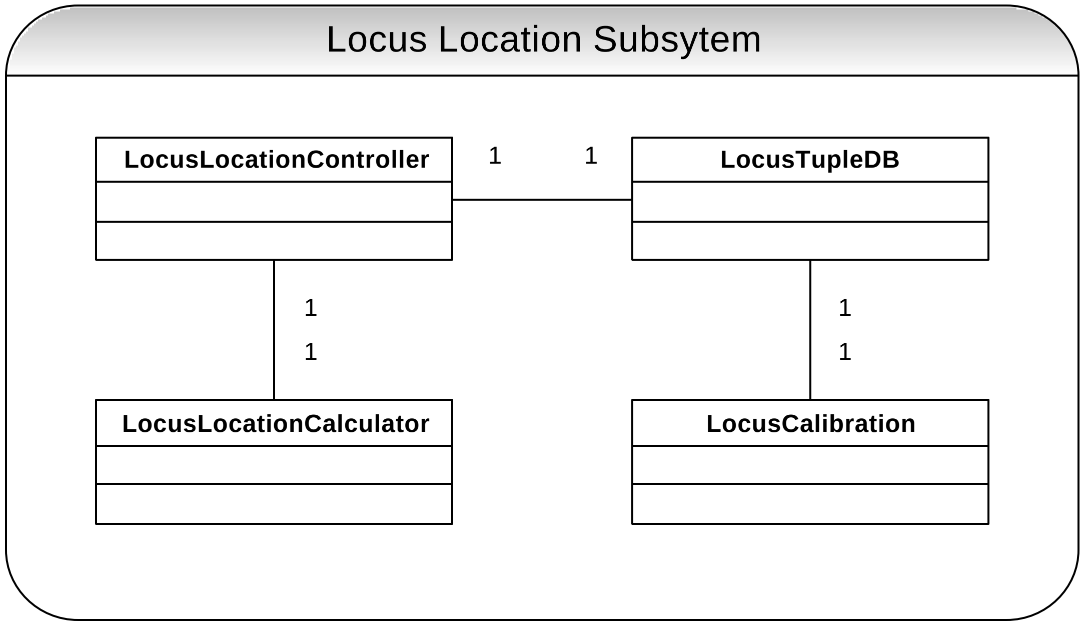

Figure 12: Locus Location Subsystem

The location subsystem shown in Figure 12 is responsible for retrieving the signal strength data from the agent subsystem in the form of a LocusTuple and then calculating the location of the client. During the calibration phase the LocationCalibration class of the subsystem is used for to drive the calibration process. The LocusLocationController is the driver for the subsystem and exposes the appropriate methods to the rest of the subsystem for requesting the client's current location. One of the most important methods of the LocusLocationController is the getLocation() method which returns the current location stored in a LocusTuple object.

The main algorithm for determining the client's location is contained inside the LocusLocationCalculator. When the controller starts up the location subsystem it provides the LocusLocationCalculator with a reference to a LocusTupleDB. Then when the getLocation() method is call the controller provides the observed tuple to the LocusLocationCalculator which then calculates the current location.

The LocusCalibration class contains the logic required for the calibration phase. During calibration the LocusCalibration class observes signal strength values and records them into a LocusTupleDB which is finally stored to disk.

3.3.4.5 Graphics Subsystem

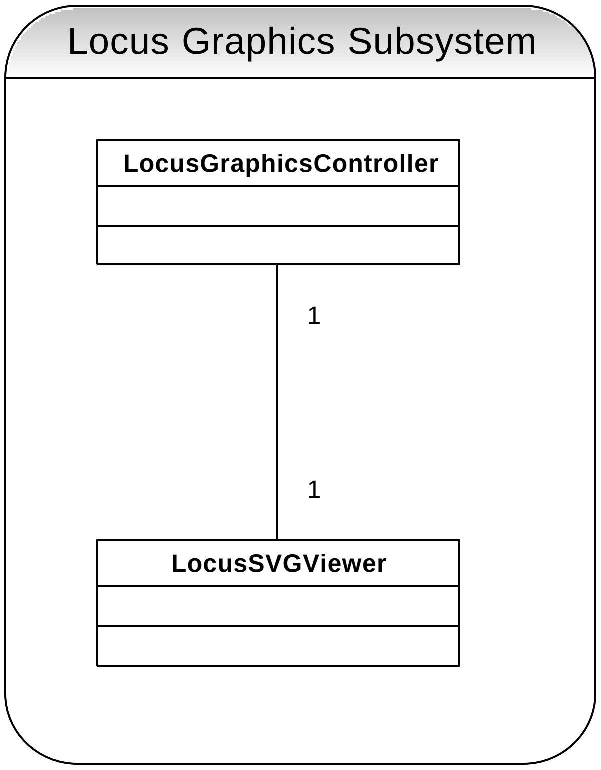

Figure 13: Locus Graphics Subsystem

The graphics subsystem shown in Figure 13 is responsible for the display of the floor map and client's coordinates. Although the LocusMainController is the main driver for setting up the Locus application the LocusGraphicsController is the main driver that sets up the user interface and calls the appropriate methods based on user input. In this project one LocusGraphicsController is to be responsible for the interface to handle calibration and run time. As part of any future works, it would be advisable to separate the controllers for calibration from the controller for the application run time interface.

The LocusSVGViewer handles the display and interaction with SVG files; in the case of Locus the SVG File that we interact with are maps. The controller handles user actions, then calls the appropriate methods to determine location, and uses the methods provided by the LocusSVGViewer to load and update the floor map.

3.3.4.6 Network Subsystem

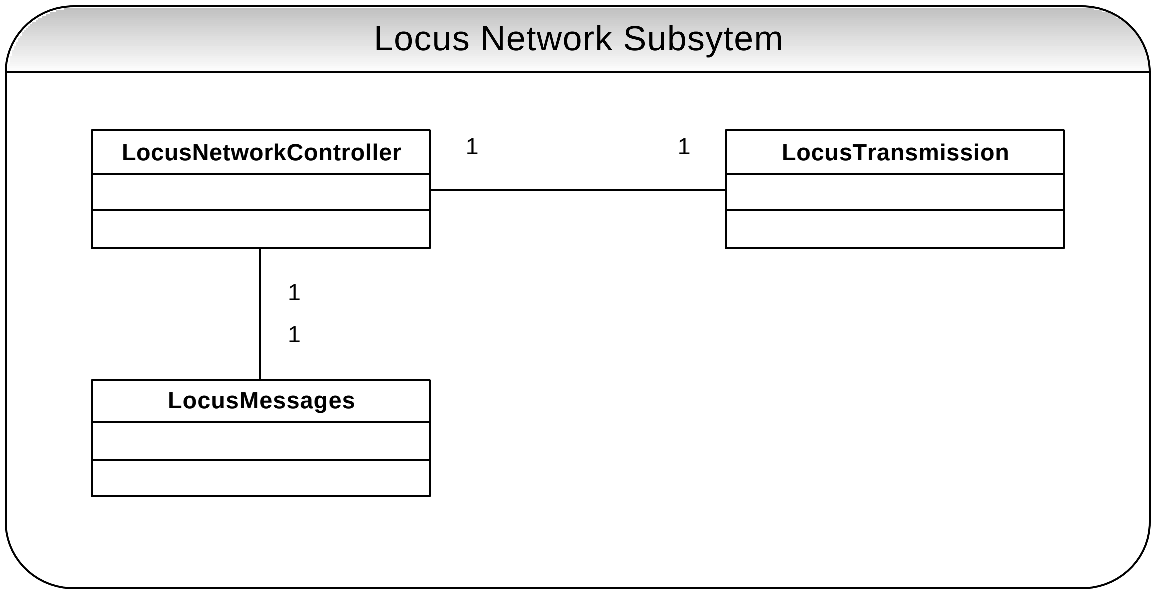

Figure 14: Locus Network Subsystem

The network subsystem design and development is primarily left for future work; however it is briefly discussed here for completeness. Figure 14 shows a possible implementation for the network subsystem. A network subsystem would be utilized if a WAY or WAT type of application was to be developed. The LocusNetworkController would communicate with the graphics controller (or another subsystem, since this portion of the design is not finalized) to get the current location and find out where to send the current location. The raw information is sent to the LocusMessages class which composes a message and passes it to the LocusTransmission class for transmission to another application. Similarly the LocusTransmission class listens for incoming location requests to be fulfilled.

4 Implementation

The development of Locus went through different stages including planning, design, and finally implementation. In this section we discuss the implementation details of our project with the attached appendices and source code providing further details.

4.1 Development Environment

This project started with goals of being a platform independent software solution. This initial design goal was taken into account while making the choice for a programming language, the development environment as well as the operating system to work on.

Java and C++ were the two preferable options for programming languages in which to develop Locus. Specifics such as the platform independent nature, language capabilities, ease of implementation for our set of needs, and available development environments were taken into consideration while making this choice. Java was selected as the language to use for this project because it was well suited to meet our object oriented and platform independent design goals. Java also had equally good GUI development capabilities when compared with C++. The Eclipse Development Environment for Java had features rivaling Visual Studio for organizing a project of this scale. Microsoft Windows was selected as the operating system to base our development on because of its wide appeal, availability of development tools, and APIs (see implementation details for agent and graphics subsystems). However, based on preliminary investigation, development on the Linux platform should also be possible.

4.2 Agent Subsystem: Using WRAPI and JNI

As stated in the design goals the agent subsystem is responsible for communication with the wireless network adaptor. In order to access the hardware, operating system specific, system calls need to be made, which makes the agent subsystem platform dependent. These system calls though similar in concept, are different in their usage on different operating systems. The core of the agent subsystem is the LocusJWRAPI class which provides a Java interface, using Java Native Interface (JNI), to the Wireless

Research API (WRAPI) written in C++. In this section we discuss the WRAPI methods that are important in Locus; further documentation on the WRAPI methods can be found at the WRAPI website [WRAPIweb].

4.2.1 Building WRAPI

During the research phase of the project we discovered WRAPI. WRAPI is an API developed in C++ for the Windows XP operating system. It provides a set of function calls that can be used to send and receive data to network adaptors. The initial appeal of WRAPI was that it claimed to be "a hardware-independent tool that works with any IEEE 802.11b wireless network hardware vendor" [WRAPIweb]. The source code for this API was also freely available. The WRAPI website provides significant documentation on the features of WRAPI. However, there is less information available for building and using WRAPI. After our experiences with WRAPI we have created our own guide and included it as an appendix to this report. In this section we will point out some of the major issues that we faced in using WRAPI as well as an overview of how WRAPI works.

WRAPI provides WLAN monitoring tools for a Windows XP based system. The WRAPI documentation describes the tools and their implementation as such [WRAPIweb]:

> The WRAPI software library (wrapi.dll) allows applications running in user space on mobile end stations to query information about the IEEE 802.11 network they are attached to. WRAPI works with any IEEE 802.11b wireless network hardware vendor.
>
> WRAPI functions obtain information about the wireless LAN using the NDIS User Mode I/O Protocol (NDISUIO).
>
> NDISUIO is a connection-less, NDIS 5.1 compliant protocol driver. It allows user-mode applications to establish and tear-down bindings to network adapters (Ethernet, WLAN etc.) Further, it also supports setting packet filters, sending and receiving data, and handling plug-and-play events. Therefore, as an NDIS_aware component, NDISUIO can directly open an NDIS miniport driver (i.e. network card driver) to send requests, set, and query information. NDISUIO provides an interface between a user-mode application and NDIS using DeviceIoControl (similar to the UNIX ioctl). The NDISUIO driver (ndisuio.sys) is already installed in your system under:
> C:\WINDOWS\system32\drivers

We started with creating a test application, in C++, using the sample code provided by WRAPI (this application and development project are provided as the "app" project in the source code). The first thing we found out was that we could not simply include the WRAPI binaries provided on the website because they were built with a different version of the C Runtime Library than the version that was available on our development environment. Any application that will use the WRAPI DLL has to be linked with the same version of the libraries as the DLL was linked against. The cleaner option was to rebuild the DLL on our system and link it with the available library version as opposed to attempting to replace the version of the library on our system.

The WRAPI DLL needs to be built by including the "nuiouser.h" file from the Windows XP DDK (Driver Development Kit). It is important that the latest version of the DDK, Windows XP DDK SP1 is used. Previous versions of the DDK will not have the right definitions in the header file that needs to be included; older versions will result in improper runtime behavior.

The older version of the "nuiouser.h" file contained the following define statement:

```
#define IOCTL_NDISUIO_OPEN_DEVICE   \
_NDISUIO_CTL_CODE(0x200, METHOD_BUFFERED, FILE_ANY_ACCESS)
```

The current version of the "nuiouser.h" file contains the following statement. This is the version that should be used for compiling WRAPI.

```
#define IOCTL_NDISUIO_OPEN_DEVICE   \
_NDISUIO_CTL_CODE(0x200, METHOD_BUFFERED, FILE_READ_ACCESS | FILE_WRITE_ACCESS)
```

The test application first calls WRAPI methods to bind with the correct device and attach to the appropriate network. It then calls a WRAPI method which retrieves the list of visible APs which contains each AP's MAC address and signal strength. All memory to store the list of network devices and the list of APs is allocated and freed inside of the application and not the DLL. Once we were able to rebuild the DLL we were then able to build the test application by linking it against the new DLL. There are several Visual Studio project options that have to be set in order to build the DLL and the app that calls the DLL; these options are detailed in the appendix. Before running the application the user needs to verify that the NDISUIO windows service has been started and the Wireless Zero Configuration Service has been stopped. “Wireless Zero Configuration is a Windows service that is normally started at boot time. It provides auto configuration for the 802.11 wireless adapters (NICs) by scanning for available access points and associating with the strongest signal. This service automatically binds to NDISUIO and does not allow other apps/dlls like WRAPI to bind to it” [WRAPIweb].

4.2.2 WRAPI Hardware Dependency Issues

We chose to adopt WRAPI for the core of the agent subsystem because of its hardware independent nature, when used with the Microsoft Windows XP Professional operating system. The ideal implementation would have every subsystem be platform independent and have no operating system or hardware dependencies. For certain types of applications one hundred percent platform independency is possible; however, for Locus it is not possible at this time.

The information returned by the agent is ultimately retrieved from the wireless network adaptor. For the Microsoft Windows XP Professional operating system WRAPI provides hardware independent access to this information. Event though WRAPI is supposed to be hardware dependent, we saw different results under different hardware configurations that met the WRAPI software requirements. We have provided the results of our investigation into the WRAPI hardware independency issues in the appendix.

4.2.3 JNI

Locus is primarily developed using the Java programming language; however the WRAPI portion of the agent subsystem was written in C++. Accessing WRAPI functionality in Locus means having Java code which calls a DLL written in C++. In java this is accomplished by using the Java Native Interface (JNI). To use JNI, class methods that need to access native code are marked with the “native” keyword and their implementation is provided inside of a DLL. Calling one of these methods will then result in a call to the DLL method which could be written in any language including C++. Detailed information on how to use JNI is available in a web tutorial by Beth Stearns

[STEARweb] and in a JNI book by Sheng Liang [LIANG99]. The LWW DLL project and LocusJWRAPI.java contain the Locus JNI implementations.

With Locus we had two choices when it came to using JNI with WRAPI with the difference being if we wanted to make any changes to the WRAPI DLL. In order for Java native methods to directly call WRAPI code the WRAPI DLL source could would have needed to be modified to conform to the JNI guidelines for native method implementation. The second option was to create a native implementation inside a new DLL; the implementations of the DLL would simply load and call the WRAPI DLL methods. The second approach acts as a wrapper for the WRAPI DLL and requires no changes to be made to the WRAPI source code. In Locus the second option was adopted, and the wrapper is implemented as the LocusJWRAPI class which calls methods in the LWW DLL (Locus WRAPI Wrapper), which then calls methods in the WRAPI DLL. The rest of the agent subsystem then calls methods of the LocusJWRAPI class.

LocusJWRAPI provides methods to attach to a specific network dynamically, get the MAC address of the AP the station is associated, and to get the list of visible APs and signal strengths. Sending and receiving data from native methods requires special handling and in the Locus implementation is achieved by sending string buffers across the Java/C++ boundary. This class also includes two special methods; WRAPIOpen() must be called before calls to any other LocusJWRAPI methods and WRAPIClose() must be called after the application has finished using WRAPI. The two methods are responsible for initializations and manual memory de-allocations that must be carried out by any application that uses WRAPI.

4.3 Graphics Subsystem: integrating Java and SVG

4.3.1 Selecting a Graphics Standard

The aim of the graphics component of the project was to create and manipulate the main program GUI and display the floor maps in a suitable format. The first step to achieve this was deciding on a graphics format that could display Maps without distortion and allow interactivity from a Java Application. As discussed in Section 2.7.3 SVG is an excellent format for displaying high quality graphics. SVG images are based on XML grammar and manipulating an image in SVG only requires changing the code appropriately. Imaging software such as Adobe Illustrator allows direct conversion from CAD to SVG which is a major requirement in this project since the floor plans of buildings exist in CAD format.

Another important requirement for using SVG was its integration capabilities with Java which can be done using Batik. Batik is a Java based toolkit which is part of an open-source project between ILOG, Kodak and Sun [BATIKweb]. Batik provides a set of modules which allow several SVG creation and manipulation capabilities from Java. These resources made SVG a perfect choice for this project.

4.3.2 Using Batik

Batik provides us with a toolkit in the form of a collection of JAR files that can be used to create and modify SVG content. Batik allows the user to manipulate SVG in many different ways. For example, by using Batik's SVG processor and SVG Viewing component, it is easy to create a basic SVG viewer. Batik has features which even allow extending this basic SVG viewer to include advanced features such as zooming, panning and rotating. Batik's SVG generator module gives the capabilities to a Java application to export its graphics, such as Java2D, into the SVG format. Batik can also be used to convert SVG into other graphics formats such as JPEG or PNG. Figure 15 describes the important components of Batik.

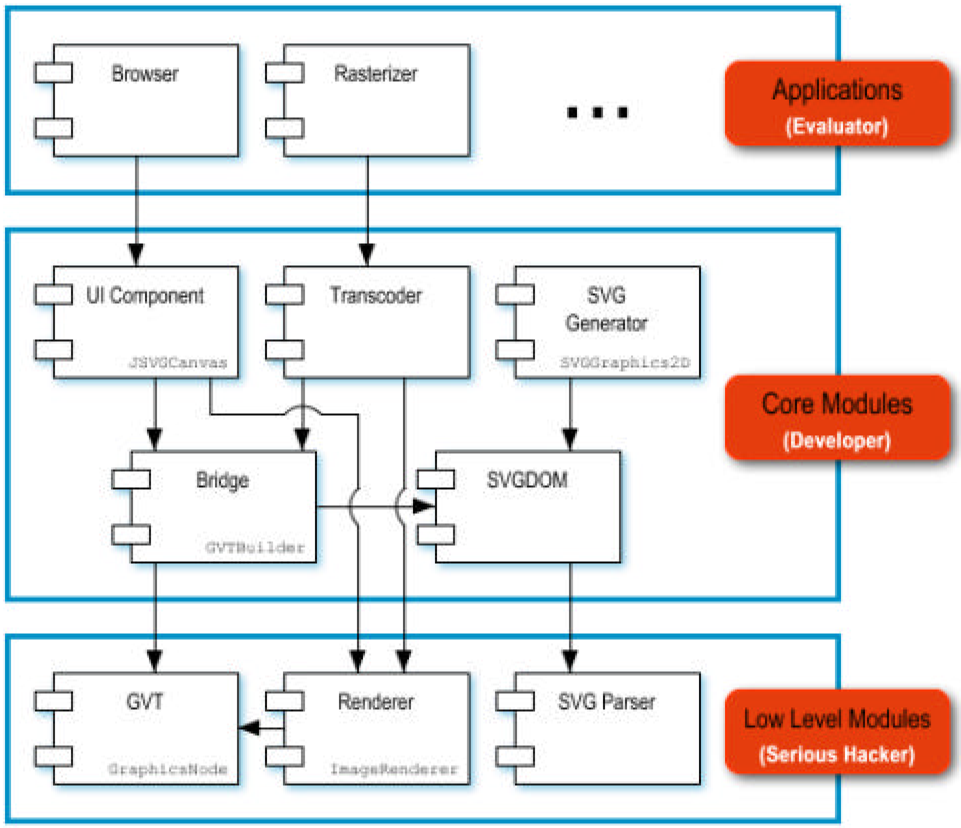

Figure 15: Batik Architecture Overvi ew [BATIKweb]

The first step in using Batik was to create a basic application that would load an SVG map and display it. The detailed steps for installing Batik and using it in Java are described in Appendix 9.4.

The SVG graphics in our final application had two basic requirements: to display a map stored physically as an SVG file, as well as dynamic generation of a dot for depicting position on the map. We decided to implement these two requirements as two separate layers for performance reasons. The first requirement, displaying the map, involved using the JSVGCanvas and pointing it to the file we needed to display. The logic behind displaying this can be followed in Figure 15. First the browser or the program points out the SVG file to be rendered. Then the UIComponent (JSVGCanvas) is given the information about this file as a URI (Universal Resource Indicator). The JSVGCanvas then sends this file to the Renderer and finally the image is loaded, built and finally rendered using the GVT and Renderer. This image is attached to the

GraphicsNode so that it can be displayed. This procedure is handled by the createComponents() method in the LocusSVGViewer class.

The next step was to dynamically display a dot on top of the displayed map. The first approach taken to do this was to load the Map, convert it into an SVGDOM object, and add code to display the dot onto the SVG map. Since Batik does not allow dynamic SVG content, the image would have to be reloaded to display the modifications. This mechanism worked out to be too slow for the purpose of this project.

The final solution to Displaying the dot was to have two separate SVG Canvases. The bottom Canvas would be the one with the static map while the canvas on top would be a transparent canvas of the same size with only a dot displayed at the appropriate coordinates. To achieve pseudo-dynamic updates for the dot, each time a dot had to be shown, a new SVG Document was created and the dot added on to it as an SVG Element. This Document was then set as the top Canvas' content. This mechanism is handled in the DisplayDot() method of the LocusSVGViewer class.

4.3.3 Application GUI

The other task of the LocusGraphicsController was to tie the SVG Canvases with the main GUI, and add appropriate graphic components and listeners to support application functionalities. The LocusGraphicsController class has a method called setupSVGGUI() which handles the initialization for the SVG Viewer, as well as creates the required components for the GUI. This method also handles all the listeners required for Locus.

There is a text field, a label area and three buttons on the main GUI. The label is used to display runtime instructions or information to the user. The text field takes information from the user during runtime. The first button is used for loading the Map of the floor a user is on. This calls the getCurrentFloor() method of the LocationController which calculates and returns the current floor name as a string. This string is then sent to the SVG Canvas so that the correct map can be displayed.

The second button updates a user's location after the Map has been loaded. Clicking this button calls the getLocation() method of the LocationController. This returns a LocusTuple corresponding to the user's current location. The X and Y coordinates are extracted from this tuple and sent to the displayDot method to update the current location on the map.

The third button is used for calibrating a floor before location sensing can be done. This button calls the start() method in the LocusCalibration class so that the user can begin calibrating. The mouse listener for the SVG Canvas is also included in the setupSVGGUI method. This is primarily used for calibration and its main function is to return the X and Y coordinates when the user clicks on the map during calibration.

4.4 Main Subsystem: Application Structure

The Locus design calls for a main application controller and controllers for each of the subsystems. The entry point for the application is the LocusMainController which is implemented as the Locus class inside of Locus.java. The design of the subsystems allowed us to initially create stub classes and integrate the entire application. Using this approach we were able to develop subsystems and simply “plug them in” as they became ready.

The first two subsystems to be developed were the agent and the graphics subsystem. For both of these systems the initial testing was done by calling the methods that would simulate normal run time inside of their respective constructors. The agent initially printed out text to the command line for the AP lists and the graphics subsystem loaded a map and displayed random dots. Later as button and event handlers were added to the graphics subsystem and the location subsystem was developed, the main file remained unchanged.

The main method starts with instantiating instances of LocusGraphicsController and LocusAgentController. Upon instantiation the graphics controller will cause the GUI to come up and the agent will cause native method declarations and allocations to be made. Next the startAgent() method of LocusAgentController is called which is currently a place holder for any statements that might have to be called in the beginning manually. After the agent has been started the setNetwork() method is called to attach WRAPI to a particular network. For the purposes of this project the two different networks used where the WPI Wireless Network and a home network. Changing the network name to attach to is currently done by changing a constant value at the top of the Locus class.

The location controller needs access to an agent controller, which can take some time to start. For this reason the location controller is not instantiated until after the agent controller has completed loading. After the location controller has been instantiated two methods of the graphics controller are called to set internal references in LocusGraphicsController and give it access to the agent and location controllers.

4.5 Location Subsystem: Calibration and Location Algorithm

The location subsystem forms the core of our application. It utilizes information obtained from the agent to calculate the location of a user and finally display it using the graphics subsystem. The location subsystem also contains the logic for calibration. The following sections discuss the implementation of these functions in detail.

4.5.1 Calibration

The logic for the calibration part of Locus is handled by the LocusCalibration Class. LocusCalibration contains a reference to an instance of the agent to enable it to obtain required information from the Access Points. LocusCalibration has a record() method, that takes in X and Y coordinates as integers and records Tuples for that position.

Calibration is initiated by clicking on the Calibration button on the GUI. The name of the floor to be calibrated is also entered in the text field in the correct format (e.g. wpi-library-floor-1). The button listener creates a new instance of a LocusCalibration class and then calls the start() method of that object. The start() method resets all the counters in the calibration class. Then the entered floor name is used to display the correct SVG map. The user is then instructed to go to a point to record measurements at and click on the map. The user has to take four measurements per point facing at four different directions. Initially we used one calibration for every point but tests showed that there was a substantial variance in the measured signal strengths for different directions, so we decided to take four measurements and average their values.

On clicking the map the record() method of the LocusCalibration class is called. The record() method keeps a count of the number of points recorded and the number of measurements taken per point. Each time the user clicks the map, the main counter 'totCount' is incremented and the user is instructed to go to the next point or the next direction. The record() method uses the agent to get AP information and stores it in temporary variables. After all four recordings have been taken for a point the AP data is averaged using the average() method and added to a LocusTupleDB object. The X and Y coordinates are recorded only the first time a user click on a point. This process is repeated for ten different points which results in an object containing ten tuples corresponding to the ten points. The LocusTupleDB object is stored onto the hard disk by calling the storeDB() method of the LocusTupleDB class. The data is stored onto file by using the ObjectOutputStream.

This procedure can be done for different floors. Each time a new floor is calibrated the old data from the file is loaded as a LocusTupleDB object and the new data is added to it. Then this new LocusTupleDB object is stored onto the disk.

4.5.2 Location Algorithm

The location algorithm is driven by the LocusLocationCalculator class. This class has three main methods for calculating location: filterByFloor(), filterByMAC() and match(). The main function of the location algorithm is to observe a tuple and compare it against the existing database. There are two main scenarios when the location algorithm gets called. When a user clicks on the "Load Map" map the location algorithm is used to determine the floor the user is on, and when the user clicks on the "Update" button, his location on the floor is calculated.

In the first scenario, when the user tries to "Load Map" his observed tuple and the pre-calibrated tuple database are known, but the floor is unknown and has to be calculated. To achieve this filterByMAC() is called followed by a call to match(). filterByMAC() takes in a tuple database in the form of a Java ArrayList, and returns a similar ArrayList. The main purpose of this function is to extract the three MAC addresses from the observed tuple and then select all the tuples in the stored database that have the APs with same three MAC addresses. This function treats the tuples as unordered tuples since the MAC addresses within each tuple may be stored in different orders. filterByMAC() returns us a new ArrayList containing only the tuples with the same three MAC addresses as the observed tuple. This ArrayList is then passed to the match() function which finds the closest match to the observed tuple in the stored database.

The match() function requires the filtered ArrayList and a threshold value as parameters. Each of the three observed signal strengths are then compared to the corresponding signal strengths for each tuple in the database. A tuple is selected for further matching only if each of the three corresponding differences between the observed and stored signal strength values are less than the threshold. This filtering gives us a new ArrayList which is then passed recursively to the match() function but with a threshold value of one less than original value. This recursive process is continued until one of the three exit conditions is matched. The first exit condition is when the passed ArrayList is empty, in which case no matches were found or an empty ArrayList was passed to the function. The second exit condition is when the threshold reaches zero. In this case there maybe more than one matches and simply the topmost tuple in the ArrayList is considered to be the match. The third and most important exit condition is when only one item remains in the ArrayList. This indicates that a match has been found in the form of the one remaining tuple in the list. Finally to get the floor, this tuple is looked up and the floor name extracted from it. This name is then passed on to the GUI and the correct map is displayed.

The second scenario when the location algorithm may be used is to update the user's location on a floor. In this case the observed tuple, the stored tuple database and the current floor are all known values. The user's coordinates are the unknowns which need to be calculated. This is achieved in a similar fashion to that of the first scenario. Since we know the current floor name, the only extra step that is performed is calling the filterByFloor() method. This method requires an ArrayList of tuples as well as the floor name as a String. The floor names stored in the database are compared to the floor name parameter, and only the tuples with the matching floor name are returned as a new ArrayList. This is done to eliminate error where the location is calculated for an invalid floor.. The other two steps are same as the methodology to determine the floor, i.e.

calling the filterByMAC() and match() functions. After match() returns a matched tuple, the X and Y coordinates are extracted from it, and sent to the Graphics Controller to display a dot at the corresponding (x, y) position. Table 8 and Table 9 sum up the location methodology for both the scenarios.

Scenario 1: Get Map		
Method	**Input**	**Output**
filterByMAC()	Observed Tuple, Original ArrayList	MAC filtered ArrayList
match()	MAC filtered ArrayList, Threshold	Matched ArrayList - Floor name extracted from tuple in Matched ArrayList

Table 8: Location Algorithm - Get Map

Scenario 2: Update Location		
Method	**Input**	**Output**
filterByName()	Original ArrayList, Floor Name	Floor filtered ArrayList
filterByMAC()	Observed Tuple, Floor filtered ArrayList	MAC filtered ArrayList
match()	MAC filtered ArrayList, Threshold	Matched ArrayList - X, Y values extracted from tuple in Matched ArrayList

Table 9: Location Algorithm - Update Location

5 Results

The purpose of this Major Qualifying Project was to create a platform independent, software-only, indoor geolocation solution. We have achieved many of our goals and created a base from which the realization of our vision should be possible. In the course of the project we also faced some challenges. In this chapter we discuss our achievements and the challenges we faced.

5.1 Achievements

We have been able to achieve our primary goal of creating a software-only indoor geolocation system using an 802.11b WLAN. The system has been implemented according to the system requirements and design that was laid out in the previous sections. All subsystems with the exception of the agent subsystem are entirely written in Java and should be platform independent, though we have not been able to test this extensively due to time constraints. The agent subsystem itself is layered so that the Windows specific WRAPI code can be easily replaced with other system specific code. In addition to implementing the stated requirements, Locus also lays out a framework with potential for additional application development.

Our methodology and algorithms as described in previous chapters allow us to locate on what floor of a building a client is located. The floor level accuracy varied during our development testing, but a majority of the trials resulted in the system loading the correct floor map or not loading a map (as opposed to loading an incorrect map which we consider to be less accurate than not loading a map). In some of the cases, repeated request would eventually result in the right map being loaded. After review of our implementation we believe that system faults lie in calibration and not necessarily in the location sensing algorithm. Once the system was able to load the correct floor map the system was able to determine the clients coordinate approximately half of the time.

There are two categories of inaccuracy with the system: inherent and runtime. The inherent inaccuracy is due to the "calibration resolution". The calibration resolution is the number of points that the user goes to and records signal strength values at. Currently this is fixed at 10 points for a given floor which splits up the floor into a grid composed of 10 blocks. Therefore the best case results would place the client in the correct grid whose dimensions are dependent on the size of the floor. The more points at which a user goes to during calibration, the higher the resolution (i.e. more blocks), and the closer the center of a block will be to the client's actual location. Runtime inaccuracy causes the client's location to be displayed in the wrong block. Two examples of runtime inaccuracies that we observed were the location updating incorrectly, or the location not updating even if the user was moving. In the following section we discuss some of the challenges we faced which we believe to be some of the sources for the inaccuracy issues. We are satisfied with the results we achieved for a proof-of-concept; however further testing and investigation into the challenges are recommended.

5.2 Challenges

5.2.1 1-AP Problem

During the implementation of the agent subsystem we ran into a problem which became known as the 1-AP bug. We faced some difficulty initially with compiling and building WRAPI which mostly had to do with the proper files being included and correct options being set in Visual Studio. The application was simply supposed to print out the list of visible APs and their respective signal strengths. When we ran the application we only received data for one AP and after subsequent executions we repeatedly received the same one item list with no updates to the signal strength.

Upon consultation with Network Operations we discovered one of the security precautions that they employ is to disable broadcast messages sent by all APs at WPI. These messages are essentially announcements to any nearby clients that a wireless network is present and there is an AP to associate with. Disabling these broadcasts can prevent certain unauthorized users from detecting that a network is present. Following up on this lead, the broadcast messages were turned on by Network Operations for all the APs in the WPI library. This time when we tested the application, we were able to get a list from WRAPI with the same number of items as there were visible APs. All APs which WRAPI is meant to work with must have broadcast messages turned on; otherwise WRAPI will not function properly and will result in a 1-AP problem. We were unable to find any information in the WRAPI documentation stating this requirement. Once we had solved the 1-AP problem we received the right number of items from WRAPI; however, we received garbage values for all but the first item in the list. This problem was addressed in detail in the appendix.

5.2.2 Location by Associated AP

In the location subsystem of the implementation chapter we discussed our current approach for calibration and for determining the floor that the client is located on. Determining the floor is important because it is a prerequisite to being able to calculate the coordinates. Our first attempt at finding the correct floor used the AP that the client was associated with.

When Locus is started, first the agent is started, and then a network is specified to attach to. By specifying a network to attach to WRAPI also associates with an AP and provides a method to return the MAC address of the associated AP. We initially made the assumption that WRAPI would always associate with an AP that is physically located on the same floor as the client. The reasoning behind this assumption was that there should be less interference between the client and APs on the same floor as opposed to APs on another floor of the same building. Before calibration we created a list that contained the relationship between each AP and the floor that they were located on. Once we retrieved the MAC address of the associated AP, we did a reverse lookup to determine the floor. The logic in this approach is sound except that our assumption did not hold true and WRAPI did not always associate with an AP on the same floor as the client. This resulted in Locus determining that the client was on the second floor when it was really on the third floor. Identifying the correct building using this assumption is more likely but is still not guaranteed.

In order for this approach to work a set of criteria must be developed that would identify APs which are most likely to be on the same floor as a client. Such criteria would likely start with a check to find out which AP has the strongest signal strength value. Other characteristics of the wireless medium such as signal to noise ratio might be worth investigating. Once the criteria has been defined, then new method, for example associateSameFloorAP(), would have to be added to WRAPI . This method would return a MAC address of the AP or APs that are on the same floor.

5.2.3 Calibration Issues

During the test runs of Locus we observed a strange problem reminiscent of the 1-AP problem and the WRAPI hardware independence investigation. When Locus is being calibrated, the user goes to a fixed number of points and the software records the coordinates of that point and the signal strengths of the visible APs. On many occasions after recording data for a several points we noticed that the observed values were not refreshing. When using our HP laptop with the Microsoft wireless adaptor the observed values would stop refreshing after a while and would not start to refresh again until Locus was restarted. On our Dell laptop Locus does not work properly because WRAPI does not work properly. However once we take the Microsoft adaptor form the HP laptop and use it in the Dell laptop then WRAPI works as expected and so does Locus. This laptop and adaptor configuration also results in better refreshing of the values. We noticed that we did not have to restart Locus all the time to get the values to refresh. Once they have stopped refreshing they would start refreshing again after a short time.

Our calibration problems have two major impacts. First, because of the refreshing problem not all the stored values are going to be correct. If we have an invalid database to compare against during run time, then we are sure to have inaccuracies when determining the location. The second impact of this problem is on our design. The agent subsystem is supposed to be platform and hardware independent; however, they the agent behaves differently under different hardware configurations. The solution might involve using a different core instead of WRAPI for the agent subsystem or modifying WRAPI so that it can handle the different hardware configurations. This solution would only be practical if the number of special cases were small.

5.2.4 Signal Strength Value Precision

WRAPI and the configuration utility supplied with our wireless adaptors both report signal strength values with two significant digits. The signal strength value precision is another type of inherent inaccuracy. The location algorithm that we have used makes the assumption that the set of values measured, for a given set of observable APs, at two different points, will be unique. If the sets are not unique, then the client that observes a signal strength matching one of those two sets, could be located at (or near) either of the points. This problem could be addressed by finding a way to increase the precision of the signal strengths values, because then there would be less of a chance of having identical sets. Other ways to decrease the likelihood of identical sets include taking into account other properties of the wireless medium or implementing a method that involves probability distributions (as discussed in the literature review).

6 Future Work

In the previous chapter we discussed the achievements and some of the challenges that we faced. In this chapter we present ideas for future work that could be done to further enhance and extend Locus. Enhancing and extending Locus provides opportunities for further academic project work and potential commercial development.

6.1 Enhancing Locus

One of the first steps in enhancing Locus would be an in-depth evaluation and feasibility study of possible solutions to some of the challenges discussed in the previous section. The first issue to consider is WRAPI. Since retrieval and storage of signal strengths are an important part of Locus, the WRAPI issues must be resolved. WRAPI claims to be hardware independent but our experience showed otherwise. Were the hardware dependency problems unique to this project? If not, then can the WRAPI code be modified so that it does work on any Windows XP platform meeting the stated hardware and software requirements?

The second enhancement suggestion is to investigate the possibility of implementing the previously discussed associateSameFloorAP() method. This method would provide Locus developers an alternative method of determining the floor a client is located on. This method could potentially improve performance in a large scale Locus implementation.

The third and fourth enhancement suggestions are aimed towards improving location sensing accuracy. The possibility of getting more precise signal strength values is definitely worth investigations. More significant digits in the signal strength values should help increase the accuracy of Locus by increasing the odds of unique measurements. Finally we would like to note that we have implemented one possible location sensing algorithm. We suggest development and comparison of results for other algorithms that employ probability distributions (discussed in literature review) and mathematical modeling.

6.2 Extending Locus

There are many possibilities and opportunities for extending Locus and we have only discussed some the extensions that we are particularly interested in. Bringing the vision described at the beginning would be the ultimate goal for anyone planning on adding to what we have created.

Implementation of Phase 2 and Phase 3 requirements, utilizing the Locus modular design, would be the next natural step. Some of the outcomes of this implementation are outlined here:

- Implement the locus network subsystem. This would be the basis for any type of WAY or WAT configuration.
- Implement a plug-in for an instant messenger application. The plug-in would be a WAY type of application.
- Package the main, agent and network subsystem so that they can run as a background process or a Windows service. This would be the lead-in to a WAT implementation.
- Implement a WAT application that installs the package service on clients. Then an administrator application queries clients for data and determines their locations.
- Research and implement an agent subsystem for the Linux and Macintosh operating systems.

7 Conclusions

We set out to create a software only solution for determining the physical locations of mobile devices on a Wireless Local Area Network (WLAN) by using only the existing WLAN infrastructure and no additional hardware. We have been successful in creating Locus which is a proof of concept and a detailed architecture for a location sensing solution. The object oriented and modular design of Locus allows it to be easily extensible, making a whole array of different applications possible. There are issues that need to be further investigated with regards to accuracy and platform independence; however, the ability to use a WLAN for indoor geolocation has been demonstrated.

8 References

[BAHL1web] Bahl, Paramvir, Padmanabhan Venkata and Balachandran, Anand. *A Software System for Locating Mobile Users: Design, Evaluation, and Lessons.* Microsoft Research. <http://www.research.microsoft.com/~padmanab/papers/radar.pdf>

[BAHL2web] Bahl, Paramvir, Padmanabhan Venkata. *An In-Building RF-based User Location and Tracking System*. Proceedings of IEEE INFOCOM Conference, March 2000. <http://systems.cs.colorado.edu/grunwald/MobileComputing/Papers/radaer-infocom2000.pdf>

[BATIKweb] *The Batik SVG Toolkit project website.* The Apache XML Project. <http://xml.apache.org/batik/>

[CASTweb] Castro, Paul, Chiu, Patrick, Kremenek, Ted and Muntz, Richard. Probabilistic *Room Location Service for Wireless Networked Environment.s* <http://www.fxpal.com/people/chiu/cckm-UBICOMP01.pdf>

[CHIPSweb] *WLAN Adapter Chipset Directory*. Absolute Value Systems. <http://www.linux-wlan.org/docs/wlan_adapters.html>

[DLL1web] *About Dynamic-Link Libraries*. Microsoft Developer Network (MSDN). http://msdn.microsoft.com/library/en-us/dllproc/base/about_dynamic_link_libraries.asp

[DLL2web] *TN033: DLL Version of MFC.* Microsoft Developer Network (MSDN). <http://msdn.microsoft.com/library/default.asp?url=/library/en-us /vclib/html/_MFCNOTES_TN033.asp>

[DLLJNIweb] *C++ DLLs and Java Native Interface*. Legacy Java Answers Forum. August 2000. <http://www.artima.com/legacy/answers/Aug2000/messages/345.html>

[EKAweb] *Ekahau Positioning Engine website.* Ekahau, Finland. <http://www.ekahau.com/products/positioningengine/>

[EKAweb2] *Ekahau Positioning Engine Review website.* Wi-Fi Planet. <http://www.wi-fiplanet.com/reviews/SW/article.php/1560261>

[ESOweb] *European Southern Observatory website.* <http://www.eso.org>

[HARDCweb] *Hardware Comparisons.* Seattle Web. <http://www.seattlewireless.net/index.cgi/HardwareComparison>

[HIG01] Hightower, Jeffrey and Borriello, Gaetano. “Location Systems for Ubiquitous Computing.” *IEEE Computer*. August 2001: 57-61.

[ISPweb] *Indoor Sensing Project website.* Princeton University. <http://www.cs.princeton.edu/~kazad/resources/cs/cs398.htm>

[KOTZweb] Kotz, David. *Dartmouth archive of wireless-network trace data.* Center for Mobile Computing at Dartmouth College. <http://cmc.cs.dartmouth.edu/data/>

[LADD1web] Ladd, Andrew, et al. *On the Feasibility of Using Wireless Ethernet for Indoor Localization.* Department of Computer Science. Rice University. <http://www.cs.rice.edu/~aladd/pubdir/journal/w_tra.pdf>

[LADD2web] Ladd, Andrew, et al. *Using Wireless Ethernet for Localization*. Department of Computer Science. Rice University. <http://www.cs.rice.edu/~aladd/pubdir/conference/iros2002.pdf>

[LADD3web] Ladd, Andrew, et al. *Robotics-Based Location Sensing using Wireless Ethernet.* Proceedings of the 8th annual international conference on Mobile computing and networking. <http://www.cs.rice.edu/~aladd/pubdir/conference/mobicom2002.pdf>

[LIANG99] Liang, Sheng. *The JavaTM Native Interface: Programmer's Guide and Specification.* Published by Addison Wesley Longman, Inc. June 1999.

[MOUSweb] Youssef, Moustafa, Agrawala Ashok, and Shankar Uday. *WLAN Location Determination via Clustering and Probability Distributions.* University of Maryland. <http://www.cs.umd.edu/~moustafa/papers/percom03.pdf>

[NEWweb] *Newbury Networks website* <http://www.newburynetworks.com/>

[NIBLweb] Castro, Paul. *Nibble Location System website.* UCLA Computer Science Department. <http://mmsl.cs.ucla.edu/nibble/>

[OCO03] O'Connor, Sean and Krzeszewski, Joseph. Personal Interview. 2 September 2003.

[PAL02] Pahlavan, Kaveh and Krishnamurthy, Prashant. *Principles of Wireless Networks.* New Jersey: pp 417-448, Prentice Hall PTR, 2002.

[PAN99] Pandya, Raj. *Mobile and Personal Communication Systems and Services*. New York: IEEE Press, pp 15-26, 1999.

[SMI98] Smith, Albert A. Jr. *Radio Frequency Principles and Applications*. New York: IEEE Press, pp 43-46, 1998

[STEARweb] Stearns, Beth. *Trail: Java Interface.* <http://java.sun.com/docs/books/tutorial/native1.1/index.html>

[STO02] Stojmenovic, Ivan, ed. *Handbook of Wireless Networks and Mobile Computing*. New York: John Wiley & Sons, pp 16 – 20, 2002.

[WEA02] Weatherall, James and Jones, Alan. "Ubiquitous Networks and Their Applications." *IEEE Wireless Communications*. February 2002: 18-29.

[WPEDweb] Webopedia website. <http://www.webopedia.com>

[WPINOweb] *WPI Network Operations website*. Worcester Polytechnic Institute. 9 October 2003. <http://www.wpi.edu/+netops>

[WRAPIweb] Balachandran, Anand. *Wireless Research API (WRAPI) website*. University of California, San Diego. <http://ramp.ucsd.edu/pawn/wrapi >

9 Appendix

9.1 Network Operations Interview

WPI Network operations or NetOps maintains the vast WPI network. Since many of the design choices to be made during the initial phases of the MQP depended on the structure of the WPI Wireless Network, a Question-Answer session was set up with NetOps. We interviewed Sean M. O'Connor and Joseph M. Krzeszewski of the NetOps to gain a better understanding of the WPI Network, especially specifics about the Wireless LAN.

Date: September 2, 2003 2:00PM

Location: WPI NetOps Office

Attendees: Sean M. O'Connor, Joseph M. Krzeszewski, Ali Taheri, Arvinder Singh

1. Question to NetOps:

Our MQP involves the measurement of Signal Strengths from different Access Points. We have two architectural choices for achieving this.

a. A local client measures signal strengths of multiple access points

b. Multiple access points are queried by a third party for signal strength of client

Which design do you think will be a better choice and why?

Response from NetOps:

The ideal solution would be the second option [APs are queried for signal], but for purposes of the project it would be easier to implement the first option [client locally measures signal]. There are currently two types of APs on campus: Symbol 4121 and Symbol 4131, and WPI is introducing a new AP, the Symbol Mobius Axon. Each of these APs have different interfaces for the programmer to query them for signal information. The Axon interface is the least developed, and would present a significant challenge to implement. Additionally the Axon AP uses multiple antennas connected to each AP. The mobile clients associate with an antenna and in turn with an AP. The problem is that the antennas could be quite far from the AP, and there is no easy way at this point to determine which antenna the client is communicating with. One suggested solution would be to try and use the unique MAC addresses of the antennas.

2. Question to NetOps:

If a client associates with an AP can other APs be queried for info about that client even though they are not associated with it?

Response from NetOps:

In order for a mobile client to be connected it associates with an AP. After the user roams away from an AP, the AP marks the client as “away” for an hour. After that the client is cleared from the AP list.

3. Question to NetOps:

For this purposes of this project we may need data from 3-4 APs. What security clearances do we need in order to access information from the APs?

Response from NetOps:

Further discussion and planning would be required on the part of NetOps; however there are two possible approaches. Once approach would be the creation of temporary-read-only host strings that would be used to access the APs through the SNMP protocol. The second approach would involve NetOps creating an interface that would be used for the purpose of this project.

4. Question to NetOps:

How many AP are there on campus? We estimate about 160 AP according to the naming convention used.

Response from NetOps:

There are currently about 60 APs active consisting of the 4121 and 4131 models. Once the new Axon APs become activated, mostly in the dormitories, there will be about 160 Access Points.

5. Question to NetOps:

How can we best identify the locations of all the access points? Do you have a map that marks all the locations?

Response from NetOps:

Some access points in the library will be easily identified by their antennas. For other areas of the campus there are maps that can be used to identify APs.

6. Question to NetOps:

Which region on campus has the highest concentration of APs and least amount of interference?

Response from NetOps:

Most of the major buildings have APs and the dormitories will soon have them activated. Fuller labs has the most number of APs; however, the design of the building does not make it a suitable candidate for preliminary work. The library has seven APs, two on each floor and one in the lower level. This building would be the most suitable to start working with because of the layout, design, and construction of the building. The access points are all the same model, the Symbol 4121, which should make initial work easier.

7. Question to NetOps:

What would be the best way to measure the precise location of each AP? Does WPI have a GPS device that we could use to determine each location?

Response from NetOps:

The precise location of the APs on the WPI campus is not documented anywhere. For the purpose of this MQP, the best way to measure the co-ordinates of an AP is to stand under it with a GPS unit and record the location.

For buildings like the library, another generic way to estimate the location is by counting the tiles on the ceiling from end to end.

8. Question to NetOps:

What precautions would you like us to take to safeguard the locations of the APs? We are considering using longitude/latitude for determining locations of AP and then map that coordinate system onto our own local coordinate system. Depending on whether an administrator or general version of the software is being used the locations of the APs could be hidden.

Response from NetOps:

Network Operations does not take any specific steps to safeguard this information, and it is not a very high priority to protect. But for obvious security concerns some steps should be taken in the architecture to give some level of protection to the AP list.

Making this information viewable for an Administrator but not for a general user, or doing something similar to this is a good idea.

Also, the list of APs should be designed such that it can be edited for adding or removing APs easily.

9. Question to NetOps:

Can we use software like NetStumbler and eventually our own software, and what kind of clearance do we need from you?

Response from NetOps:

For the purpose of this project, utilities like NetStumbler can be used to get information about the wireless network. Although, for anything trying anything new or different, Network Operations should be informed first.

9.2 WRAPI and LWW Build Guide

We compiled and built the WRAPI DLL from the source code provided on the WRAPI website [WRAPIweb]. In this section we go through where to acquire the files and how to build the WRAPI DLL and a Win32 Console application using WRAPI. There are also instructions for building the Locus LWW (Locus WRAPI Wrapper) DLL that also uses WRAPI. These steps have been tested for Visual Studio .NET 2003, but they should be similar for Visual Studio .NET 2002 (VS.NET). The CD-ROM supplement includes all the mentioned files as well as the VS.NET project files.

9.2.1 Required Files

The following files can be downloaded from the WRAPI website located at http://ramp.ucsd.edu/pawn/wrapi/download.html or can be found on the CD-ROM supplement. The Zip file contains the source code for the DLL and the CPP file is source code for a test application.

- WRAPIb2.0.zip

You will need the following files from the Zip file:

- WRAPI.cpp
- WRAPI.h
- WRAPIExports.h

In addition to the WRAPI files you will need the following file from the Windows XP SP1 DDK:

- nuiouser.h

To build the LWW DLL you will need the following files from the Locus source code found on the CD-ROM supplement:

- WRAPITest.cpp
- LWW.cpp
- edu_wpi_cs_mqp_locus_LocusJWRAPI.h

9.2.2 Steps for Building the WRAPI DLL

1. Create a new MFC DLL project named WRAPI
2. Copy these files to the project directory and add them to the as existing items:
    a. WRAPI.cpp
    b. WRAPI.h
    c. WRAPIExports.h
3. Add nuiouser.h to project directory or set the include path to point to it
4. In stdafx.h comment out section for _WIN32_WINDOWS
5. Under the Project Properties : Configuration Properties:
    a. Under C/C++ : Code Generation:
       Specify Multi-threaded Debug DLL
    b. Under C/C++ : Preprocessor : Preprocessor Definitions:
       Add "WRAPI_EXPORTS"
    c. Under C/C++ : Code Generation : Enable Minimal Rebuild:
       Set to No
    d. Under C/C++ : Language : Treat wchar_t as Built-in Type:
       Set to No
6. You should now be ready to build the project by selecting build

9.2.3 Steps for Building the Test Application

1. Create a new win32 console project named app
2. In project wizard check empty project
3. Copy these files to the project directory and add them to the project as existing items:
    a. WRAPITest.cpp
    b. WRAPIExports
    c. stdafx
4. Under the Project Properties : Configuration Properties:
    a. Under General : Use of MFC :
       Specify use MFC in a Shared DLL
    b. Under C/C++ : Code Generation :
       Specify Multi-threaded Debug DLL (same as in WRAPI DLL project)
    c. Under C/C++ : Precompiled Headers : Create/Use Precompiled Header:
       Specify Not Using Precompiled Header

   d. Under Linker : Input : Additional Dependencies:
      Specify WRAPI.lib
7. You should now be ready to build the project by selecting build
8. Before running the application type these two command at the command prompt: “net start ndisuio” and “net stop WZCSVC”

9.2.4 Steps for Building the LWW DLL

1. Create a new MFC DLL project named LWW
2. Copy these files to the project directory and add them to the as existing items:
   a. LWW.cpp
   b. WRAPIExports.h
   c. edu_wpi_cs_mqp_locus_LocusJWRAPI.h
3. In stdafx.h comment out section for _WIN32_WINDOWS
4. Under the Project Properties select Configuration Properties:
   a. Under General: Configuration Type:
      Specify Dynamic Library (.dll)
   a. Under C/C++ : Code Generation:
      Specify Multi-threaded Debug DLL
   b. Under C/C++ : Preprocessor : Preprocessor Definitions:
      Add "WRAPI_EXPORTS"
   c. Under C/C++ : Code Generation : Enable Minimal Rebuild:
      Set to No.
   d. Under C/C++ : Language : Treat wchar_t as Built-in Type:
      Set to No.
   e. Under Linker : Input : Additional Dependencies:
      Specify WRAPI.lib
5. You should now be ready to build the project by selecting build.
6. Refer to LocusJWRAPI.java for an example on how to integrate this DLL into Java.

9.3 WRAPI Hardware Independence Investigation

WRAPI is supposed to be a hardware independent API. However, during the development of Locus we observed behavior that would indicate otherwise. The results of our investigation are presented here, and based on these results we recommend further testing of WRAPI.

9.3.1 WRAPI Hardware and Software Requirements

The following overview and requirements are taken directly from the WRAPI website is taken from the WRAPI web site [WRAPIweb]:

> WRAPI is a software library that allows applications running in user space on mobile end stations to query information about the IEEE 802.11 network they are attached to. WRAPI 1.0 is implemented on the Windows XP operating system and is a hardware-independent tool that works with any IEEE 802.11b wireless network hardware vendor.
>
> **The hardware requirements for running WRAPI are as follows:**
> - A wireless LAN based on the IEEE 802.11b standard, configured with one or more access points in infrastructure mode.
> - A high-performance laptop or workstation with an X86 processor.
> - Wireless Network adapters (NICs) from any hardware vendor for an IEEE 802.11b-based wireless LAN.
>
> **The software requirements are as follows:**
> - Windows XP operating system.
> - Windows XP miniport drivers for the NIC
> - Windows XP DDK (driver development kit)

9.3.2 Laptop Specifications

Locus was developed and tested on two different laptops both meeting the WRAPI hardware and software requirements. The specifications of the laptops are as shown in Table 10.

Dell	HP
Dell Inspiron 8500	HP ZT1000
Intel Pentium 4-M 2.4 GHz	Intel Pentium 4-M
1.00 GB RAM	256 MB RAM
Microsoft Windows XP Pro, SP1	Microsoft Windows XP Pro, SP1
Dell TrueMobile 1300 (Broadcom chipset)	MS Wireless Adaptor MN-520 (Prism2 chipset)

Table 10: Development and Test Laptop Specifications

9.3.3 Details of the WRAPI GetAPList() method

The WRAPI DLL provides a method called WRAPIGetAPList() which calls the method GetAPList() of the CWRAPIApp object (this object implements the WRAPI code). GetAPList() creates a buffer in which it stores the results returned from the DeviceIOControl() call. After successfully building the application it was tested on both the Dell and the HP laptops. The test application makes a call to WRAPIGetAPList() which is supposed to return a list of APs and their signal strengths. Our tests returned different results on each system. On the Dell, GetAPList() returns the correct number of APs; however, it only displays the correct MAC address and signal strength value for the first AP in the list. When the same exact code is tested on the HP laptop, it returns the results expected: a list of AP MAC addresses and signal strengths.

When researching how to retrieve signal strength data we discovered that similar projects, have had issues in the past when dealing with wireless network adaptors with different chipsets. After looking up the adaptors in the Dell and the HP we discovered that they had different chipsets [HARDCweb], [CHIPSweb]. Dell's TrueMobile 1300 adapter used the Broadcom chipset, while the Microsoft MN-520 adapter used the Prism2 chipset. We then tested application on the Dell but this time using the Microsoft MN-520 adaptor instead of the built in Dell adaptor. This time the results returned were as expected.

When we debugged the GetAPList() function we discovered that the buffer in the function contains different blocks data depending on which of the two systems we were debugging. The function assumes that the DeviceIOControl() method returns data in a certain format and stores it in the buffer. GetAPList() assumes that the buffer has one block of good data corresponding to one AP followed by another good block of data. What we observed is that on the Dell the buffer will contain a block of good data at the beginning but following it, there will be a mix of good and bad blocks. However, on the HP the buffer will contain good blocks followed by good blocks. There is a casting inside GetAPList() which converts the buffer to a list, which is subsequently passed on to the calling function.

Debugging Steps:

1. GetAPList(AP_DATA **ppAP_data, long *plItems) is called.
2. We have these local variables:

UCHAR	QueryBuffer[1024];
PNDISUIO_QUERY_OID	pQueryOid;
PNDISUIO_SET_OID	pSetOid;
PNDIS_802_11_BSSID_LIST	pBssid_List;
3. After two calls to DeviceIoControl() the QueryBuffer is filled in.
4. QueryBuffer contains the AP mac addresses that we want plus other stuff.
5. The following line creates a local list from the buffer:
   pBssid_List = (PNDIS_802_11_BSSID_LIST)pQueryOid->Data;
6. And the following line gets the number of items in that list:
   *plItems = pBssid_List->NumberOfItems;
7. Some memory is allocated in the calling function (that calls GetAPList()).
8. Elements of pBssid_list are copied to the calling function list.
9. When QueryBuffer contains invalid data then pBssid_List will contain invalid data, and so will the list in the calling function.

9.3.4 Buffer Data Organization

The data in the buffer is made up of blocks with each block corresponding to an observed AP. Each block of data is 104 bytes long. The MAC address of each AP is stored starting at the 13th byte of each block, and is 6 bytes long. Table 11: Buffer Data Organization summarizes, for four consecutive blocks, the organization of the buffer returned by the DeviceIOControl() method.

Block	Block Range	MAC Address Blocks
1	0 – 103	12 – 17
2	104 – 207	116 – 121
3	208 – 311	220 – 225
4	312 – 415	324 – 329

Table 11: Buffer Data Organization

The following are the set of MAC addresses that could be contained in the buffer. *At the request of network operations the AP MAC addresses that we worked with have been scrambled to meet security and privacy concerns.*

0.32.105.2.97.104
0.103.107.146.27.45
0.32.105.2.90.101
0.103.107.146.28.106
0.32.105.2.93.102
0.32.105.2.89.100
0.103.107.54.119.159

9.3.5 HP Laptop Buffer

```
[0]   23 '?'      [54]  0         [108] 0         [162] 0
[1]   2 '?'       [55]  0         [109] 0         [163] 0
[2]   1 '?'       [56]  1 '?'     [110] 177 '±'   [164] 206 'Î'
[3]   13 '?'      [57]  0         [111] 199 'Ç'   [165] 255 'ÿ'
[4]   4 '?'       [58]  0         [112] 104 'h'   [166] 255 'ÿ'
[5]   0           [59]  0         [113] 0         [167] 255 'ÿ'
[6]   0           [60]  189 '½'   [114] 0         [168] 1 '?'
[7]   0           [61]  255 'ÿ'   [115] 0         [169] 0
[8]   104 'h'     [62]  255 'ÿ'   [116] 0         [170] 0
[9]   0           [63]  255 'ÿ'   [117] 103 'g'   [171] 0
[10]  0           [64]  1 '?'     [118] 107 'k'   [172] 32 ' '
[11]  0           [65]  0         [119] 146 '’'   [173] 0
[12]  0           [66]  0         [120] 28 '?'    [174] 0
[13]  103 'g'     [67]  0         [121] 106 'j'   [175] 0
[14]  107 'k'     [68]  32 ' '    [122] 0         [176] 100 'd'
[15]  146 '’'     [69]  0         [123] 0         [177] 0
[16]  27 '?'      [70]  0         [124] 20 '?'    [178] 0
[17]  45 '-'      [71]  0         [125] 0         [179] 0
[18]  0           [72]  100 'd'   [126] 0         [180] 0
[19]  0           [73]  0         [127] 0         [181] 0
[20]  20 '?'      [74]  0         [128] 87 'W'    [182] 0
[21]  0           [75]  0         [129] 80 'P'    [183] 0
[22]  0           [76]  0         [130] 73 'I'    [184] 50 '2'
[23]  0           [77]  0         [131] 45 '-'    [185] 95 '_'
[24]  87 'W'      [78]  0         [132] 87 'W'    [186] 0
[25]  80 'P'      [79]  0         [133] 105 'i'   [187] 0
[26]  73 'I'      [80]  56 '8'    [134] 114 'r'   [188] 16 '?'
[27]  45 '-'      [81]  94 '^'    [135] 101 'e'   [189] 0
[28]  87 'W'      [82]  0         [136] 108 'l'   [190] 0
[29]  105 'i'     [83]  0         [137] 101 'e'   [191] 0
[30]  114 'r'     [84]  16 '?'    [138] 115 's'   [192] 0
[31]  101 'e'     [85]  0         [139] 115 's'   [193] 0
[32]  108 'l'     [86]  0         [140] 45 '-'    [194] 0
[33]  101 'e'     [87]  0         [141] 78 'N'    [195] 0
[34]  115 's'     [88]  0         [142] 101 'e'   [196] 0
[35]  115 's'     [89]  0         [143] 116 't'   [197] 0
[36]  45 '-'      [90]  0         [144] 119 'w'   [198] 0
[37]  78 'N'      [91]  0         [145] 111 'o'   [199] 0
[38]  101 'e'     [92]  0         [146] 114 'r'   [200] 0
[39]  116 't'     [93]  0         [147] 107 'k'   [201] 0
[40]  119 'w'     [94]  0         [148] 0         [202] 0
[41]  111 'o'     [95]  0         [149] 0         [203] 0
[42]  114 'r'     [96]  0         [150] 0         [204] 1 '?'
[43]  107 'k'     [97]  0         [151] 0         [205] 0
[44]  0           [98]  0         [152] 0         [206] 0
[45]  0           [99]  0         [153] 0         [207] 0
[46]  0           [100] 1 '?'     [154] 0         [208] 130 '‚'
[47]  0           [101] 0         [155] 0         [209] 132 '„'
[48]  0           [102] 0         [156] 0         [210] 11 '?'
[49]  0           [103] 0         [157] 0         [211] 22 '?'
[50]  0           [104] 130 '‚'   [158] 0         [212] 0
[51]  0           [105] 132 '„'   [159] 0         [213] 0
[52]  0           [106] 11 '?'    [160] 1 '?'     [214] 254 'þ'
[53]  0           [107] 22 '?'    [161] 0         [215] 107 'k'
```

```
[216] 104 'h'       [273] 0             [330] 0             [387] 0
[217] 0             [274] 0             [331] 0             [388] 0
[218] 0             [275] 0             [332] 20 '?'        [389] 0
[219] 0             [276] 32 ' '        [333] 0             [390] 0
[220] 0             [277] 0             [334] 0             [391] 0
[221] 32 ' '        [278] 0             [335] 0             [392] 44 ','
[222] 105 'i'       [279] 0             [336] 87 'W'        [393] 96 '`'
[223] 2 '?'         [280] 100 'd'       [337] 80 'P'        [394] 0
[224] 90 'Z'        [281] 0             [338] 73 'I'        [395] 0
[225] 101 'e'       [282] 0             [339] 45 '-'        [396] 16 '?'
[226] 0             [283] 0             [340] 87 'W'        [397] 0
[227] 0             [284] 0             [341] 105 'i'       [398] 0
[228] 20 '?'        [285] 0             [342] 114 'r'       [399] 0
[229] 0             [286] 0             [343] 101 'e'       [400] 0
[230] 0             [287] 0             [344] 108 'l'       [401] 0
[231] 0             [288] 44 ','        [345] 101 'e'       [402] 0
[232] 87 'W'        [289] 96 '`'        [346] 115 's'       [403] 0
[233] 80 'P'        [290] 0             [347] 115 's'       [404] 0
[234] 73 'I'        [291] 0             [348] 45 '-'        [405] 0
[235] 45 '-'        [292] 16 '?'        [349] 78 'N'        [406] 0
[236] 87 'W'        [293] 0             [350] 101 'e'       [407] 0
[237] 105 'i'       [294] 0             [351] 116 't'       [408] 0
[238] 114 'r'       [295] 0             [352] 119 'w'       [409] 0
[239] 101 'e'       [296] 0             [353] 111 'o'       [410] 0
[240] 108 'l'       [297] 0             [354] 114 'r'       [411] 0
[241] 101 'e'       [298] 0             [355] 107 'k'       [412] 1 '?'
[242] 115 's'       [299] 0             [356] 0             [413] 0
[243] 115 's'       [300] 0             [357] 0             [414] 0
[244] 45 '-'        [301] 0             [358] 0             [415] 0
[245] 78 'N'        [302] 0             [359] 0             [416] 130 ','
[246] 101 'e'       [303] 0             [360] 0             [417] 132 '„'
[247] 116 't'       [304] 0             [361] 0             [418] 11 '?'
[248] 119 'w'       [305] 0             [362] 0             [419] 22 '?'
[249] 111 'o'       [306] 0             [363] 0             [420] 0
[250] 114 'r'       [307] 0             [364] 0             [421] 0
[251] 107 'k'       [308] 1 '?'         [365] 0             [422] 138 'Š'
[252] 0             [309] 0             [366] 0             [423] 45 '-'
[253] 0             [310] 0             [367] 0             [424] 204 'Ì'
[254] 0             [311] 0             [368] 1 '?'         [425] 204 'Ì'
[255] 0             [312] 130 ','       [369] 0             [426] 204 'Ì'
[256] 0             [313] 132 '„'       [370] 0             [427] 204 'Ì'
[257] 0             [314] 11 '?'        [371] 0             [428] 204 'Ì'
[258] 0             [315] 22 '?'        [372] 168 '¨'       [429] 204 'Ì'
[259] 0             [316] 0             [373] 255 'ÿ'       [430] 204 'Ì'
[260] 0             [317] 0             [374] 255 'ÿ'       [431] 204 'Ì'
[261] 0             [318] 1 '?'         [375] 255 'ÿ'       [432] 204 'Ì'
[262] 0             [319] 34 '"'        [376] 1 '?'         [433] 204 'Ì'
[263] 0             [320] 104 'h'       [377] 0             [434] 204 'Ì'
[264] 1 '?'         [321] 0             [378] 0             [435] 204 'Ì'
[265] 0             [322] 0             [379] 0             [436] 204 'Ì'
[266] 0             [323] 0             [380] 32 ' '        [437] 204 'Ì'
[267] 0             [324] 0             [381] 0             [438] 204 'Ì'
[268] 174 '®'       [325] 32 ' '        [382] 0             [439] 204 'Ì'
[269] 255 'ÿ'       [326] 105 'i'       [383] 0             [440] 204 'Ì'
[270] 255 'ÿ'       [327] 2 '?'         [384] 100 'd'
[271] 255 'ÿ'       [328] 89 'Y'        [385] 0
[272] 1 '?'         [329] 100 'd'       [386] 0
```

9.3.6 Dell Laptop Buffer

```
[0]   23 '?'        [54]  0             [108] 0             [162] 11 '?'
[1]   2 '?'         [55]  0             [109] 0             [163] 22 '?'
[2]   1 '?'         [56]  1 '?'         [110] 0             [164] 3 '?'
[3]   13 '?'        [57]  0             [111] 0             [165] 1 '?'
[4]   2 '?'         [58]  0             [112] 0             [166] 6 '?'
[5]   0             [59]  0             [113] 0             [167] 7 '?'
[6]   0             [60]  221 'Ý'       [114] 0             [168] 6 '?'
[7]   0             [61]  255 'ÿ'       [115] 0             [169] 85 'U'
[8]   204 'Ì'       [62]  255 'ÿ'       [116] 0             [170] 83 'S'
[9]   0             [63]  255 'ÿ'       [117] 0             [171] 32 ' '
[10]  0             [64]  1 '?'         [118] 0             [172] 1 '?'
[11]  0             [65]  0             [119] 0             [173] 11 '?'
[12]  0             [66]  0             [120] 88 'X'        [174] 30 '?'
[13]  103 'g'       [67]  0             [121] 0             [175] 173 '-'
[14]  107 'k'       [68]  32 ' '        [122] 0             [176] 15 '?'
[15]  146 '’'       [69]  0             [123] 0             [177] 0
[16]  28 '?'        [70]  0             [124] 142 'Ž'       [178] 103 'g'
[17]  106 'j'       [71]  0             [125] 151 '—'       [179] 107 'k'
[18]  0             [72]  100 'd'       [126] 51 '3'        [180] 2 '?'
[19]  0             [73]  0             [127] 76 'L'        [181] 0
[20]  20 '?'        [74]  0             [128] 1 '?'         [182] 0
[21]  0             [75]  0             [129] 0             [183] 0
[22]  0             [76]  0             [130] 0             [184] 11 '?'
[23]  0             [77]  0             [131] 0             [185] 0
[24]  87 'W'        [78]  0             [132] 100 'd'       [186] 0
[25]  80 'P'        [79]  0             [133] 0             [187] 0
[26]  73 'I'        [80]  136 'ˆ'       [134] 17 '?'        [188] 21 '?'
[27]  45 '-'        [81]  47 '/'        [135] 0             [189] 179 '³'
[28]  87 'W'        [82]  37 '%'        [136] 0             [190] 232 'è'
[29]  105 'i'       [83]  0             [137] 20 '?'        [191] 63 '?'
[30]  114 'r'       [84]  16 '?'        [138] 87 'W'        [192] 221 'Ý'
[31]  101 'e'       [85]  0             [139] 80 'P'        [193] 18 '?'
[32]  108 'l'       [86]  0             [140] 73 'I'        [194] 0
[33]  101 'e'       [87]  0             [141] 45 '-'        [195] 80 'P'
[34]  115 's'       [88]  0             [142] 87 'W'        [196] 242 'ò'
[35]  115 's'       [89]  0             [143] 105 'i'       [197] 1 '?'
[36]  45 '-'        [90]  0             [144] 114 'r'       [198] 1 '?'
[37]  78 'N'        [91]  0             [145] 101 'e'       [199] 0
[38]  101 'e'       [92]  0             [146] 108 'l'       [200] 0
[39]  116 't'       [93]  0             [147] 101 'e'       [201] 80 'P'
[40]  119 'w'       [94]  0             [148] 115 's'       [202] 242 'ò'
[41]  111 'o'       [95]  0             [149] 115 's'       [203] 0
[42]  114 'r'       [96]  0             [150] 45 '-'        [204] 0
[43]  107 'k'       [97]  0             [151] 78 'N'        [205] 0
[44]  0             [98]  0             [152] 101 'e'       [206] 1 '?'
[45]  0             [99]  0             [153] 116 't'       [207] 0
[46]  0             [100] 1 '?'         [154] 119 'w'       [208] 0
[47]  0             [101] 0             [155] 111 'o'       [209] 103 'g'
[48]  0             [102] 0             [156] 114 'r'       [210] 107 'k'
[49]  0             [103] 0             [157] 107 'k'       [211] 0
[50]  0             [104] 130 '‚'       [158] 1 '?'         [212] 204 'Ì'
[51]  0             [105] 132 '„'       [159] 4 '?'         [213] 0
[52]  0             [106] 11 '?'        [160] 130 '‚'       [214] 0
[53]  0             [107] 22 '?'        [161] 132 '„'       [215] 0
```

```
[216] 0
[217] 103 'g'
[218] 107 'ø'
[219] 146 '’'
[220] 27 '?'
[221] 45 '-'
[222] 0
[223] 0
[224] 20 '?'
[225] 0
[226] 0
[227] 0
[228] 87 'W'
[229] 80 'P'
[230] 73 'I'
[231] 45 '-'
[232] 87 'W'
[233] 105 'i'
[234] 114 'r'
[235] 101 'e'
[236] 108 'l'
[237] 101 'e'
[238] 115 's'
[239] 115 's'
[240] 45 '-'
[241] 78 'N'
[242] 101 'e'
[243] 116 't'
[244] 119 'w'
[245] 111 'o'
[246] 114 'r'
[247] 107 'k'
[248] 0
[249] 0
[250] 0
[251] 0
[252] 0
[253] 0
[254] 0
[255] 0
[256] 0
[257] 0
[258] 0
[259] 0
[260] 1 '?'
[261] 0
[262] 0
[263] 0
[264] 194 'Â'
[265] 255 'ÿ'
[266] 255 'ÿ'
[267] 255 'ÿ'
[268] 1 '?'
[269] 0
[270] 0
[271] 0
[272] 32 ' '
[273] 0
[274] 0
[275] 0
[276] 100 'd'
[277] 0
[278] 0
[279] 0
[280] 0
[281] 0
[282] 0
[283] 0
[284] 224 'à'
[285] 205 'Í'
[286] 36 '$'
[287] 0
[288] 16 '?'
[289] 0
[290] 0
[291] 0
[292] 0
[293] 0
[294] 0
[295] 0
[296] 0
[297] 0
[298] 0
[299] 0
[300] 0
[301] 0
[302] 0
[303] 0
[304] 1 '?'
[305] 0
[306] 0
[307] 0
[308] 130 '‚'
[309] 132 '„'
[310] 11 '?'
[311] 22 '?'
[312] 0
[313] 0
[314] 0
[315] 0
[316] 0
[317] 0
[318] 0
[319] 0
[320] 0
[321] 0
[322] 0
[323] 0
[324] 88 'X'
[325] 0
[326] 0
[327] 0
[328] 25 '?'
[329] 54 '6'
[330] 78 'N'
[331] 201 'É'
[332] 140 'Œ'
[333] 10 '?'
[334] 0
[335] 0
[336] 100 'd'
[337] 0
[338] 17 '?'
[339] 0
[340] 0
[341] 20 '?'
[342] 87 'W'
[343] 80 'P'
[344] 73 'I'
[345] 45 '-'
[346] 87 'W'
[347] 105 'i'
[348] 114 'r'
[349] 101 'e'
[350] 108 'l'
[351] 101 'e'
[352] 115 's'
[353] 115 's'
[354] 45 '-'
[355] 78 'N'
[356] 101 'e'
[357] 116 't'
[358] 119 'w'
[359] 111 'o'
[360] 114 'r'
[361] 107 'k'
[362] 1 '?'
[363] 4 '?'
[364] 130 '‚'
[365] 132 '„'
[366] 11 '?'
[367] 22 '?'
[368] 3 '?'
[369] 1 '?'
[370] 1 '?'
[371] 7 '?'
[372] 6 '?'
[373] 85 'U'
[374] 83 'S'
[375] 32 ' '
[376] 1 '?'
[377] 11 '?'
[378] 30 '?'
[379] 173 '-'
[380] 15 '?'
[381] 0
[382] 103 'g'
[383] 107 'k'
[384] 1 '?'
[385] 0
[386] 0
[387] 0
[388] 13 '?'
[389] 0
[390] 0
[391] 0
[392] 21 '?'
[393] 179 '³'
[394] 232 'è'
[395] 63 '?'
[396] 221 'Ý'
[397] 18 '?'
[398] 0
[399] 80 'P'
[400] 242 'ò'
[401] 1 '?'
[402] 1 '?'
[403] 0
[404] 0
[405] 80 'P'
[406] 242 'ò'
[407] 0
[408] 0
[409] 0
[410] 1 '?'
[411] 0
[412] 0
[413] 103 'g'
[414] 107 'k'
[415] 0
[416] 204 'Ì'
[417] 204 'Ì'
[418] 204 'Ì'
[419] 204 'Ì'
[420] 204 'Ì'
[421] 204 'Ì'
[422] 204 'Ì'
[423] 204 'Ì'
[424] 204 'Ì'
[425] 204 'Ì'
[426] 204 'Ì'
[427] 204 'Ì'
[428] 204 'Ì'
[429] 204 'Ì'
[430] 204 'Ì'
[431] 204 'Ì'
[432] 204 'Ì'
[433] 204 'Ì'
[434] 204 'Ì'
[435] 204 'Ì'
[436] 204 'Ì'
[437] 204 'Ì'
[438] 204 'Ì'
[439] 204 'Ì'
[440] 204 'Ì
```

9.4 Batik Guide

9.4.1 What is Batik?

Batik is Java based toolkit which is part of an open-source project between ILOG, Kodak and Sun. The documentation and downloads for Batik can be found on the Apache Batik website (http://xml.apache.org/batik). All links this document refers to can be accessed through the Batik website, unless specified otherwise.

9.4.2 Downloading and Installing Batik

Since Batik is based on Java, it requires the Java Development Kit (JDK) to be installed on the computer prior to installing Batik. Also 'Java' should be included in the PATH environment for any Batik based programs to run.

Batik can be downloaded on its main project web site by clicking on the download link. The detailed instructions for downloading Batik are provided on the webpage as well as the CD-ROM supplement provided with this document. In order use the batik toolkit one needs to download the binary distribution containing compiled executable JAR files. This file is named as 'batik-version.zip' (e.g. batik-1.5.zip). After this file has been downloaded it can be extracted to a folder following the guidelines set forth on the webpage.

In order to use batik for reading SVG files, the executable JAR file for the Crimson XML parser is also required. The binary files for crimson are available for downloading at Crimsons project distribution page (http://xml.apache.org/dist/crimson/). The required file is named as 'crimson-version-bin.zip'. This file should be extracted to an appropriate path.

9.4.3 Using Batik with Java on the Command Line

As mentioned earlier Batik is a collection of a large number of JAR files. To successfully compile and run a Java program utilizing Batik features, these JAR files need to be included in the class path variable (CLASSPATH). The following batch file can be used to set the CLASSPATH after changing the Batik, Crimson and Java paths appropriately. This batch file sets up the class paths, compiles the Java application, runs it and then sets the class paths to what they originally were. The lines you may need to change are printed in boldface.

```
@echo off

REM Setting up paths
REM ----------------
set javapath=C:\j2sdk1.4.2_02
set batikpath=C:\batik\batik-1.5
set crimsonpath=C:\batik\crimson-1.1.3

REM Storing current classpaths
REM --------------------------
set tempClasspath=%CLASSPATH%
set tempCP=%CP%

REM Emptying current classpath
REM --------------------------
set CLASSPATH=
set CP=

REM Adding java tools.jar
REM ---------------------
set CLASSPATH=%javapath%\lib\tools.jar;

REM Adding main batik and crimson JARS
REM --------------------------------
set
CLASSPATH=%CLASSPATH%;%batikpath%\batik.jar;%crimsonpath%\crimson
.jar

REM Adding more JARS
REM -------------------
set CLASSPATH=%CLASSPATH%;%batikpath%\lib\batik-awt-
util.jar;%batikpath%\lib\batik-bridge.jar;%batikpath%\lib\batik-
css.jar;%batikpath%\lib\batik-dom.jar;%batikpath%\lib\batik-
ext.jar;%batikpath%\lib\batik-extension.jar;

REM Adding more JARS
REM -------------------
set CLASSPATH=%CLASSPATH%;%batikpath%\lib\batik-gui-
util.jar;%batikpath%\lib\batik-gvt.jar;%batikpath%\lib\batik-
parser.jar;%batikpath%\lib\batik-
script.jar;%batikpath%\lib\batik-svg-dom.jar

REM Adding more JARS
REM -------------------
set CLASSPATH=%CLASSPATH%;%batikpath%\lib\batik-
svggen.jar;%batikpath%\lib\batik-swing.jar;%batikpath%\lib\batik-
transcoder.jar;%batikpath%\lib\batik-
util.jar;%batikpath%\lib\batik-xml.jar

REM Adding current folder
REM ---------------------
set CLASSPATH=%CLASSPATH%;.
```

```
REM Settign CP = Classpath
REM ----------------------
set CP=%CLASSPATH%

echo Compiling...
pause
javac SVGApplication.java

echo Running...
pause
java SVGApplication

echo Terminating...
set CP=%tempClasspath%
set CP=%tempCP%
```

9.4.4 Using Batik with Eclipse

Batik can be used with Eclipse by making a few changes to the project properties. In the project properties window go to "Java Build Path" on the menu on the left. Go to the "Libraries" tab and click on "Add External JARS". The required Batik JAR files can be added using the JAR selection browser that opens. Once the Batik JARs are included Eclipse will be able to recognize batik includes and compile and run the programs.

9.4.5 Sample Programs

Sample programs can be found on the Batik project website under the tutorial web pages. After setting up the initial environment as described in this document any of the provided sample codes should successfully compile and run.

9.5 Locus Runtime Guide

Locus is an application written in the Java programming language, and it requires the Java Runtime Environment. The following sections provide detailed steps for running Locus.

9.5.1 Launching Locus

Locus can be launched from the DOS command line. To start up a DOS command window (console), Click 'Start', and then click 'Run'. In the 'Run' window, type 'command' or 'cmd' and press 'OK'. This will bring up the console. Change the working directory to the path where Locus was installed using the 'cd' command, for e.g. 'cd c:\somepath\Locus\'. Type 'runLocus' and press enter. This will launch the main locus GUI. Figure 16 shows a screenshot of Locus during runtime.

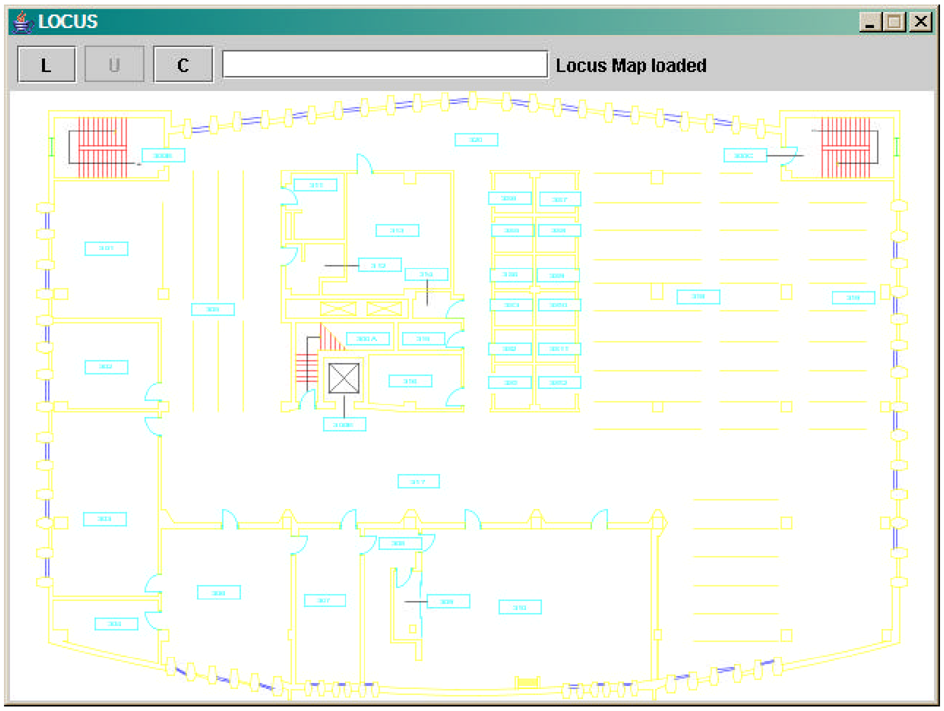

Figure 16: Locus Screenshot

The main GUI has two main components. The interactive panel on top consists of three buttons for performing different tasks, a text field for entering data and a label area for displaying messages. Below the panel is the graphics area that displays the floor maps. Once Locus starts up, the user has an option of either entering the Calibration mode or the Location mode. Both these modes are discussed further.

9.5.2 Calibration

The Calibration phase is necessary to "train" the system and provide it with recorded data which the location algorithm requires. To begin calibration, the user needs to enter the floor name in the text box on the GUI panel. The floor names follow a strict naming convention of "<institution-name>-<building-name>-floor-<floor number>". For example, the third floor of the library building in WPI would be named as, "wpi-library-floor-3". The application assumes that a map for the floor exists in the maps folder under the main Locus folder in SVG format with the same naming convention as described above (for example maps/wpi-library-floor-3.svg).

Calibration requires the user to go to ten different points on the floor, which are evenly spaced out, and take measurements with the application. After the user walks to the first point, he needs to click the "C" button on the panel. This would load the map of the floor that the user specified in the text box. After the map is loaded, the user is instructed to click on the map and indicate where the first point is. The user has to take four measurements facing in four different directions for every point. The text on the label area is updated to instruct the user of the next point number and the direction number whenever the user clicks on the map. Each time the user moves to a new location, the current position has to be indicated by clicking on the corresponding point on the map. After the user has gone through all the ten points, the calibration phase for this floor is complete and the user can either shut down the application or calibrate another floor following the same instructions as above.

9.5.3 Location Sensing

Before a user can use Locus for actual location sensing, the floor he is on must have been calibrated and the calibrated data should exist on the local machine. To begin Location sensing the user has to press the “L” button which would determine the floor on which the user is located. The map for this floor is then loaded and displayed in the graphics area. Clicking on “L” also enables the “U” button which is used for updating the user’s location on the map. On clicking the “U” button the user’s location is determined and displayed on the floor map with a red dot. The “U” button can be clicked each time that the user wants to update his location on the map.

9.6 Locus Javadoc