Location Fingerprinting on Infrastructure 802.11 Wireless Local Area Networks (WLANs) using Locus*

Ali Taheri Arvinder Singh Emmanuel Agu

Department of Computer Science,
Worcester Polytechnic Institute,
Worcester, MA 01609, USA

Abstract

Location fingerprinting is a technique for location sensing on 802.11 Wireless Local Area Networks (WLANs), using commodity WLAN cards and no additional hardware tags. Location fingerprinting is a two-phase process. First, a radio map of observed Signal Strength (SS) values from different locations are recorded during an offline calibration phase. Then, in real time, SS values observed at a users mobile device are compared to the radio map values using proximity-matching algorithms in order to infer current user locations. We present Locus, a software-only, platform-independent tool for location fingerprinting on 802.11 WLANs. Locus has an object-oriented design, and is implemented in Java with graphical display in Scalable Vector Graphics (SVG). While several proximity-matching algorithms have been proposed, very little research has evaluated their performance on existing wireless networks. Using Locus as a framework, we experimentally compared the performance of two proposed proximity-matching algorithms and also quantified the variance of observed SS values on five mobile devices. We find that in practice, due to issues such as access point occlusion from certain locations, in-building interference effects on signal strengths, calibration and signal strength detection difficulties on certain mobile platforms, the behavior of proximity-matching algorithms can be mobile platform and wireless network dependent, and can not always be generalized.

1. Introduction

IEEE 802.11 Wireless Local Area Networks (WLANs) have become widely deployed and are fuelling a wide range of location-aware computing applications. Accurate user location information enables a wide range of location-dependent applications. Users may request to print to the closest printer, tourists may receive rich contextual information about their current location and in geocasting, network packets may be routed to users at a specified location.

The Global Positioning System (GPS) is currently the de facto standard for location sensing in outdoor wireless environments. However, GPS does not work well in indoor WLAN environments. Moreover, GPS requires dedicated hardware. Location sensing on 802.11 networks using commodity Network Interface Cards (NICs) and no additional tags or hardware is attractive due to its reduced cost, easier deployment and use, and improved indoor functionality. Additionally, a software-only solution can be integrated as a location sensing module of a larger context-aware application on infrastructure wireless LANs.

Location fingerprinting is an increasingly popular location sensing technique which involves a two-phase process. First, received signal strength values (or radio map) from all Access Points (APs) at selected locations in a building are recorded during an offline calibration phase. Then, during the online phase, proximity-based matching algorithms are used to infer a user's location by comparing the current observed signal strength value to the pre-recorded values in the radio map database.

We present Locus, a software-only platform independent tool for location sensing on 802.11 wireless LANs with display in Scalable Vector Graphics (SVG). Locus was designed using object-oriented techniques and implemented in Java. Specific platform-independent modules are clearly abstracted and wrapped in Java Native Interface (JNI) classes. Specifically, we integrated WRAPI, a third party library for retrieving signal strength into our framework using JNI. Using Locus as a framework, we implemented and experimentally compared the accuracy of two previously proposed proximity-matching algorithms for inferring user locations, namely, a simple threshold elimination method and

* Funding for this work was provided in part by the National Science Foundation grant number 0303592

a euclidean-distance-based algorithm. We also measured the variance of observed signal strength values on different laptops.

Locus was extensively tested in the Worcester Polytechnic Institute (WPI) library building. We found that in practice, due to issues such as access point occlusion from certain locations, in-building interference effects on signal strengths, calibration and signal strength detection difficulties on certain mobile platforms, the behavior of proximity-matching algorithms can be mobile platform and wireless network dependent, and can not always be generalized. We are currently using Locus as a foundation to develop a wide range of location-aware applications and modules within our ubiquitous computing university campus environment and plan to release Locus to the research community.

The rest of this paper is as follows. Section 2 presents a taxonomy of location sensing techniques, section 3 gives an overview of the location fingerprinting technique used in our Locus tool. Section 4 presents an overview of Locus including a usage scenario, our design goals, envisioned levels of localization, design and implementation. Section 5 outlines our main results including performance analysis of Locus. Section 6 discusses our results, section 7 outlines related work and section 8 are our conclusions.

2. Location Sensing Taxonomy

There are several different approaches for determining the location of users on a wireless network. The most popular and commercially adopted approach is GPS which works best in outdoor locations [8].

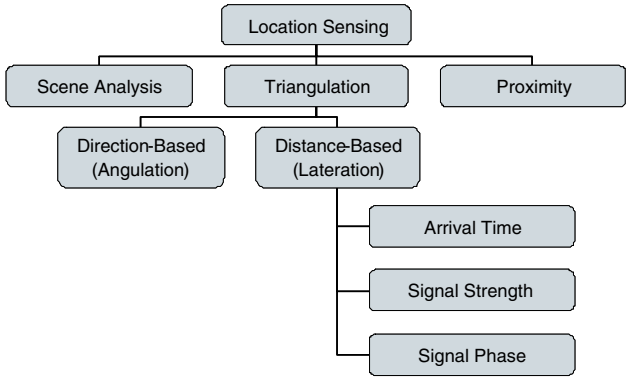

Figure 1. Location Sensing Taxonomy

Location sensing techniques can be divided into three general categories: scene analysis, triangulation and proximity. Figure 1 presents a categorization of the location sensing techniques. [5] and [8] provide a good background and explanation of these location sensing techniques.

3. Location Fingerprinting

Our goal was to utilize a location sensing technique that worked well indoors and did not require additional hardware as is required by GPS and some other tag and sensor based approaches. The approach that we adopted was location fingerprinting where we used different proximity algorithms to calculate the users location. An overview of the location fingerprinting approach is now presented.

Location Fingerprinting or proximity matching is a location sensing technique in which a users location is determined by using a known set of values and comparing them against an observed value. Our approach is two phased: offline or calibration and online or run time phases. During the calibration phase of this process, the MAC address and signal strength of up to three visible access points are measured by a user using a notebook computer. These values are stored in a tuple data structure and the tuple is associated with a known coordinate. The point-tuple association is stored in a database or radio map for future use. During the online phase, to determine user location, the signal strengths of visible access points are again measured and stored into a tuple. The tuple observed during runtime is then compared against tuples stored in the database. When a match is found then the coordinates of the stored tuple are returned. Various proximity-matching algorithms can then be used to compare the stored signal strength values to the observed real time values in order to infer a user's location. This approach allows us to determine the users location to within proximity of a pre-calibrated known location. By measuring more tuples during the calibration phase and minimizing the average distance between calibrated locations, the resolution of the process can be increased. Additionally, interpolation and signal propogation modeling can be used to infer locations between two pre-calibrated points, and improve accuracy.

Location fingerprinting is not immune to the effects of radio interference. However, since the pre-calibration values already include the effects of interference, building material construction and radio propagation effects, these effects are implicitly accounted for. In our system we assume that there will be sources of interference but for the most part the sources and amounts will not vary. For example we assume that the building structure will not change and new walls or large pieces of furniture will not be added or moved. In the event that such a change does occur that particular floor where the change occurred can be recalibrated. If the sources and amounts of interference do not change then it is reasonable to assume the same signal strength reading will be measured at a given point during different

times, though this value will represent the effects of interference caused by walls and furniture between the AP and the user. The amount of interference is irrelevant to our approach as long as the amount stays consistent. Furthermore the signal strength tuples for a given point should be unique for a given point and the same combination of three MAC addresses and signal strengths should not occur for more than one point. Once we were able to store unique tuples for each point that we went to during calibration then we were able to observe a tuple during run time and find the closest match amount the prerecorded tuples. The user location was then inferred to be the location at which the prerecorded tuple was measured.

Sources of error in location fingerprinting include the fact that observed signal strength fluctuates over time, and varies with location and user direction especially for the non-directional antennae that are common in mobile devices. Finally, since different mobile devices will return different nominal values for signal strength at the same location, calibration on one computer and inference on another will introduce errors.

4. Locus

Locus is a tool that we developed for indoor location sensing system using location finger printing techniques. In this section we present a typical usage scenario, our design goals and requirements including envisioned levels of localization, the object oriented system design, and details on the implementation of Locus.

4.1. Usage Scenario

In order to make our vision of Locus more concrete, a typical usage scenario is now presented.

John Smith is a new freshman at 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. Johns 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 Johns 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.

4.2. Design Goals

In designing our location sensing system, we laid out several design goals to guide our efforts. First of all, we wanted a *software-only*, *platform-independent* solution which could be easily configured, utilize commodity Network Interface Cards (NICs) and run on existing wireless LANs since many organizations such as WPI have already invested large amounts of money and time on the deployment of wireless 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.

We also wanted a *modular design* to ease concurrent development of components, design flexibility and expandability, and easier testing. In this project the graphics modules and location sensing modules should be developed and tested concurrently but also independently. 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. Finally, a modular design encourages the use of Locus as a location module in a larger context-aware application.

An *intuitive GUI* which would display the (x,y) coordinates of the users' locations in an intuitive and concise manner, was also desirable. Finally, since, it was expected that the system will require a certain amount of configuration before it is ready for run time, we wanted a system with a *simple setup and generation of building and floor maps*

4.3. Levels of Localization

We envisioned several that Locus would support multiple levels of localization. Specifically, Where Am I (WAI)?, Where Are You (WAY)?, and Where Are They (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. The usage scenario outlined in section 4.1 is based on WAI localization 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 clients 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 primarily focussed on the development of a WAI system which is the foundation for the other systems. WAY and WAT development is envisaged as future work.

4.4. Object-Oriented Design

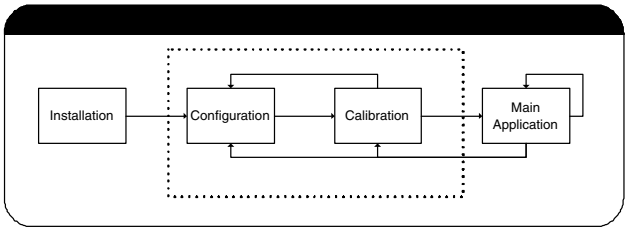

Figure 2. Flow of Locus Application

Figure 2 shows the flow of the Locus application. First a user retrieves and installs the Locus code base on her mobile device. Next comes a configuration phase in which she loads the necessary maps of the buildings of interest and assigns them names which conforms to a format which is described in [1]. Next comes the calibration phase in which the radio map, or database of signal strengths from different Access Points (APs) from different locations in the building/floor, is populated. Finally, Locus enters the main application in which it is able to infer the user's location based on received signal strengths.

In order to meet our goal of having a modular and flexible system, Locus was designed as different components or subsystems that interact with each other and can be implemented, tested, and changed independently with no impact on the rest of the system as long as the subsystem interfaces do not change. More details about the design of Locus can be found in [1].

4.5. Implementation Details

The Locus design was primarily implemented in Java in order to meet the platform independence goal. The agent subsystem uses the Wireless Research API (WRAPI) [9], a third-party module for retrieval of signal strengths. WRAPI was developed in C++ for the Microsoft Windows XP operating system. To clearly separate this system dependent component from the rest of the system, a Java Native Interface (JNI) class was created to handle the communication. If hardware communications or signal strength detection components are developed for other operating systems, they can be easily integrated into the system as long as they use the same JNI interface. The Locus graphical interface uses the Batik API for manipulating SVG files and content. Figure 4.5 shows Locus' architecture as implemented.

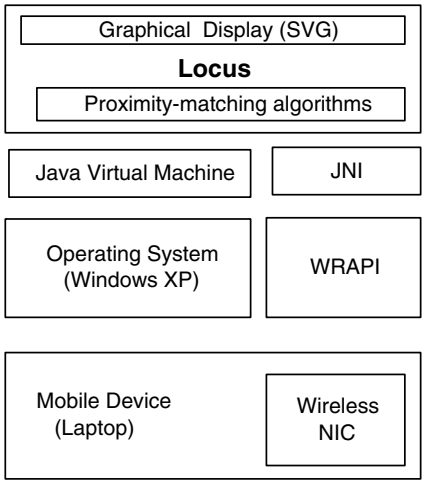

Figure 3. Locus Architecture

5. Results

In this section, our main results are summarized. Our methodology and algorithms allowed us to locate on what part of a floor of a building a user 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.

In addition to achieving our location sensing goals we were also able to create a high quality interface using SVG to display location. The user interface for Locus is designed using Java swing components and the Batik SVG Toolkit. The map area of the map is a mix of dynamic and static SVG content and is implemented as multiple layers. The bottom layer displays a static map of a building floor which has been converted from CAD drawings of actual floor plans. The layers on top of the map are transparent and used to

Subsystem	Function
Main	Application entry point and driver
Agent	Communications with wireless network card to retrieve AP signal strengths (SS)
Location	Algorithms for calculating location based on recorded and observed AP SS.
Graphics	Recording of coordinate during calibration and coordinate display at runtime
Network	Communication between different clients. Future development

Table 1. Locus Subsystem Overview

Figure 4. Locus Screenshot

display non-static information. Each layer can be dedicated to unique elements, and newer layers can be added on top as need be. For instance, resources such as printers could be placed on different layers. The composite SVG image is then generated at runtime using java calls to the Batik toolkit. Additionally there are zooming and panning features that allow users to easily navigate the map. This results in a high quality, interactive and highly customizable interface for displaying maps and other context related information. Figure 4 displays the current look of Locus.

5.1. Performance Analysis

This section contains results of performance tests of Locus at WPI. Worcester Polytechnic Institute (WPI) has an extensive data network and has been consistently ranked among Americas most wired colleges. [1] The Universitys network consists of various computers in labs, offices, dormitories and fraternities, and also other devices such as file and print servers and network printers.

1 Based on results of Yahoo Internet Life survey

Figure 5. Map of WPI Wireless LAN coverage

The WPI Wireless LAN currently has approximately 60 Access Points throughout campus, with the number soon expected to reach about 160. The WPI Wireless LAN uses the 802.11b standard. Most buildings on WPIs main campus are covered by the WPI Wireless LAN. Figure 5 highlights the areas on campus with wireless coverage. All buildings with wireless access are shown in red. 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.

We configured and tested Locus in the WPI Gordon library (building 13), which has seven APs, two on each floor and one on the lower level. Our performance tests included studying the variance in signal strengths in indoor locations since that is a possible cause for error in location sensing. We concentrated on two possible cases where such errors in signal strength could be critical to determining exact location. The first case focused on measuring the variance of signal using the same hardware setup on the same location but facing different directions. In the second case we used different laptops and measured signal on the same location and direction.

Signal strength variance measurements: Figures 6 and

	Model	Operating System	Wireless NIC	Wireless NIC chipset
Laptop 1	HP Pavilion ZT1000	Windows XP Professional	Microsoft MN-520	Prism2/2.5/3
Laptop 2	HP Pavilion ZT1185	Windows XP Professional	Trendware - TEW-226PC	Realtek
Laptop 3	Dell Inspiron 8500	Windows XP Professional	Dell TrueMobile 1300	Broadcom
Laptop 4	IBM ThinkPad	Windows XP Home	Linksys 802.11b PCMCIA	Prism2/2.5/3
Laptop 5	Dell Inspiron 8500	Windows XP Professional	Microsoft MN-520	Prism2/2.5/3

Table 2. Configuration of Laptops Tested

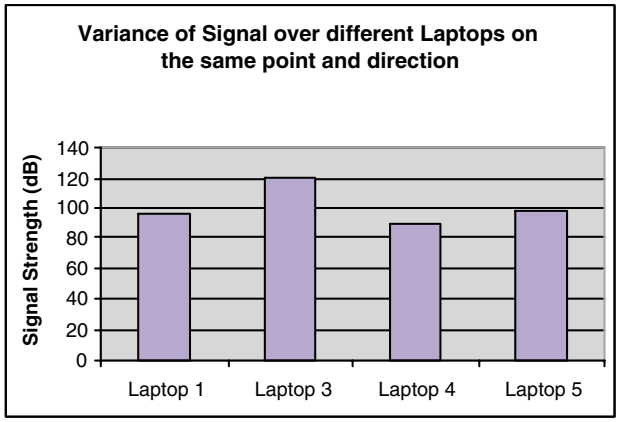

Figure 6. Signal variance on different laptops

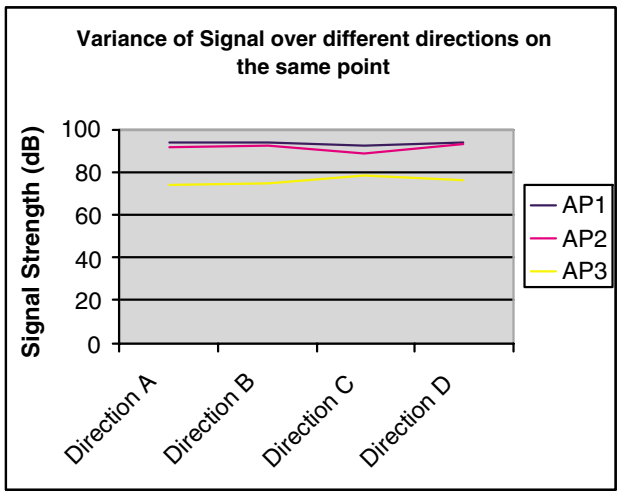

Figure 7. Signal variance with direction

7 illustrate the results of the tests that we conducted to investigate the variance of observed signal strength values on five different laptops. Table 2 gives the configurations of the laptops tested. Each laptop showed different signal strength values with the difference varying from around 3 units up to 31 units in the illustrated test case. The variance of signal strength over different directions was not relatively lower but still enough to affect accuracy. The data clearly indicated that signal strength does vary over direction, time and hardware and future algorithms for indoor location sensing should try and account for this error in order to improve accuracy.

Proximity-matching algorithm comparison: We also compared two matching algorithms to select one for use in locus, namely, least threshold and Euclidean distance [2] with plans to implement a third using joint probability distribution in the near future. The first two are similar in nature and measure distance between signal strength values for the relative Access Points. Starting with a high threshold value, the least threshold method eliminates all signal fingerprints that have a difference from the observed signal strength greater than the assumed threshold value. This threshold value is gradually reduced until only one match is left from the recorded signal fingerprints, which is the closest match. In the Euclidean distance method we measure the distance between signal values using the Pythagorean distance formula and the fingerprint having the least distance is the closest match. The formula for the Euclidean distance between two sets of signals $[x1, x2, x3]$ and $[y1, y2, y3]$ is given as $\sqrt{(x_1 - y_1) + (x_2 - y_2) + (x_3 - y_3)}$

Table 2 illustrates a test case comparing the two algorithms. Surprisingly, the results were identical for Euclidean Match as well as Threshold Matching with both giving the same error for the same test. The Euclidean method was still preferred over the Threshold match, since over a larger set of test runs the Threshold algorithm sometimes failed to bring up any matches at all. The joint probability method can further reduce errors shown by the previous algorithms. The joint probability method [6] compares the observed values against a larger set of data per fingerprint. At each point during the offline fingerprinting phase, multiple signal values are recorded and a probability value is attached to each unique signal values observed over a period of time. This way we have actual signal strength values instead of an average value per point. If most of the signal strength varia-

Test Number	Actual closest fingerprint	Euclidean match	Threshold match
T1	2	2	2
T2	2	2	2
T3	4	4	4
T4	5	5	5
T5	7	7	7
T6	9	**12**	**12**
T7	10	10	10
T8	11	11	11

Table 3. Comparison of proximity matching algorithms

tions are accounted for in the recorded data, then the online matching becomes less prone to errors.

6. Discussion

In this section we discuss some of our main achievements in this project and share some lessons learned and challenges including sources of error and mobile device and WLAN-specific difficulties encountered in the development process. We also 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.

Sources of error: 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 clients actual location. Runtime inaccuracy causes the clients 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. Next, some of the challenges we faced which we believe to be some of the sources for the inaccuracy issues, are discussed. The challenges we faced included the 1-AP problem, location by associated AP and callibration issues. These are now discussed in more detail.

1-AP Problem: During the implementation of the agent subsystem we ran into a problem which became known as the 1-AP bug. The WRAPI 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 in [1]

Location by Associated AP: In [1], 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, 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 rela-

tionship 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.

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, 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.

Overall, We are satisfied with the results we achieved for a proof-of-concept; however further testing and investigation into the challenges are recommended.

7. Related Work

Due to the the widespread adoption of 802.11 WLANs, indoor location sensing is currently an active area of research. The Ekahau Positioning Engine [4] and Newbury Networks [7] are two of the leading commercial products on the market. In academic literature, Radar [2] and Nibble [3] are two location sensing systems.

8. Conclusions

We have presented Locus, a software-only, platform-independent tool for tag-less location sensing on infrastructure 802.11 WLANs. Locus implements the location fingerprinting technique in Java with display in Scalable Vector Graphics (SVG). Locus was extensively tested in the Worcester Polytechnic Institute (WPI) library and was used as a framework for implementing and experimentally comparing two proposed proximity-matching algorithms. The observed signal strengths on different mobile devices were also compared to quantify cross-platform issues. It is our hope that the Locus tool shall serve as a good framework for comparing the performance of various proposed proximity-matching location fingerprinting algorithms on existing wireless LANs, as well as present a location sensing module for building context-aware applications on WLANs.

References

[1] Ali Taheri and Arvinder Singh, "LOCUS: Wireless LAN Location Sensing", Major Qualifying Project, Worcester Polytechnic Institute (WPI), January 2004.

[2] P Bahl, V Padmanabhan, "An In-Building RF-based User Location and Tracking System", Proc. INFOCOM, 2000.

[3] P Castro, P Chiu, T Kremenek and R Muntz, "Probabilistic Room Location Service for Wireless Networked Environments", in Proc. UBICOMP 2001

[4] Ekahau Positioning Engine, Ekahau, Finland. http://www.ekahau.com/products/positioningengine/

[5] J Hightower and G Borriello, Location Systems for Ubiquitous Computing. IEEE Computer,Aug 2001: 57-61.

[6] M Youssef, A Agrawala, and U Shankar, WLAN Location Determination via Clustering and Probability Distributions, in Proc. IEEE PerCom 2003.

[7] Newbury Networks, http://www.newburynetworks.com/

[8] K Pahlavan and P Krishnamurthy, "Principles of Wireless Networks", Prentice Hall PTR, 2002, pp 417-448.

[9] A Balachandran, Wireless Research API (WRAPI), University of California, San Diego, http://ramp.ucsd.edu/pawn/wrapi