Design and Implementation of an Extensible Sensing Service for Home Networks

by

Sayed Ahmed

A dissertation submitted in partial satisfaction of the requirements for the degree of

Master of Science

Department of Computer Science
Faculty of Graduate Studies
University of Manitoba

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES
*****
COPYRIGHT PERMISSION

DESIGN AND IMPLEMENTATION OF AN EXTENSIBLE SENSING SERVICE FOR HOME NETWORKS

BY

Sayed Ahmed

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of

Manitoba in partial fulfillment of the requirement of the degree

OF

MASTER OF SCIENCE

Sayed Ahmed © 2006

Abstract

Location determination is a basic, yet critical component for smart homes to provide customization of automated smart home applications to each user's needs. To date, there has been much research in location determination but little aiming at future automated smart home applications. Hence, this thesis focuses on the design, development, and evaluation of an effective location sensing system for smart home applications. The location sensing system consists of three components: a sensor network-based location determination system incorporated with user identities, a prototype location data storage system, and a middleware interface to the location determination system to provide interoperability among home automation applications. The sensing system has been deployed in a prototype home environment in one floor of TRLabs Winnipeg. The location accuracy, the system provides is quite reasonable for smart home applications. This thesis also identifies metrics affecting location accuracy, studies effects of those parameters on location accuracy, and shows the effects in graphical plots.

Acknowledgements

I would like to express my sincere gratitude to all who have contributed to complete this thesis. I would like to thank my supervisor Dr. Rasit Eskicioglu for his continuous guidance and encouragement to develop this work. I would definitely pay thanks to Professor Dr. Peter Graham for his valuable suggestions for my thesis. Also, I am thankful to professor Dr. Neil Arnason for his valuable guidance for my thesis proposal. I would also like to thank Dr. Jeff Diamond and Dr. Peter Graham for serving on my examining committee.

I would be greatly thankful to TRLabs Winnipeg, Canada for supporting me with a scholarship for my research.

Finally, a profound gratitude goes to all of my family members who provided continuous encouragement and emotional support to accomplish this endeavor.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Visions of the future often feature completely automated systems in homes, offices, industrial manufacturing, and similar environments where most of the activities and computing are operated automatically based on current user or environmental contexts, user habits, needs, and preferences. Very recently, this vision is becoming possible through the emergence of context-aware, ubiquitous, and pervasive computing. Pervasive computing provides distributed tools in our environment through which we can access information anytime, anywhere [10]. Ubiquitous computing uses intelligent computing devices embedded in the physical world and enables people to interact with computers more naturally even while moving. Ubiquitous computing enables devices to sense changes in their environments, and then to automatically adapt and act, based on these changes [44]. Context-aware computing, where context refers to the physical or social situations or states [30], aims to use the surrounding environment and user contexts to automate computing and activities to improve user experiences; . Advances in hardware for sensor nodes have made wireless sensor networks (WSNs)[1]

[1]A Wireless Sensor Network (WSN) is a mesh network of tiny wireless sensor nodes. Sensor nodes communicate among themselves using RF communication and sense the physical world. Sensor networks contain active sensors who have processing, sensing, and communication capabilities [44].

a promising underlying technology for pervasive and ubiquitous computing supporting the systems of various user and environmental contexts. Hence, context-aware computing along with ubiquitous and pervasive computing implemented using WSN technology has the potential to improve human experiences with computers and enable more automation in homes, offices, and similar environments [36].

These smart technologies, when applied to home environments, will enable many useful applications; including: home security, child and personal care, remote monitoring of homes, automatic operation and control of various household devices, entertainment devices, and temperature and lighting control units as well as personalized customization of user interfaces in the home. In an application scenario, a sensor network at home will track the location and movement of the home residents. The sensor network will also keep track of key in-home devices, and monitor environmental parameters (humidity, temperature, and light).

A home network that connects all devices, servers, and the sensor network, and which exploits knowledge of the locations of the home residents, will be better able to control the devices in the environment to improve each home occupant's experience. For example, a person might be watching channel X in the bedroom and then move to the kitchen and work there for 30 minutes; however, he does not want to miss the TV program. On moving to the kitchen, the TV in the kitchen could be automatically turned on and switched to channel X, guided by a sensor network and a context-aware ubiquitous computing system built into the home. Similarly, the temperature, light intensity and humidity of the kitchen might be set to match the user's personal preference. In this scenario, the sensor network tracks the user's location and the home network uses it to control various devices. Such smart homes and applications require several components such as:

- A home network that will connect all home appliances, sensors, and servers using wired or wireless technologies and, possibly, varying protocols
- Sensing devices to sense users and environmental contexts
- Identification of devices and users
- Determination and tracking of device and user locations
- Storage and retrieval systems for the sensed data, such as location, environmental parameters
- Awareness of the functionalities of the devices
- Awareness of user habits, preferences, and priorities among users
- Effective and efficient access to user-preferences
- Interoperability[2] among devices, services, and applications that use different protocols and specifications.

The Home Technologies Group (HTG) at TRLabs Winnipeg has adopted a framework for future smart homes as depicted in Figures 1.1 and 1.2 to do research on different aspects of smart homes. In the assumed environment, future smart homes will be built with sensor networks formed using many different kinds of sensors, intelligent devices, and protocols to monitor user locations, user movement, temperature, light level, humidity, and user activities. Home residents will carry tiny sensors as indicated by the mobile motes in Figure 1.1, whereas infrastructure sensors will be fixed as indicated by the fixed motes in Figure 1.1. Fixed motes will help in determining users' (mobile sensors) locations, monitoring the environment, and collecting

[2]The ability of devices and applications with different standards to communicate with each other in a heterogeneous computing environment.

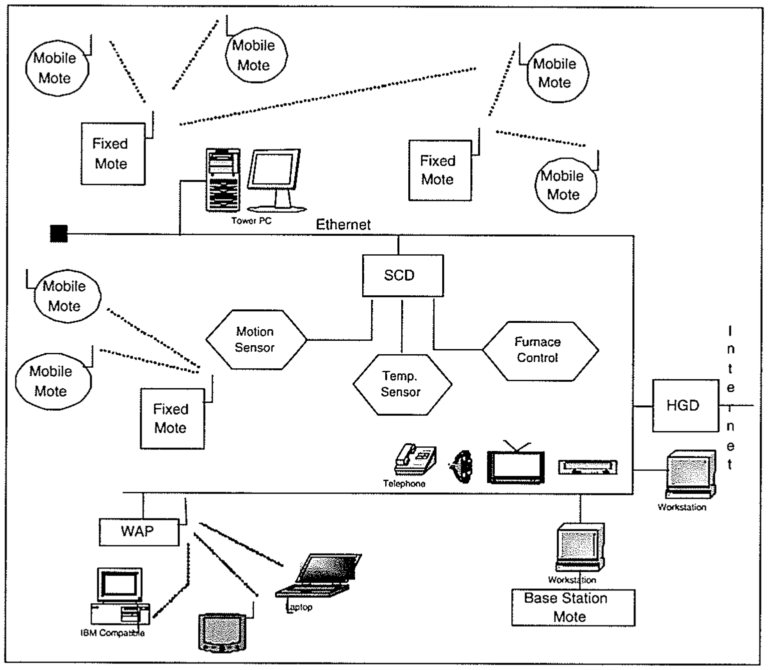


Figure 1.1: High Level View of the Envisioned Future Smart Home

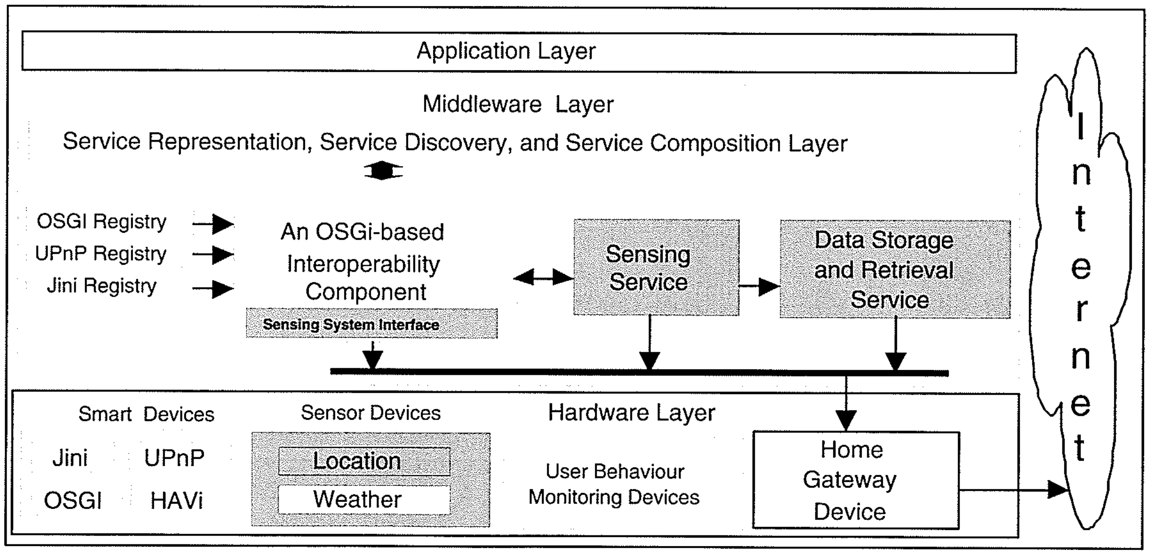


Figure 1.2: Envisioned Smart Home Framework

different context data concurrently. The sensed data will be fed to a base station sensor that is connected to a workstation. This workstation or the Home Gateway Device[3](as indicated by HGD) in Figure 1.1 can serve as a context server to control home automation applications. A Home Gateway Device, will also provide access to the home automation system from the Internet. The servers will store the sensed data for real-time querying and future use. Further, the servers will also store derived user-preferences. The servers will use environmental, and location data, and the stored user-preferences, to support context-based applications and control devices. Over longer periods, as users' behavior changes, the sensor system will use the collected data to update the stored information (database) as required.

Future smart homes will likely contain devices using different standards, protocols, and technologies. A smart interoperability system will provide integration among them so that one service or application can use the functionalities provided by services or applications based on different standards. The interoperability component will run on the home gateway devices. In the envisioned framework as depicted in Figure 1.2, interoperability will be provided as a middleware[4] service. In the framework, we propose the use of OSGi [1] to provide interoperability among Jini [28], UPnP [8], HAVi [19] and other middleware-compatible devices and services. A "Services" system will also run in the middleware to provide functionalities such as service discovery in home networks, and service composition. "Service" in this context means any software process or hardware functionality that can be used by applications or other

[3]A Home Gateway Device is a generic term for a computer or embedded device that supports and, possibly, controls various systems in a home, including the Internet access and home entertainment services.

[4]Middleware is a software component that acts between the operating system layer and the application software layer to provide some common but required services such as easy integration, scalability, and transparency to all kinds of applications to reduce burden from the application software or operating systems.

services.

Location determination is a very basic, essential, yet challenging component of smart home applications. When user identities, presence, and locations are known, many applications can be customized to the users' needs. Based on current user location, nearby home devices can be adapted to work according to the preferences of the users. For many applications, such as locating a lost object, past locations need to be stored [21]. To fit the location sensing system to the overall envisioned smart home framework as depicted in Figures 1.1 and 1.2, implementing a middleware interface component for the location sensing and storage system will be important.

There is little or no research in sensor network-based location sensing systems aimed at smart home applications and none targetted to the envisioned smart home framework. Hence, this thesis will have to develop all the important aspects of a location sensing system for home networks. The contributions of the thesis are as follows:

a) Designing and implementing a sensor network-based location determination service to provide user location within a home offering sufficient location accuracy to support the development of a wide range of location and user-preference aware home applications. User identities are also associated with their locations so that home applications can be customized to meet user preferences.

b) Implementing a prototype system for historical location store that stores users' past locations to support past location based applications [5] and application customization based on previous behavior patterns.

c) Implementing a prototype middleware interface to the location sensing and storage service to provide convenient access for context-aware home applications.

[5]Example past location based applications might be locating frequently lost objects (glasses) or determining user's past movements in homes.

1.1 Organization

The organization of the remainder of this thesis is as follows: Related work is presented in Chapter 2. In Chapter 3, an overall problem description and motivation are provided. Chapter 4 presents solution methodologies and implementation algorithms. Performance results and an evaluation of the work is provided in Chapter 5. Finally, Chapter 6 concludes this thesis and presents some directions for future research.

Chapter 2

Related Work

This section presents a representative sample of research work from the related literature on location determination approaches. The work on location determination can be categorized according to different aspects; much research addresses algorithmic aspects of location determination, some work addresses different strategic aspects to improve location accuracy while some other work applies location determination approaches and algorithms in different applications and context-aware environments. The aim of my research is providing a location sensing system for context-aware and ubiquitous smart home environments. Hence, this chapter primarily presents representative projects utilizing location determination in context-aware environments, and analyzes them with respect to suitability to smart home environments. Lastly, this chapter will describe how my work differs from other related work.

Many location determination projects address location determination in outdoor environments while other works address location determination in indoor environments. A representative sample of work among them are provided below

2.1 Outdoor Location Determination Systems

Most early work addressed the issues of location determination in an outdoor open space, usually over a large geographical area. These types of systems use protocols such as Wide-Area Cellular [41], Global Positioning System (GPS) [29], Enhanced Observed Time Difference (E-OTD), Time Difference of Arrival (TDOA), and Angle of Arrival (AOA) [6]. GPS is satellite-based and the most widely used outdoor location determination system. GPS is primarily based on line of sight communication with the satellites that is not appropriate for indoor environments. Moreover, the location accuracy, the infrastructures required for indoor environments, and the cost of GPS do not make it an attractive solution for indoor applications [20].

Unlike outdoor systems, indoor location determination systems face more challenges because of signal disturbances due to obstacles such as building fixtures, furniture, walls, doors, and moving objects. These obstacles introduce dense multipath effect effects in indoor and home environments. Multipath Effect is defined as the situation when signals from the same source reach a location via different transmission paths. This effect usually occurs in an environment where there are many obstacles in the signal propagation path and usually happens because of the reflection of the signals by the obstacles. Signals reach the destination as a combination of the direct and reflected signals. Multipath effects can result in significant signal loss due to mutual cancellation of the signals especially indoors and in areas where many metallic surfaces are present [32].

Hence, outdoor protocols based on angle, time, and time difference are not feasible or suitable enough for indoor applications due to dense multipath characteristics in homes [42, 27]. Hence, more recent work address these challenges and aims to develop accurate and low-cost location determination systems for indoor environments [26].

2.2 Indoor Location Determination Systems

Due to the current and future indoor applications' needs, very recently, much research is on-going in indoor location determination. This section presents representative location determination works for indoor environments.

2.2.1 Infrared-based Systems

- **Active Badge:** The Active Badge System [43] is an early system to determine the locations of individuals in indoor office environments. The principle component of this system is a small badge. The individuals to be identified wear these badges. The badges periodically send a pulse-width modulated infrared (IR) signal bearing the identity of the person carrying it. The room or building is usually equipped with many receivers to sense signals from these badges. The receivers are interconnected to a central server. The accuracy of the Active Badge system is not satisfactory for some applications, typically, the location of an object is identified only by general room numbers rather than using finer units, such as coordinates. Moreover, IR is quite limited by nature because it does not transmit through walls, and because it is directional.

2.2.2 Radio Frequency and Ultrasound-based Systems

Later projects have extended the IR approach by using radio frequency and other signals such as ultrasound. Some representative works are presented below:

- **Active Bat:** Active Bat [15] is a context-aware platform that permits tracking mobile users' locations in indoor building environments. Active Bat uses both Radio Frequency (RF) and ultrasound signals to detect object locations. The

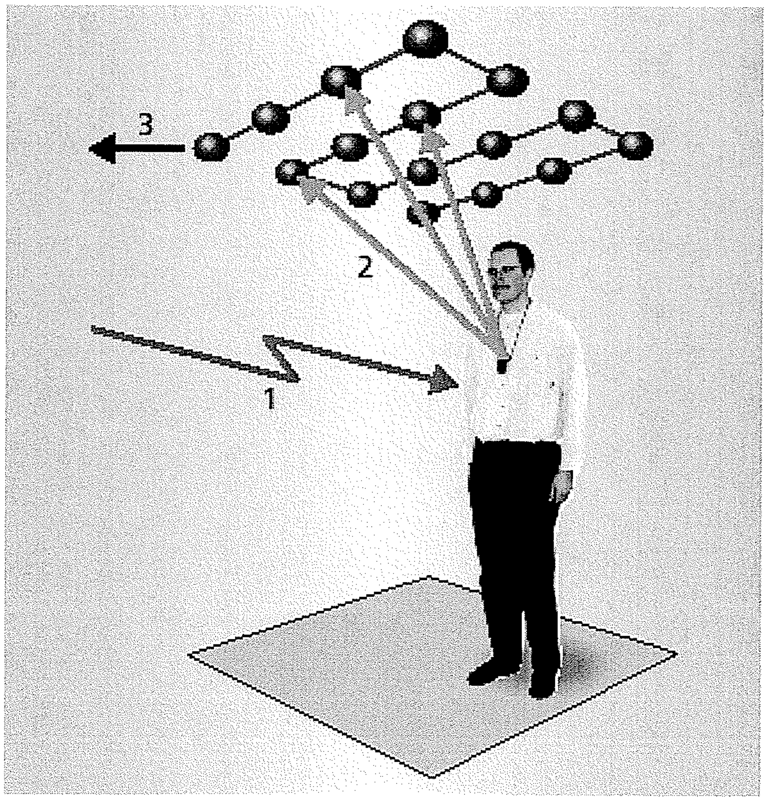


Figure 2.1: Active Bat System [40]

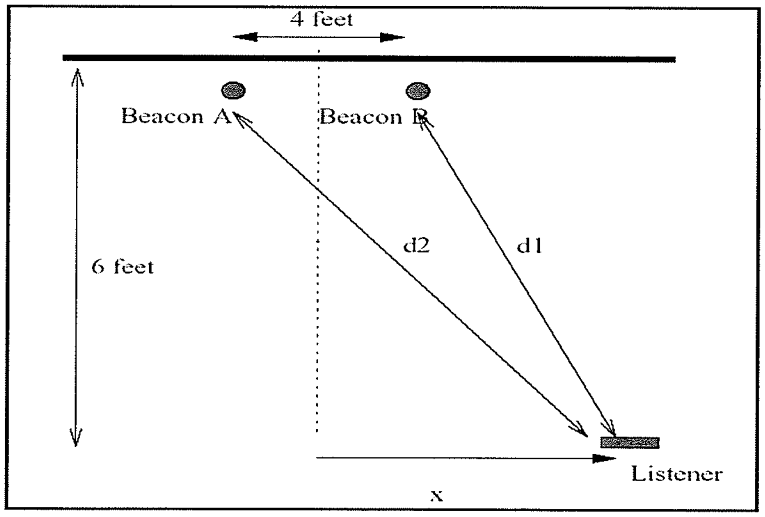


Figure 2.2: Cricket System [33]

user is required to carry a small sensor tag (bat) that identifies and locates him accurately in three dimensions. These devices transmit ultrasound signals to the signal receivers that are installed on the ceiling. The receivers measure the time of propagation of the ultrasound signals from the devices. Then by triangulation the position of the device is determined. A wireless, RF network synchronizes the Bats with the ceiling receivers to facilitate accurate time of flight measurements. The Active Bat system is illustrated in Figure 2.1. Unfortunately, ultrasound-based techniques require line-of-sight and will impose large errors in distance measurements if line-of-sight paths are blocked between transmitters and receivers. Further, ultrasound signaling is vulnerable to sound noise in home (e.g. party situations). Ultrasound systems based on Time of Flight is a more accurate and robust distance measurement approach than RSSI (Received Signal Strength Indicator) in the indoor environment, but it faces two major challenges for home applications; the measurable range is short and measurement errors may be introduced due to the multipath effect in obstructed environments [11]. Further, active bat sensors can not form a data network nor they can be directly controlled and integrated with computer networks. They also lack capability in sensing extra contexts such as temperature in the environment, do not provide interfaces to integrate additional sensors, lack the ability to concurrently run application code (i.e. are not programmable).

- **Cricket:** Cricket [33, 37] also uses both Radio Frequency (RF) and ultrasound signals to detect objects' locations. Cricket is a location-support system for in-building, mobile, and location dependent applications. Cricket allows mobile or static nodes to determine their own physical locations by listening and analyzing information from beacons deployed throughout the building. Cricket does not explicitly track user locations, rather Cricket helps devices learn about their locations and lets them decide to whom to advertise their locations [33].

The basic idea behind the projects is that signal transmitters send out RF signals and a mobile receiver acknowledges messages by emitting ultrasound signals. The transmitter devices picks up the ultrasound messages and calculate the distance between itself and the mobile receiver based on the time difference of the travelling speed of RF and sound. The Cricket system is illustrated in Figure 2.2.

Cricket system location estimation depends heavily on the propagation properties of light (RF) and sound (ultrasound). In a home environment, modelling radio propagation is quite difficult. Problems with RF are: first, there are large fluctuations in Received Signal Strength Indicator (RSSI), and, second, radio signal strength has no clear correlation with distance due to the multipath effect in indoor environments [27]. Further, ultrasound is not robust to noise.

- **Bluetooth-based:** A Bluetooth-based positioning system is presented by Kotanen et al. [23]. Their work project studies the feasibility of different location determination techniques using Bluetooth, such as Angle of Arrival (AOA), Cell Identity (CI), Time of Arrival (TOA), Time Difference of Arrival (TDOA), and RX power levels. They then implement a local positioning system based on received power levels. The received power signals are converted to distance estimates using a simple propagation model. The location accuracy is reported to be 3.76 meters. An architecture for their Bluetooth positioning system is provided in Figure 2.3. However, Bluetooth technology is limited by the number of supportable devices, and limitations on measuring received power levels precisely.

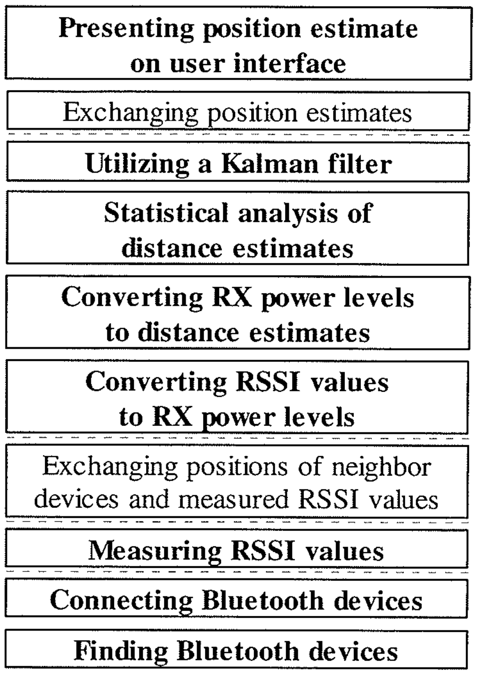

Presenting position estimate on user interface
Exchanging position estimates
Utilizing a Kalman filter
Statistical analysis of distance estimates
Converting RX power levels to distance estimates
Converting RSSI values to RX power levels
Exchanging positions of neighbor devices and measured RSSI values
Measuring RSSI values
Connecting Bluetooth devices
Finding Bluetooth devices

Figure 2.3: Architecture of a Bluetooth-based Positioning System [23]

- **DOLPHIN:** DOLPHIN is an Ultrasound-based System [12] that uses a hop-by-hop location determination mechanism with a few manually-configured reference nodes. All nodes can send and receive ultrasonic and radio signals, and can measure the distance between two objects. These distances are used to determine the actual position of an object. The DOLPHIN system is illustrated in Figure 2.4.

In the dolphin system, there are a few reference nodes. All other nodes whose locations are to be determined start internal pulse counters when they receive RF signals from reference nodes (A, B, C in Figure 2.4). These RF signals contain the pre-determined positions of reference nodes. After a while reference nodes transmit ultrasonic pulses. When other nodes receive the ultrasonic pulses, they stop their internal counters and compute their distances from reference nodes

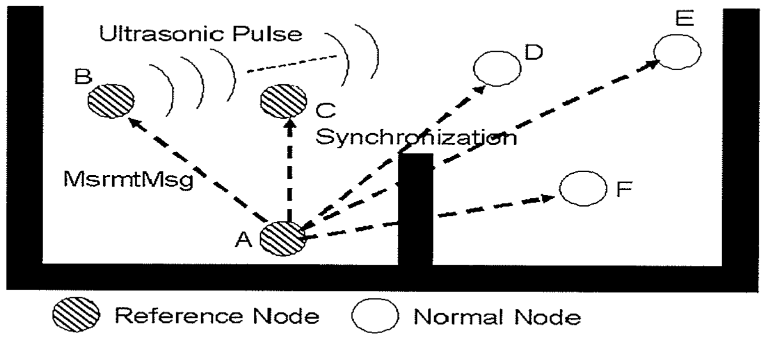


Figure 2.4: DOLPHIN Location System [12]

based on the speed of sound. When a node can collect such distances from 3 or more reference nodes, it can easily compute its location using triangulation [12].

- **Wireless LAN-based Systems:**
  - **RADAR:** RADAR [2, 4, 3] is a Wireless LAN-based (WLAN) project by Microsoft Research that uses RF signal strengths and two-phase approach (data collection and deployment) to estimate the location of a mobile target. RADAR gathers information from an existing RF data network to determine locations. Radar uses RF signal strength as an indicator of the distance between a transmitter and a receiver. RADAR then uses this distance information to locate a person by triangulation [34]. The RADAR system is illustrated in Figure 2.5.

    In the initial setup phase, signal properties at well-known points are gathered and stored in a database. These known points are called reference-points. In actual deployment, each mobile wireless device physically measures the received signal strength of beacon signals emitted by multiple 802.11 access points. The mobile devices correlate these readings with pre-existing signal strength measurements (of the reference-points) to de-

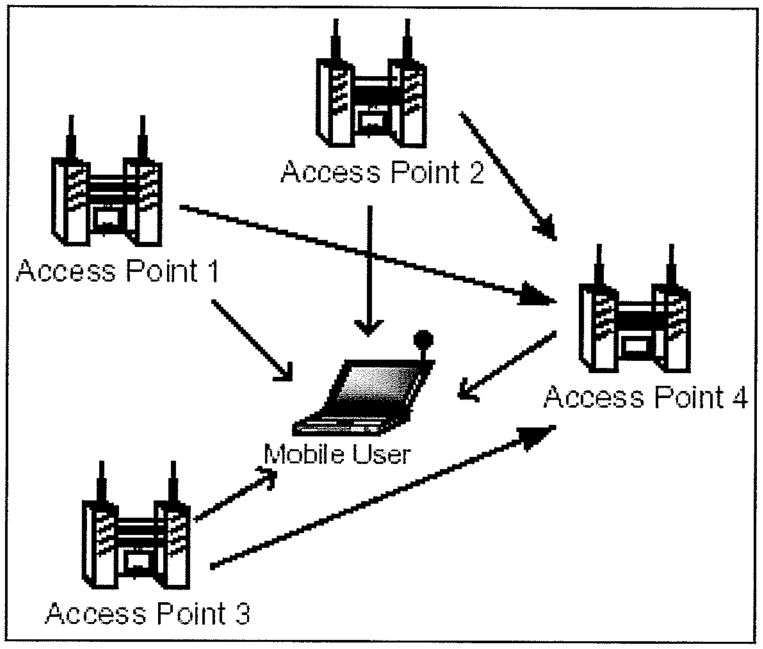


Figure 2.5: Microsoft RADAR System [3]

termine the mobile nodes' estimated location. The RADAR system provides two major advantages: it requires very few base stations and it can use the general purpose wireless networking infrastructure of the building. However, the objects being tracked must support a wireless LAN. In smart homes and in many ubiquitous applications, tiny, power, and resource constrained sensors are preferable where wireless LAN support is impractical. Additionally, RADAR is not easily expandable to a multi-floored environment or three dimensional space [6].

- **EPE** The Ekahau Positioning Engine (EPE) [17] is another positioning system based on WLAN signals which utilizes signal strength and calibration. EPE provides floor, room, and door-level accuracy. EPE is solely a software-based system that is very similar to RADAR system. The EPE system initially collects a few signal samples in known places, and based on the sample properties, and other system features, EPE constructs a signal strength model of the environment. The model is then stored in a computer and in actual deployment received signal strengths are fed to this model to determine their locations.

- **Sensor Network-based Systems:**
  - **MoteTrack:** MoteTrack [26, 27] is a sensor network-based system that extends concepts from the RADAR system and has location accuracy of two to three meters in indoor/building environments. MoteTrack uses a sensor network-based system based on Mica2 [18] sensors, whereas RADAR uses 802.11 wireless networks. MoteTrack is targeted at emergency-response applications such as fire fighters and rescuers entering a building who may use a heads-up display to track their locations and monitor safe exit routes. In the MoteTrack prototype, the target nodes (i.e. mobile nodes that represent fire-fighters) are always connected to a workstation having a display unit. The locations and safe exit routes are displayed in the display unit (carried by the target node/fire-fighter). Hence, the fire-fighter can determine the nearest or the safest exit routes. Unfortunately, carrying a computing device and a display unit in a home application will not be acceptable. Rather, I propose home-residents will carry only the sensors and no display unit.

    Like RADAR, MoteTrack has two phases: a data collection phase and an actual deployment (operation) phase. In the data collection phase, signal properties of the reference-points are collected. In the deployment phase, signal properties gathered from the target nodes are matched with those of the reference-points. The location of the target node is the centroid of the

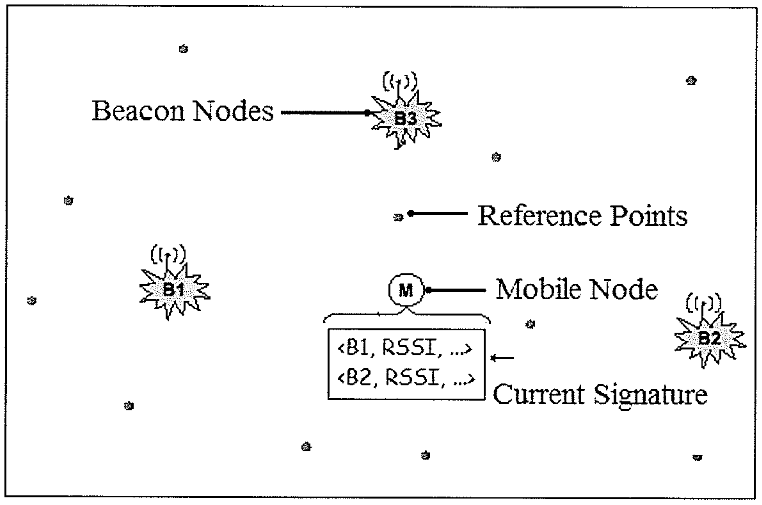


Figure 2.6: MoteTrack Prototype [26]

closely matched reference-points. In the MoteTrack application scenario, the buildings need to be pre-mapped for data collection. Hence, MoteTrack proposes GPS enabled sensors or GPS technologies for the automation of the data-collection phase. This thesis adopts two phase operations similar to RADAR and MoteTrack but adapts the deployment of the sensors, the data collection and operation phase appropriate to home application needs. A prototype MoteTrack deployment is depicted in Figure 2.6.

- **Other Mica2 Sensor Network-based Systems:** Recently some other research has also addressed the use of Mica2 sensor network-based location determination. Most of these projects address different algorithmic or deployment aspects for localization. Most such application projects are much different than smart home application scenarios.

  VICom [42], for example, uses Mica2 sensors and proposes several analytical models and extensions to angle or time difference-based algorithms to reduce multipath effects in indoor environments. Hii et al. [17] have

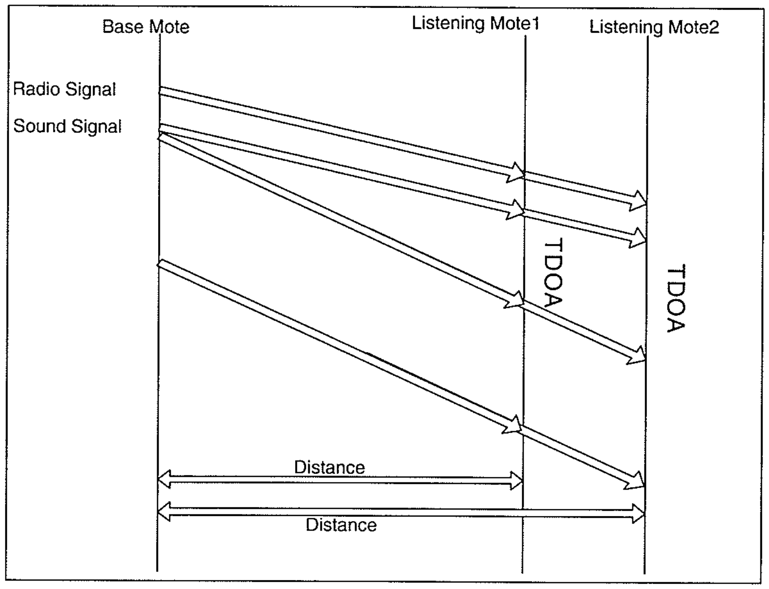


Figure 2.7: TDOA used by EPE system [17]

developed a location determination system for robotic applications that uses Mica2 sensors, 4KHz sound signals, and time difference of arrival. However, acoustic positioning needs line of sight for accurate location determination as a 4KHz tone can not penetrate walls and can be easily absorbed by obstacles. Moreover, the 4KHz tone is audible and hence can cause irritation.

Work such as that of Hii and Zaplavsky integrates EPE and Acoustic positioning to improve location accuracy. The TDOA algorithm, as used by the Mica2 motes in EPE, is depicted in Figure 2.7. Very recently, the Cricket algorithm was implemented on the Mica2 sensor network platform using a combination of RF and ultrasound signals. Probability Grid [39] is a Mica2 sensor network-based location determination system implemented for an outdoor environment. Liu et al. [25] has developed another Mica2 sensor network-based location determination method for outdoor environments that mainly deals with physical attacks[1] on location determination schemes in hostile environments. Walking GPS [38] is another sensor network-based outdoor location determination system that is implemented mainly on Mica2 and XSM motes.

2.2.3 Other Location Sensing Technologies

Several other technologies, such as electromagnetic sensing, image recognition, and pressure sensing, are also used to determine location. However, these technologies are not universal and were usually designed to address specific problems or environments.

Electromagnetic technologies can provide very accurate location detection, although the technology is very costly. The sensors should be placed close to the target objects, and accuracy degrades with the presence of metallic objects in the environment.

Computer vision is also used for exact location determination such as in Microsoft's Easy Living System [7]. However, computer vision-based technologies are too costly to be used for ubiquitous deployment in smart environments. Such vision-based systems might, however, be good candidates for monitoring and detecting the presence of users in specific portions of a smart environment, e.g. at the entrance to a home or corporate office.

The location determination system in Georgia Tech's Smart Home project [16], uses pressure sensors to determine users' locations. The smart floor can identify users via their foot pressure characteristics. However, such technology is not suitable for deployment in existing homes, as the floors have to be embedded with pressure sensors. The system also provides poor scalability and incremental deployment costs [6].

[1]To destroy a location determination system deployed in a hostile environment such as in military applications, attacks such as false beacon signals might be introduced by some parties.

2.2.4 Triangulation-based Systems

Most of the location determination techniques employ some form of triangulation, trilateration, multilateration (extension of trilateration), or measurement of radio, infrared, or ultrasonic signals. Triangulation is a way of determining an object's location using the locations of other objects (the centroid of three or more locations) [34]. Cotroneo et al. [9] argue that in an indoor office environment, such triangle-based approaches are inadequate for the following reasons:

- In indoor environments, triangle-based approaches are not robust to radio interference of similar frequencies, and the presence of walls. To eliminate the effect of interference and presence of wall, additional devices are required. This approach increases the cost of the infrastructure. A more complex estimation or calculation model, as in [23, 31, 5], can also reduce the distance error (the difference between actual and calculated distances) introduced by similar frequencies and walls.

- Triangle-based approaches suffer from measurement errors and depend on measurement techniques. Measurement errors are also linearly dependent on the strength of the received signal [23].

In view of this, Cotroneo et al. [9] proposed a zone-based location determination technique to reduce the errors related to triangulation. The architecture was tested over a Bluetooth and Wi-Fi infrastructure. Reference points-based approaches, such as RADAR and MoteTrack, although they use concepts similar to triangulation/trilateration, can easily reduce location errors by using more reference-points.

From the above discussion, I concluded that there is little or no research addressing location determination for smart home applications that are based on the use of sensor networks. Home networks provide additional challenges for location determination because there are more dense multipath effects than most outdoor and some indoor areas. Hence, providing significant location accuracy in a home environment is difficult. Further, sensor networks are widely used in many recent academic and industry sensing applications, and provide promise for use in future context-aware applications. Hence, my thesis work focused on designing and implementing a location sensing system using sensor networks. A detailed overview of my location sensing system is presented in section 3.

Chapter 3

Problem Statement and Motivation

The problem addressed in this thesis is to design, develop, and assess a location determination system for use in home network environments that offers quick query responses and adequate location accuracy. In addition to being able to provide the current location of specific individuals in the home, such a system must be able to track movements over potentially long time periods (to enable context sensitive home applications). Further, all of these services must be made readily accessible to other systems within the home.

3.1 System Requirements and Challenges

An appropriate location sensing system for smart home environments poses the following requirements and challenges:

- **Location Accuracy:** The determined locations need to be very close to the actual locations so that the actions based on user locations are appropriate. The locations need to be accurate at least to the detailed in room level and/or within a couple of meters.

- **Location Determination System:** The hardware platform, deployment of the hardware, interconnection technology, operation and interaction of the hardware and home residents in the location determination system need to be appropriate for smart home environments. A proper algorithm is also required to achieve the necessary location accuracy. The system needs to cover the total home area so that it can detect locations for users in all areas. Moreover, the system needs to support devices/badges that are small, light weight so that they may be easily carried by home occupants.

- **Location Storage System:** Designing and implementing an effective location storage system is also important to facilitate past location-based applications. Location storage can also facilitate mobility tracking for home users. To integrate user preferences into home applications the same storage mechanism can be extended to store user preferences as well. A flexible and efficient retrieval mechanism to query the location database from the applications is also in utmost importance.

- **User Identification:** To customize applications based on user identities and preferences, user identification is an important feature of a location sensing system.

- **Accessible by other home applications (Interoperability):** To support customization of home applications based on individual users needs, most home applications (using different standards, and protocols) will need to know location of individual users. Hence, an interface to the location sensing service is essential to allow it to be utilized by other home applications.

- **Multipath Effects in Indoor Environment:** In home environments, within a short distance signals will face many obstacles such as doors, walls, home furniture, and moving residents. Signals emitted from various household and entertainment devices may also cause interference/multipath effects with sensor radio signals. Hence, the location sensing system needs to adopt an approach that addresses this issue.

3.2 Desirable Characteristics

Recall from the discussion in Chapter 2, that an effective location determination system for a smart home should meet the following criteria:

- The system should be wireless so that there is no requirement for additional wiring, or extensive modification to existing infrastructure.

- The system should use media that (i) is omnidirectional, (ii) does not require line of sight, (iii) can penetrate walls, (iv) is not sensitive to variations of light and temperature, (v) is not sensitive to infrastructure (power or network) failures, and (vi) is not vulnerable to electromagnetic noise. RF signals provide such features. Radio or sound wave-based technologies are more popular, since the hardware for measuring or modulating these signals is relatively inexpensive and requires little modification of standard hardware used in wireless communications [6].

- The sensing should be sensor network-based [1], as sensor networks provide intelligent context sensing, interface to integrate external sensors and devices, and can

[1]Sensor network sensors are active sensors that can sense and process data when required. These active sensors can form a network like computer networks and be integrated with computer networks. Traditional sensors are passive and do not have processing or filtering power. These passive sensors can only sense phenomena but can not process the sensed data when required.

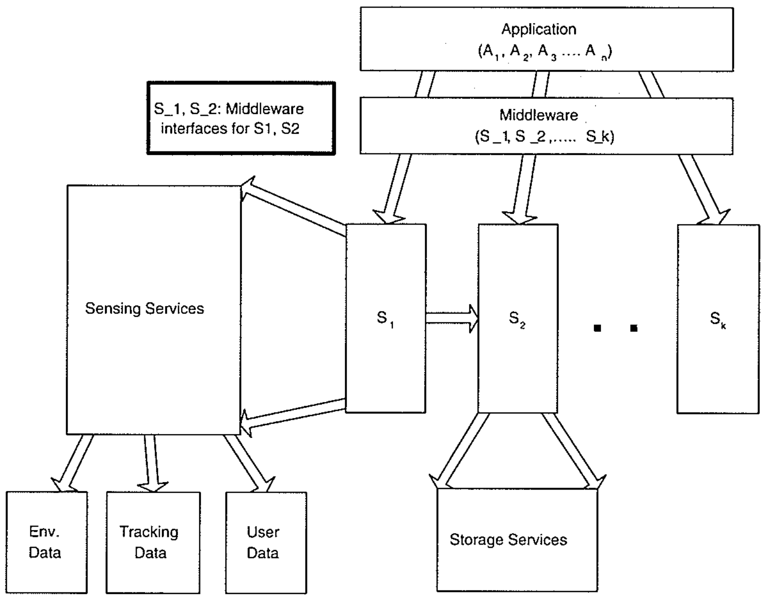


Figure 3.1: Sensing Service Framework

run multiple applications concurrently. In future ubiquitous computing, sensor networks will be an integral part of applications, hence sensor network-based systems will also be most suitable for smart home applications [36, 35].

- The sensors should be tiny to be easily deployable in homes and carried by occupants.

- The system should be reference points-based. This is because, it is still more practical to use reference points whose location and signal properties are known to model a home, as there is no proper analytical model to model radio propagation in homes and other indoor environments. Additionally, RF has no clear correlation with distance [26].

3.3 High Level Architectural Model

As shown in Figure 3.1, future smart homes will have a number of applications, indicated by A_1, A_2,...A_n in the figure, representing applications such as home security, child care, care of the elderly, as well as home device operation and control. These applications will use services, indicated by S_1, S_2,... S_k, representing services such as location determination, location tracking, location storage, user preferences storage and retrieval, home environmental monitoring, user activities and habit monitoring. Among these components, I have designed and implemented location determination and location storage services. To provide interoperability through a middleware, I have developed interfaces to the services such as S_1 and S_2 that run in the middleware framework.

Chapter 4

Design and Implementation

This section presents the design of my location sensing system as well as the implementation details. The implementation involved three separate, sequential steps: implementation of a location determination system that was integrated with user identities, implementation of the location storage system, and implementation of the interoperability component of the location determination system.

The design and implementation of my sensing system aims to provide interaction well suited to home application scenarios. In smart homes, the locations of the home residents must be tracked to operate various home devices automatically at appropriate times (e.g. turning on lights when someone enters a dark room). Hence, in a smart home scenario, there will be mobile nodes that will be carried by the home residents. These mobile nodes will calculate their current locations and send location data to the base station. The base station will transmit locations to a service in a server or a home gateway device. This service will then provide location information to other home applications and services. My location sensing system works exactly the same way to facilitate home applications.

The design and implementation of my sensing system also aims to provide significant location accuracy to at least room level and preferably within a couple of meters. I analyzed existing location determination projects in terms of suitability to home use, discovered how to address the limitations and challenges they impose, and finally designed a location sensing system suitable to home applications. My solution has adapted existing (reference points-based) algorithms to implement a prototype home environment and assessed and fine tuned the technique's location accuracy for home applications. The system developed is an extension to the RADAR and Mote-Track project, but uses the Mica2 sensor platform and is targetted to smart home applications. A prototype location storage component was also developed to store users' current and past locations. Moreover, to fit the location sensing system to the smart home framework, I created an interoperability component to allow in-home applications to access the location sensing system.

4.1 Design and Implementation of the Location Determination System

For the implementation of the location determination system, the selection of a platform was the first step. Hence, the implementation platform is first explained in Section 4.1.1. I adopted a sensor network-based system to determine locations and built and deployed a sensor network. Different components of the sensor network utilize different software modules. The sensor network, its hardware and software components are presented in Section 4.1.2. I then deployed the sensor network and the phases, operation methods, algorithms etc. that I implemented for the sensor network to determine locations. These are outlined in Section 4.2.

4.1.1 Operating and Programming Environment

I used the Mica2 sensor network platform based on Mica2 and Mica2Dot sensors as the hardware for the location determination system. Mica2 and Mica2Dot sensors were developed by the University of California Berkeley and are marketed by Crossbow Inc. Mica2 platform sensors are of small size and hence easy to place anywhere in the environment. Mica2Dot sensors are smaller in size than Mica2 sensors and are button shaped. Both Mica2 and Mica2Dot sensors operate in the same frequency range and use the same radio (CC1000 radio chip) [18]. As these sensors are battery-operated and have radio interfaces with significant noise immunity, they are resilient to infrastructure failures. Mica2 sensors consume little power, operate at multiple power levels, support sleep modes to save power, and have sufficient radio range to cover the area of a typical home. With a single AA battery, the lifetime of Mica2 sensors can be up to 17.35 months [18]. Moreover, both Mica2 and Mica2Dot sensors have interfaces for additional sensing hardware. Mica2 sensors can be used to provide concurrent operation of multiple services and/or applications. Both Mica2 and Mica2Dot sensors can operate as mobile and beacon nodes. However, because of their smaller button-like shape, Mica2Dot sensors are preferable for use in badges worn by home occupants.

Mica2 sensors run the TinyOS operating system [24]. TinyOS is an open-source operating system specially designed for wireless sensor networks. It is a component-based operating system. Hence programmers can integrate only the needed components to keep the code size small. Thus, it is appropriate for resource-constrained sensor networks. TinyOS supports an event-driven execution model, and includes power management and flexible scheduling to support unpredictable wireless communications.

nesC (nested C) is the programming language for the Mica2 platform [13]. nesC is an extension to the C programming language specially designed to fit with the structuring concepts and execution models of TinyOS. The nesC programming model integrates reactivity to the environment, concurrency, and communications, corresponding to the model for TinyOS and sensor networks.

I have also implemented several components (as described in the next section) using Java that run on the HGD to provide independence in respect of workstation hardware and operating system platforms.

4.1.2 Sensor Network Components

The sensing system consists of the following sensor network hardware and software components.

- **Beacon Nodes(B):** These are the infrastructure sensor nodes that are deployed throughout the home at fixed locations and are shown as "B" in Figure 4.1. Beacon nodes, with the help of a software component, transmit beacon signals over the radio connection. The beacon nodes use multiple frequencies and multiple power levels in transmitting beacon signals to improve location accuracy. I used Mica2 sensors for the beacon nodes.

- **Mobile Nodes(M):** These are the sensors that will be worn by the home residents (shown by "M" in Figure 4.1). During testing, to imitate home users, these sensors are placed in different locations over time. Mobile nodes receive messages from the beacon nodes, and calculate their current locations. I used Mica2Dot sensors for the mobile nodes.

- **Base Station Sensor(BS):** A Mica2Dot sensor (BS in Figure 4.1) is connected to a central workstation (serving as the HGD) using a serial port of the workstation. This base station sensor collects data from the mobile nodes and then transmits it to the workstation through a serial port.

- **Server:** This is a personal computer (shown by "HGD" in Figure 4.1) to which the base station sensor is connected. This PC collects data from the base station sensor. Several components such as the data collection component, and the location display component (described in the next section) run on this computer.

4.1.3 Other Software Components

Other software components implemented for the location determination system are as follows:

- **Serial Port Data Receiver:** A component that runs on the server workstation to capture the data coming through the serial port.

- **Data Collection Component:** During the data collection phase (explained in Section 4.2.1), the data collection component runs on the server and collects signal strengths as measured by the mobile nodes. This component also builds the reference signature database.

- **Location Display Component:** Runs on the server and displays the location data on a graphical map of the home. Additionally, this component calculates room numbers from the co-ordinates of the locations. Further, based on the user locations, this component operates devices. This component can easily be extended to provide location based applications. A prototype data storage component implemented using a Java vector is also integrated with the location display component.

4.1.4 Deployment

The beacon nodes are deployed in a Grid topology equally spaced throughout the home. The nodes are placed in such a way that the total area of the home under consideration is covered by the radio range of the nodes. In the testbed implementation (discussed in Section 5.1), the total area is divided into four (approximately) equal rectangles and a beacon node is placed at the center of each rectangle as shown in Figure 5.1. Mobile nodes represent the home residents, and hence, can be placed anywhere in the home. The base station sensor is connected to a workstation located at one end of the testbed. The serial port data receiver, the data collection, and the location display components all run on the server.

4.2 Operations for Location Determination

I have implemented the following phases and operations as part of the location sensing system:

4.2.1 Phases

The location determination system has two operational phases. First, is the data collection phase, and the second is the actual deployment (operation) phase[1]. In the data collection phase, I collected signal signatures (signal properties containing received signal strength from beacon nodes) for many different but known locations

[1]The operation phase is the phase when the location determination system actually runs in the home and other applications access location service to customize themselves to each user's need.

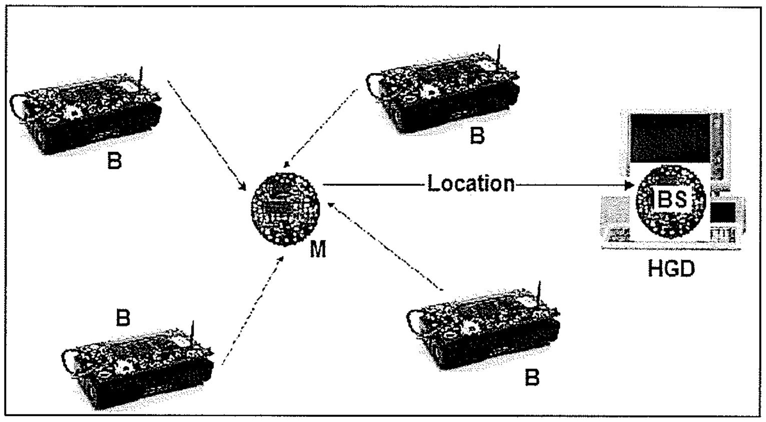


Figure 4.1: Sensor Network Components

in the prototype home and stored them in a database. These points are referred to as "reference points" and the database is called the "reference database". During the operation phase, the reference database is used by the mobile sensors.

4.2.2 Operation

During the data collection phase, the beacon nodes are deployed in the environment and a mobile node is placed at different known locations. The data collection software component is used in this phase, that provides an interface to indicate the locations of the mobile node positions in a map of the home. Afterwards, the data collection component determines the co-ordinates of those positions. In this phase, the beacon nodes broadcast beacon signals periodically. A beacon signal contains data including the beacon node's id, the sequence number of the signal, the power level used to send the signal, and the frequency channel used to send the signal. Whenever the mobile nodes receive a beacon signal, they collect the information embedded in the signal and measure the strength of the signal. Mobile nodes send this information to the base station sensor over the radio connection. The base station sensor forwards the information to the serial port data receiver. The data collection component in the workstation collects the data from the serial port data receiver and records the data to the reference database.

During the deployment phase, beacon nodes broadcast signals in the same way as in the data collection phase. One or more mobile nodes may move through the home environment. The mobile nodes receive signals from the beacon nodes for a period and create a signal signature for the received signals. The signal signature contains multiple tuples where each tuple contains the strongest signal strength as received by the mobile nodes for the current position along with the corresponding beacon node IDs. The signal signature is then matched with the reference signature database which is stored locally in each mobile node[2]. Location is calculated as the center of the matching reference points. The location data is supplied to the base station sensor through the radio interface. The base station sensor forwards the location data to the serial port data receiver. A location display component in the workstation collects the data from the serial port data receiver and displays the data in a graphical interface. The location is also stored by the data collection component.

4.2.3 Location Estimation Algorithm

Let s, is the signal signature [3] of a mobile node whose location is to be calculated. A mobile node first calculates the distance of the signature s, from s to each reference signature r_i in the reference signature database R. Let T be the set of tuples represented in both signatures (s and r_i), according to the Manhattan metric [14], the

[2]For 12m*12m area I used 150 reference points that occupy only 46 KB space. The program code is around 20KB. Mica2Dot sensors have 512 KB of Flush memory. Hence, my approach can support a very large house as well

[3]The signal signature contains multiple tuples where each tuple contains the strongest signal strength as received by the mobile nodes for the current position along with the corresponding beacon node IDs.

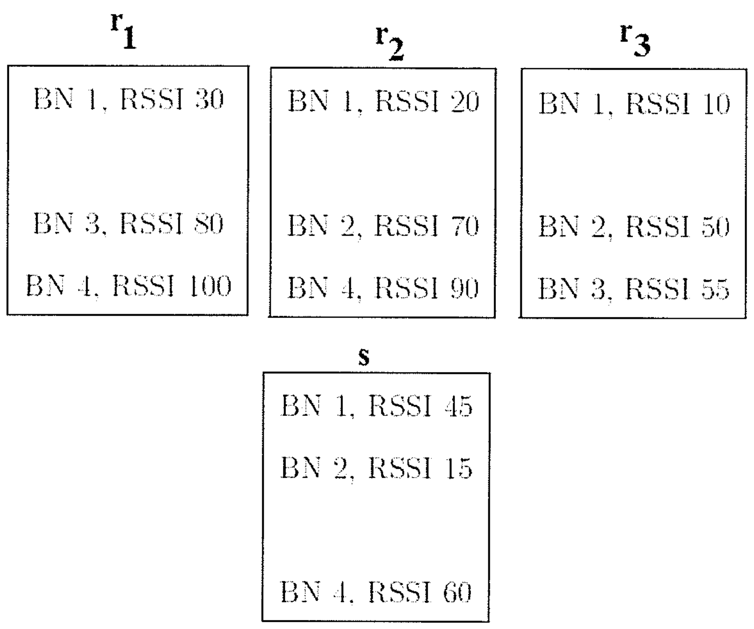


Figure 4.2: Reference Points and Signal Signature

distance set ($M(r, s)$) is:

$$M(r,s) = \Sigma_{t \epsilon T} \, |meanRSSI(t)_r - meanRSSI(t)_s|$$

where t is an individual element of the set T, and $meanRSSI(t)_r$ is the mean Relative Signal Strength Indicator (RSSI) value in the signature tuple t appearing in signature r. For example, r_1, r_2, and r_3 are three reference points in the reference signature database and s is the mobile node's signal signature as depicted in the Figure 4.2. The Manhattan distance between s and r_1 is

$$M(r_1, s) = |30 - 45| + |100 - 60|$$

Manhattan distance equation is further extended to adapt to beacon node failure as described later in this Section. The Euclidean distance metric [14] is very similar to the Manhattan metric. However, Manhattan distances are more appropriate for resource-constrained sensor networks as they require less computation and, hence less battery resources [27]. Using this set of signature distances, the location of a mobile node can be calculated in several ways. The simplest approach(the k nearest algorithm) is to take the centroid of the geographic location of the k nearest (in signature space) reference signatures. The base station finds the k ($k = 1, 2, 3, ..., n$) nearest matching signatures to the mobile node's signature from the reference signature database. The location of the mobile node is determined to be the centroid of the reference locations. If the value of k is at least three, this algorithm can provide significant accuracy. A large value of k incurs additional computational effort and delay. However, a small value of k might not always give a very accurate location. A small and fixed value of k does not account for cases where the density of reference signatures is not uniform. For example, in a physical location where few reference signatures have been taken, using the k nearest reference signatures may lead to comparison with signatures that are very distant. There is an alternate approach (kth nearest) [27] where reference signatures only within a ratio (1-1.3) of the nearest reference signature are considered. If the nearest signature is $r^* = arg_{r \epsilon R}\ min M(r, s)$, all reference signatures $r \epsilon R$ that satisfy $\frac{M(r,s)}{M(r*,s)} < c$ for some constant c are considered. The geographic centroid of the locations of this subset of reference signatures is then taken as the mobile nodes' position. In the later case, only the nearest reference signatures are considered. If we take the value of c to be very close to 1 such as 1.1 or 1.2, only the very nearest reference signatures are considered. Hence, small values of c (i.e. very close to one) that match at least 4 reference points, would give the most accurate locations possible [27].

4.2.3.1 Adaptive Signature Distance Metric

The set of signature tuples in the reference signature r and mobile nodes signature s are not always identical. Hence, a strategy is required to account for missing data in

r	s
BN 1, RSSI 30	BN 1, RSSI 45
	BN 2, RSSI 15
BN 3, RSSI 80	
BN 4, RSSI 100	BN 4, RSSI 60

Table 4.1: Example Mobile Node and Reference Database Signal Signatures

one signature or the other. If r contains a signature tuple not found in s, then s might have been taken at a different location, or a beacon node may have failed. First, when there are no beacon node failures, missing tuples between two signatures indicate that they are at different locations. For this instance, the bidirectional signature distance metric is defined as [27]:

$$M_{bidirectional}(r,s) = M(r,s) + \beta \, \Sigma_{\ t \varepsilon (s-r)} \ meanRSSI(t)_s + \beta \, \Sigma_{\ t \varepsilon (r-s)} \ meanRSSI(t)_r$$

Each RSSI tuple not found in $(r \bigcap s)$ adds a penalty to the distance, proportional to signatures' RSSI values. In a home environment, I assumed there will be little failure of the beacon nodes. Hence, I adopted this bidirectional signature distance metric. In my experiment, I used $\beta = 1$.

When beacon nodes fail, a larger number of RSSI tuples in the set $(r - s)$, causes an explosion in error. To minimize the errors due to failed nodes, the unidirectional distance metric is defined as:

$$M_{unidirectional}(r,s) = M(r,s) + \beta \, \Sigma_{\ t \varepsilon (s-r)} \ meanRSSI(t)_s$$

This metric only penalizes tuples found in s and not in r. Here, the assumption is that the reference signatures were acquired while all beacon nodes were operational. The unidirectional metric only compares signatures between operational nodes [27].

As an example, consider the signatures in Table 4.1:

The bidirectional and unidirectional distance metrics are as follows:

$$M_{bidirectional} = |30 - 45| + |100 - 60| + 15 + 80 = 140$$

$$M_{unidirectional} = |30 - 45| + |100 - 60| + 15 = 70$$

4.2.4 Design and Implementation of the Location Storage

A proof of concept location storage system has been implemented using a Java vector object. The store contains as many of the past locations of a person as can be stored. One row or entry in the Vector contains the UserID, timestamp, and location co-ordinates. Whenever a new location is determined by the sensing system, the existing location in the store is checked to determine whether it is the location with the same co-ordinates as in the previous timestamp for this user; if not, the new data is inserted; otherwise the data is discarded. As the storage system is based on a Java vector, it can store only as much data as the memory (RAM) or page file of the workstation supports. Hence, the time period that this system can support for past location-based applications will be highly dependent on the RAM and Page File size of the workstation, number of home-residents, and the frequency of user movements.

4.2.4.1 Extension of the Location Sensing System

A long term location store will enable past location based applications such as locating frequently lost objects (missing glasses, watch, mobile phone) in homes. Further, long term storage can be mined to identify rules or patterns about user behaviors and device operations. This will help to create user profiles to store user behavior patterns that will facilitate automation based on the derived patterns. Noticing changes in user behavior can also be used to trigger the automation system to adapt itself. Long term storage is required for some of these purposes.

For the extension of the storage system, relational or XML based database technologies can be used. However, in the future, home applications may be accessed from the Internet as a service. Hence, I believe XML based semi structured technologies will be used extensively. Further, using query languages such as XPATH and XQuery, user profiles can be easily generated and manipulated from a future XML based location database.

4.2.5 Design and Implementation of the Interoperability Component

I have implemented an OSGi [1] based interoperability component utilizing the Knoplerfish [22] OSGi framework to deploy and test the OSGi component. OSGi is a middleware framework that can provide interoperability among services based on a variety of middleware "standards". In the Knoplerfish framework, services are deployed as bundles. The bundles are Java jar files according to OSGi specifications. The bundles usually contain a manifest file formatted in attribute-value pairs to store metadata about the bundle such as the classpath of the bundle, java packages imported and exported by the bundle, and java native libraries required by the bundle. Each bundle has an activator interface. When bundles are deployed in the Knoplerfish framework and started, this activator is called to initialize and register the bundle into the framework. Also, the bundle needs start, stop, update, and uninstall methods to allow it to be managed by the framework. Each bundle has a context through which it can install other required bundles, interrogate other bundles, obtain persistent storage, lookup and retrieve services, and subscribe for various event notifications. For the location sensing service, I have implemented a bundle with these features. The location bundle provides services such as location in realtime and current location of a person. Another OSGi bundle was developed and deployed in Knoplerfish framework that calls these services offered by the location sensing bundle. This second bundle provides a graphical interface with options to call different methods of the OSGi location component and displays the results.

4.3 Implementation Details

This section presents some details of the algorithms used by beacon nodes and mobile nodes, respectively:

- **Beacon Nodes:** Beacon nodes transmit beacon signals at regular intervals in the both data collection and operation phases. For this purpose, beacon nodes use the same algorithm both in the data collection and operation phases. Beacon nodes transmit signals at different frequencies and power levels to achieve significant location accuracy[4]. Beacon nodes initially select a frequency level and then switch over all power levels at regular intervals. Afterwards, beacon nodes move to the next frequency channel and again switch between all power levels. In my experiments (described in the next chapter), I used up to three power levels and one frequency. Beacon node messages contain data including Beacon Node IDs, sequence numbers of the messages, frequency channels, and the power levels. The algorithm just described are presented in Algorithm 1 and Algorithm 2.

- **Mobile Nodes:** During the data collection phase, whenever a mobile node receives a beacon signal, it extracts the information (beacon node ID, sequence

[4]Varying beacon signals over multiple frequencies and transmission powers increase accuracy. [27]

Algorithm 1 Beacon Node Operation

```
procedure BeaconNodeMain()
1: Initialize
2: Start Timer
3: while true do
4:     when timer fires
5:         SendBeaconMessage()
6:     end when
7: end while
end procedure
```

Algorithm 2 Send Beacon Messages

```
procedure SendBeaconMessage()
1: Increase sequence number for the beacon message
2: Embed current frequency, current power level, message sequence number, and
   beacon node ID into beacon message
3: Send the beacon signal
4: if Current Transmission Power Index==NBR_POWER_LEVELS then
5:     Current Frequency Channel Index = (Current Frequency Channel Index + 1)%
       NBR_FREQ_CHANNELS
6:     Current Transmission Power Index = 0;
7: end if
8: Current Transmission Power Index = Current Transmission Power Index + 1
end procedure
```

Algorithm 3 Actions On Beacon Message Receive

```
procedure ActionsOnBeaconMessageReceive()
1: if Mobile Nodes Receive a Beacon Signal then
2:     if phase==data collect then
3:         Extract information such as beacon node ID, sequence number, power level,
           and frequency channel from the signal
4:         Measure the received signal strength
5:         Send the extracted and measured data to the Base Station Sensor
6:     else if phase==operation then
7:         Store the data into the hash table for signal signature creation
8:     end if
9: end if
end procedure
```

number, power level, and frequency channel) from the signal. Mobile nodes also measure the strength of the signal. Mobile nodes then supply the extracted and measured data to the base station sensor over the radio connection. The data is collected by the Data Collection Component and recorded in the reference signature database (Algorithm 3).

During the operation phase, whenever mobile nodes receive a beacon signal, they extract information from the beacon signal and insert the data into a hash table (where Beacon Node ID is used as the key). The hash table data is sorted in descending order by received RSSI for each beacon node. These data from the storage table are used to create signal signatures whose locations are calculated. The actions taken by mobile nodes in the event of receiving a signal from beacon nodes are shown in Algorithm 3. The detailed algorithm executed by mobile nodes is shown in Algorithm 4. Mobile nodes use three timers for frequency change, data collection, and location estimation respectively. Whenever the frequency timer expires, mobile nodes tune to next frequency. Whenever the data collection timer fires, mobile nodes send the collected data (reference point properties/signature) to the base station, and whenever location estimation timer fires, mobile nodes calculate their own locations and send location data to the base station server.

The location estimation algorithm is shown in Algorithm 6. Mobile nodes create a signal signature for their current positions. To create the signal signature, mobile nodes use the hash table where all the received beacon messages are stored. Each Mobile node selects the beacon node IDs and the maximum RSSI values associated with the beacon nodes. These data form signal signatures whose locations are determined. Signal signature creation is described in Algorithm 7. Mobile nodes then calculate the distances from their signal signatures to all reference points in the reference database using the algorithm provided in Section 4.2.3. Mobile nodes then select the nearest 3–15 matching reference points and the node's location is calculated as the center of its matched points.

Algorithm 4 Mobile Nodes' Overall Operation

```
procedure MobileNodeMain()
1:  call Initialize()
2:  Start frequency timer
3:  if phase == operation then
4:      Start location estimation timer
5:  end if
6:  if phase == datacollection then
7:      Start data collection timer
8:  end if
9:  while TRUE do
10:     when frequency timer fires
11:         Switch to next frequency level
12:     end when
13:     if phase == operation then
14:         if Message received from Beacon Nodes then
15:             call ActionsOnBeaconMessageReceive()
16:         end if
17:         when Location timer fires
18:             call LocationEstimate()
19:         end when
20:     else if phase == datacollection then
21:         when Data collection timer fires
22:             call CollectReferenceData()
23:         end when
24:     end if
25: end while
end procedure
```

Algorithm 5 Initialize

```
procedure Initialize()
  1: Initialize hash table to store received messages
  2: Initialize frequency level
end procedure
```

Algorithm 6 Location Estimation

```
procedure LocationEstimate()
  1: call constructSignature()
  2: Find four matching reference points that have minimum distance with the constructed signal signature
  3: Location is calculated as the center of these matched points
end procedure
```

Algorithm 7 Signal Signature Construction

```
procedure constructSignature()
  1: Construct a signal signature set that will contain the RFSignals with the strongest RSSIs (from the Hash table) and sorted by source beacon IDs. All decisions regarding the strongest RSSIs are based on an arbitrary but fixed frequency channel and default transmission power used
end procedure
```

Chapter 5

Evaluation

The primary aim of the thesis is to design a location sensing system for smart home applications and implement a functional prototype. Hence, experiments were done to test different components of the implemented system functions. The metric used to measure the performance of my location determination system was location accuracy. Experiments were done to test the location accuracy. Other metrics that affect location accuracy in my system include the number of reference points, distribution of the reference points, the location determination algorithm, and the number of matching points used to calculate locations. The effects of these parameters on location accuracy are also considered. Further, the differences between my system and MoteTrack are discussed in Section 5.4.

5.1 Experimental Setup

A test environment was setup to test the functionality and appropriateness of the location sensing system for smart home environments as shown in Figure 5.1. The test environment was deployed on one floor of TRLabs Winnipeg and consisted of

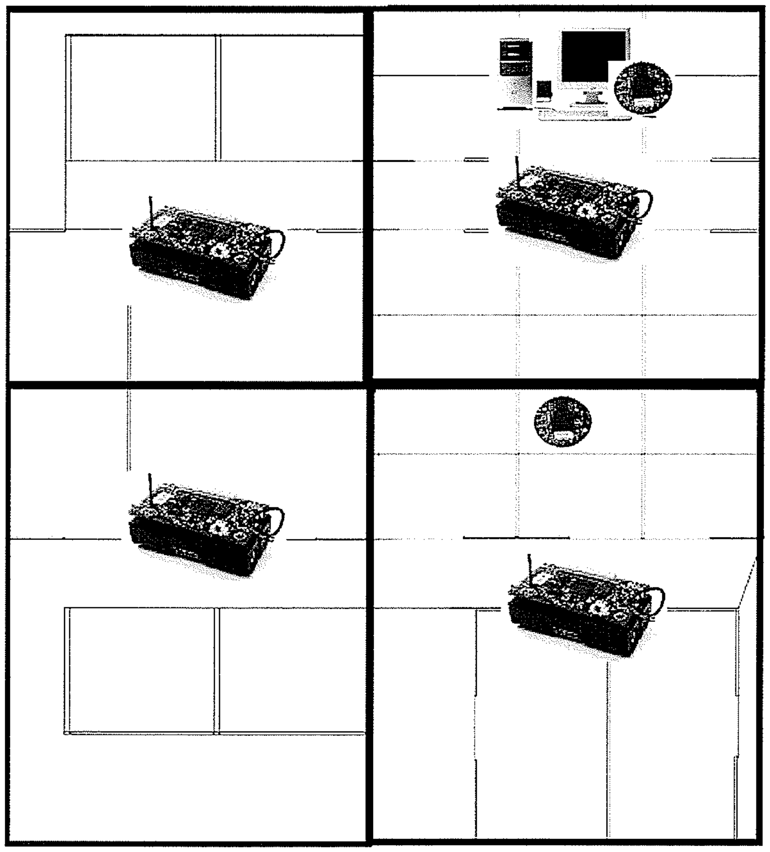
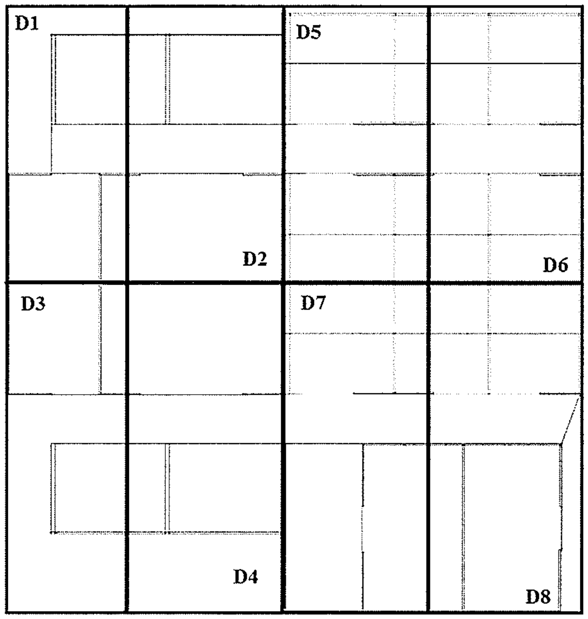


Figure 5.1: Sensor Network Deployment Area

four beacon nodes (Mica2 sensors). The total floor area was logically divided into four "rooms". The tick lines in Figure separate the virtual rooms and the thin lines display the office partitions inside the "rooms". A beacon node was placed at the center of each room. The rooms were divided with partitions and walls as might be found in a home environment. A base station sensor connected to a workstation (PC) was placed at one end of the floor.

I selected 15 positions randomly distributed in the home covering all four rooms. The base station component and the location display component were started. Then the beacon nodes were turned on. Initially, I put a mobile node in each of the 15 places one by one and collected the location readings for each of these positions. The

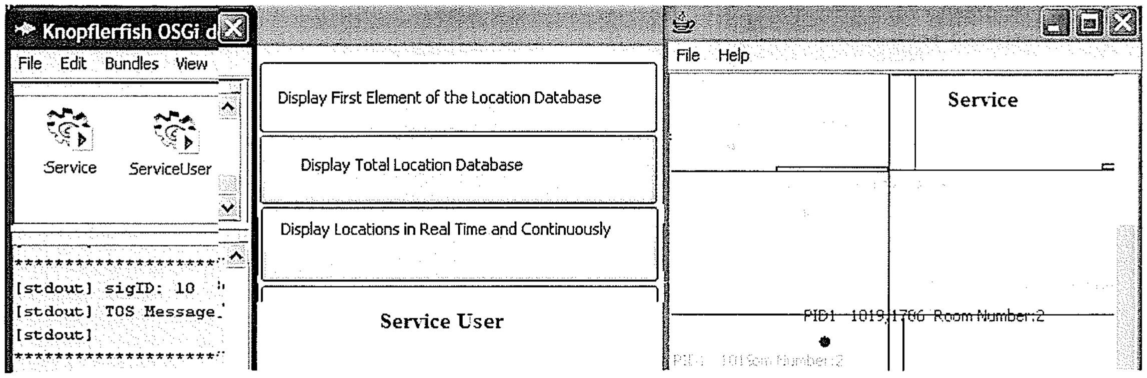


center

Figure 5.2: Device Positions in Different Rooms

graphical component displayed the coordinates of the determined positions, the IDs of the mobile nodes, and the corresponding room numbers. The average location error was 1.5 meters and room level accuracy was 90%. The location error and room level accuracy are quite reasonable for most smart home applications. Further, we can improve the location accuracy and room level accuracy using some techniques as depicted in section 5.2.2. In some random cases, location error randomly varied up to 4 meters. Usually, mobile nodes got strongest signals from the beacon nodes in the same room. However, based on user presence and movement or when there were obstacles directly between them then the mobile nodes got the strongest signals from remote beacon motes and this caused higher location errors. Further, for the positions at the edge points of two rooms, the room number also varied randomly.

To test the location integration to applications, I assumed there were two devices in each room, placed in each half of the room as indicated by $D_1 ... D_8$ in Figure 5.2.


Knopflerfish OSGi d
File Edit Bundles View
Service
ServiceUser
Display First Element of the Location Database
Display Total Location Database
Display Locations in Real Time and Continuously
Service User
File Help
Service
PID1 1019,1766 Room Number:2

```
********************
[stdout] sigID: 10
[stdout] TOS Message
[stdout]
********************
```

Figure 5.3: OSGi Service and Service User as deployed in Knoplerfish

Whenever a location was determined, based on the location, the nearest device was turned on. For most cases, it was the correct device that was turned on.

The location sensing OSGi component was also deployed in the Knoplerfish OSGi framework. This OSGi component provided several services such as most currently detected locations, and a person's most recent locations as well as his past positions based on his ID. To test this interoperability component, I developed another OSGi component to make use of the services provided, and this second component was also deployed in Knoplerfish framework.

This user component provided a graphical interface with several options to select such as Supply Most Currently Detected Locations, and Supply a Person's Most Recent Locations. Based on the selections, this component called the related functions from the location sensing OSGi component and displayed the retrieved data. The components are depicted in Figure 5.3. In a real application environment this location data can be used to operate a device. Using this code as a basis, in home environments where OSGi is used as the middleware framework, my location sensing services can be used effectively.

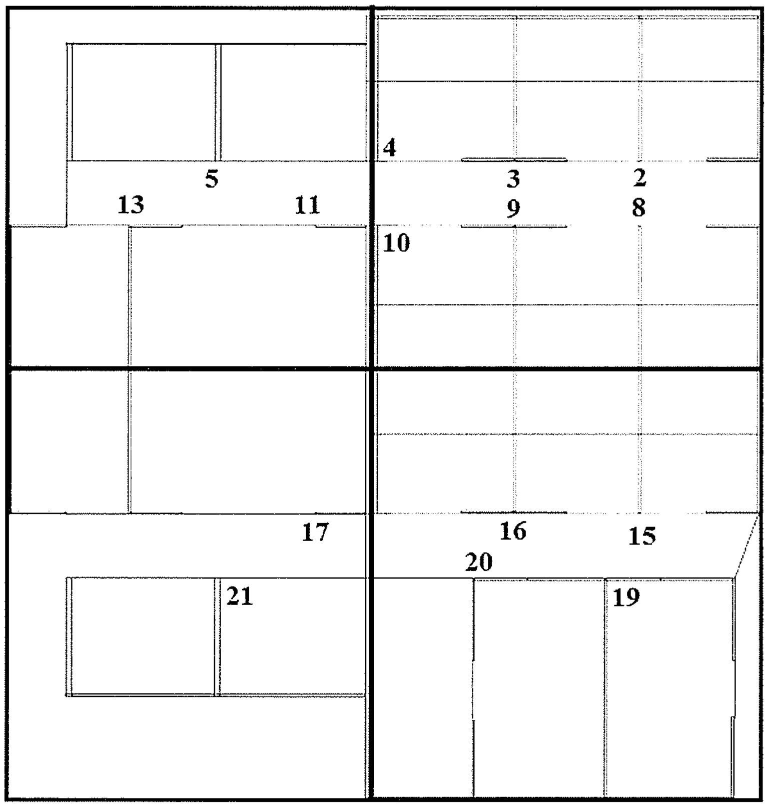


Figure 5.4: Places where Location Data were Collected

5.2 Location Accuracy

The positions where location data was collected are shown in Figure 5.4. I put a mobile node in each of the 15 places shown and collected location data as supplied by the mobile nodes. While collecting readings for each of these positions, I waited for a stable point where consecutive readings were consistent for many readings, and from that point 100 data readings were considered for analysis. Afterwards, I calculated the location errors for each of these places. Location error was calculated as the distance between the actual location and the system determined location.

I got 28^{th}, 50^{th}, 86^{th}, 90^{th}, and 97^{th} percentile location errors under 1, 1.5, 2, 2.5, and 3 meters respectively. Average location error for these positions is 1.5 meters. Room level location accuracy was 90%. The standard deviation for the location errors was 0.34 which is quite low indicating the error values are mostly concentrated near the mean. This location accuracy supports room level accuracy which is quite acceptable for most smart home applications. A table presenting my experimental results is provided in Table 5.1.

A table representing the parameters used for this experiment is provided in Table 5.2. In this experiment, beacon nodes used one power level and one frequency level for radiating beacon signals. I distributed reference points in every "room" one meter apart, and used four reference points for matching against the mobile nodes' signal signatures for location determination. Also, I used the k-nearest algorithm with $k = 4$. For each position, 10 messages from the beacon nodes were considered both for data collection and during the operation phase. Similar experiments were done with two different mobile sensors placed at two different places concurrently. The location accuracy was very similar to the previous experiment.

Pos	**Samples**	**Avg**	**1m**	**1.5m**	**2m**	**2.5m**	**3m**	**Room**
2	100	1.66	6%	43%	91%	95%	100%	100%
3	100	1.80	39%	39%	55%	55%	83%	64%
4	100	1.70	3%	37%	83%	83%	93%	60%
5	100	1.05	86%	86%	100%	100%	100%	100%
8	100	1.77	0%	10%	100%	100%	100%	100%
9	100	1.73	0%	0%	100%	100%	100%	100%
10	100	2.04	2%	53%	60%	60%	90%	98%
11	100	1.24	18%	82%	98%	100%	100%	70%
13	100	1.54	37%	37%	37%	90%	100%	100%
15	100	0.87	98%	100%	100%	100%	100%	100%
16	100	0.88	97%	99%	100%	100%	100%	98%
17	100	1.49	37%	66%	66%	72%	88%	67%
19	100	1.69	0%	0%	100%	100%	100%	100%
20	100	1.19	0%	100%	100%	100%	100%	100%
21	100	1.68	1%	1%	90%	100%	100%	100%
Avg	100	1.49	28%	50%	85.33%	90.33%	97%	90%

Table 5.1: Location Error at 15 Different Places

Parameter	Value
NBR FREQ CHANNELS	1
NBR TX POWERS	1
BEACON SEND PERIOD	100 ms
FREQ LISTEN PERIOD	100 ms
EST LOC PERIOD	1000 ms
Number of Samples considered	10
DATA COLLECTION PERIOD	1000 ms

Table 5.2: Parameters Used for Location Determination

5.2.1 Location Accuracy at Edge Points

An additional experiment was done to test the location accuracy at the edges of the four rooms. Tweleve points were selected for the experiment as shown in Figure 5.5. The parameters used for this experiment were the same as above. The average location error for these locations was 3.18 meters. Location errors were higher than the previous experiment because the edge points do not have reference points evenly distributed all around them. Having reference points evenly distributed around them would have improved the location accuracy. The individual location errors for each of the places was shown in Figure 5.5 are enumerated in Table 5.3. The room level accuracy for this experiment was 80%.

From Table 5.3, we see that the location errors for the extreme edge points such as points 1, 4, 5, 8, 9, and 12 are much higher than the others. The average and standard deviation for these six extreme edge locations were 3.87 meters and 0.48 meters. For the other six locations, the average location error was 2.47 meters and the standard deviation was 0.64 meters.

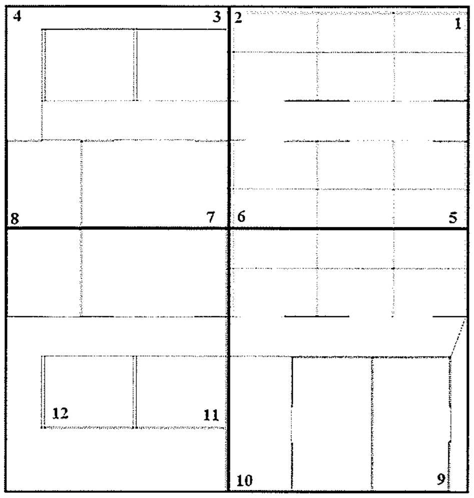


Figure 5.5: Edge Points where Locations were Collected

Positions	Location Error	Room Level Accuracy
1	3.35	100%
2	3.18	76%
3	2.38	91%
4	3.96	100%
5	3.64	66%
6	2.14	52%
7	1.70	54%
8	3.85	100%
9	4.90	100%
10	1.97	91%
11	3.49	61%
12	3.56	71%
Average	3.17	80%

Table 5.3: Location Error at Edge Points

From the discussion above, this thesis concludes that reference points-based sensor network-based location determination systems can provide reasonable location accuracy and hence are appropriate for smart home environments in terms of location accuracy.

5.2.2 Improving Location Accuracy

The location accuracy of my system is quite reasonable, however, different techniques as mentioned below will very likely improve the location accuracy

- Using multiple frequency levels along with multiple power levels for beacon signals so that reference signatures are more diverse

- Almost always, I got room level accuracy of 100%, but, for intersection points between two rooms, the room level accuracy was much lower. In such cases, we might use software calibration to determine the room numbers. The beacon node id from which the mobile node is receiving the strongest signal will likely be the desired room number. We can also count the number of signals the mobile nodes are receiving from the particular beacon node. Further, based on the user's past location sequences and timestamps as stored in the database, user movement direction might be determined, hence room level accuracy could be improved.

- Careful placement of home devices, and furniture in respect of user movement, sitting, standing, and lying areas would also improve the application experience that might arise due to intersection points' location or room level accuracy. Unfortunately, this would also be a tedious process.

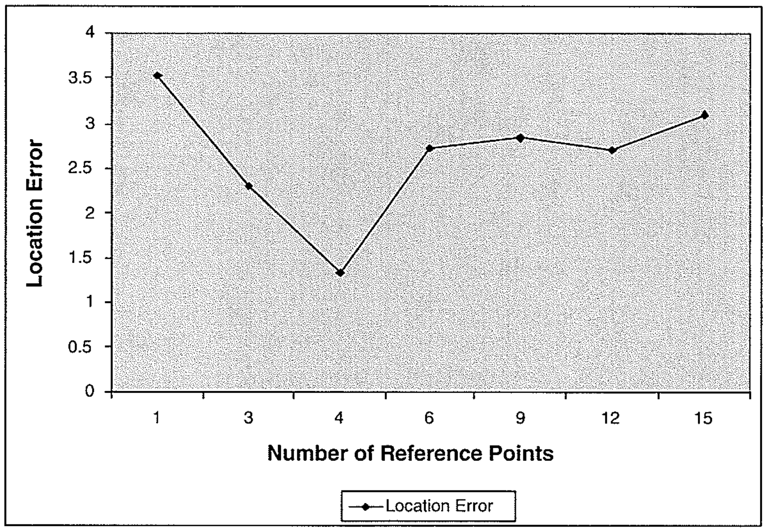


Figure 5.6: Effect of the Number of Reference Points Matched

5.3 Effects of Parameters on Location Accuracy

This section demonstrates the effects of certain parameters on location accuracy.

5.3.1 Effect of the Varying Number of Reference Points Matched against the Mobile Node Signal Signature

In this experiment, I used different numbers of reference points to be matched with the mobile nodes' signal signatures whose locations are to be calculated. I selected one place in each room near the center of each room. I selected positions 2, 5, 16, and 21 as shown in Figure 5.4. For each of these places, I varied the number of reference points to be matched from one to 15 such as, 1, 3, 4, 6, 9, 12, and 15. I collected location data for each of these places and for each of the number of reference points to be matched. Other parameters used for this experiment were the same as the previous experiments. I calculated location errors for each of these places. A graph demonstrating the effect is shown in Figure 5.6. A table demonstrating the graph

Locations	2	5	16	21	Average
No of Ref Points	m	m	m	m	m
1	2.45	4.50	3.60	3.55	3.52
3	2.35	3.89	1.37	1.56	2.29
4	1.06	1.66	0.88	1.68	1.32
6	3.30	2.51	1.43	3.65	2.72
9	2.96	2.75	2.30	3.33	2.83
12	3.11	1.94	2.21	3.58	2.71
15	3.16	2.61	2.64	3.96	3.09

Table 5.4: Data on Effect of the Number of Reference Points Matched

data is also provided in Table 5.4.

Figure 5.6 shows that location accuracy is best when the number of reference points to be matched is four. Decreasing as well as increasing the number of reference points to be matched from four decreases the location accuracy and increases location error. When the number of reference points to be matched is one, location error is 3.5 meters, with the increase of the number of reference points to be matched to three, location error reduces to 2.3 meters and when the number of reference points to be matched is four, I got location error to be 1.32 meters which is the minimum. Location accuracy is best when the number of reference points considered are evenly distributed around the actual position. When the number of reference points is too few, the location is biased only to one reference position that might be a distant position, and an even distribution is not possible in this scenario. Location errors again increase with the number of reference points to be matched. When the number of reference points to be matched is around 15, location errors increase to more than three meters. When the number of reference points to be matched is much higher,

Positions	1m	2m	4m
2	1.66	1.94	1.02
3	1.80	0.63	1.02
4	1.70	1.95	1.58
5	1.05	1.80	2.30
8	1.77	1.27	1.81
9	1.73	1.33	1.75
10	2.04	1.84	2.57
11	1.24	1.65	2.29
13	1.54	2.09	3.16
15	0.87	2.83	2.97
16	0.88	1.44	3.01
17	1.49	1.41	1.85
19	1.69	3.18	3.60
20	1.19	0.42	1.69
21	1.68	1.46	2.74
Average	1.49	1.68	2.22

Table 5.5: Data on the Effect of the Distribution of Reference Points

more distant reference positions in addition to nearby positions are considered and hence introduce significant errors into the location accuracy. Therefore, I conclude that the number of reference points to be matched needs to be around four ($k = 4$) for optimal results.

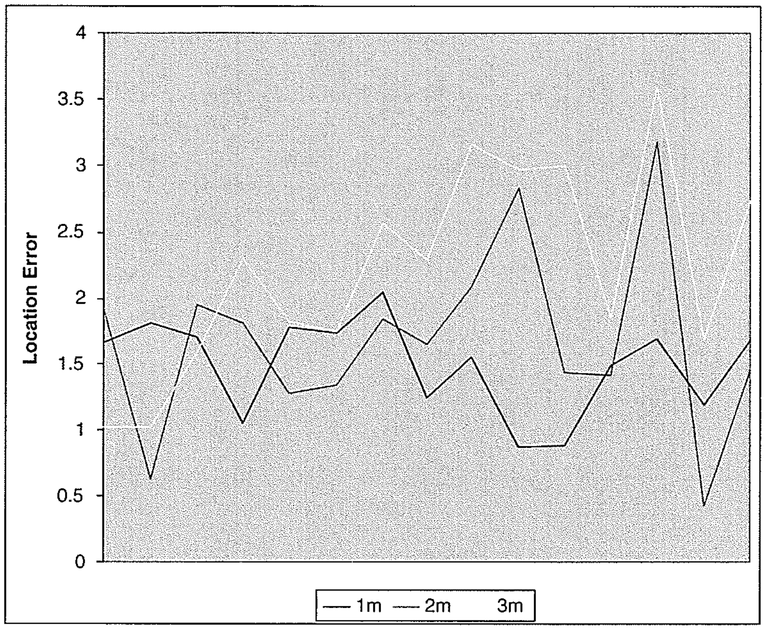


Figure 5.7: Effect of the Distribution of the Reference Points

5.3.2 Effect of the Distribution of Reference Points

In the previous experiments, the reference points were collected one meter away from one another. To study the effect of the distribution of the reference points on location accuracy, in this experiment, reference points were collected all over the home 1, 2, and 4 meters apart in three steps (trials). The numbers of reference points were 150, 48, and 16 respectively for 1, 2, and 4 meter distances. The average location errors seen for these three cases were 1.5, 1.68, and 2.22 meters respectively. I collected location data for 15 random places (as shown in Figure 5.4) and calculated the location error for these places. The individual location error for these positions are shown in Figure 5.7 and Table 5.5.

Figure 5.7 and Table 5.5 show that location accuracy is more reasonable when reference points are collected one meter apart. The possible reason behind this is that as the reference points to be matched are only one meter away from one another, the four reference points that are considered in location determination logically should be nearer than the other two cases.

When reference points are 2 meters apart, location accuracy is still reasonably good; mostly under 2 meters. The possible reason for this is that in this case the four reference points considered for location determination should be only a little further away than in the previous case (1 meter apart). However, as location is calculated as the centroid of the four reference points, the location errors do not differ much. One observation, from my experimental data is that when the reference points are 2 meters apart the location readings were more stable than 1 meter. The possible reason behind this is that when there are reference points every meter, there are more combinations of the four reference points that will match mobile nodes' signal signature.

When the reference points are distributed 4 meters apart the location error is mostly around three meters. The possible reason behind this is that in this case, the four reference points considered for location determination will be further away than in the previous two cases. However, for many positions, such as 2, 3, 4, 8, 9, 17, 20 location errors are still under 2 meters.

5.3.3 Effect of Varying Power Levels

In this experiment, reference points were collected using different number of power levels (specifically, one power level and three power levels). For both cases, location readings were collected for six positions. The location errors and the effect of the different power levels on location accuracy are shown in Figure 5.8 and Table 5.6. For this experiment, reference points were collected every 2 meters apart and only in the top right and one third of the bottom right rooms. The total number of reference

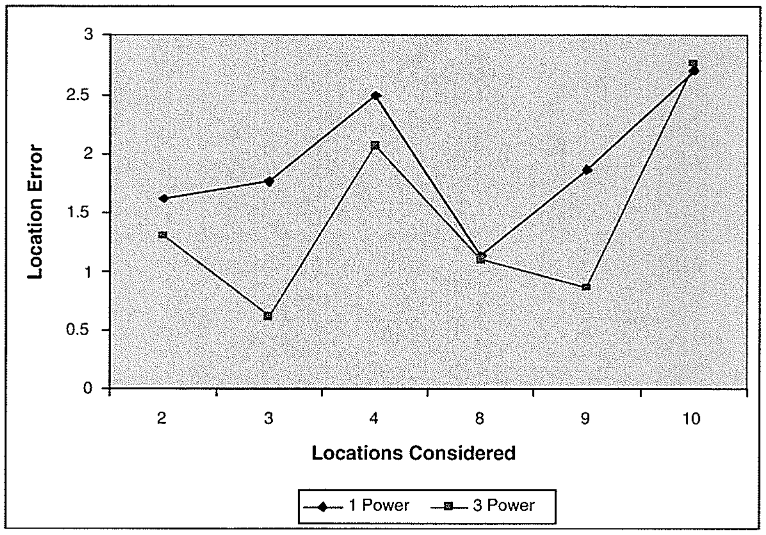


Figure 5.8: Effect of Varying the Number of Power Levels on Location Accuracy

Locations	1 Power	3 Power
2	1.62	1.31
3	1.77	0.62
4	2.5	2.07
8	1.14	1.11
9	1.87	0.86
10	2.71	2.76
Average:	1.93	1.45

Table 5.6: Data on the Effect of Varying the Number of Power Levels on Location Accuracy

points was 16.

From Figure 5.8 and Table 5.6, we see that when the number of power level is three, almost always, I got better location accuracy than with a single power level. The average location error for one and three power levels were 1.93 and 1.45 meters, respectively. Further, while experimenting, I observed that when three power levels were used, reported locations were more stable than with one power level, and also they were almost always consistent. The possible reasons behind this are that when more power levels were used the reference signatures were more diverse, would cover fluctuations of the sending power levels, would remedy the multipath effect more effectively, the beacon signal would reach more area with significant signal strength, and mobile nodes would hear from more beacon nodes.

5.4 Comparison with MoteTrack

I have adopted an approach similar to that used in the RADAR and MoteTrack projects. However, there are significant differences between them and my work. This section studies the appropriateness of my system by comparing it to RADAR and MoteTrack projects.

RADAR is a WLAN-based technology. As mentioned previously, in future smart homes and in many context-aware ubiquitous applications, tiny, sensors are preferable since wireless LAN support is impractical [6].

MoteTrack is solely a location determination system. The aim of MoteTrack is emergency response applications. My location sensing system aims at future smart home applications. To fulfill this goal, this system along with location determination system provides additional components including a location storage component and an (OSGi) interface to the location sensing and storage system to enable application access through middleware framework. Smooth integration of these components was tested in a prototype home environment to study the effectiveness of the system. Further, user identities and room numbers were also integrated into the system and reported along with determined locations.

In centralized MoteTrack, mobile nodes send the signal signature to the base station and the base station calculates locations. In decentralized MoteTrack, locations are calculated by the Beacon nodes. In my system, mobile nodes calculate their own locations and supply the locations to the base stations. This approach can also provide privacy to the home residents who arrange it so that their locations are locally determined, but not reported to the HGD.

In the MoteTrack prototype, the target nodes (mobile nodes) whose locations are to be determined are always connected to a workstation (laptop) accompanying a display unit. The display unit displays locations and safe exit routes for the fire fighter. In my system, target nodes are always mobile and are tiny sensors. These nodes are not connected to any workstation or display unit but are, instead, used as badges carried by home residents as is desirable for home applications. This deployment of the sensors in my system fits well to the working scenario in smart homes.

The location accuracy for MoteTrack was two and three meters in 20 and 80 percent of the cases in indoor building environment. I got 28^{th}, 50^{th}, 86^{th}, 90^{th}, and 97^{th} percentile location errors under 1, 1.5, 2, 2.5, and 3 meters respectively in home environment with an average location error of 1.5 meter. The room level accuracy was 90%.

Based on the above discussions, I conclude that my location determination system is more suitable for home environments than MoteTrack or RADAR in terms of location accuracy, data collection phase flexibility, and convenience of use by the home residents. Further, my location storage and interoperability component make the system easily integrable with the envisioned future home environment, and will also open the door for many useful home applications. MoteTrack and RADAR lack this potential.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Location-aware ubiquitous smart home applications are of great practical importance by virtue of the numerous applications they enable that will automate useful home activities with little or no user intervention. Though location determination is a well-researched area, location sensing in homes and similar indoor environments is fairly new, and still challenging. Moreover, sensor networks have become a new and promising sensing paradigm that is extensively used in outdoor, and indoor applications to provide context awareness.

My thesis research has designed, developed, and assessed a sensor network-based location sensing system for smart home applications. The system provides reasonable location accuracy that is appropriate for most smart home applications. I have associated user identities with their locations, which is required for customization of applications based on users' preferences. A proof of concept storage and retrieval system for locations has also been developed that will support new applications, such as locating frequently lost objects, and tracking past locations. Moreover, I implemented an OSGi-based middleware interface to the location sensing and storage services so that other smart devices, services, and applications can access location information through the middleware. These two components will make my location sensing system easy to integrate into future context aware smart home frameworks.

6.2 Future Work

This work opens several new research directions for further study. As an immediate extension, the location sensing service can be tested, assessed and adjusted in a real home to address the challenges that arise. The sensing and the storage system could also be extended to sense and store environmental contexts such as temperature and humidity. The same sensor network, can of course, be utilized to sense location and environmental data concurrently, and this could be further studied. The storage system is also extensible to store users' preferences and habits (e.g. how they operate devices both past and current). Strategies need to be developed to find appropriate persistent queryable storage techniques. Further research can be done to find useful techniques and algorithms, perhaps using data-mining or AI techniques, to analyze the database to extract user habits and device operation patterns. Developing strategies or algorithms to define rules, to represent the rules in a proper way, and to make them accessible to other home and/or remote services/applications is worth further research.

Bibliography

[1] OSGi Alliance. OSGi. http://www.osgi.org. Accessed on 01-03-2006.

[2] P. Bahl, A. Balachandran, and V. Padmanabham. Enhancements to the RADAR User Location and Tracking System. Technical Report MSR-TR-2000-12, Microsoft Research, February 2000.

[3] P. Bahl and V. Padmanabham. A Software System for Locating Mobile Users: Design, Evaluation, and Lessons. Technical Report MSR-TR-2000-12, Microsoft Research, February 2000.

[4] P. Bahl and V. Padmanabhan. RADAR: An In-Building RF-Based User Location and Tracking System. In *Proc. of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 775–774, March 2000.

[5] M. Capkun and J. Hubaux. GPS-Free Positioning in Mobile Ad Hoc Networks. In *Proc. of the 34th Hawaii Int'l Conference on System Sciences*, pages 10–10, January 2001.

[6] D. Cook and S. Das. *Prediction Algorithms for Smart Environments*, chapter Smart Environments: Technology, Protocols, and Applications, pages 175–192. Wiley-Interscience, 2005.

[7] Microsoft Corp. Microsoft Easy Living. http://research.microsoft.com/easyliving. Retrieved on 01-03-2006.

[8] Microsoft Corp. Universal Plug and Play (UPnP) Overview. http://msdn.micro soft.com/library/default.asp?url=/library/en-us/wceoak40/html/coconupnpove rview.asp. Retrieved on 01-03-2006.

[9] D. Cotroneo, S. Russo, F. Cornevilli, and V. Vecchio. Implementing Positioning Services Over an Ubiquitous Infrastructure. In *Proc. of the 2nd IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, pages 14–18, May 2004.

[10] T. Dillon. Pervasive and Ubiquitous Computing. http://www.nestafuturelab.org /viewpoint/art71.htm. Retrieved on 01-03-2006.

[11] Y. Ding, C. Wang, and L. Xiao. Virtual Ruler: Mobile Beacon Based Distance Measurements for Indoor Sensor Localization. http://www.cse.msu.edu/s̃andeep /posterworkshop2006/poster13.ppt. Retrieved on 01-06-2006.

[12] Y. Fukuju, M. Minami, H. Morikawa, and T. Aoyama. DOLPHIN: An Autonomous Indoor Positioning System in Ubiquitous Computing Enviroment. In *Proc. of the IEEE Workshop on Software for Future Embedded Systems*, pages 53–56, May 2003.

[13] D. Gay, P. Levis, R. Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proc. of Programming Language Design and Implementation (PLDI)*, pages 1–11, June 2003.

[14] Improved Outcomes Software Group. Distance Metrics Overview. http://www.improvedoutcomes.com/docs/WebSiteDocs/Clustering/Clustering _Parameters/Distance _Metrics _Overview.htm. Retrieved on 01-03-2006.

[15] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. The Anatomy of a Context-Aware Application. *ACM Wireless Networks*, 8(2/3):187–197, 2002.

[16] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The Gator Tech Smart House: A Programmable Pervasive Space. *IEEE Computer*, 38(3):50–60, March 2005.

[17] P. Hii and A. Zaslavsky. Improving Location Accuracy by Combining WLAN Positioning and Sensor Technology. http://www.sics.se/realwsn05/papers/hii05i mproving.pdf. Retrieved on 01-03-2006.

[18] Crossbow Technology Inc. Mica2/Mic2Dot: Products and Specifications. http:// www.xbow.com. Accessed on 01-03-2006.

[19] HAVi Inc. Home Audio Video Interoperability. http://www.havi.org. Accessed on 01-03-2006.

[20] justEtc. GPS Overview. http://www.justetc.net/smartspace/gps.php. Retrieved on 01-03-2006.

[21] D. Kidd, J. Robert, D. Gregory, G. Christopher, E. Irfan, B. MacIntyre, M. Elizabeth, E. Thad, and W. Newstetter. The Aware Home: A Living Laboratory for Ubiquitous Computing Research. In *Proc. of the 2nd Int'l Workshop on Cooperative Buildings*, pages 191–198, October 1999.

[22] knopflerfish.org. Knopflerfish OSGi. http://www.knopflerfish.org. Accessed on 01-03-2006.

[23] A. Kotanen, M. Hannikainen, H. Lappakoski, and T. Hamalainen. Experiments on Local Positioning with Bluetooth. In *Proc. of the IEEE Int'l Conference on Information Technology: Coding and Computing*, pages 297–303, April 2003.

[24] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler. The Emergence of Networking Abstractions and Techniques in TinyOS. In *Proc. of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, pages 1–14, March 2004.

[25] D. Liu, P. Ning, and K. Du. Attack-Resistant Location Estimation in Sensor Networks. In *Proc. of the 4th Int'l Conference on Information Processing in Sensor Networks*, volume 4, pages 99–106, April 2005.

[26] K. Lorincz and L. Li. MoteTrack: An Indoor Location Detection System for Sensor Networks. Technical Report, Harvard University-Division of Engineering and Applied Sciences, January 2004.

[27] K. Lorincz and M. Welsh. MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking. In *Proc. of the Int'l Workshop on Location and Context-Awareness*, pages 63–82, May 2005.

[28] Sun Microsystems. Jini Network Technology - Overview. http://wwws.sun.com /software/jini/overview/. Retrieved on 01-03-2006.

[29] P. Misra, B. Burke, and M. Pratt. GPS Performance in Navigation. In *Proc. of the IEEE (Special Issue on GPS)*, volume 87:1, pages 65–85, January 1999.

[30] T. Moran and P. Dourish. Introduction to this Special Issue on Context-Aware Computing. *HCI*, 16(2-4):87, 2001.

[31] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS). *IEEE GLOBECOM*, 5:2926–2931, November 2001.

[32] Smart Antenna Laboratory: Georgia Institute of Technology. Multipath. http://users.ece.gatech.edu/ mai/tutorial_multipath.htm, 2006. Retrieved on 01-03-2006.

[33] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. of the 6th Annual Int'l Conference on Mobile Computing and Networking*, pages 32–43, Aug 2000.

[34] Northwestern University Qualitative Reasoning Group. What is Triangulation? http://www.qrg.northwestern.edu/projects/vss/docs/Navigation/1-what-is-triangulation.html. Retrieved on 01-03-2006.

[35] Intel Research. Overview: Computing Woven Seamlessly into Everyday Life. http://www.windowsfordevices.com/articles/AT3234214428.html. Retrieved on 01-03-2006.

[36] SearchNetworking.com. Pervasive Computing. http://searchnetworking.techtarget.com/sDefinition/0,, $sid7\_gci759337, 00$.h tml. Retrieved on 01-03-2006.

[37] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha. Tracking Moving Devices with the Cricket Location System. In *Proc. of the 2nd USENIX/ACM MOBISYS Conference*, pages 190–202, June 2004.

[38] R. Stoleru, T. He, and A. Stankovic. Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks. In *Proc. of the IEEE Int'l Conference on Local Computer Networks*, volume 29, pages 480–489, 2004.

[39] R. Stoleru and A. Stankovic. Probability Grid: A Location Estimation Scheme for Wireless Sensor Networks. In *Proc. of the 1st IEEE Int'l Conference on Sensor and Ad Hoc Communications and Networks*, volume 1, pages 430–438, October 2004.

[40] M. Tavakoli and S. Yamamoto. Human Tracking Devices: the Active Badge/Bat and Digital Angel/Verichip Systems. http://islab.oregonstate.edu/koc/ece399/f03/explo/shiraji-yamamoto.pdf, 2006. Retrieved on 01-03-2006.

[41] S. Tekinay. Special Issue on Wireless Geolocation Systems and Services. *IEEE Communications Magazine*, 36(4):28–28, April 1998.

[42] Vicom. Radio Localization of Mobile Terminals in Wireless Networks. http://www.elet.polimi.it/dsp/tlc/position.htm. Retrieved on 01-03-2006.

[43] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.

[44] Wikipedia. Ubiquitous computing. http://en.wikipedia.org/wiki/Pervasive_Computing. Retrieved on 01-03-2006.