A Hierarchical Location Tracking System
Project Report

Dhruv Pandya
{dhruv.pandya@doc.ic.ac.uk}

Supervisor: Dr. E. C. Lupu
{e.c.lupu@doc.ic.ac.uk}

Second Marker: Professor Morris Sloman
{mss@doc.ic.ac.uk}

June 13, 2002

Abstract

The proliferation of mobile computing devices and the development of high-speed, low-cost wireless networks has created ample opportunity for location systems. Most location systems tend to develop around specific technologies, and hence are restricted by the limitations of the technology. This in turn restricts the application set that may be deployed on the system.

Future mobile devices are expected to feature multiple wireless technologies. This shall create the opportunity to use various technologies to infer the same location, and thus free applications from the constraints of technology limitations. Furthermore, applications shall be allowed to exploit a larger set of location information.

In this project, we have proposed and implemented data collection architecturesfor Bluetooth, IEEE 802.11 Wireless LAN, GPS and Terminal Login. Sensor-independent location estimation algorithms have been proposed and implemented in a prototype system for indoor location tracking. Furthermore, we propose a basic fusion algorithm with the intention of improving accuracy. All the algorithms have been evaluated experimentally using Bluetooth and IEEE 802.11 Wireless LAN.

Acknowledgements

- I would like to thank the project supervisor, Dr. Emil Lupu, for his advice on the project and final report, and for meetings without prior appointments. Thanks are also extended to the second marker, Prof. Morris Sloman for his insight into the intermediate outsourcing and review reports. Special thanks go to Susan Eisenbach, Anne Oneill and all the people on level 5 at the Computing department for keeping up with my measurement experiments.

- I sincerely appreciate all the assistance provided by Ravi Jain in creating the substance of this project and demonstrating equal enthusiasm and encouragement throughout the duration of the project. I would also like to thank Telcordia Technologies for providing the Bluetooth equipment which made this project feasible.

- A special thank you to my Mum and Dad for their continuing support and encouragement. And a very special thanks to my kid sister for cheering me up when things weren't going so well!

Contents

Chapter 1

Introduction

1.1 Motivation

"Thursday morning, 8:15. Jo leaves home for work. As she sits in her car, an LCD mounted on the dashboard comes alive and draws a road map, highlighting a route to take to work. She does not speak a word. She arrives at Rammond Park in 22 minutes; considering the traffic on the I-80, it should have taken her just over an hour. As she approaches the main lobby, the main security doors spring open and she walks away to the elevator. She waits four seconds before the elevator arrives and she is lifted to level 12. She has not pressed a single button. As she exits the elevator, the corridor leading to her room is lit up brightly. She stops by Mark's to say Hello. And pictures of their weekend skiing trip download onto her PDA. She enters her room and continues to work at her desktop on the report she started last night."

Such a scenario is not unimaginable: the pressure sensor under the car seat switches on the on-board computer on Jo's car; a location system uses Jo's mobility patterns, current time, location and its GPRS connection to the local council map and traffic database to plot an optimum route to her destination; the wireless LAN network in the office car park authenticates and tracks Jo as she approaches the main entrance doors, and opens them as she enters; the reception wireless network alerts the elevator scheduler to cater for Jo's level 12 ride, and the lighting system is put on standby as Jo approaches floor 12. In Mark's room, the Infrared network, as per Mark's instructions, uploads the pictures Jo's PDA, while the Bluetooth network around Jo's office synchronizes her work and home files to enable her to continue without further delay.

The current proliferation of mobile computing devices and the development of high-speed, low-cost wireless networks nurtures the development of such ubiquitous systems. Ubiquitous systems adapt to the changing environment or context, and hence provide a more enriching experience to the end user. Contextual information can be sourced via a sensor link, for example, a thermostat to measure temperature.

Recent attention has focussed on one element of context: location. When Jo sits in her car and starts driving, her location could be monitored using the Global Positioning System (GPS). Her location in the car park could be tracked using Wireless LAN and her position at the entrance doors serves as input to have them opened. Furthermore, when the reception system detects that she is positioned at the elevator, the doors open and take her to level 12. Her location outside the elevator on level 12 causes the corridor lights to switch on

and her location in Mark's room enables the automatic download of pictures onto her PDA. Her presence in the Bluetooth network causes an automatic file synchronization between her laptop and desktop.

This demand for location information from multiple sources shall become increasingly common, with future mobile devices being equipped with multiple technologies such as GPS, cellular cell-sector ID, Bluetooth and 802.11 wireless LAN [11]. Furthermore, the following discussion highlights the need for multiple sources of location information:

- All technologies may not operate everywhere. For example, GPS will not work inside buildings and around tall structures, whereas these environments are more suited to 802.11 wireless LAN.
- Each sensor technology operates independently, hence any location estimation is technology-specific. Furthermore, the variety of techniques and lack of standards for location representation, make it difficult to co-relate location information from different sensors, or develop estimation techniques that are technology independent. However, efforts are underway to allow such interoperatability across wireless platforms [19].
- Applications may demand various levels of accuracy. For example, to find the nearest taxi, the cell-sector ID could be sufficient, whereas Bluetooth would be useful in identifying the location of spare parts in a factory. Furthermore, some applications may require location information in a continuous data stream, whereas others may demand periodic updates at certain intervals.

1.2 This Project

In this project, an attempt is made to manage location information from multiple sources including Bluetooth, 802.11 wireless LAN, GPS and Terminal Login. For each technology, data collection architectures have been proposed and implemented, and sensor data processed to provide location information. For Bluetooth and 802.11 wireless LAN, sensor-independent and data fusion algorithms have been explored in an indoor location tracking environment to achieve higher location accuracy. The effectiveness of the algorithms has been evaluated using experimentation and empirical measurements.

1.3 Report Structure

Chapter 2 begins with an overview of location systems. Location estimation and data representation techniques are presented. A technical overview of contemporary sensor technologies is presented, followed by a discussion about related work in the field.

Chapter 3 provides details about location estimation techniques, data collection architectures and location data models used in this project.

Chapter 4 provides a detailed account of static scene analysis and sensor-independent algorithms used to improve location accuracy.

Chapter 5 describes our experimental test bed, data collection procedures, and a performance evaluation of the algorithms described in Chapter 4.

Chapter 6 presents a prototype implementation of the system.

Chapter 7 summarizes the achievements of the project, identifies limitations and proposes further work.

Chapter 2

Background

2.1 Chapter Summary

- Location estimation techniques, location data reprsentation models and other properties of location systems are described.
- An overview of sensor technologies is given.
- Contemporary location systems are described.

2.2 Properties of Location Systems

A broad set of issues arise when discussing location systems. In this section, these attributes are discussed to update the reader on the current jargon.

2.2.1 Location Models

A location system must identify a data model for representing locations of mobile and fixed objects. This data model is called a *location space*. A location space can be designed as an n-dimensional coordinate system, or more abstractly as a set of identifiers or names that are associated to physical locations. The former approach is known as a *geometric*, while the latter as *symbolic* modelling. The longitude-latitude coordinate system [20] used by GPS is an example of the geometric model, where as building, floor and room names are examples of a symbolic model.

A location system must address the requirements of both the location sensors and location-aware applications. For example, GPS adopts the geometric approach by using the longitude-latitude scale to represent locations. On the contrary, Login uses a symbolic model to map user terminals to room numbers. There exist applications that use a combined model, incorporating the features of both geometric and symbolic approaches. For example, the Intelligent Highway system defines the location of a vehicle in terms of postcode and physical quantities such as speed and heading.

Both modelling techniques share strong and weak features. Symbolic modelling makes use of identifiers that are discrete and well-structured. This makes it easier to define a hierarchical data model, which in turn enables multi-resolution processing. For example, Room 331 in Alexander Building could be identified using the hierarchical symbolic name, Alexander-331.

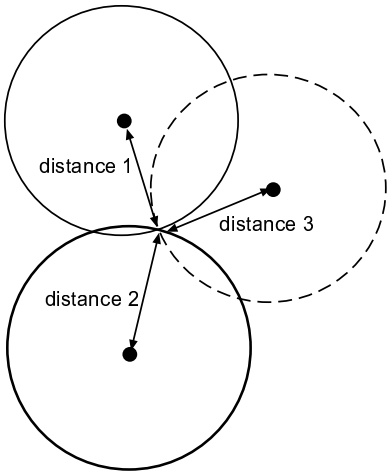

Figure 2.1: The Lateration method of Triangulation uses three distance measurements to compute a 2-D location estimate.

However, symbolic names suffer a level of indirection, for example in Login, where terminal names are mapped to room numbers.

Since geometric modelling expresses location in terms of points, areas and volumes on a coordinate system, accuracy of location information is preserved. Furthermore, it is possible to define spatial queries based on coordinates for the purposes of co-relation and containment. On the weak side, computations might need to be performed to satisfy a query. Apart from that, a translation mechanism is required to present information in human-readable form of symbolic data.

2.2.2 Location Estimation Techniques

Raw sensor data consists of physical quantities, for example, signal strength in dBm for Radio Frequency (RF) connections. A variety of techniques have been proposed to use sensor information and produce location information particular to a location space. This section discusses three techniques: Triangulation, Proximity and Scene Analysis.

Triangulation

Triangulation is based on the geometric properties of triangles. There are two methods used to estimate a user location.

- Lateration computes the location of a user by measuring its distance from multiple reference points. A two-dimensional position requires three distance measurements, and four distance measurements are required to compute position in three-dimensions. Figure 2.1 shows the estimation of a 2-D user location using three distance measurements. The challenge then arises in determining the distance between the reference point and object of interest. Three approaches include:
  - Autonomous systems obtain a *direct measurement* using a physical action or movement. For example, robots extend their arm until they make contact with a solid

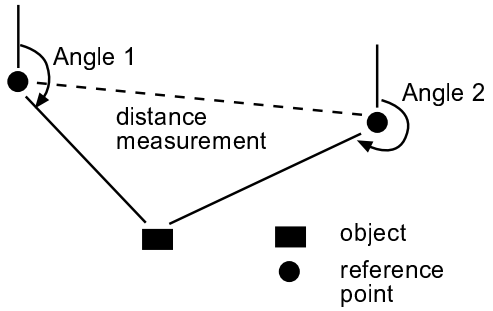

Figure 2.2: Computing the two dimension position using two angle and one distance measurement, using Angulation.

object. Such measurements are easier to understand but difficult to obtain due to the complexities involved in coordinating autonomous physical movement.

- A *time-of-flight* analysis involves the measurement of the time taken to travel to an object at constant velocity. The resulting time and velocity values are used to compute distance. This technique has gained popular use with ultrasound and RF based location systems. In such cases, an RF or ultrasound pulse is transmitted and the time to reach the receiver is noted. A common problem has been multiple reflections in the environment which distort the times at the receiver. Systems using this technique have adopted statistical pruning solutions that select the time with the strongest pulse etc. Furthermore, it is necessary to synchronize the time between the transmitter and receiver, especially for RF based systems. It is possible to avert this problem by measuring the round-trip time at the transmitter using its clocks. This can be achieved by configuring the receiver to transmit a pulse on receiving one from the transmitter. On the other hand, systems like GPS, enclose a satellites' local time in the transmission. This time is used by a GPS receiver to compute the time of flight of the radio signal from the satellite to the receiver, and then used to estimate distance.
- The intensity of a signal reduces as the distance from the source of emission increases. For example, a radio signal is known to undergo an attenuation of $1/r^2$ at a distance r from the source. Mathematical functions correlating attenuation and distance, for a particular emission, can be used to estimate distance. This technique is usually not very effective in closed spaces such as office locations because the attenuation patterns are influenced by reflection, refraction and multipath effects. Hence, the resulting distance estimates are not very accurate.

- Angulation determines the location of an object based on angles, rather than distances. Two angle measurements and the distance between the reference points is used to compute the estimate. The angle measurements is between the reference point and object, as shown in Figure 2.2. An enabling technology for this technique is the phased antenna array, where multiple antennas with known separation measure the time of arrival of a signal. The difference in arrival times and geometry of the receiving array is then used to determine the direction from which the signal originated. The VHF Omnidirectional Ranging aircraft navigation system uses this method to find the three dimensional

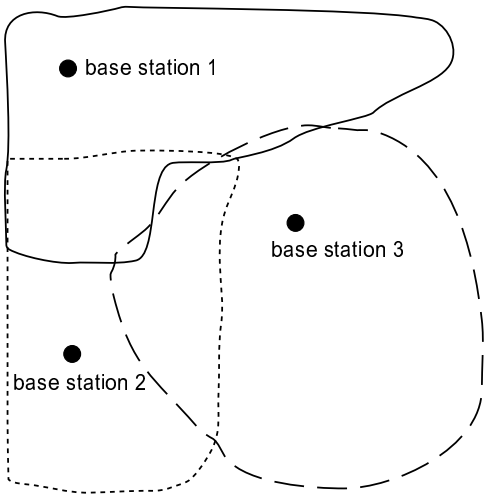

Figure 2.3: An example wireless LAN coverage setup. By determining the strongest connection to a base station, it is possible to infer the area where the user object is located.

location of an airplane [14].

Lateration has been more successful in finding application due to the variety of techniques that exist to estimate distance. Furthermore, Angulation requires knowledge of the geometry of its neighbouring reference points, making it a poor choice for commercial applications that demand quick deployment.

Proximity

This technique infers location when an object is 'near' a known location. The presence of an object in the vicinity is sensed by the physical medium of the sensor technology. Three general method include:

- Detecting physical contact. Technologies for detecting contact include pressure, touch and motion sensors. For example, motion sensors are available in commercial products triggering alarms, corridor lighting etc.
- Monitoring wireless access points. When a mobile device enters the coverage area of a wireless access point, the effective location of the device can be known by the definition of the coverage area. For example, figure 2.3 shows the possible coverage areas of three 802.11 wireless LAN base stations. A user with a wireless LAN equipped laptop could be in a region defined by the coverage of an access point that demonstrates the strongest wireless connection in terms of signal strength.
- Observing personal ID systems. Access to personal user information such as login histories, telephone records and credit card transactions make it possible to determine the approximate location of a user. For example, the electronic tags in E-Toll highway systems may be used to identify the most recent station passed by a user.

  Proximity-based systems may need to be combined with identification systems to provide symbolic or geometric location information. In the above wireless LAN example,

recognition of the strongest base station should be followed by the identification of the floor where the station is located. Furthermore, the proximity information available from this approach is limited to the technology in use. For example, in the wireless LAN case, the area identified by proximity methods could be as large as range of the wireless LAN implementations, whereas a region identified by Infrared could not be larger than a room.

Scene Analysis

This technique uses features of a scene observed from a particular point to infer location of objects in the scene. *Static* scene analysis compares features observed during system operation phase with predefined tables that maps features to locations. The tables are constructed through the observation of features in a simple scene. *Differential* scene analysis observes the difference in features between successive scenes and estimates location. For example, if furniture is a feature of an image of a room, comparison with a predefined set of furniture features can help locate a person in the room. Furthermore, differential analysis of the images can track user movements in the room. The nature of the comparison feature is determined by the technology used to extract the feature from the scene. The RADAR [3] project uses signal strength measurements on 802.11 wireless LAN connections as the characteristic feature to perform the analysis.

Scene Analysis permits passive observation of the scene, contrary to Triangulation and Proximity where participants are required to transmit and receive signals. The transmission of signals may in turn compromise user privacy and consume power on mobile devices. However, changes in the scene e.g. moving the furniture, may force the re-construction of the feature comparison tables.

2.2.3 Place of Computation

Computations involved in estimation techniques discussed in Section 2.2.2 can take place on the local object or a remote machine. The local object could take the form of a mobile device, for example, cell phone or ID tag. The remote machine could be the tracking access point or a server responsible for collecting and storing location information.

Processing on the local device places constraints in terms of minimum processor power, memory and battery life. However, user privacy is maintained because location is computed and stored on the local device, and published only if required by a user application.

On the other hand, remote processing places the computational burden on the infrastructure. This reduces the resource demand on the local device, which in turn opens up the system to a wider set of applications. In this model, location information would be stored at the server end. Hence access to information must be protected using control policies that may be set by users or administrators. In the example scenario, Jo can log on to the system and set up a policy where her location will not be available for the weekend period when she visits her aunt in Connecticut.

2.2.4 Accuracy, Precision and Granularity

Accuracy defines the error in distance of the calculated position from the actual position. *Precision* defines the statistical probability of achieving a desired level of accuracy. *Granularity*

is used to define the smallest possible physical space whose boundaries can be detected accurately. A concise summary of a location system is usually expressed in terms of some or all of these quantities. A trade-off may be achieved between these quantities to meet application requirements.

Various technologies provide different level of accuracy and the concept of data fusion is an open research topic. However, statistical error merging has been proposed to assess the effectiveness of these techniques.

2.2.5 Other Attributes

The attributes discussed next play a decisive role in the implementation and deployment of commercial systems.

Scalability The scale of a system may be assessed in terms of the coverage area per unit of infrastructure and number of objects that can be located per unit infrastructure per time interval. For example, GPS can serve an unlimited number of users worldwide, while electronic tags readers can only read one tag at a time. Besides that, time consideration is important due to the limited bandwidth available in sensing objects. For example, RF-based technology can tolerate a certain maximum number of communications before the channel becomes congested.

Cost Cost assessment of a location system usually considers time, capital and computational costs. Time costs cover issues such as infrastructure installation, administration and maintenance times. Capital costs include the price per mobile unit, cost of an infrastructure element and remuneration for personnel. Computational costs deal with the processing overhead required to compute a position in a location space from raw sensor data. For example, time and capital costs associated with the installation of an office wide infrared network depend on the office space and the scale of the required coverage. However, computational costs depend on the number of users of the system, but this may be catered for by using a high powered server system.

Identification An identification scheme is required to recognize or classify a located object. A common technique is to assign globally unique identifiers (GUID) to objects, and this GUID is revealed to the system when the object is located. A database access can retrieve further information pertaining to the object and allow further system specific functions. In the example scenario, Jo can be automatically authenticated with the office network using her GUID, and airport baggage can be routed to collection centres for all airports around the world.

Limitations Limitations of a sensor usually restricts the application set that may effectively use the technology. For example, call forwarding based on GPS is challenging because GPS signals are too weak to penetrate indoor locations. Similarly, line of sight is required for communication between infrared devices and this makes it necessary to install more infrared receivers per room to cater for the cases when the user body blocks a pulse.

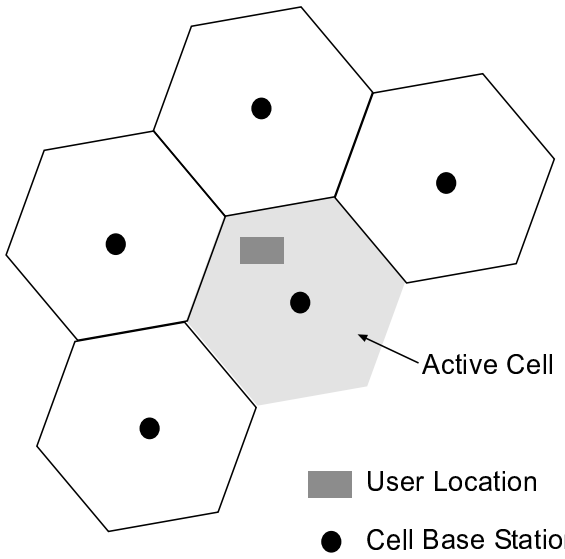

Figure 2.4: Proximity analysis in cell technology. The current active cell is the geographical location in which the user is present. The base station can be searched in a database to determine the exact street address or GPS coordinates of the station.

2.3 Sensor Technologies

Section 2.2 discussed various attributes of a location system that are affected by the choice of sensor technology. This section provides an overview of technologies that are currently being used to estimate location information. An effort is made to restrict the discussion to standards that have gained acceptance both in industry and academia.

2.3.1 Cell Technology

Mobile or cell phones [7] operate at radio frequencies between 824-849MHz. The technology is based on the idea of dividing a geographical location e.g. a city, into a number of smaller regions or cells. Each cell is served by a base station which connects to a Mobile Telephone Switching Office (MTSO). The MTSO is responsible for storing a subscribers active cell and routing calls to land lines.

It serves well as a location sensing technology with proximity analysis, discussed in Section 2.2.2, as a candidate technique to obtain the active cell of the user, as shown in Figure 2.3.1. The symbolic location, for example street address, or geometric location, for example, GPS coordinates, of the base station serving that cell may be obtained via a database search. However, since the size of the cell is sometimes as large as 10 miles, the granularity of location information obtained is high. Finer grained information may be possible to obtain using signal attenuation techniques discussed in Section 2.2.2. However, this technique may not be effective in cities because attenuation patterns may be distorted by tall buildings and large structures.

Despite these technical difficulties, Government regulations are requiring mobile phone operators to provide location estimation accuracy to within 150 metres in the case of emergency [17].

2.3.2 Global Positioning System

The Global Positioning System [2, 8] operates at Radio Frequencies between 1,227 and 1,575 MHz. A US Government project, it was deployed on 17 July 1995 and consists of 24 satellites (21 in active use, 3 backups). On the ground, five monitoring stations collect and provide ranging information to the master control. The master control calculates satellite orbits and forms navigation messages that are transmitted to the satellites via three ground antenna stations located around the world.

GPS uses the concept of *trilateration* to locate a user on the longitude-latitude coordinate system [20]. The use of this technique means that the receiver requires to know the distance to three different satellites to calculate longitude-latitude coordinates and four satellites to calculate a further quantity of altitude. The receiver calculates the distance to the satellite using the time of flight of RF wave from the satellite to the receiver. Time of flight analysis, discussed in Section 2.2.2, requires the time taken for a radio frequency signal to travel from the satellite to the receiver. This requires the satellite and receiver to be synchronized. Complex methods are used to achieve this synchronization and also account for reduced speed of radio waves in the Earth's atmosphere.

GPS measures location to within 100m. However the Wide Area Augmentation System [18] provides offers a location accuracy of less than 3m. depending on the complexity of the receiver. Since the orbit of the satellites is 11,000 miles above the surface of Earth, the radio signal received at the receiver is weak. Hence GPS does not work effectively in cities shielded by tall structures, and inside buildings. Furthermore, GPS does not provide any direction information.

2.3.3 Bluetooth

The Bluetooth wireless technology [9, 15] was originally inspired at Ericsson Mobile Communications in 1994 and with the help of an industry-wide effort, the first specification was released in early 1998. Although initially intended to replace cables connecting mobile nodes to static hosts, Bluetooth allows point-to-point communication and consequently encourages mobile ad-hoc networking and the concept of Personal Area Networks.

Operating between 2.4-2.48GHz (the Industrial, Scientific and Medical band), it is a low-power, low-cost, short-range (approximately 10m) Radio Frequency solution providing synchronous voice communication at 64kbps and asymmetric data communication at 723kbps.

As Figure 2.5 explains, each Bluetooth device can act as a *master* device and support 7 active *slave* devices simultaneously. A slave device can also acts a master and support its own network, hence the network of the master and the slave form a *piconet*, shown in Figure 2.6. In addition to this, the specification also defines *parking* modes that enables one device to connect to a maximum of 255 devices. All these feature enables the technology to support a wide variety of mobile network topologies.

Apart from allowing the formation of a hierarchy of connections, the technology helps reduce interference by using frequency hopping among 79 frequencies at 1MHz intervals. Security features afforded by the technology include authentication, encrypted connection and session key generation.

Bluetooth may be used like any other Radio Frequency based sensor, estimating locations in small spaces provided by office locations or family homes. Proximity, scene analysis, signal attenuation models and time-of-flight analysis are potential distance estimation techniques.

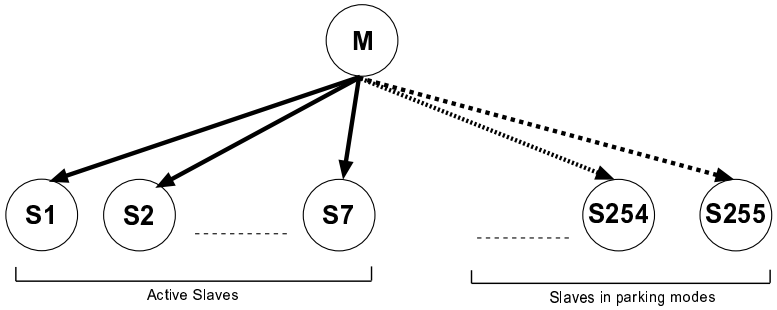

Figure 2.5: The diagram shows a Bluetooth piconet. One master can have a maximum of 7 simultaneous active connections. A maximum of 255 slaves can be supported by allowing the slaves to enter special parking modes. Each slave is restricted to only one master.

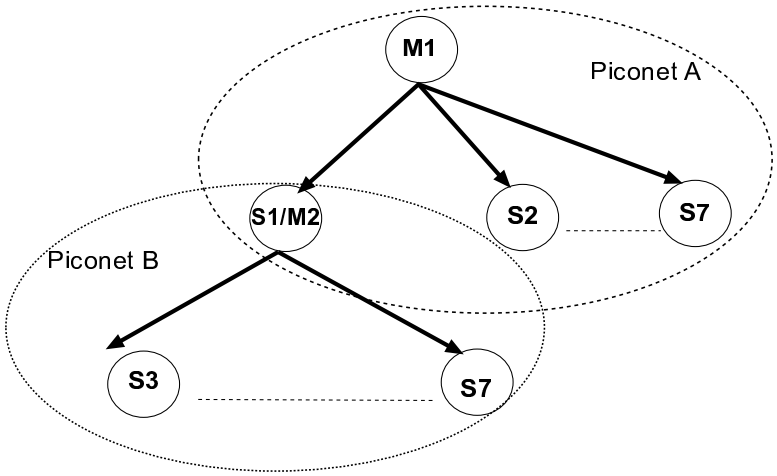

Figure 2.6: The diagram shows that a slave participating in one piconet, can master another piconet. S1 is a slave in piconet A and the master of piconet B.

Signal attenuation models may not provide accurate estimates due to attenuation patterns being influenced by reflection, refraction and multipath effects from room walls. Time-of-flight analysis also may not fare well because short distances and high speed of radio waves will require very accurate clocks and place strong constraints on transmitter-receiver clock synchronization. However, its low range makes it particularly useful with proximity-based solutions. For example, if a user device is found to be in an area covered by a Bluetooth access point, the user can be estimated to be located within a 10m radius of the access point. Static scene analysis also presents interesting opportunities, although the initial calibration required by the analysis could significantly increase time costs.

2.3.4 802.11

In July 1992, the 802.11 Working Group decided to concentrate its RF studies and standardization efforts on the 2.4 GHz spread spectrum ISM bands. The resulting specification [12] formalized the physical and medium access layer (MAC).

The physical layer consists two RF and one infrared specifications. The RF supports 1 Mbps and 2Mbps using different modulation techniques. The MAC supports all three physical layer specifications using Carrier Sense Multiple Access with Collision Avoidance(CSSMA/CA) [Ethernet uses CSMA with Collision Detection]. The MAC is responsible for access control functions such as coordination, error checking and data unit delimiting.

The standard supports two network topologies [21]: mobile stations are able to communicate with each other in ad-hoc fashion, or the stations may communicate via access points which are part of a distribution system e.g. Ethernet. The latter case allows integration with other networking technologies, hence the development of wide-area networks.

The standard provides wireless connectivity of fixed, portable and moving stations moving at pedestrian and vehicular speeds within a local area. Other features include multicast and broadcast services, authentication, time-bound delivery and power management.

Techniques applicable to Bluetooth may also be found useful with 802.11. Static scene analysis was found particularly effective by RADAR, discussed in Section 2.4.3, for indoor location tracking. However, since the technology provides a range of approximately 100m, proximity may not be an ideal candidate as the granularity of the estimates may increase by a large proportion compared to the 10m offered by Bluetooth. In contrast, attenuation techniques may use the technology in open spaces such as college campuses to provide accurate estimates at low time and capital costs.

2.3.5 Non RF-based Sensors

This discussion focusses on technologies that do not involve RF waves. Many technologies successfully work with radio frequency based solutions to offer better location sensing opportunities.

Ultrasound Ultrasound are high-frequency sound waves with a speed of 355 metres per second in 21 degrees Celsius air. The high-frequency, low-wavelength nature of the wave prevents large interaction with obstructions and hence enables directional penetration without suffering loss in strength. Furthermore, ultrasound does not require line of sight because it undergoes both reflection and diffraction. Successfully used with RF-based sensors, it has gained popular use in time-of-flight analysis: Radio is used to synchronize the clocks on the devices, before the transmitter signals an ultrasound pulse. The time

taken for the pulse to reach the receiver is used to estimate distance, which contributes to the lateration process of triangulation, discussed in Section 2.2.2 to estimate a location.

Infrared The Infrared Data Association (IrDA) specification details the infrared physical layer: a serial, half-duplex, asynchronous connection with transfer rate of 115.2kbps - 4Mbps, having a range of 1m and viewing angle between 15-30 degrees. Higher level protocols allow control and sharing of the infrared medium, discovery of services on a peer device, and segmentation and re-assembly of application data-units. Like Bluetooth, infrared is a compact, low-cost and low-power solution initially aimed at cable replacement. However it provides a higher-bandwidth connection than Bluetooth. Infrared requires line of sight and cannot penetrate walls. This makes it particularly suited for proximity-based location estimation techniques. For example, the Active Badge system discussed in Section 2.4.1 provides symbolic location information representing an infrared constrained volume like a room. Like ultrasound, infrared may also be used to provide direction information.

Login Login information may be used to approximate the location of a user in indoor environments of offices. In a networked environment, many operating systems allow the query of terminal information where a user is logged on. For example, the *finger* command in a networked Linux environment returns information such as login name, login time, idle time and terminal name. The terminal may be identified to a room or laboratory using a database search. Furthermore, the activity of a user at the terminal may be used to infer the whereabouts of the user.

Video Capture Video processing fits well with the Scene analysis location estimation technique described in Section 2.2.2. Various technologies being pursued include gray scale [10], omnidirectional [4] and multiple colour stereo. Active research areas include face detection [6], gesture understanding, body-part, whole-body and group tracking. In addition to this, other primary areas of interest have been outdoor location tracking [24, 5] and trajectory estimation [22]. The Easy Living project discussed in Section 2.4.6) uses cameras as the primary sensor technology.

2.4 A Survey of Location Systems

This section reviews location systems that being developed commercially and in the research field. For each system, the sensor technology, location estimation technique and architectural inventions are outlined.

2.4.1 The Active Badge Location System

One of earliest projects involving location tracking, the Active Badge Location System [26], developed at Olivetti Research Ltd, aims for providing location information for call forwarding. The Active Badge is an infrared device transmitting an IR signal that is received by a network of Badge Sensors joined on an Ethernet. The sensors are polled by a master sensor to record the sighting. The system achieves an accuracy of 9cm, 95% of the time.

Since infrared cannot penetrate walls it serves well in defining office boundaries. However, this also requires the installation of Badge Sensor per room, and there also exists the possibility

of dead spots where infrared cannot penetrate. In addition to this, sunlight and fluorescent light are known to interfere with infrared.

2.4.2 Cricket Location-Support System

Cricket is a location system developed at the Laboratory for Computer Science at Massachusetts Institute of Technology, aimed at providing location information to resource discovery services such as Jini [13] of Sun. It focusses on indoor location estimation, with a sensor infrastructure consisting of custom-built cheap radio frequency hardware. Stationary devices or beacons transmit radio pulses that contain short strings describing the geographical location in which they are present. The radio transmissions from the beacon are picked up by mobile clients termed listeners.

Time of flight analysis is used to estimate the distance between a beacon and a listener. A radio transmission from the beacon is accompanied by a concurrent ultrasound pulse. The listener measures the time between the first receipt of the radio and ultrasound to infer the distance to the transmitter. However, the ultrasound is expected to arrive while the radio transmission is being received.

This distance estimation technique may be simple, but it does present further challenges: There is no coordination between the beacons, hence radio and ultrasound transmissions between neighbouring devices interfere, causing incorrect readings at the listener. To avoid such interference effects, the following solutions have been proposed:

- A randomized transmission schedule is adopted where the frequency of transmission varies uniformly between a time interval. For example, four transmissions are made in one second distributed in [150,350]ms. This avoids repeated synchronization between neighbouring transmissions and hence prevents persistent collisions.
- The radio transmission rate is dropped so that a fixed size string would take sufficiently long for the ultrasound to arrive from the same beacon. This would enable a correct correlation between the ultrasound and the radio.
- It is possible that an ultrasound signal to interfere with the radio of a neighbour. This scenario is prevented by positioning the beacons to avoid such interference patterns. Furthermore, if two radio signals interfere, the listener would opt for the stronger signal.
- Finally, the listener undertakes a statistical analysis whereby the beacon with the highest frequency or shortest mean distance is selected.

Cricket is able to achieve a location granularity of 4 x 4 square feet. User privacy is a key strength, and is achieved by allowing a client application to initiate location estimation when required. The use of cheap hardware and lack of prior knowledge of beacon positions pushes down time and capital costs respectively, making it suitable for large scale deployment. However, the system places the computational and consequent power burden on the resource-constrained mobile receivers, which can lead to shorter mobile usage and high application response times.

2.4.3 RADAR

RADAR [3] is an indoor location tracking system developed at Microsoft Research. The project explores the use of static scene analysis in indoor location tracking using the IEEE

802.11 wireless LAN standard as its sensor technology. As discussed in Section 2.2.2, scene analysis involves the measurement of certain features of interest of a simple scene. The same features are measured at system operation time and compared against the simple scene features to infer location.

In the case of RADAR, the feature of interest is the signal strength of a radio connection between a 802.11 wireless LAN equipped mobile and base station. Measurement samples in the simple scene, termed offline, consist of signal strength data being collected at various locations in an indoor floor environment. The signal strength information collected at run-time is then compared with the offline tables to estimate location. The comparison involves the computation of the Euclidean distance between the run-time and offline samples, and the offline sample that minimizes this measure are chosen as the estimate location.

Since the base stations are arranged to provide overlapping coverage of an area, the above comparison process yields estimates from more than one base station. RADAR calculates the average of all the estimate locations, and the accuracy for the averaged estimate is found to improve.

Further experimentation has concluded with the following observations:

- Depending on the physical dimensions of the region of interest, the number of locations where offline signal strength measurements are conducted may be reduced. However to maintain good accuracy, the chosen locations should be distributed uniformly over the area.
- The number of signal strength measurements collected and averaged at run-time for comparison purposes does not hold a major impact on the location accuracy.
- User orientation while collecting both offline and run-time information, significantly affects estimation accuracy.

RADAR has also explored radio attenuation modelling, discussed in Section 2.2.2, as a distance estimation technique. The empirical offline measurements are replaced by data generated using the propagation models. In particular, the Floor Attenuation Factor propagation model is used, and the parameter of the model have been determined experimentally. The reader is directed to [25] for the mathematical details on this model.

This effort at employing scene analysis to indoor location tracking using radio-based sensors has produced encouraging results with the empirical and propagation modelling techniques generating an accuracy of 2m to 3m and 4.3m respectively. The use of 802.11 implementations is becoming increasingly common and the RADAR concept of exploiting existing infrastructures for location tracking helps reduce capital costs and enables quicker deployment. In contrast, the task of collecting initial offline signal strength data is impractical for large areas. The use of propagation modelling may potentially solve this problem, although it does require initial experimentation to establish model-specific parameters.

2.4.4 Sentient Computing using Bat

AT&T's sentient system [1] uses a model to represent the real world. Every object of interest is included in this model. The model is updated by sensors and used by applications to control the environment and query its state.

The mobile is a small device consisting of a RF receiver and an ultrasound transmitter, and termed as Bat. A wireless cellular network sends an RF pulse to synchronize Bats with ceiling

receivers and the time-of-flight of the ultrasound from the bat to the receiver is measured to compute the 3D position of the Bat. This location technique assists in maintaining an accurate model by achieving accuracy within 3cm in 95% of the readings. In addition to this, erroneous location updates are filtered by a set of rules.

2.4.5 Cooltown

Boomtown [16] is a research program at Hewlett-Packard Laboratories, focussing on mapping physical locations to Web resources. Infrared beacons, RF ID tags and bar codes are used as bridging technologies to provide a URL that is used by a client device to download the web page via a RF wireless link. The web page provides a list of services associated with the physical object to which the bridge is attached. Accuracy, precision and granularity are device-dependent, and vary per object and location.

Scalability and cost issues arise during the initial deployment and maintenance of the devices. Apart from that, the limited range of the bridging devices forces the user to be close to the actual location or object.

2.4.6 Easy Living

The Easy Living [23] project at Microsoft Research uses colour stereo cameras for locating people within an area. The combined depth information and colour images, provided by the camera, are used to build a background model. This model is built in the absence of people in the area covered by the cameras. During run-time, background subtraction is conducted on a live camera image to bring out the pixels corresponding to a person. Further visual processing is conducted before the image used in person tracking.

The system is able to achieve an accuracy of 10cm. It has been tested to work satisfactorily tracking 3 people with 2 cameras.

The advantage of visual tracking is that, unlike the RF, ultrasound and infrared based systems, the user does not have to carry any devices in order to be located. However, the issue of user privacy will require significant effort as a person will be tracked at the pixel level irrespective of any access control. Furthermore, the use of expensive hardware suggests high installation, if not maintenance, costs. The system could suffer with increasing scene complexity and occlusive motion.

2.5 Conclusion

Properties of Location Systems This section placed emphasis on the general attributes of location systems. In particular, Triangulation, Proximity Analysis and Scene Analysis were discussed as possible location estimation technques. Furthermore, the role of geometric and symbolic models in data represention techniques was discussed. Accuracy, precision and granularity were introduced as metrics for comparing location systems. The section also outlined that scalability and costs associated with location systems play a dominant role in building, deploying and maintaining commercial systems.

Sensor Technologies This section provides a technical overview of various sensor technologies and discusses the use of each sensor with the different location estimation techniques. In particular, radio based sensors are used with all the three location estimation techniqes, where as infrared was found to work effectively with proximity analysis.

Location Systems This section provides an overview of various location systems. The Active Badge system uses infrared with proximity analysis, Cricket combines ultrasound and radio to perform a time-of-flight analysis and RADAR uses the IEEE 802.11 Wireless LAN standard in indoor location tracking. Other project like Easy Living use scene analyis using video-tracking.

Chapter 3

System Architecture and Design

3.1 Overview

The discussion on location systems in Chapter 2 outlined the use of various sensor technologies in location estimation. Each system offered a certain set of attributes in terms of accuracy, precision, granularity and scale. This in turn determined the range of applications that can make effective use of the available location information. For example, the Cricket system may only provide location information for the area covered by the sensor infrastructure, resulting in the application set being designed to cater for that particular area.

We propose a hierarchical location tracking system that sources location information from a variety of technologies, and offers a rich API to applications to exploit the benefits of different technologies and achieve their function in the best possible way. Figure 3.1 gives an overview diagram of the system. The project has focussed on the use of standard technologies to allow inter-operatability across all hardware implementations. Furthermore, the choice of radio technology allows for different levels of granularity to be incorporated in the location information.

3.2 Location Estimation using RF-based sensors

Bluetooth, 802.11 wireless LAN and GPS form are the radio-based sensors used in this project. Success enjoyed by related projects like RADAR and Cricked make Bluetooth and 802.11 wireless LAN ideal candidates for indoor location tracking. Furthermore, a Bluetooth network was relatively easy to establish in our experimental testbed already covered by an existing 802.11 wireless LAN infrastructure. Several location estimation techniques, discussed in Section 2.2.2 have been explored. The following is the reasoning on selecting Scene Analysis as our distance estimation technique:

- Time-of-flight analysis, discussed in Section 2.2.2 is not particulary suitable because of the difficulties in synchronizing the clocks on transmitter and receiver Bluetooth and 802.11 wireless LAN devices. While the software efforts that synchronize clocks on a network to within 1ms-5ms are respected, the short distances of indoor locations and high speed of radio-frequency waves place a stringent requirement on clock synchronization accuracy. For example, a radio wave travelling at 3 x $10^8 ms^{-1}$ would cover a typical indoor distance of 20m in 6.7 x 10^{-5}ms. Hence the 1ms-5ms error present in the software clock synchronization would not be acceptable.

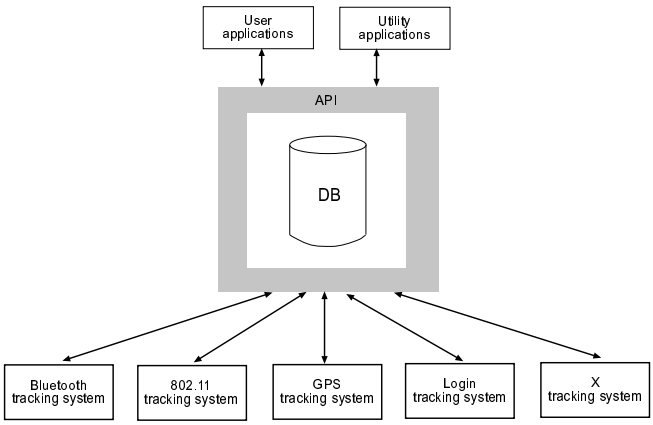

Figure 3.1: A hierarchical location tracking system sourcing location information from GPS, 802.11 wireless LAN, Bluetooth and Terminal Login. The generality of the system makes it easier to add another X tracking system. An API is used by both applications to query for location information, and sensor-dependent tracking systems to feed raw sensor data or estimated location information.

- Signal attenuation mathematical modelling, discussed in Section 2.2.2, is not very effective for distance estimation in closed environments of offices and homes. This is because the attenuation patterns undergo interference from reflection, refraction and multipath effects caused due to the closed nature of the surrounding environment.
- Proximity is a suitable technique for distance estimation using radio-based sensors. By determining the strongest or closest Bluetooth or 802.11 wireless LAN base station, it is possible to infer the space in which the mobile device is present. For example, two Bluetooth devices can communicate if they are separated by a distance of 10m. Similarly a mobile is able to communicate to a 802.11 wireless LAN base station if it is within a distance of 100m. However, to achieve higher levels of accuracy, systems like RADAR have successfully employed Static Scene Analysis, discussed in Section 2.2.2, to indoor location tracking.

3.2.1 Static Scene Analysis

Static scene analysis involves the examination of certain features in an environment containing the location system. The details of this procedure are dependent on the sensor technology in use. For Radio Frequency (RF) based technologies, a static analysis involves three steps:

Measurement of *offline static* data: Signal strength of a radio wave is measured at various locations in the area of interest e.g. floor in a building. This data is referred to as *offline* because it is collected when the tracking system is not operational, and *static* because the signal strength at a particular location remains constant, ignoring any random interference.

Measurement of *run-time* or *real-time* information: With an operational system, the signal strength of an active radio connection is measured. This data is termed *run-time* or *real-time* because it is collected during live operation of the system.

Comparison between the offline and run-time information: The comparison between the offline and run-time information based purely on signal strength information. Various levels of interference cause run-time signal strength to be distorted from the offline samples, introducing errors in this raw matching process.

Static and run-time data may be collected from multiple access points or base stations to make more distance estimates available and these may processed further to obtain a more accurate location estimate.

3.2.2 Filtering

Signal strength information collected during the offline and run-time phase of scene analysis is subject to interference effects. The following is a discussion into the physical properties related to such interference phenomena:

- Radio waves are electromagnetic waves, and hence their speed is determined by the density of the medium in which they travel. The temperature of air affects its volume, and hence density. This causes radio waves to undergo refraction when moving between regions of varying temperature. The change of speed involved in refraction, changes the path of the wave and hence its attenuation pattern. This leads to varying signal strength readings for a single radio connection at the same location.
- Furthermore, similar refraction behaviour is observed when a radio wave travels through a person walking down a corridor. Water has a different density compared to air, and the large quantities of water contained in human bodies is also responsible for refraction.
- In addition to these, the presence of any obstruction e.g. moving an office desk, also has significant properties on radio attenuation.

Thus it is important to protect both the offline and run-time signal strength information. Chapter 5 describes the data collection process undertaken to build the offline static information. In brief, signal strength samples were collected in four directions for each physical location, and the averaged value was noted as the signal strength for that location.

In the case of run-time information, one signal strength sample is collected in a fixed interval of time. To make effective use of every new signal strength information, a *moving window fixed sample low pass filter* has been explored. The filter works by averaging a fixed number of raw samples, before moving by a certain window of samples and calculating another average. Figure 3.2 shows an example filter that averages over five samples and moves by a one sample window.

3.2.3 Sensor-Independent Location Estimation Algorithms

The comparison of filtered offline and run-time information in the third phase of scene analysis produces a location estimate per access point or base station. Hence, four estimates would be proposed for an infrastructure with four access points. Common approaches in systems like RADAR, average the estimates and produce one location result. We have proposed secondary algorithms algorithms, detailed in Chapter 4, to further process these estimates with the intention of improving accuracy over the simple averaging algorithm.

The algorithms are termed *sensor-independent* because in our approach to scene analysis described in Chapter 5, the comparison between run-time and offline signal strength data

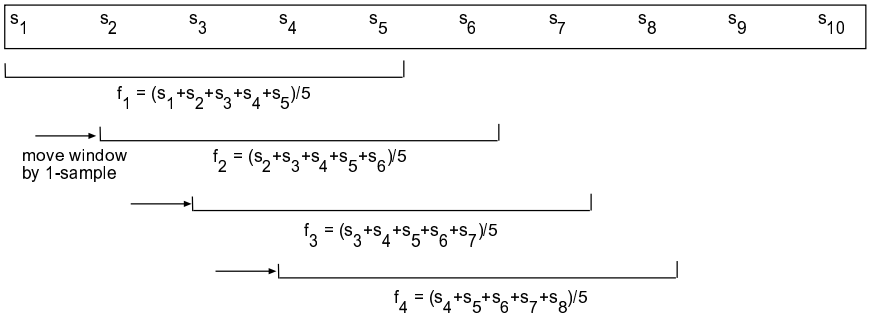

Figure 3.2: A moving average 1-window 5-sample filter. The filter averages 5 raw samples to form 1 filtered result. After moving by a window of 1 sample, the next 5 samples are averaged to form the next filtered result.

produces estimates represented in a grid system. Thus all the processing undertaken by the algorithms is relative to the grid and hence independent of the sensor technology.

3.2.4 Summary

The process of computing user location from raw signal strength data is divided into three stages, as illustrated in Figure 3.3:

1. Filtering is carried out on the raw signal strength to reduce random interference effects.
2. The filtered run-time data is compared against the filtered static signal strength information to obtain a set of location estimates for each access point.
3. The estimates are then processed using sensor-independent location estimation algorithms to produce location information with improved accuracy.

3.3 Tracking Architectures

A location system uses sensors to provide technology-specific information which may be used estimate location. This section describes the various challenges encountered in collecting such sensor information. In the case of radio-based technologies, the discussion focusses on the measurement of signal strength, while login information data collection describes general architectural issues.

3.3.1 Bluetooth Tracking

The design of the run-time data collection architecture is influenced by certain characteristics of the Bluetooth specification. These include:

Bluetooth connection setup time Table 3.1 shows the minimum, average and maximum Bluetooth connection setup times for a typical Bluetooth implementation. The inquiry and paging procedures to make a Bluetooth connection may take a maximum of 12s. The same procedures take a maximum time of 3s on the Toshiba Bluetooth implementations

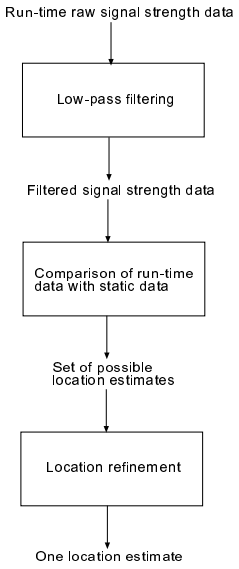

Figure 3.3: Three phase processing of raw signal strength data to obtain one location estimate

Procedure	Minimum Time	Average Time	Maximum Time
Inquiry	0.00125s	3-5s	10.24-30.72s
Paging	0.0025s	1.28s	2.56s
Total	0.00375s	4.28-5.28s	12.8-33.28s

Table 3.1: Theoretical Bluetooth connection set up times. For any typical implementation, a maximum delay of approximate 13 seconds may be experienced, while the minimum may fall to as little as 3.75ms.

used in this project. The disconnection of a Bluetooth link on our hardware takes a maximum of 0.5s.

The total connection time involved in setting up connections to multiple access points follow from Equations 3.1 and 3.2.

$$TotalConnectionTime = 3s \times Numberof AccessPoints \tag{3.1}$$

$$TotalDisconnectionTime = 0.5s \times Numberof AccessPoints \tag{3.2}$$

Nature of a Bluetooth piconet A Bluetooth piconet is formed between two or more Bluetooth devices. The device initiating a connection is termed the master while the device receiving a connection request is termed the slave. A device participating in a Bluetooth piconet is allowed to have several outgoing connections, but only one incoming connection. Furthermore, the order in which the connections are set up determines subsequent connections. For example, a slave with an existing incoming connection is not able to initiate an outgoing connection i.e. a slave cannot become a master. However, a device with an existing outgoing connection is able to accept an incoming connection request i.e. a master in one piconet is able to participate as a slave in another piconet.

Appendix A details the investigation into to these two issues. Considering that run-time data collection involves the formation and disconnection of Bluetooth connections, the delay associated with these procedures can be significant in an environment with multiple access points. For example, in an infrastructure consisting of three Bluetooth access points, the total connection and disconnection time between all access points and a mobile device could amount to a maximum of 9s and 1.5s respectively. If connections were repeatedly set up and destroyed between all the access points and a mobile user for the purposes of tracking, the frequency of signal strength updates would be reduced to one distance estimate per 10.5s. In such an amount of time, a fair distance can be covered by a user walking at a reasonable pace, and it is possible that he or she walks through a small region, such as one defined by a Bluetooth setup, and be detected only a few number of times. To avoid such a scenario it is necessary to observe a high rate of signal strength information updates, for example 1 update per second.

Taking these factors into account, two architectures are proposed to provide link quality updates at frequent intervals of time.

Architecture 1 with a Dormant Client device

As shown in Figure 3.4, this architecture features a dormant client and active access points.The access points are responsible for:

1. Searching for client Bluetooth devices in the coverage area and building a client list.
2. Initiating a Bluetooth connection with each device in the client list.
3. Measuring the link quality of the active connection.
4. Updating the database with the most recent link quality via a Local Area Network (LAN) connection.

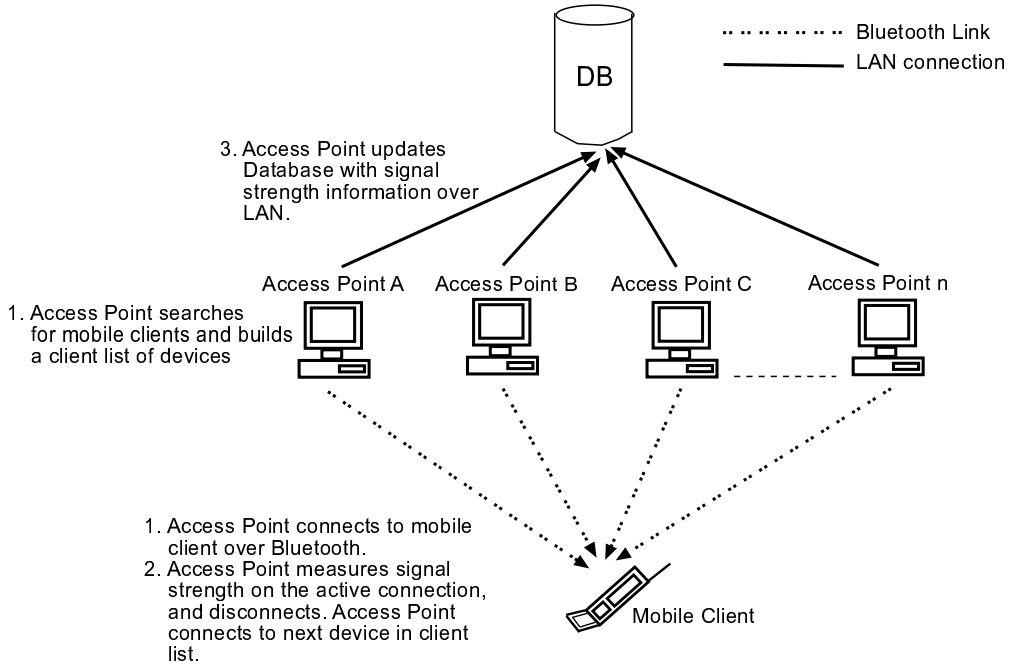

Figure 3.4: Architecture 1 with a Dormant Client device. This allows the client to be inactive, with all the business logic programmed into the servers at the access points. The access points are responsible for searching for client devices, forming a connection with the client, measuring the signal strength on the connection and updating the database via a LAN link. This forces all the access points to have LAN connections.

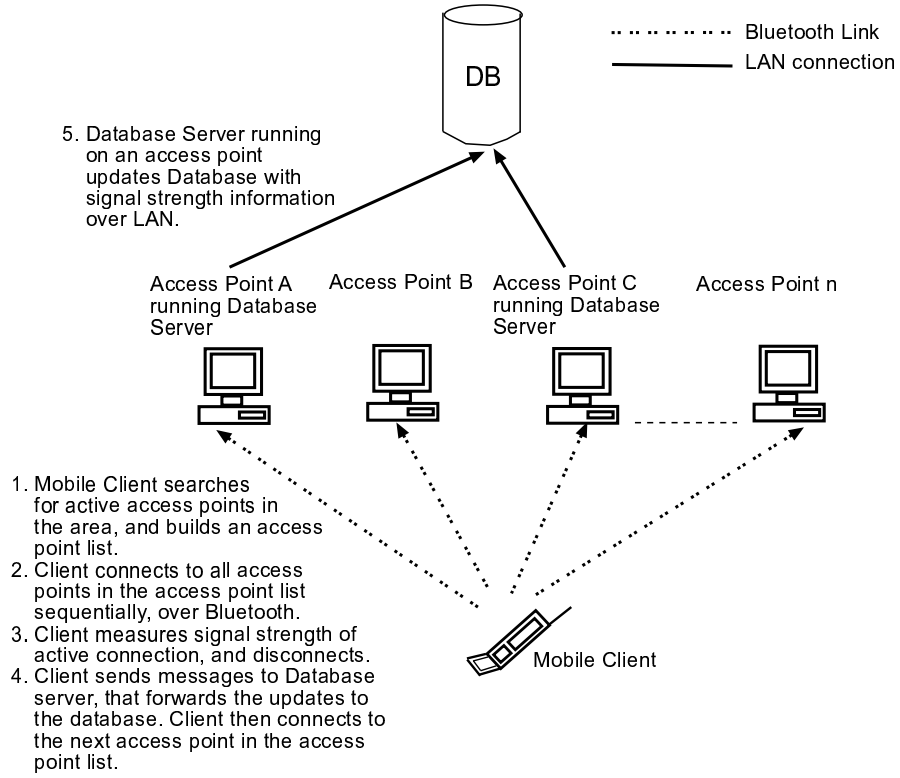

Figure 3.5: Architecture 2. In this system, the client is responsible for searching for access points, initiating connections with the access points in range, measuring signal strength on the formed connection, and updating the database via messages sent through a server running on one of the access points.

Limitations: At stage 1 above, there is competition between the access points to connect to the same client. This gives rise to the possibility of *starvation* i.e. an access point never obtains a connection with the client and hence never registers a signal strength update in the database. This in turn adversely affects distance estimation and produces inaccurate location results. However, it is possible that the presence of many devices in an area enables the access points to fall into a connecting rhythm such that an access point competes less often with its neighbour. The possibility of such a connection pattern is determined by the number of clients and access points in the area. Secondly, the architecture does not address the problem of long connection set up times leading to low frequency of distance estimates.

Advantages: The system is able to handle a wide variety of Bluetooth-enabled devices including mobile phones, laptops, PDAs, headsets etc. Furthermore, there is no processing involved on the client side allowing it to be resource limited device in terms of battery life, memory capacity and processing power. Finally, the dormancy of the client enables changes in system design without affecting the client. This enables the system to be scaled to a larger area.

Architecture 2 with an Active Client device

Figure 3.5 shows an alternative approach, where the signal strength data collection responsibility is assigned to the client. The client undertakes the following tasks:

1. Searching for all the access points in the area and building a access points list.
2. Connecting to all the devices in the access points list.
3. Measuring link quality of the connection.
4. Sending the link quality measurement in a message to a server running on an established access point. The message is structured as shown in Equation 3.3.

$$Message = AccessPointID + LinkQuality + LocalClientTime \tag{3.3}$$

Limitations: The client is now required to be a resource rich device conducting a fair amount of processing. This in turn reduces the variety of devices that the system can support. Secondly, the server that updates the database must be set up to run on various access points so that the client maintains constant connectivity to the database, even in the case when it moves out of range of one access point. Finally, competition amongst clients for a connection with the access points leads to the the possibility of starvation of a client device.

Advantages: Similar to the Cricket approach, this architecture fits well in the model to protect user privacy. A client device is never tracked without user consent, unless the user configures the client software to participate in the tracking process. More importantly, *if the presence of only one client is assumed*, it is possible for the client to maintain simultaneous connections to all the access points in range and provide signal strength updates for each access point at a very high rate of at least 1 sample per second. In this case, the long connection times of the Bluetooth technology may be avoided. Furthermore, the issue of starvation does not arise because the presence of one client is assumed.

3.3.2 802.11 Wireless LAN Tracking

The design of the 802.11 wireless LAN tracking system is made simpler by the immediate connection of a client to the nearest base station. Thus it is possible to enable run-time signal strength data collection as soon as a user is detected in an area covered by a base station. Furthermore, since a base station is not restricted to one client, it is possible to measure signal strength data for a number of clients simultaneously. Based on these observations, two run-time signal strength data collection architectures are proposed.

Querying the Base Station

Figure 3.6 illustrates an architecture where a daemon queries remote base stations for signal strength information associated with all its mobile clients connections. The information returned from the queries is then propagated to the database by the daemon. The base stations need to be configured to allow such queries.

Limitations: User privacy is compromised as any user connecting to the wireless LAN infrastructure is subject to tracking. Furthermore, the user will not be aware of the back end signal strength data processing being undertaken infer his or her location.

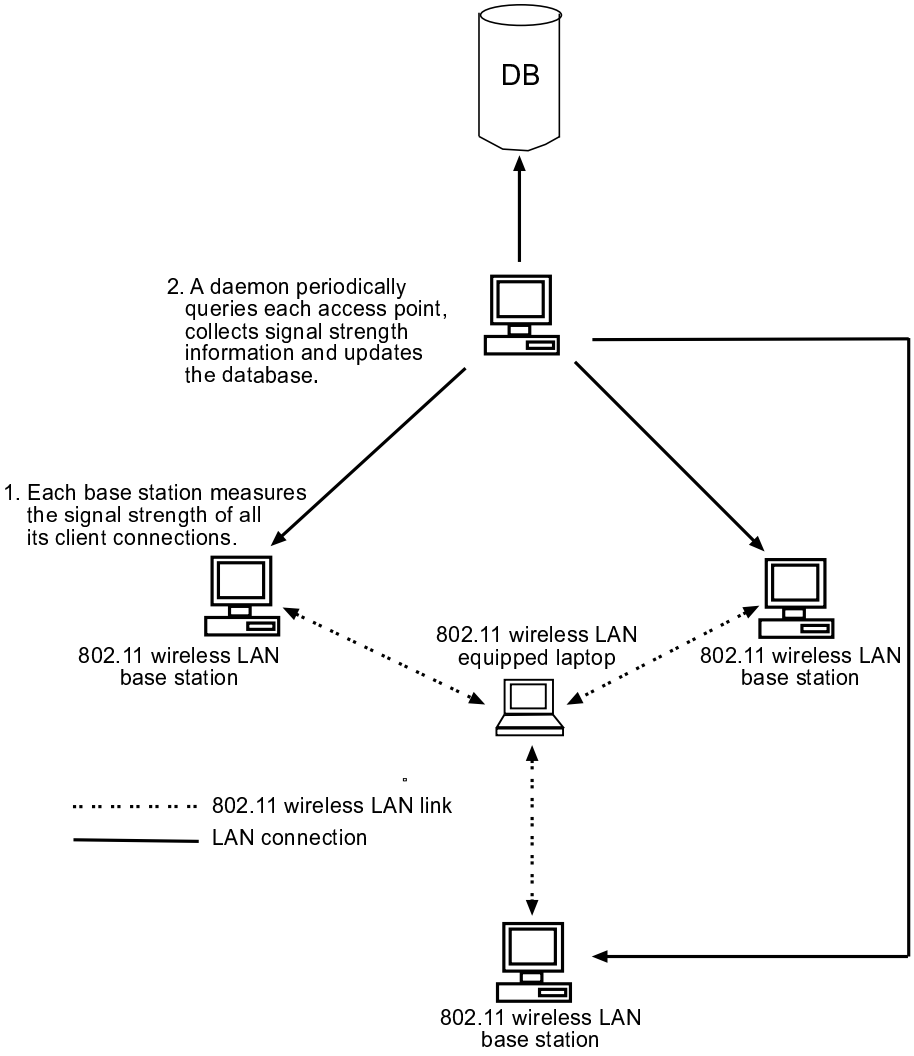

Figure 3.6: Querying the Base Station. In this system, a daemon queries remote base stations for signal strength information associated with all its mobile clients connections. The information returned from the queries is then propagated to the database by the daemon. The client remains inactive and conducts no processing activities.

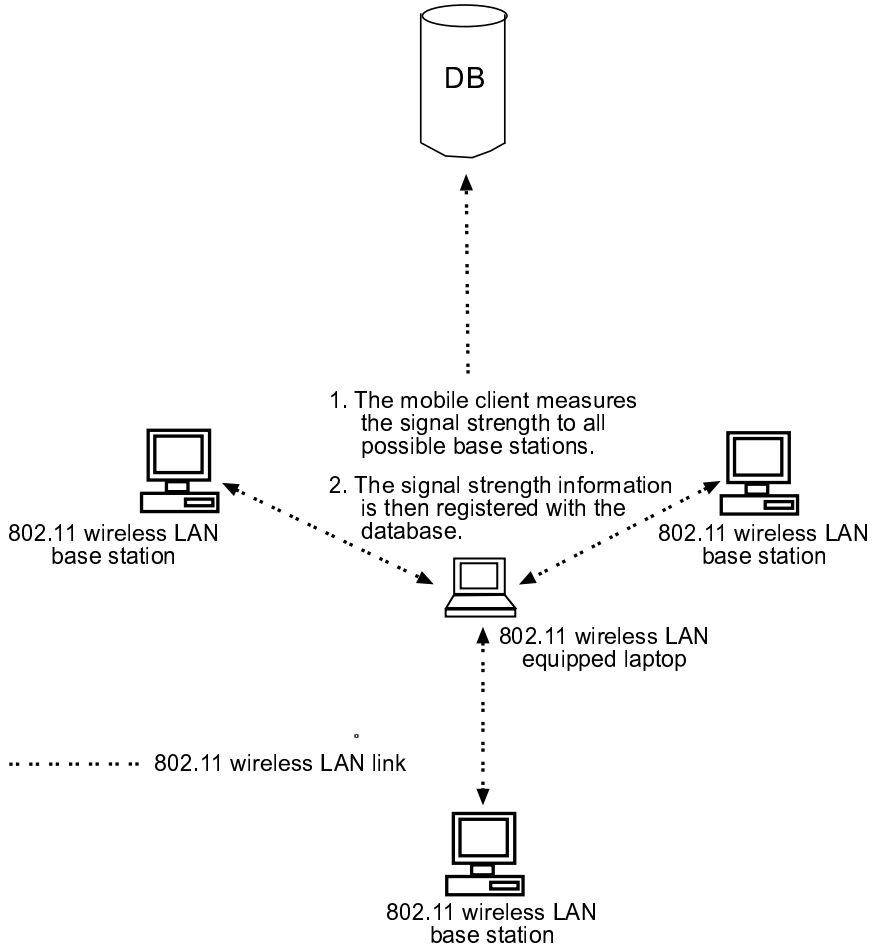

Figure 3.7: Querying the Client. In this system, the signal monitoring logic is programmed into the client. The client is responsible for measuring signal strength for all the base stations in the area, and periodically updating the database with signal strength information.

Advantages: The architecture maybe scaled to cover a wider geographical location by configuring the daemon to query a larger number of base stations. Furthermore, no processing is involved at the client end, allowing the clients to be resource constrained devices. Finally, any device with an 802.11 wireless LAN interface is subject to tracking, and this increases the set of devices that can be served by the tracking system.

Querying the Client

Figure 3.7 illustrates an architecture where the client monitors signal strength to all base stations in the area, and periodically updates the database with signal strength information.

Limitations: Since signal monitoring logic is programmed into the client, the device is expected to contain a minimum processing capacity. This may exclude certain devices with an 802.11 wireless LAN interface to participate in the tracking system. Furthermore, mobile devices that meet the processing requirements have significant strains placed on their battery because the technology consumes a lot of power.

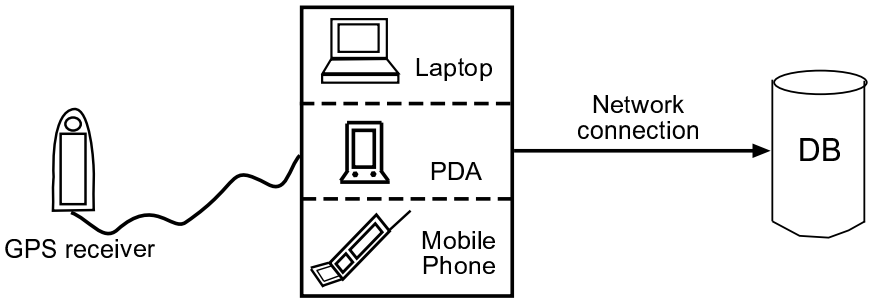

Figure 3.8: GPS tracking. The diagram shows a GPS receiver connecting as a peripheral device to a PC, PDA or mobile phone. Standard hardware interfaces such as COM and USB ports allows such connections. The receiver may be periodically queried for GPS readings. In the presence of a network connection, the host PC, PDA or mobile phone device may then propagate the readings to the system database.

Advantages: On the other hand, like Cricket, the architecture helps in assuring user privacy because signal strength monitoring and updates to the database only take place when required by a user application. The architecture still enjoys high scalability as the client may enter any area with wireless LAN coverage, monitor signal strength and send updates to the database via the nearest base station.

3.3.3 Global Positioning System

The Global Positioning System (GPS) consists of a receiver that employs time-of-flight analysis of radio signals transmitted by orbital satellites to compute location information represented on the longitude-latitude coordinate system. Some receivers offer standard hardware interfaces, for example COM and USB, to enable data transfer from the receiver to a PC. This facilitates collection of GPS location information which may be registered with a database in the presence of a network connection.

Figure 3.8 illustrates a possible architecture. The PC may be programmed with a daemon that periodically queries the GPS receiver for location information and propagates updates to a database. If there does not exist a network connection, location information may be buffered on local disk, and the database updated with the buffer information when network connection is restored. For example, if Jo's laptop is equipped with a GPS receiver and daemon, her location may be monitored while she travels from home to work. When her laptop network connection is restored, the daemon propagates buffered information to the system database. The GPS location information may be used to build mobility patterns that assist Jo's travel the following day on her way to work.

Limitations: The architecture assumes that the GPS receiver connects as a peripheral to a device rich in resources like disk capacity for location history, standard hardware ports for receiver connection and processing power and battery life for extended running of the daemon. Furthermore, common GPS receivers are bulky devices and having them as peripheral devices may not be user-friendly.

Advantages: User privacy may be provided by allowing user configuration of the daemon. For example, Jo could configure the daemon to stop collecting GPS readings while visiting a friend over a weekend, or instruct it to purge location history before it is forwarded to the

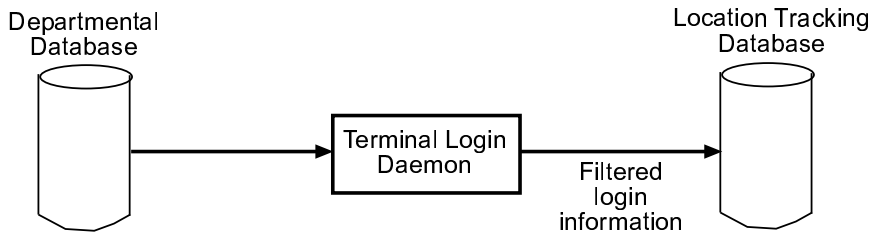

Figure 3.9: Filtering Login information. A daemon is responsible for periodically collecting login information from the departmental database, filtering it to match the requirements of our tracking system and updating the appropriate tables.

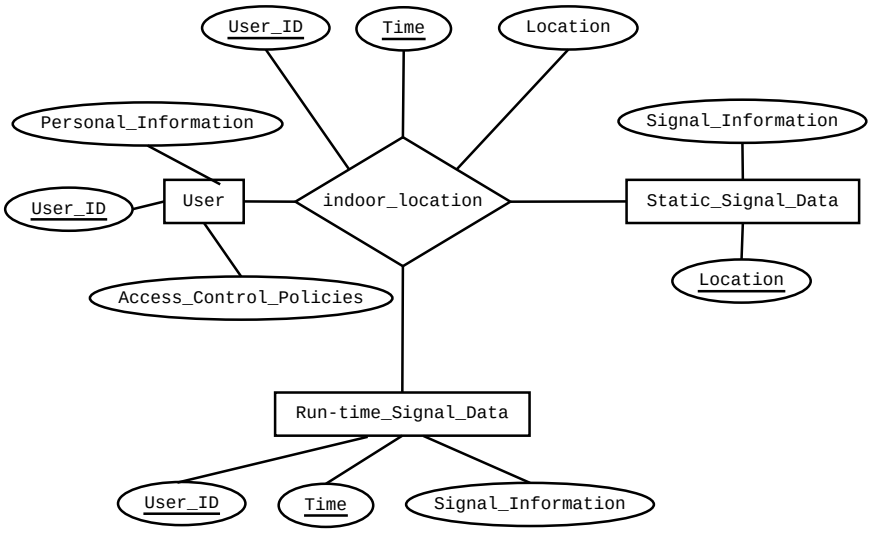

Figure 3.10: An ER diagram for holding signal and location information for radio-based sensors.

system database.

3.3.4 Terminal Login

This project has access to a departmental database that stores login information on users logged onto the system. A daemon runs periodically to filter out irrelevant information and updates our database with more relevant data. Figure 3.9 illustrates the process.

3.4 Location Model

This section discusses the data model used to house sensor and location information generated by each technology. The models are specified using the Entity-Relationship (ER) approach.

3.4.1 Radio-Based Sensors

The ER diagram in Figure 3.10 specifies the data model for the radio-based technologies i.e. Bluetooth and 802.11 wireless LAN. The User entity models the user information including a user identifier, personal information such as home address and system specific information such as access control policies to provide user privacy. The user identifier, User_ID, is the primary key of the entity because it is unique throughout the system.

The Static_Signal_Data entity models the offline static data collected during the first phase of Static Scene Analysis, described in Section 3.2.1. The entity holds the measured signal information and the location where the measurement was carried out. Location is the primary key because only one set of signal data is measured for each location. Hence it is not possible to have two sets different signal data for the same location.

The Run-time_Signal_Data entity models the signal data collected at run-time. Hence the time attribute signifies the time at which the signal data was collected. Since the system is made up of more than one user, the model also features the User_ID attribute to identify the owner of the signal data. The combined set of User_ID and Time form the primary key. User_ID is included in the primary key because it is possible that the system registers signal data for two different users at the same time, and hence Time alone cannot identify a unique signal data sample. The Time attribute is included in the primary key because signal data is collected from the user device on a continuous basis and hence is required to uniquely identify a data sample.

In both the Static_Signal_Data and Run-time_Signal_Data entities, the instances of signal information and location information are specific to the radio technology being modelled. Furthermore, the Run-time_Signal_Data entity is existent dependent on the User entity because it is not possible to generate run-time information if there is no user registered with the system.

3.4.2 GPS

The ER diagram in Figure 3.11 specifies the data model for tracking using GPS. The design of the User and Run-time_GPS_Coordinates entity follow exactly from the model above for radio-based sensors. In the case for Run-time_GPS_Coordinates entity, latitude-longitude coordinates are stored rather than signal strength as in Run-time_Signal_Data entity of radio-based tracking.

The Static_GPS_Map entity holds a translation between GPS coordinates and physical locations that are described using symbolic names. The Symbolic_Location attribute is the primary key because each geographical location may only have one set of GPS coordinates. Finally, the Run-time_GPS_Coordinates entity is existent dependent on the User entity because it is not possible to generate run-time GPS information if there is no user registered with the system.

3.4.3 Terminal Login

The ER diagram in Figure 3.12 specifies the data model for monitoring user terminal information. The design of the User entity follows exactly from the model above for radio-based sensors. The Static_Terminal_Map entity is responsible for holding the locations of terminals. Since a terminal may be present at only one physical location, Terminal_Name is the primary key for this entity.

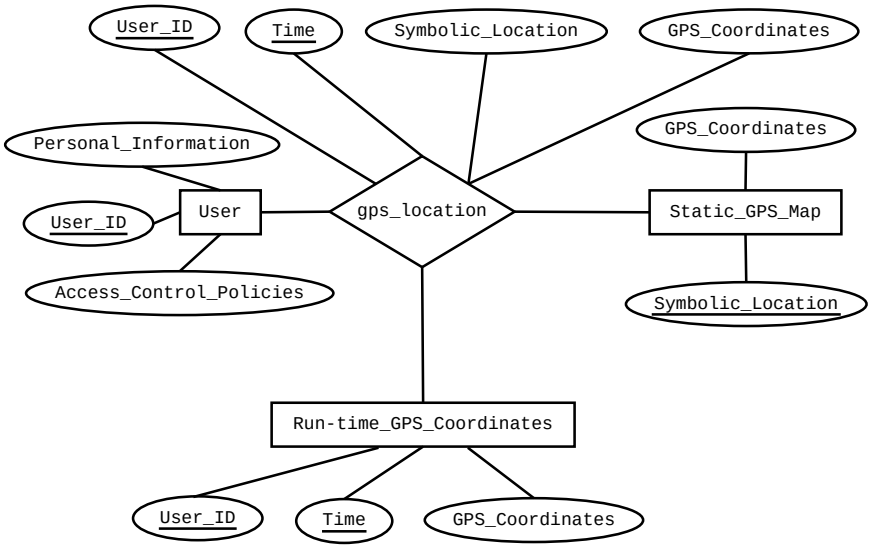

Figure 3.11: An ER diagram for holding longitude and latitude coordinates for tracking using GPS.

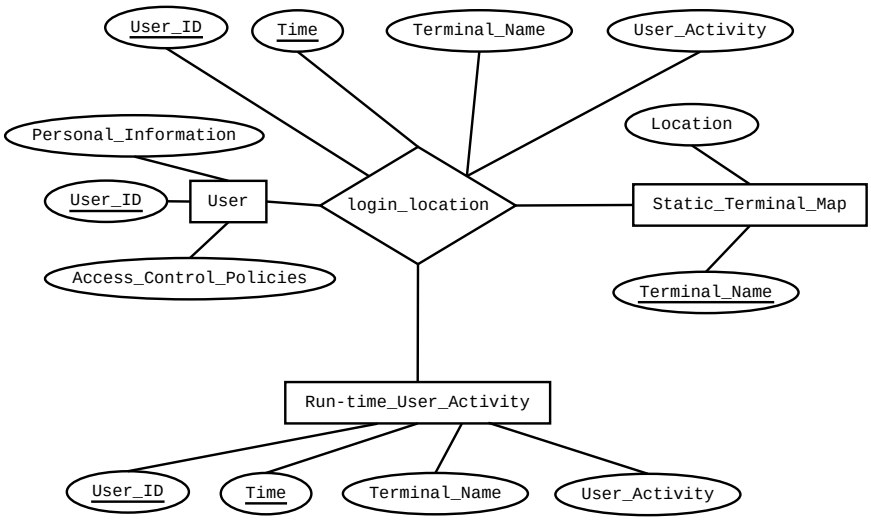

Figure 3.12: An ER diagram for holding longitude and latitude coordinates for tracking using GPS.

The Run-time_User_Activity entity holds real-time user terminal information. The combined set of User_ID and Time forms the primary key. User_ID is included in the primary key because it is possible that the system registers login information for two different users at the same time, and hence Time alone cannot identify a piece of Login information. The Time attribute is included in the primary key because login information is collected from the user device on a continuous basis and hence is required to uniquely identify information at a particular time.

Chapter 4

Sensor-Independent Location Estimation Algorithms

- An adaptation to static scene analysis for indoor location tracking is presented.
- Sensor-independent location algorithms are described.
- Location data fusion is introduced with a simple averaging algorithm.
- A basic fusion algorithm is presented.

4.1 Static Scene Analysis

Section 3.2 explained the principles of Static Scene Analysis used in indoor location tracking using radio-based sensors. The first step of the analysis involves the measurements of signal strength at various locations in the area of interest to build the offline static signal strength tables. The second step involves the collection of signal strength information in real time when the system is operational. Finally, the real-time or run-time sample data is compared against the static tables to offer location estimates.

To achieve better accuracy, signal strength information may be collected from multiple access points or base stations. Hence the static scene analysis may produce one estimate per access point. The algorithms detailed in this chapter process individual estimates in a bid to achieve improved accuracy.

Chapter 5 details our experimentation technique. In brief, an imaginary grid is placed on a map of the Bluetooth and Wavelan infrastructure, and offline and run-time samples are collected at various locations defined in terms of grid coordinates. Consequently, the estimates from static analysis are also expressed as grid coordinates. The algorithms are termed sensor-independent because they solely operate on the grid coordinates and do not use any technology-specific information.

4.2 An Adapted Static Scene Analysis

Section 3.2.2 discussed filtering techniques that are used in this project to reduce the impact of interference on location estimates computed from signal strength information. To further

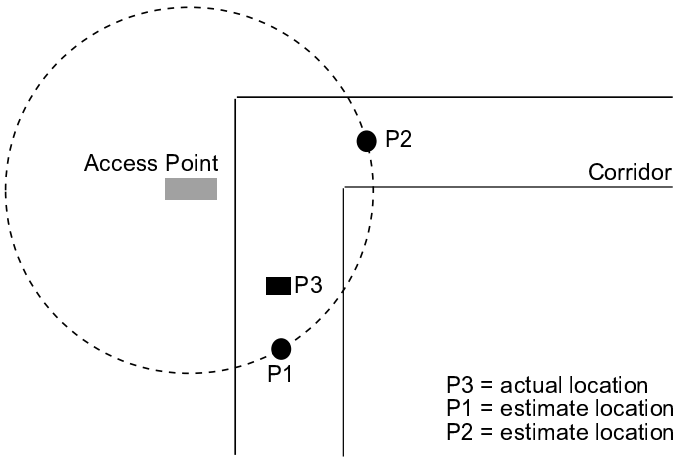

Figure 4.1: The diagram shows that it is possible different physical locations that equidistant from an access point to measure equal signal strength. Hence a bracketing search of the offline static tables can result in two different physical locations matching the same run-time signal strength data.

guard against such random interference, an adaptation to the third step of static analysis is proposed.

Normally, the third step involves a one-to-one comparison between run-time and offline data. Based on experimentation and statistical analysis described in Chapter 5, the comparison is modified to include run-time signal strength samples that are within a fixed *bracket of samples* off the measured run-time value. For example, if a measured run-time sample was -62dBm, the first comparison will involve searching the static tables for a location where a signal strength of -62dBm was measured. If a match does not exist, the run-time sample is increased and decreased by one signal strength unit, and the search repeated with the modified run-time sample of -61dBm and -63dBm. The range of units through which the run-time sample is modified is dependent on the bracket value. In the example, if the bracket value is 5dBm, the last search would allow the run-time samples having values -67dBm and -57dBm.

However it should be noted that if a match is found, the search is discontinued before the bracket limit is reached as location estimates are available for further processing. In the above example, if the search returns locations that measured -64dBm for that access point or base station, then no further searches are carried out with other possible signal strength values.

On the other hand, if for a particular access point, the comparison process does not return any locations for the signal strength samples within the bracketing range, then the client is assumed to be out of the coverage area provided by the access point, and does not participate in the location estimation process.

Furthermore, it is possible that one run-time signal strength sample may generate more than one *candidate estimate* per access point. This is because two different physical locations in the offline static tables may match the run-time signal strength. For example, in Figure 4.1, P1 and P2 are equidistant from the access point, and hence could potentially measure the same signal strength. Therefore, if signal strength were measured at these two locations during the offline data collection phase, they will be proposed as estimates for the same run-time signal

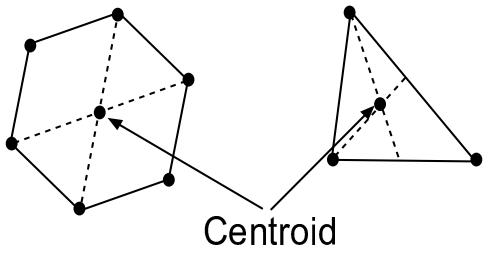

Figure 4.2: The diagrams show how individual location estimates from access points are used to find one average location. The algorithm finds the centroid or centre of mass of the polygon. In the first case, we imagine that 5 access points produce 5 estimates, hence the centre of the pentagon is the more accurate location. Similarly, in the second case, three access points are imagined, and they produce three estimates. The more accurate location estimate is the centre of triangle.

strength.

4.3 Location Estimation Algorithms

This section discusses the location estimation algorithms that process the output from static analysis. The output of static analysis is expressed in terms of the grid system. All algorithms have been implemented and evaluated against a set of run-time data samples. The grid system and run-time experimentation are detailed in Chapter 5.

4.3.1 Centroid

Centroid is an averaging algorithm that finds the geometric centre of a polygon formed from candidates contributed by all the access points. Each access point contributes at most one location candidate to the formation of the polygon. This is done to allow a fair contribution by all access points and helps maintain a regular polygon.

If multiple candidates are generated by bracketing for an access point, then only one is randomly selected and contributed to the polygon, while the other candidates are discarded. Since each candidate holds the same offline signal strength it is not possible to favour one candidate over the other. A random selection enables this choice to be made and forms a benchmark for comparison with other algorithms.

When only one access point is in range, a single estimate will be available. Hence it is not possible to construct a polygon. In this case, the estimate from the algorithm will be the same as the estimate from the single access point.

Figure 4.2 shows two potential ways in which centroid could operate. This technique has been used by RADAR in approximating an improved location from the estimates generated by all the access points.

A Weighting Heuristic

A radio signal is known to attenuate by a factor proportional to $1/r^2$, where r is the distance between the transmitter and receiver. Since signal strength is a representative of the intensity

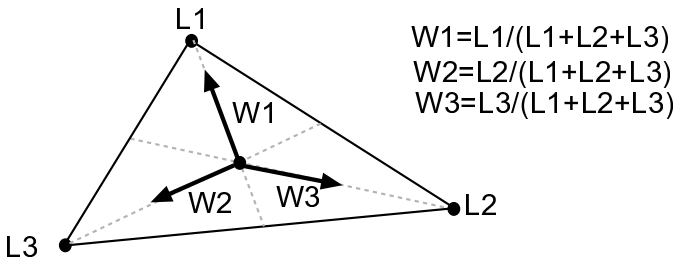

Figure 4.3: A weighting adaptation to centroid. The geometric centre of the polygon is shifted towards a candidate with stronger signal strength. This is achieved by assigning a weight to each candidate. The weight is equal to the ratio of the signal strength of the candidate to the total signal strength of all candidates.

of a radio signal, a weak signal strength measurement indicates the weak intensity of a radio signal. The $1/r^2$ attenuation pattern suggests small changes in intensity and hence signal strength at large distances from the receiver. In some cases, weak signal strength may be taken to be an indication of large distances, and hence does not serve well as a distance indicator.

Taking this fact into account, a weighting heuristic may be added to the otherwise purely geometric centroid calculation. The heuristic aims to pull the geometric centroid towards the candidate with higher signal strength, as shown in Figure 4.3. This is achieved by assigning a weight to each candidate. The weight is equal to the ratio of the signal strength of the candidate to the total signal strength of all candidates. The adaptation breaks the sensor independence norm by making use of technology-specific radio information.

4.3.2 Smallest Polygon

For each access point, the centroid algorithm randomly selects a candidate and contributes it to polygon formation, while the other candidates are discarded. It is possible that a candidate in the discarded set could be a better estimate than the one selected. For example, in figure 4.1, P1 is a better candidate than P2, but it is possible that the random nature of centroid may select P2 over P1. Smallest Polygon attempts to eliminate this possibility.

The algorithm follows from the theory that in an interference-free environment, run-time and offline link quality would match exactly and hence all the candidate locations from an access point would equal the actual location. When slight interference is allowed, the candidates would be very close to the actual location, and hence very close to each other. Thus the algorithm attempts to find the smallest polygon that can be formed from the set of candidates proposed by each access point. Perimeter of the polygon is used to compare the size of different polygons. Finally, the centroid of the smallest polygon is then calculated to offer an improved estimate.

When more than two access points contribute candidates, the algorithm finds the shortest corresponding polygon. For example, assume a case where three access points return the following distance estimates, as shown in Figure 4.4:

$$AP_a = a_1, a_2, a_3$$
$$AP_b = b_1$$

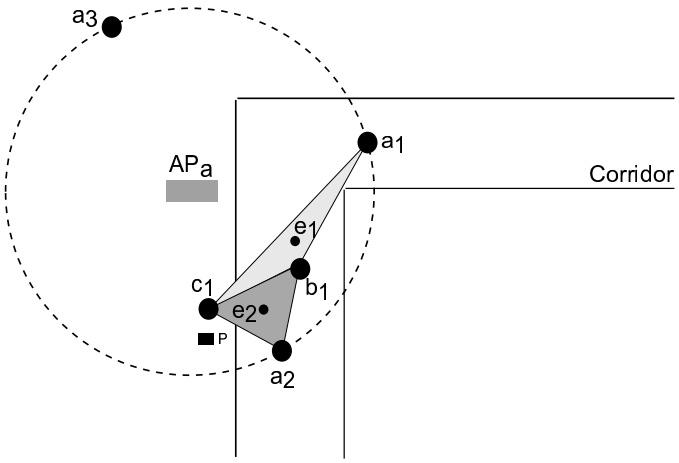

Figure 4.4: The diagram shows the formation of triangles (a_1,b_1,c_1) and (a_2,b_1,c_1). It is obvious that (a_2,b_1,c_1) is shorter than (a_1,b_1,c_1), and the centroid of this triangle, e_2 is much closer to the actual position, P, than e_1. Hence e_2 is a more accurate estimate than e_1.

$$AP_c = c_1$$

The algorithm finds the polygon, in this case a triangle, with the shortest perimeter. To achieve this, triangles are formed between all the points in the set. For the case above the three triangles formed are: (a_1,b_1,c_1), (a_2,b_1,c_1), (a_3,b_1,c_1). The centroid of the smallest triangle is calculated to offer a more accurate location.

For the case when exactly two access point propose candidate sets, Smallest Polygon attempts to find the candidates that may form shortest line, and the result proposed will be the mid-point of the line. For the case of one access point coverage, a small adaptation to centroid is adopted: a polygon is formed between all the candidate locations of that one access point and proposed as the smallest polygon.

4.3.3 Convex Hull

Smallest Polygon ignores candidate locations and corresponding centroids that do not belong to the smallest possible polygon. Convex Hull attempts to consider a larger data set by taking into account the centroid of all polygons formed during Smallest Polygon operation. The centroid from all the polygons is used to build a *secondary polygon structure*. The centroid of the secondary structure is presented as the estimate from the algorithm.

For example, in Figure 4.5, a secondary structure is constructed from e_1, e_2 and e_3 which are the centroids of triangles (a_1,b_1,c_1), (a_2,b_2,c_2) and (a_3,b_3,c_3) respectively. For the purposes of clarity other triangles e.g. (a_1,b_1,c_2) are not shown.

4.3.4 Polygon Intersection

The previous algorithms have concentrated efforts on individual points. Polygon Intersection investigates the processing of regions formed between candidate locations, to produce an

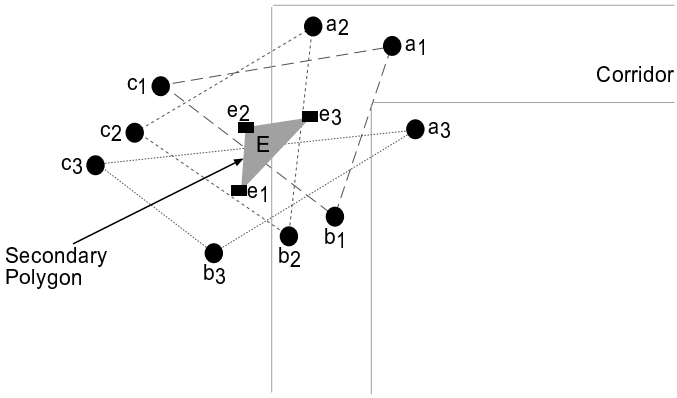

Figure 4.5: The Convex Hull algorithm.

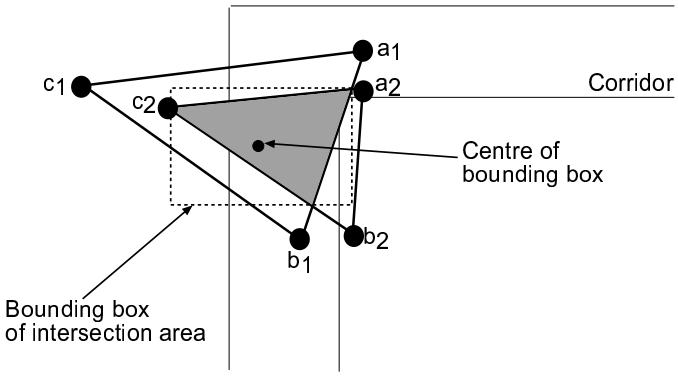

Figure 4.6: The Polygon Intersection algorithm. Triangles are formed between the candidates $\{a_1, a_2\}$, $\{b_1, b_2\}$ and $\{c_1, c_2\}$ of access points A, B and C respectively. The intersection between the triangles (a_1,b_1,c_1) and (a_2,b_2,c_2) is then calculated. The estimate proposed by the algorithm is the centroid of the bounding box outlining the final intersection area. Two polygons are shown for clarity purposes.

improved location estimate. As in Smallest Polygon, candidate locations are used to form polygons. The polygons are ordered in descending order of polygon length i.e. longest polygon first, and starting from the longest polygon, the area of intersection of polygons is calculated. The ordering is intended to achieve the maximum number of intersections between all the polygons. Since the final intersection result may be heavily skewed and could potentially introduce errors in the estimate, the bounding box of the final intersection area is calculated, and the centre of the box is proposed as the estimate.

For example, in Figure 4.6, triangles are formed between the candidates $\{_1, a_2\}$, $\{b_1, b_2\}$ and $\{c_1, c_2\}$ of access points A, B and C respectively. The intersection between the triangles (a_1,b_1,c_1) and (a_2,b_2,c_2) is then calculated. The estimate proposed by the algorithm is the centroid of the bounding box outlining the final intersection area.

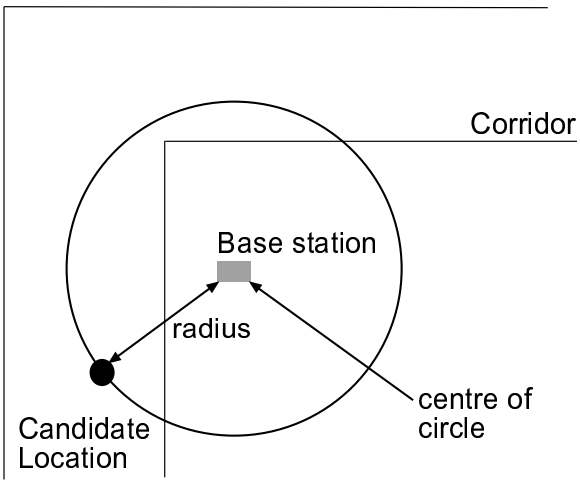

Figure 4.7: Formation of a circle for triangulation. A circle is formed using the location of the access point as the centre of circle and the distance between a candidate location and access point as the radius.

4.3.5 Triangulation

Triangulation is a classical location estimation technique that has been popularly used in various location systems including the widely accepted GPS. It is a generalization to two methods: lateration and angulation. Lateration computes the location of a user by measuring its distance from multiple reference points. Angulation determines the location of an object based on angles, rather than distances. A detailed account of the technique is presented in Section 2.2.2.

We have implemented the lateration method of triangulation. As shown in Figure 4.7, a circle is formed using the location of the access point as the centre of circle and the distance between a candidate location and access point as the radius of the circle. Since the access point and candidate locations are represented in a grid system, the radius value is calculated as the Euclidean distance between the two gird coordinates. Each access point contributes the largest circle that may be formed from all its candidate locations. This is done to achieve a maximum number of intersections between circles contributed by other access points.

Figure 4.8 shows two possible scenarios involving circle intersections. In part (a), the two circles intersect at a_1 and a_2, and the estimate returned by the algorithm is the average of the two intersections. In part (b), there are six intersections between the three circles. The intersections are grouped into sets depending on the intersection to which they belong. For example, the three sets formed are $\{a_1, a_2\}$, $\{b_1, b_2\}$ and $\{c_1, c_2\}$. The Smallest Polygon algorithm then attempts to find the smallest polygon that may be formed by locations from each set. In this case, the smallest triangle is formed by a_1, b_1 and c_1. Finally, the centroid calculates the geometric centre of the polygon.

A Rings Heuristic

The Triangulation algorithm attempts to find the maximum number of circle intersections by selecting the largest circle that may be formed from all the candidate locations for each access

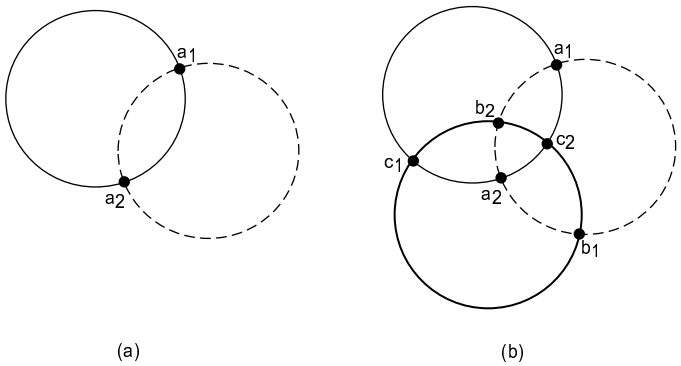

Figure 4.8: Circle intersection scenarios for triangulation. In part (a), the two circles intersect at a_1 and a_2, and the estimate returned by the algorithm is the average of the two intersections. In part (b), there are six intersections between the three circles. Three sets formed are $\{a_1, a_2\}$, $\{b_1, b_2\}$ and $\{c_1, c_2\}$. The Smallest Polygon algorithm then attempts to find the smallest polygon that may be formed by locations from each set. In this case, the smallest triangle is formed by a_1, b_1 and c_1, and the estimate proposed by the algorithm is the centroid of the smallest triangle.

point. Since the largest circle is formed using the furthest candidate location, the resulting intersections and centroid of the intersections will be further away from the actual location. This has been confirmed by experimentation analysis.

A Rings heuristic is proposed that attempts to avoid this scenario by considering both the smallest and largest circles, thereby accounting for the closest and furthest candidate locations. Intersections are then calculated between all the circles.

For example, figure 4.9 shows the intersection of minimum and maximum circles formed for two access points. In that case, there may be a maximum of eight intersections. Much like Triangulation, the intersections are grouped into sets according to the rings which are used to generate them. Smallest Polygon then finds the smallest possible polygon that may be formed using a point from each set. Centroid follows by averaging the polygon coordinates to provide one location estimate. For the case where only two rings intersect, the heuristic allows the formation of a polygon between the intersection points belonging to the same set, and the centroid of this polygon is the proposed estimate of the algorithm. In the example above, the centroid of the polygon formed by all the intersection points ($a_1..a_8$) is the result of triangulation.

4.4 Location Data Fusion

Data Fusion is a technique that uses location information from a variety of sources to infer a more accurate location estimate, that would be unavailable if the sources were individually. The technique may work well for sensors with overlapping coverage areas. This section describes a basic averaging algorithm aimed at improving location accuracy.

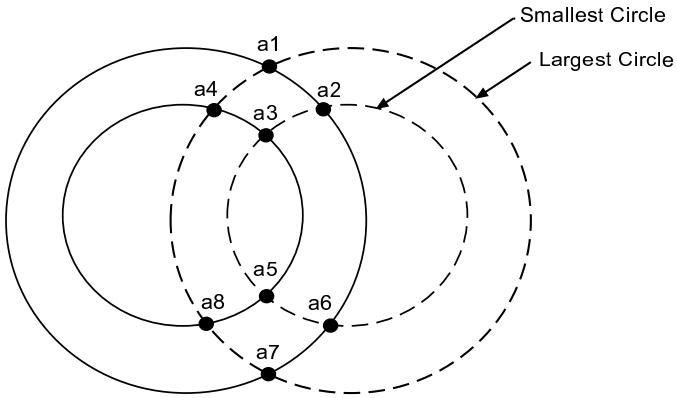

Figure 4.9: Triangulation with Rings. The figure shows that a maximum of eight intersections are possible from the intersections of two rings. In this case, the centroid of the polygon formed by (a_1..a_8) is the result of triangulation.

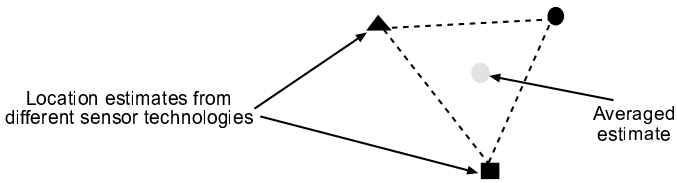

Figure 4.10: A basic algorithm for location data fusion. Location estimates from a variety of algorithms are used to form a polygon whose centroid is proposed as an estimate.

4.4.1 A Basic Averaging Algorithm

The algorithm processes data generated by the location estimation algorithms detailed in Section 4.3, and hence is sensor-independent. It works by averaging estimates produced by the location estimation algorithms processing data from different sensor technologies. As Figure 4.10 shows, estimates from each technology are used to build a polygon and the geometric centre of that polygon is proposed as the estimate.

Adding a weighting heuristic is a challenging task because the calculation of the weights will require a formal method for integrating the various technology-specific information. If a new technology becomes available, the method will need to be re-defined to achieve the optimum weighting criterion.

Chapter 5

Empirical Measurements and Performance Evaluation

5.1 Chapter Summary

- Our empirical testbed is presented.
- Offline and run-time data collection procedures are outlined to perform Static Scene Analysis using Bluetooth and 802.11 wireless LAN.
- A low pass filtering technique is evaluated and optimum performance parameters determined.
- The algorithms described in Chapter 4 are evaluated using experiments with Bluetooth.
- The effect of filtering on location accuracy is explored.
- Adapted Static Scene Analysis is investigated to reduce distance error variation.
- The basic data fusion algorithm is compared against the location accuracy provided individually by Bluetooth and 802.11 wireless LAN.

5.2 Experimental Testbed

Our experimental testbed is located on the fifth floor of a six-storey building. The experimental area measures 32.2m by 25.7m, and is shown in Figure 5.1.

5.2.1 Bluetooth Infrastructure

The Bluetooth infrastructure features three stationary *access points*, and one *mobile client*. The access points consist of three laptops having a Pentium 233 MHz processor and 64MB of RAM. The mobile client is laptop equipped with a Pentium 500MHz and 128MB of RAM. Each of the machines is equipped with a Toshiba PCMCIA Bluetooth card, shown in Figure 5.2. The Bluetooth hardware is configured by default to operate at full power and provide an maximum range of 30m in open environments. However, the effective range is reduced in the presence of walls and other obstructions present in the closed environment of the floor

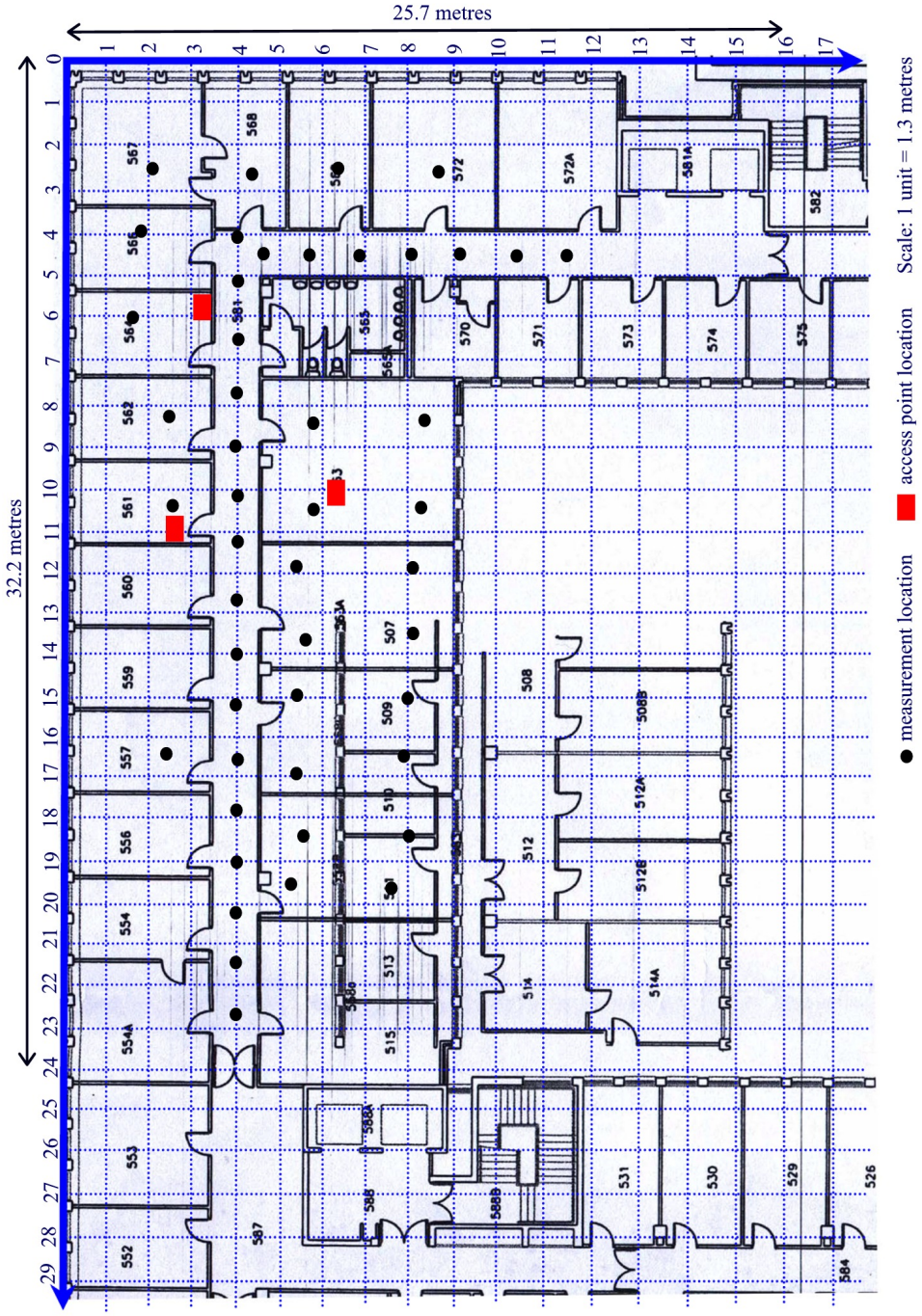

Figure 5.1: Level 5, Huxley Building, Dept. of Computing, Imperial College. The diagram shows the floor plan of the Bluetooth experimental setup. The diagram also features an imaginary grid placed on top of the plan, and the positions where offline link quality data was collected.

Figure 5.2: The Toshiba Bluetooth card used in this project.

plan. The access points are positioned to provide overlapping coverage and form mutual Bluetooth connections to aid run-time or real time data collection.

5.2.2 802.11 wireless LAN infrastructure

The 802.11 wireless LAN consists of three Orinoco base stations mounted at the locations indicated in Figure 5.1. The wireless LAN setup is an extension of the departmental network and the base stations are located to provide coverage to all the users. The range of the network is 100m, but like Bluetooth, the effective range is reduced due to the presence of walls and other obstructions.

The wireless LAN client consists a Compaq Evo laptop equipped with a Lucent Silver 802.11 wireless LAN card and the Orinoco Client Manager software package.

5.2.3 An Abstraction Methodology

Static scene analysis has been adopted as the location estimation technique to convert raw sensor data to location information. A detailed description and an explanation of the adaptations are presented in Section 3.2 and 4.2 respectively. In brief, the analysis requires the measurement of offline signal information at various locations in the area of interest to build static tables that translate signal information to locations. At run-time, raw signal data is compared against the static tables to estimate locations.

The RADAR project described in Section 2.4.3 has successfully experimented with this approach for indoor location tracking. The RADAR experience suggests that the uniform distribution of the physical locations, where signal strength information is collected for building the offline static tables, is more important than the actual number of locations. To capitalize on this observation and achieve accurate uniformity, an imaginary grid is placed on a scaled map of the floor plan as shown in Figure 5.1. The coordinate system is used as a reference point to select the physical locations where offline data is collected.

The grid also serves several side-benefits: it acts as an abstract layer and enables the development of sensor-independent location estimation algorithms, discussed in Chapter 4, that process grid coordinates rather than distances in the physical environment or sensor specific information. Furthermore, the grid provides a translation mechanism for converting

between symbolic data and geometric data. For example, in Figure 5.1, Room 560 could be identified using a rectangle with coordinates {(0,11.2), (3.4,11.2), (3.4,13.2) , (0,13.2)}. This translation enables the straightforward implementation of containment and co-relation algorithms. For example, to determine if Jo is in Room 560, the system has to check if her grid coordinates are geometrically contained within the rectangle defined by Room 560.

5.2.4 Bluetooth Data Collection

Access to the Bluetooth services and low-level hardware features is achieved through a Java API implemented as Java Native Interface (JNI) wrapper [1] built on a COM API [2]. The Java API enables low level and profile-level connections to be established and broken. The Toshiba Bluetooth device used in this project is one of the first commercial releases of the technology and hence not completely error-proof. The firmware programmed on the device does not provide signal strength information. However a data value called *Link Quality* is available via the Java API. Link Quality has a range of 0 to 255; 0 represents no or weak connection, and 255 represents a strong connection. Experiments have shown that Link Quality varies both with distance and the presence of obstructions, hence is a potential substitute to signal strength.

Offline Data Collection

Figure 5.1 shows the locations in our experimental setup where offline link quality data samples were collected. 49 uniformly distributed locations were chosen on the coordinate system and translated to physical dimensions in terms of metres. At each physical location, 100 link quality samples were measured in each of four directions (north, south, east and west), for each of the three access points. The link quality stored in the offline static tables for a particular location was the average of 400 samples. Thus for each access point, the static data consists of a set of 49 locations and their associated link quality values.

Raw link quality information was collected at 30 different physical locations to simulate run-time data collection. In order to investigate the behaviour of the algorithms over different access point coverage scenarios, the locations were chosen to allow an equal distribution of varying access point coverage. Hence 10 of the 30 locations received three access point coverage, 10 received two access point coverage and the remaining 10 were covered by one access point. At each location, 1500 samples of link quality data were measured while facing the north direction. This large data set allowed the flexibility of experimenting with a high-sample filters discussed in Section 5.3.1, and observing the behaviour of the algorithms under each filter.

5.2.5 802.11 wireless LAN Data Collection

Since this project does not have the access permissions to the wireless LAN base stations, it is not possible to extract signal information at the base station. Furthermore, we do not have access to the proprietary Lucent API that enables signal extraction at the client. Chapter 6 describes the system used to extract signal information at the client. In brief, the Orinoco Client Manager software is configured to extract signal information from the driver and log the data into a shared file. The file is then parsed to collect the appropriate information.

[1] The Java API was built at Telcordia Technologies during placement work of the author.
[2] This COM API is a product of Digianswer, and is shipped with the card.

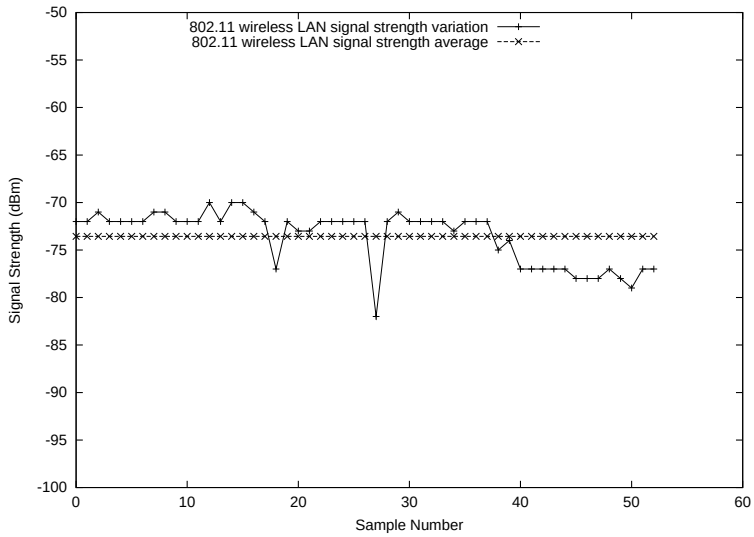

Figure 5.3: Signal strength variation of a typical 802.11 wireless LAN connection.

Offline Data Collection

Signal strength data was collected at the 49 locations marked in Figure 5.1. At each physical location, 50 signal strength samples were measured in each of four directions (north, south, east and west) for all the visible base stations. The signal strength stored in the offline static tables for a particular location was the average of 200 samples.

Online Data Collection

Signal strength data was collected at the same 30 locations as for Bluetooth. For each location, 50 raw signal strength measurements were taken while facing the north direction.

The physical locations locations where offline and run-time signal strength data is collected are the same as for Bluetooth. This was done to maintain consistency with Bluetooth measurements and enable comparison of the the location estimates from both the technologies. However, the setup of the wireless LAN base stations and the large range afforded by the technology makes all the 30 run-time locations receive three base station coverage. In Section 5.3.3, the Bluetooth data is appropriately selected to enable a fair comparison between the two technologies for location estimation.

Apart from that, in the offline case we measured 100 link quality samples for Bluetooth and 50 signal strength samples for wireless LAN because the observed variation in signal strength for wireless LAN was considerably less, as shown in Figure 5.3. Furthermore, since the data is collected using parsing, it was more practical to collect 50 offline samples for wireless LAN.

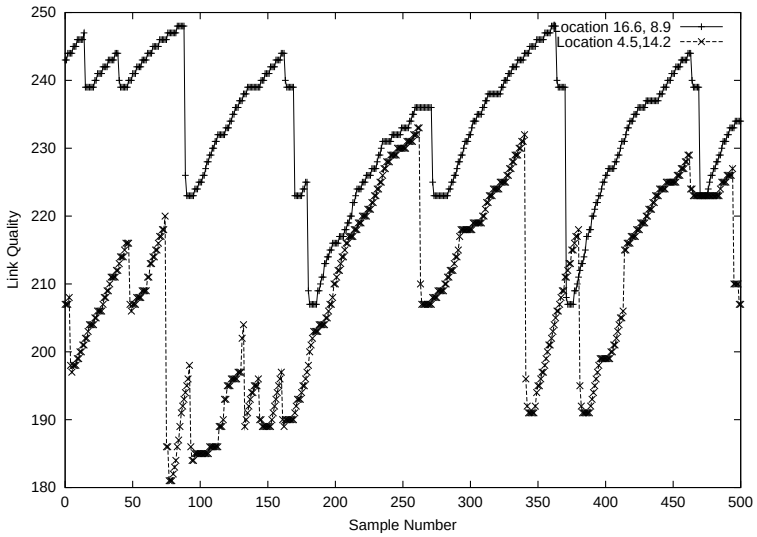

Figure 5.4: Raw link quality variation: The graph shows the link quality variation at two different locations.

5.3 Performance Evaluation

This section presents and analyzes the results of the various experiments that have been conducted on the location estimation algorithms. Initially, the Bluetooth and 802.11 wireless LAN results are described in Section 4.3. A quantitative comparison between the two technologies follows and the section ends with a discussion on the performance of data fusion algorithms.

Note: All distance error values are expressed in metres.

5.3.1 Filtering

Section 3.2.2 highlighted the need for filtering processes to guard against random interference effects, and presented a low pass digital filtering algorithm. In brief, the filter works by averaging a fixed number of raw signal data samples, before moving by a certain window of samples and calculating another average. Figure 5.4 shows link quality variation of a typical Bluetooth connection at a physical location in the floor plan. Experiments have been conducted to identify the optimum sample size that would achieve significant stabilization. In all the experiments, the moving window has been limited to one sample to correctly simulate the one sample per second data collection frequency of the run-time architecture.

Figures 5.5 and 5.6 show the effect of a 10-sample and 200-sample filter respectively, with a one sample moving window. Link quality samples processed through a 10-sample filter show the similar variation as the non-filtered raw link quality. On the other hand, a high 200-sample filter achieves a certain degree of stabilization.

Note: A high-sample low pass filtering technique may suit well to stationary devices whose signal strength or link quality variation can be attributed purely to interference. In contrast, both interference and changing user location are responsible for the variation in signal data observed for mobile devices. This infers that a high-sample filter may not the best

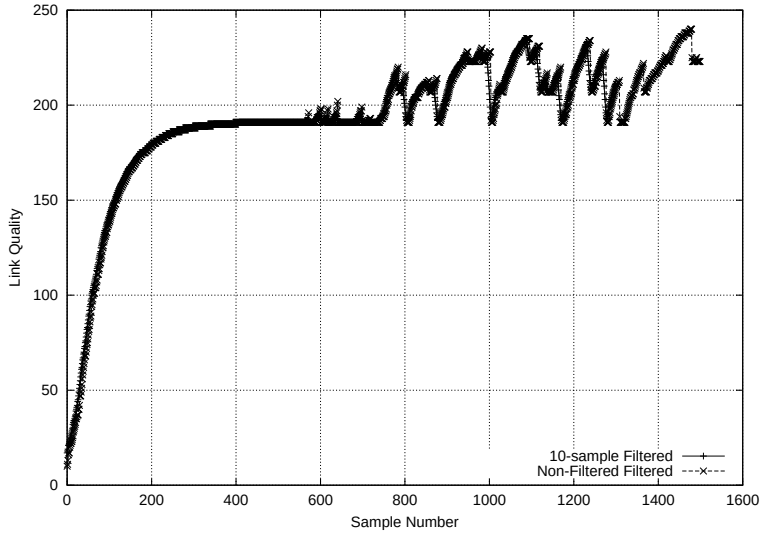

Figure 5.5: Filtering effect of a 10-sample filter. The graph shows effect on link quality variation of a 10-sample filter measured against raw non-filtered link quality.

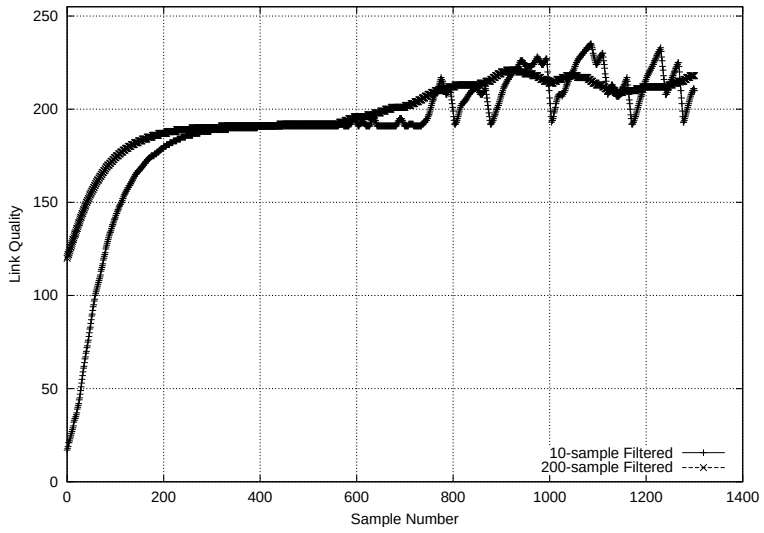

Figure 5.6: Filtering effect of a 200-sample filter. The graph shows effect on link quality variation of a 200-sample filter measured against raw non-filtered link quality.

No. of Access Points	Centroid
Three	5.34
Two	10.16
One	17.06

Table 5.1: The table summarizes the distance error involved with the Centroid algorithm.

possible choice since the filtering technique may consider signal information that is obsolete in terms of time and location. For example, in our prototype system, the Bluetooth data collection architecture registers link quality at a rate of one unit per second. A 200-sample filter would be averaging 200 samples of link quality data. Since each sample is one second old, the earliest sample being considered in the averaging would have been taken 200seconds or 3minutes 20seconds ago. A user moving at a constant pace could possibly be in a different physical location to that whose samples are being considered. This may introduce large errors in location estimation for mobile users.

5.3.2 Location Estimation using Bluetooth

Section 3.2 introduced the principles of Static Scene Analysis used in indoor location tracking using radio-based sensors. In Section 4.2, an adaptation to the conventional scene analysis approach was presented. The adaptation involved the modification of the comparison phase of static analysis to include run-time signal strength samples that are within a fixed *bracket of samples* off the measured run-time value. Finally the results from the bracketing phase are processed by the sensor-independent algorithms described in Section 4.3.

This section begins with an evaluation of the performance of the location estimation algorithms. Next, descriptions of the investigation into bracketing, and filtering to achieve location accuracy are given. Finally, a method to achieve stable distance error at run-time is discussed.

In the discussions that follow, the bracket value in the adapted static scene analysis has been set to 20 units of link quality. This value is obtained by taking three random run-time physical locations, computing the standard deviation in link quality for each and averaging the standard deviations to offer a bracket value. This allows for interference effects that may be present during offline or run-time data collection.

Note: The distance error is calculated using the Euclidean distance between the actual location and estimated location from the algorithm. Both the locations are represented as grid coordinates.

Centroid

Centroid is an averaging algorithm that finds the geometric centre of a polygon formed from candidates contributed by all the access points. Figure 5.7 shows the performance of the centroid algorithm. The distance error increases from locations with three access points to locations with one access point. The average error over each coverage area is 5.34m, 10.16m and 17.06m for three, two and one access point coverage areas respectively, and is summarized in Table 5.1. This trend is in agreement with the findings of the RADAR project where distance error was observed to increase with reduced access point coverage. Analysis of the data suggests that the increase in error associated with two point covered locations is due to

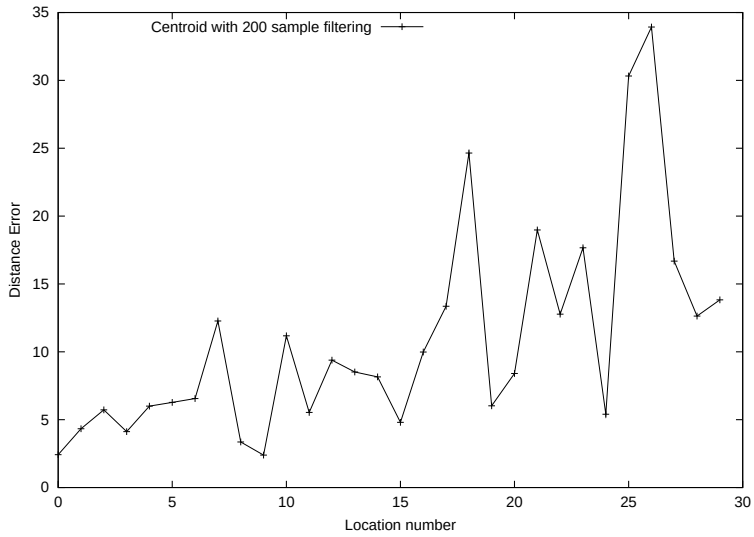

Figure 5.7: Centroid. The graph shows the distance error for each location using the Centroid algorithm. Locations 0 to 9 have three access point coverage, 10 to 19 have two access point coverage and 20 to 29 have one access point coverage.

the reduced number of candidates being used to calculate the average. In the case of locations being covered with one access point, only one candidate will be available and hence the lack of averaging causes a large error.

For each access point, Centroid randomly selects one candidate from all the proposed candidates. Experimentation with the weighting heuristic was deferred to Smallest Polygon because this random element could distort any improvements offered by the technique.

Smallest Polygon

Smallest Polygon attempts to find the shortest polygon that may be formed from all the candidates proposed by all the access points. Each access point may contribute at most one candidate to the polygon. The centroid of polygon is calculated and returned as the location estimate.

Figure 5.8 compares the effectiveness of Smallest Polygon over Centroid. Smallest Polygon does not show any major improvements, with an increase in error evident at some locations.

- Analysis of data for location 17 shows that an outlying candidate distant from the actual location, was used in the averaging computation of centroid. Hence the average was skewed away from the actual location.

- Locations labelled 20 to 28, covered by one access point, show no change suggesting that the majority of bracketing results consisted of one candidate. The minor reduction at location 29 is due to the averaging of all the candidates proposed by one single access point, a break of the norm where one access point may contribute only one candidate towards polygon formation.

Figure 5.8 shows that that weighting the centroid has no effect on the distance error. This may be expected for locations 20-29 where the candidates used in the centroid belong to the

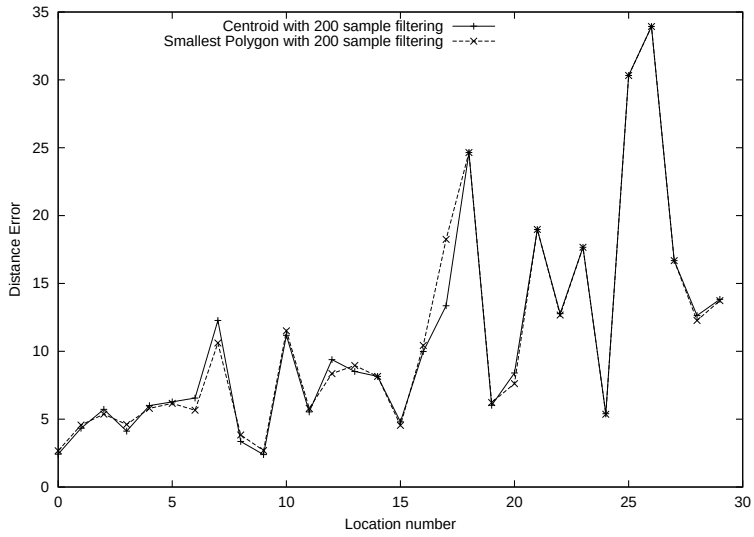

Figure 5.8: Smallest Polygon against Centroid. Locations 0 to 9 have three access point coverage, 10 to 19 have two access point coverage and 20 to 29 have one access point coverage. Minor improvements are observed at locations 5 and 6 for three access point, location 12 and 20 for two access points, and 28 for one access point coverage areas. A dominant increase in error is observed at location 17 under two access point coverage.

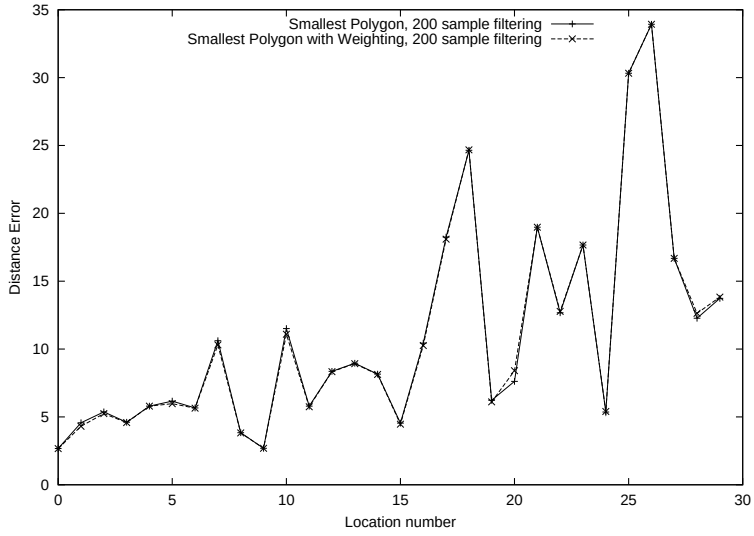

Figure 5.9: Smallest Polygon with Weighting against Smallest Polygon. The graph shows the effect of centroid weighting being used in Smallest Polygon. No improvements are observed at any of the locations.

No. of Access Points	Centroid	Smallest Polygon	Weighting
Three	5.34	5.20	5.10
Two	10.16	10.69	10.60
One	17.06	17.06	16.95

Table 5.2: The table summarizes the distance error involved with Centroid, Smallest Polygon and its Weighting heuristic.

same access point and have the same link quality, thus making no difference to the centroid position.

For locations covered by two and three access points, analysis of data reveals that the weighting ratios are nearly equal, thus causing minor changes to the distance error. The reason for this is that the distance between the Bluetooth access points is small: 5.6m, 3.6m, 5.5m. This causes the distance between a run-time experiment location and each of the access points to vary between a range of 1m to 5m. The link quality demonstrates minor variation over such distance ranges, hence each of the access points register link qualities of near-equal magnitude. This may be attributed to the fact the Bluetooth hardware is operating at full power and hence does not undergo much attenuation over such distances in the absence of obstructions.

Table 5.2 summarizes the averages error for Centroid, Smallest Polygon and its Weighting heuristic.

Convex Hull

Convex Hull considers a larger data set by taking into account the centroid of all polygons formed during the Smallest Polygon operation. The centroid from all the polygons is used to build a *secondary polygon structure*. The centroid of the secondary structure is presented as the estimate from the algorithm.

Figure 5.10 shows that Convex Hull is performs poorly when compared to Smallest Polygon. This holds true for all locations across all coverage areas except location 7 where a reduction in error is experienced. Analysis of intermediate data reveals a scenario illustrated in Figure 5.11 that explains the poor performance of convex hull: a secondary polygon (1,2,3) is constructed from the centroids of the polygons (a_1,b_1,c_1), (a_2,b_1,c_1), and (a_3,b_1,c_1). The location estimate is the centroid of the secondary polygon, e_1. The possibility of an outlying primary centroid distorts the secondary polygon, and hence drifts the average estimate away from the true location. In contrast, the Smallest Polygon algorithm is immune to this affect as it considers the centroid of the smallest polygon, e_2. Hence an outlying centroid does not affect its estimate. Table 5.3 compares the average distance error across Smallest Polygon and Convex Hull.

Polygon Intersection

Polygon Intersection operates on regions formed between candidate locations. As in Smallest Polygon, candidate locations are used to form polygons. The polygons are ordered according to length, and starting from the largest polygon, the area of intersection of polygons is calculated. This is done to achieve the maximum number of intersections between all the polygons. The bounding box of the final intersection area is calculated, and the centre of the

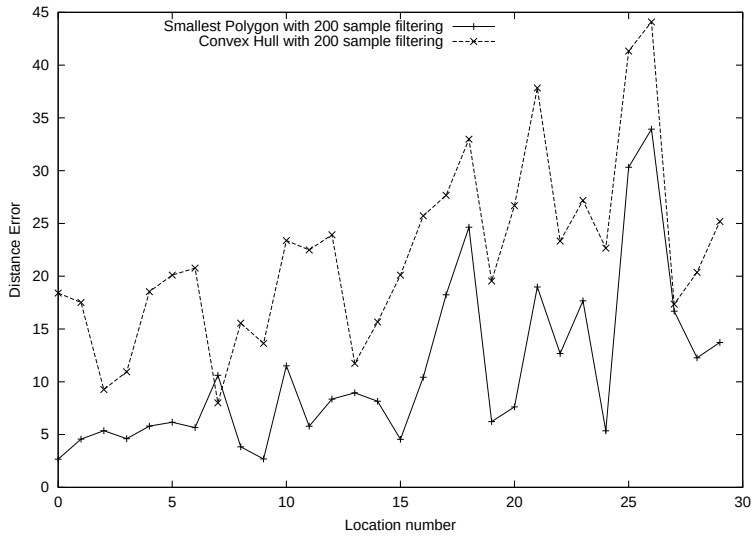

Figure 5.10: Convex Hull against Smallest Polygon. For all locations except 7, Convex Hull performs poorly.

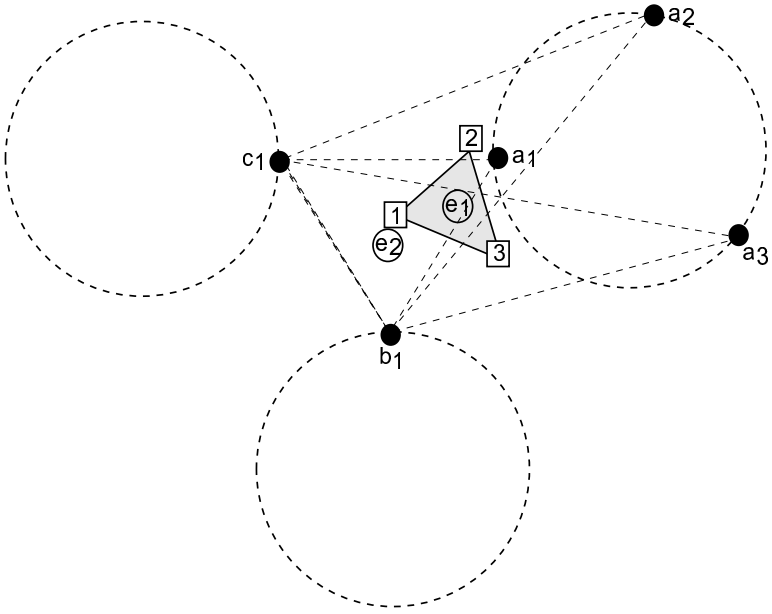

Figure 5.11: The diagram offers an explanation to the adverse effects shown by the Convex Hull algorithm. A secondary polygon (1,2,3) is constructed from the centroids of the polygons (a_1,b_1,c_1), (a_2,b_1,c_1), and (a_3,b_1,c_1). The location estimate is the centroid of the secondary polygon, e_1 and is further away from the centroid of the smallest polygon, e_2.

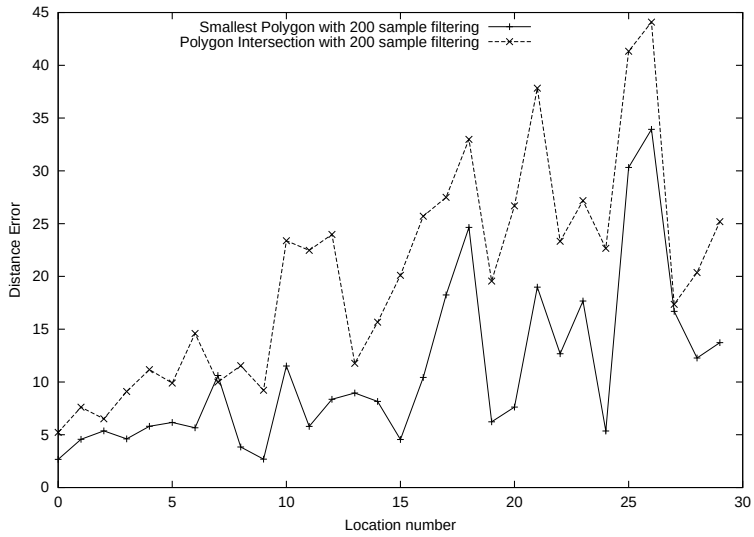

Figure 5.12: Polygon Intersection against Smallest Polygon. The graph shows that Polygon Intersection does not offer any improvement compared to Smallest Polygon irrespective of coverage area.

No. of Access Points	Smallest Polygon	Convex Hull	Polygon Intersection
Three	5.10	15.28	9.49
Two	10.58	22.32	22.31
One	16.95	28.60	28.60

Table 5.3: The table summarizes the average distance error involved with the Smallest Polygon, Convex Hull and Polygon Intersection algorithms.

box is provided as the estimate.

Figure 5.12 shows that Convex Hull performs poorly against Smallest Polygon. Analysis of intermediate data reveals that intersections from the first three polygons produces an area that holds no further intersections with the remaining polygons. Furthermore, since the initial intersections begin with large polygons where the candidates are much further apart and hence poor choices under the theory of Smallest Polygon, the final intersection is an area that is further away than the polygon formed under Smallest Polygon. In contrast, Polygon Intersection is observed to perform better than Convex Hull in locations covered by three access points, as shown in Table 5.3.

Triangulation

In Triangulation, a circle is formed using the location of the access point as the centre of circle and the distance between a candidate location and access point as the radius of the circle. Intersection is calculated between the largest circles contributed by each access point. Smallest Polygon then attempts to find the shortest polygon among the intersection points. The centroid of the polygon is proposed as the location estimate.

Figure 5.13 shows the performance of Triangulation over Smallest Polygon. For locations

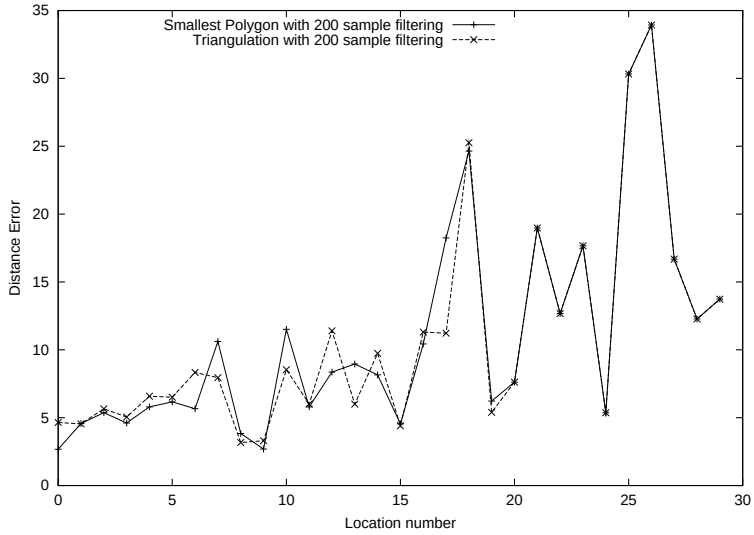

Figure 5.13: Triangulation against Smallest Polygon. Locations 0 to 9 have three access point coverage, 10 to 19 have two access point coverage and 20 to 29 have one access point coverage. Improvement is observed at locations 7 and 8 for three access point coverage locations, 10, 13, 17 and 19.

No. of Access Points	Smallest Polygon	Triangulation	Triangulation with Rings
Three	5.10	5.57	5.36
Two	10.58	9.93	10.03
One	16.95	16.92	17.06

Table 5.4: The table summarizes the average distance error involved with the Smallest Polygon, Triangulation and and Triangulation with Rings.

covered by three access point, improvement is observed at two locations, while the error increases for seven locations. Similarly, for two access point coverage locations, error reduces at four and increases at four locations. For locations covered by one access point, Triangulation will behave similar to Smallest Polygon because candidates from only one access point are available and hence the centroid of the polygon formed between these points will be proposed as the estimate.

Analysis of intermediate data reveals that the policy to use the furthest candidate to obtain the largest circle and hence the maximum number of intersections, causes the increase in error. This holds true for the cases where run-time locations are close to the access points i.e locations 0-6. The large circle forms intersections which are further away from the access point and hence further away from the actual location. Table 5.4 summarizes the average distance error between Smallest Polygon, Triangulation and its heuristic.

An Investigation into Offline and Run-time Data Comparison

The third phase of static scene analysis involves the comparison of run-time signal data with offline static tables. For each access point, the static tables contain the link quality measured

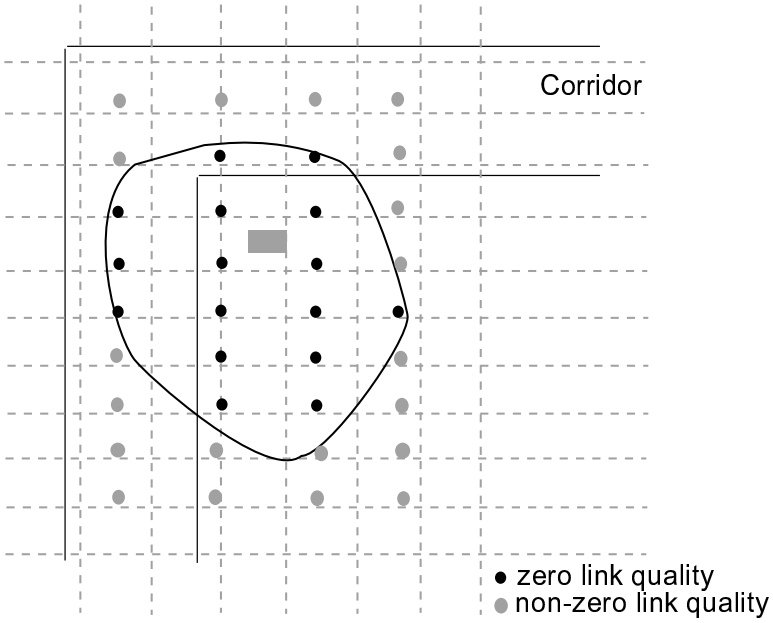

Figure 5.14: A scenario where an access point does not cover all static locations.

at each of the 49 physical locations in the floor plan in Figure 5.1. Since it is not possible for each access point to cover all the 49 locations as in Figure 5.14, some entries in the static tables contain zero link quality registered for an access point, physical location pair.

At run-time, it is possible that the same access point provides no coverage and hence a zero link quality is registered. This investigation explores the idea of matching run-time zero link quality against the zero link quality entries in the static table. Two ideas can be entertained:

Theory 1 Zero link quality should not be considered in location computations because there are many candidates outside the coverage area, which are far apart. This is against the theory of Smallest Polygon where the closest candidates are used to average an estimate. This holds true for outdoor technologies, e.g. Cell.

Theory 2 Zero like any other link quality provides information that the user is present in particular area or not. In the case of zero link quality registration by an access point, the user is confirmed to be outside the coverage of that access point and this leaves room for considering candidate information that would otherwise be ignored.

The two theories have been experimented with the Centroid, Smallest Polygon, Convex Hull and Triangulation algorithms.

- Figure 5.15 shows the two cases being implemented on the Centroid algorithm. The graphs show no clear trend with a large increase and decrease in error observed at locations 26 and 28 respectively. This re-enforces the random element of the Centroid algorithm: a candidate is randomly selected from a list of proposed candidates.

- Figures 5.16 and 5.17 compare the effect of the ignoring zero link quality on Smallest

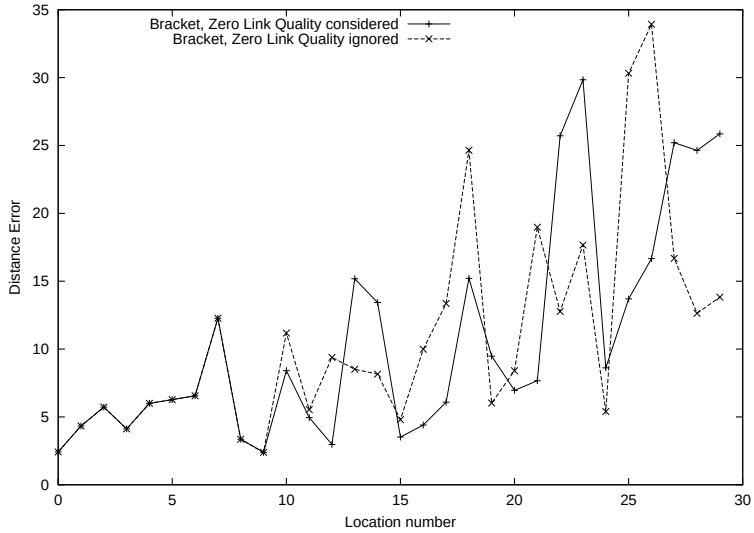

Figure 5.15: The graph shows the effect of ignoring zero link quality on Centroid. Locations 0 to 9 have three access point coverage, 10 to 19 have two access point coverage and 20 to 29 have one access point coverage. There appears to be no clear trend over all the locations. A high increase in error occurs at location 26 and a large decrease is observed at location 28. This re-enforces the random element of the Centroid algorithm: a candidate is randomly selected from a list of proposed candidates.

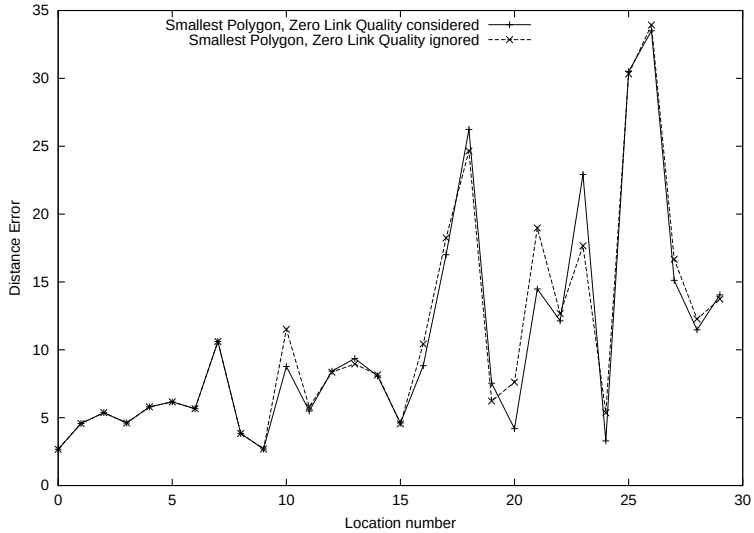

Figure 5.16: The graph shows the effect of ignoring zero link quality on Smallest Polygon. Locations 0 to 9 have three access point coverage, 10 to 19 have two access point coverage and 20 to 29 have one access point coverage. Increase in error is observed at locations 10, 20 and 21, while the only reduction in error occurs at location 23.

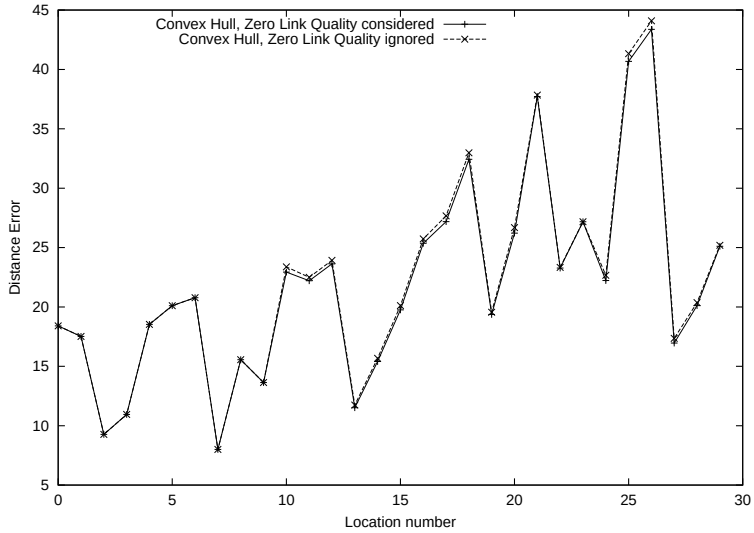

Figure 5.17: The graph shows the effect of ignoring zero link quality on Convex Hull. Locations 0 to 9 have three access point coverage, 10 to 19 have two access point coverage and 20 to 29 have one access point coverage. Small increase in error is evident at locations 10, 11, 12, 16, 17, 18, 25 and 26, while there is no observed reduction in error.

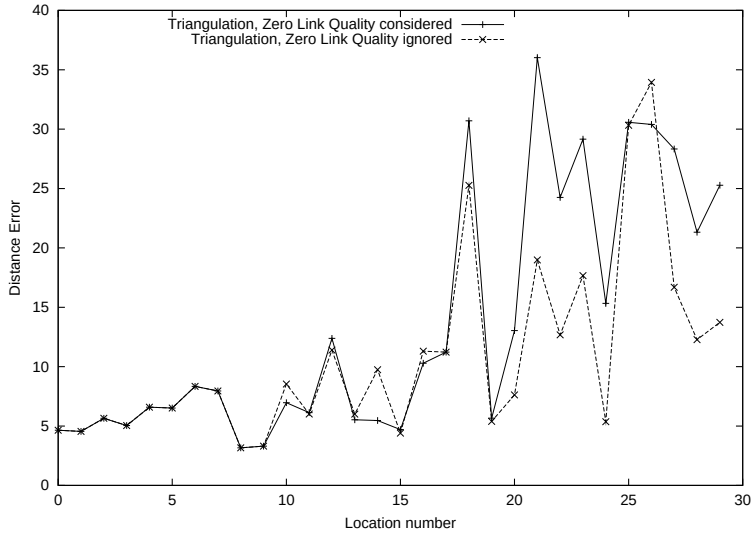

Figure 5.18: The graph shows the effect of ignoring zero link quality on Triangulation. Locations 0 to 9 have three access point coverage, 10 to 19 have two access point coverage and 20 to 29 have one access point coverage. An increase in distance error is observed at locations covered by two or three access points. Some locations covered by one access point show a reduction in error because of the Triangulation falls back to Smallest Polygon in such cases.

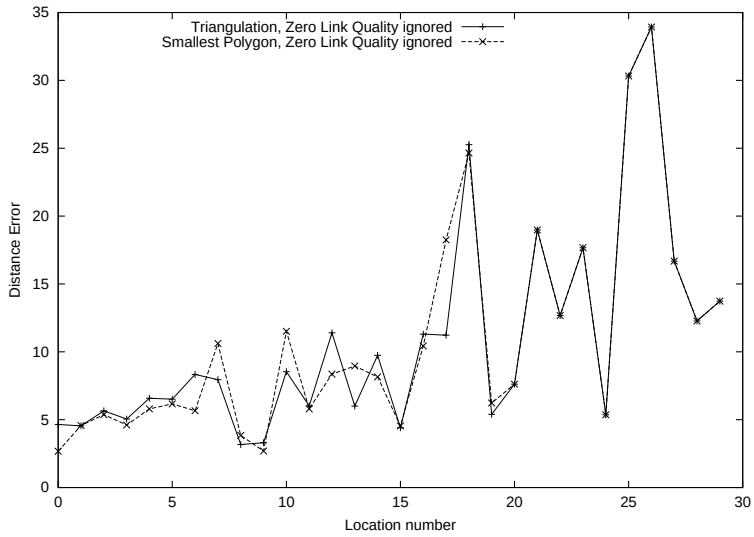

Figure 5.19: Fall back mechanism of Triangulation. The distance error of the two algorithms matches exactly for locations 20 to 29. This shows that Triangulation behaves like Smallest Polygon in one access point cases.

Polygon and Convex Hull respectively. In both the cases there distance error is observed to increase irrespective of access point coverage.

- In contrast, the results of Triangulation in Figure 5.18 show that locations covered by one access point experience a reduction in error by ignoring zero link quality. This is because is not possible to find intersections with one circle. The fall back mechanism of the algorithm forms a polygon between the candidates, and the centroid of this polygon is returned as the estimate. This may be seen in Figure 5.19, where the distance error for Triangulation and Smallest Polygon follow exactly for location 20 to 29.

The results of Smallest Polygon and Convex Hull show a strong bias towards Theory 1. Furthermore, an increase in error is observed with Triangulation for locations with two or three access point coverage. This gives evidence that Theory 1 is possibly a stronger method for indoor location tracking.

Effect of Filtering on Accuracy

Section 5.3.1 described the investigation into determining the parameter set of the *moving window fixed sample low pass filter* that would achieve the most effective stabilization in link quality. This section explores the effect of filtering on distance error using the same low pass filter. Smallest Polygon has been used for the purposes of this experiment.

Figure 5.20 shows the effect of a 100-sample low pass filter over the raw data. A reduction in distance error is observed at locations four locations, while the error increases at three locations. With the 200-sample filter in Figure 5.21, distance error is found to increase at six locations, but performs well at the remaining others. As may be expected, there appears to be no relation between access point coverage and filtering effect on distance error. The summary in Table 5.5 observes a minor improvement in average distance error for one and

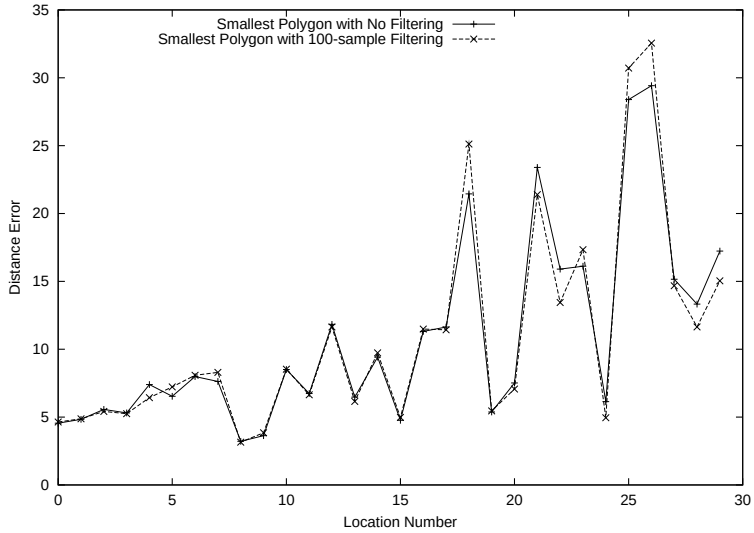

Figure 5.20: The graph shows the effect of a 100 sample low pass filter on the distance error when using Smallest Polygon to estimate location. The error decreases at locations 4, 21, 28 and 29, and increases at 18, 25 and 26.

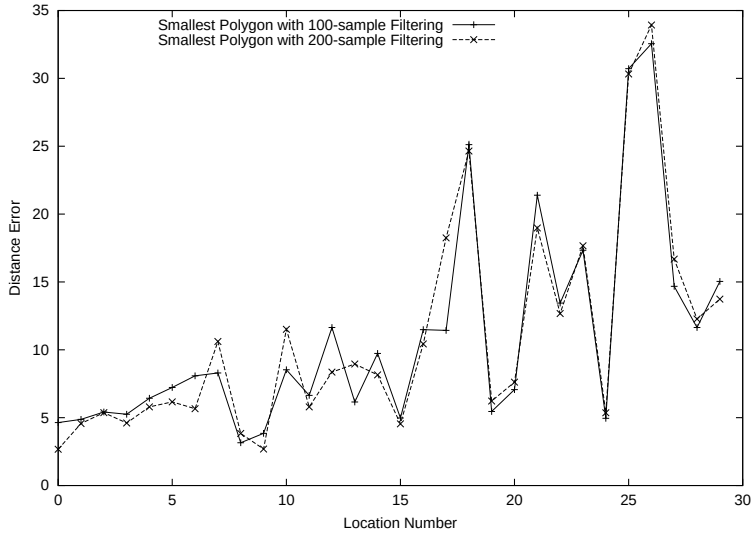

Figure 5.21: The graph compares a 100 and 200 sample low pass filter under Smallest Polygon. With he 200-sample filter significant increases in error are observed at locations 7, 10, 13, 17, 26 and 27. Apart from these, the error is seen to improve especially for locations 0 to 6.

No. of Access Points	0-sample(m)	100-sample(m)	200-sample(m)
Three	5.89	5.93	5.48
Two	9.75	10.12	10.69
One	17.26	16.88	16.92.

Table 5.5: The table summarizes the average distance error involved with the Smallest Polygon, Triangulation and and Triangulation with Rings.

three access point coverage locations. However an increase in average error is experienced for two access point coverage locations. The improvements for one and three access point coverage locations is too small and may be attributed to interference. Furthermore, the increase in error distance for two access point coverage location suggests the lack of concrete evidence to base firm conclusions. This is in agreement with the findings of RADAR where increasing the number of run-time samples did not have a strong impact on the distance error.

Reducing Error Variation

Over the course of experiments and analysis of algorithm intermediate data, three characteristics of link quality have been identified:

- In Section 5.3.2, it was noted that link quality does not vary largely over distance ranges of 1m to 5m.
- Interference causes a large fluctuations in link quality which in turn affects the initial comparison phase and hence the distance error.
- The presence of obstructions e.g. concrete walls in our infrastructure, can cause severe changes in link quality. A peculiar observation was the increase in distance error from 0.8m to 9m with a change in link quality of one unit.

These fragile characteristics of link quality make it a poor candidate for location estimation. However this is the only link information available from our current hardware. This section outlines an adaptation to the bracketing algorithm introduced in Section 4.2. The adaptation aims to reduce the large variation in distance error caused by interference and magnified by the fragile nature of link quality, as shown in Figure 5.22.

Normally, the third step of static scene analysis involves a one-to-one comparison between run-time and offline data. The bracketing algorithm allows the run-time data to vary between a fixed bracket till an offline match is found. If a match is found, bracketing discontinues and provides the coordinates corresponding to the match as an estimate. As mentioned above, in some cases the fragile nature of link quality causes a unit change to represent physically distant locations. In this cases, if all the locations that fall within the bracket range are considered, it is possible that an accurate estimate may be contained in the estimate set, and identified by the sensor-independent algorithms. On the other hand, it is possible that a run-time link quality is accurate and some locations that lie within the bracket value are far from the actual location.

The bracketing algorithm was adapted to consider, not the first match, but *all* the locations that match link quality values in the bracket range. Figure 5.23 shows the variation in distance error for a 5 link quality unit bracket. The curve for the variant algorithm with

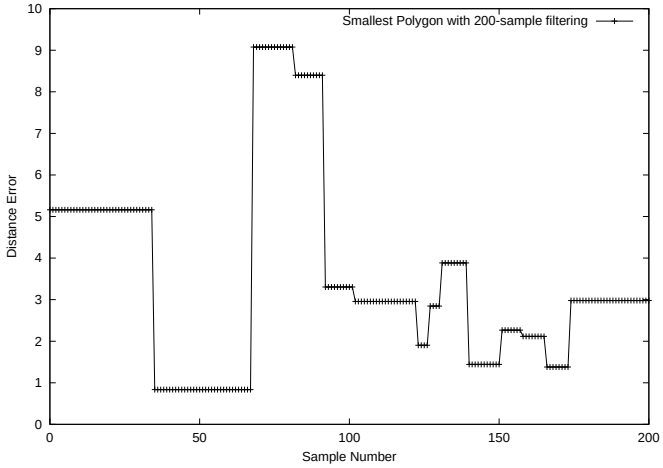

Figure 5.22: Variation in distance error under Smallest Polygon while standing stationary. The change in link quality causes the location estimate and hence the distance error to undergo large variations.

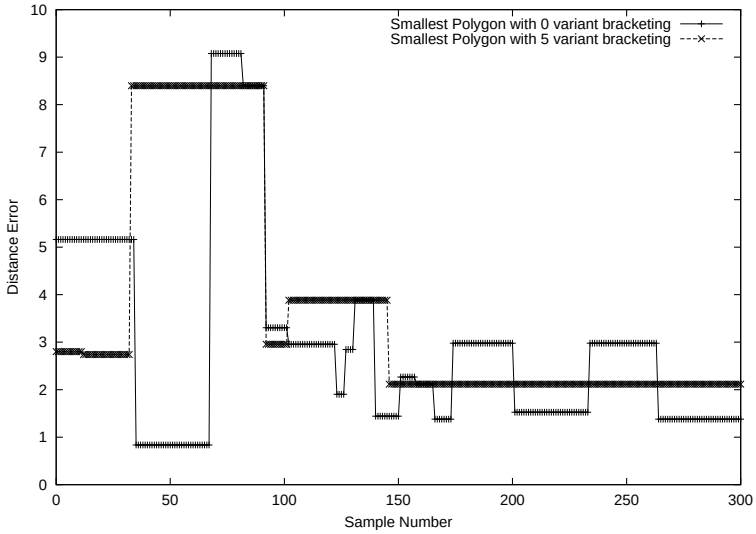

Figure 5.23: The graph shows the stabilization that is achieved by using a 5 unit bracketing range. The original bracketing curve has 7 peaks, while the variant with a 5 unit bracketing range has 2 peaks. Furthermore, the variant algorithm demonstrates an approximately similar average error as the original bracketing algorithm.

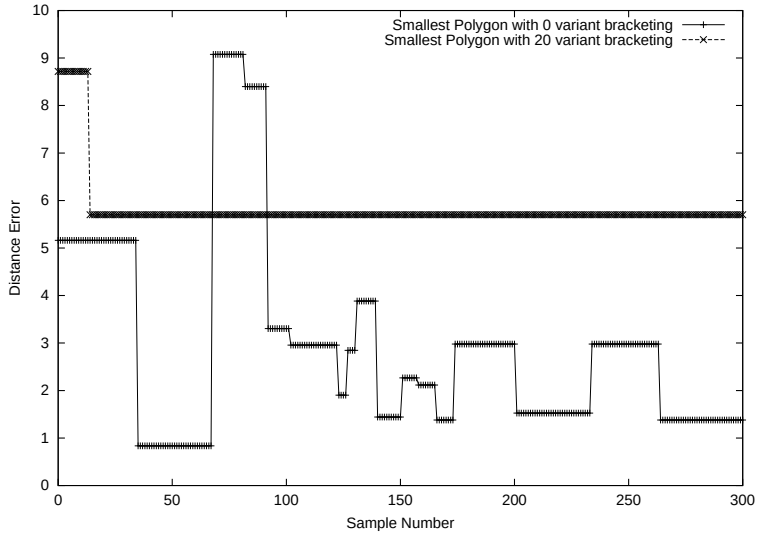

Figure 5.24: The graph shows the stabilization that is achieved by using a 20 unit bracketing range. The original bracketing curve has 7 peaks, while the variant with a 20 unit bracketing range has 1 peaks. However, the average error increases from an average of 2.5m to 5.5m.

the adaptation experiences reduced variation while maintaining an approximately similar distance error. However, Figure 5.24 shows that increasing the bracket value of the variant may achieve greater stabilization, but at the cost of increased distance error. Hence a trade off is involved when trying to stabilize distance error variation.

5.3.3 Data Fusion

This section discusses the performance of the data fusion algorithms described in 4.4 is discussed. The evaluation also presents a quantitative comparison of the Bluetooth and 802.11 as location sensor technologies. However, the following modifications on corresponding signal data have been made to enable accurate comparison:

The following signal data modifications were carried out:

Bluetooth data modification The number of Bluetooth link quality samples measured for the offline and run-time case were reduced to match the sample number of 802.11 wireless LAN. Thus, for the offline case, 50 link quality samples from each direction are averaged and stored in the static tables for that physical location. For the run-time case, the number of samples was reduced from 1500 to 50 to match the 802.11 sample number.

802.11 wireless LAN data modification The Bluetooth infrastructure consists of three access points, and the 802.11 infrastructure consists of three base stations. However each 802.11 base station is mounted with two network cards and this effectively increases the number of access points to six for 802.11. Experimentation was conducted to calculate the average difference in signal strength between two cards mounted on the same base station for all the 49 measurements. The difference was found to be 3dBm. Since the signal strength readings for a pair of cards mounted on the same base stations differs by

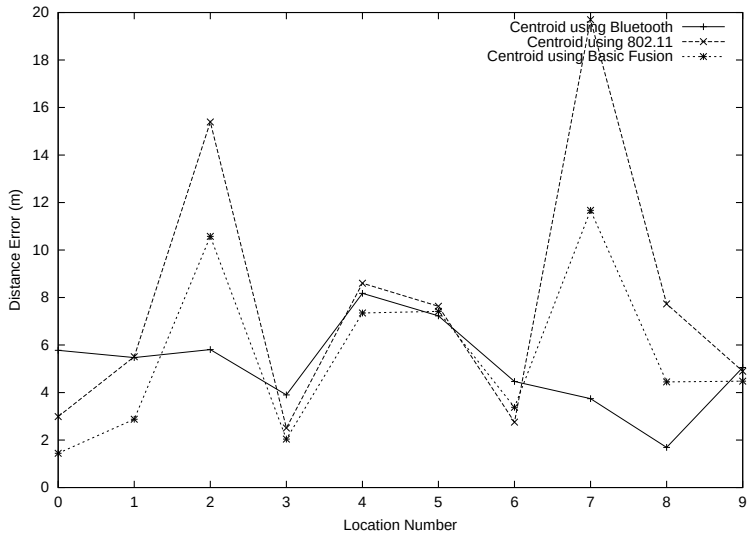

Figure 5.25:

an average of 3dBm, all the signal strength measurements for pair of cards belonging to the same base stations were averaged and stored in the static tables to represent one base station.

The Basic Averaging Algorithm

The modified data sets described above were processed through the adapted static scene analysis using a bracket value of 20 link quality for Bluetooth and 6dBm for 802.11. Both these values are calculated from the average of the standard deviations of the run-time data at all the 30 physical locations. All the 30 run-time locations have three 802.11 base station coverage. However for Bluetooth 10 locations have three, two and one access point coverage. Hence the 10 locations with both three Bluetooth access point coverage and 802.11 wireless LAN base station coverage have been selected for comparison.

The results of static scene analysis are then processed through Centroid, Smallest Polygon, Convex Hull and Triangulation. For each algorithm, the estimates from Bluetooth and 802.11 are averaged according to the description of Section 4.4. In brief, this essentially involves finding the mid-point between the two individual estimates.

- Figure 5.25 compares the distance error involved with Bluetooth, 802.11 and Fusion using the Centroid algorithm. At locations 0, 1, 3, 4 and 9, the distance error for Basic Fusion is less than that of Bluetooth and 802.11. For the remaining locations, Basic Fusion helps reduce the error involved in either Bluetooth or 802.11.

- Figure 5.26 compares the distance error involved with Bluetooth, 802.11 and Fusion using the Smallest Polygon algorithm. Locations 0, 3 and 8 experience a smaller distance error with Basic Fusion than Bluetooth or 802.11. For the remaining locations, Basic Fusion helps reduce the error involved in either Bluetooth or 802.11.

- Figure 5.27 compares the distance error involved with Bluetooth, 802.11 and Fusion

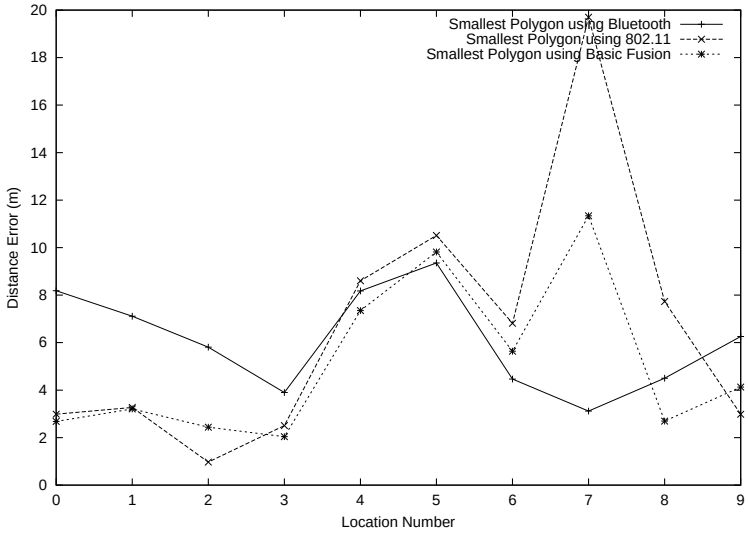

Figure 5.26:

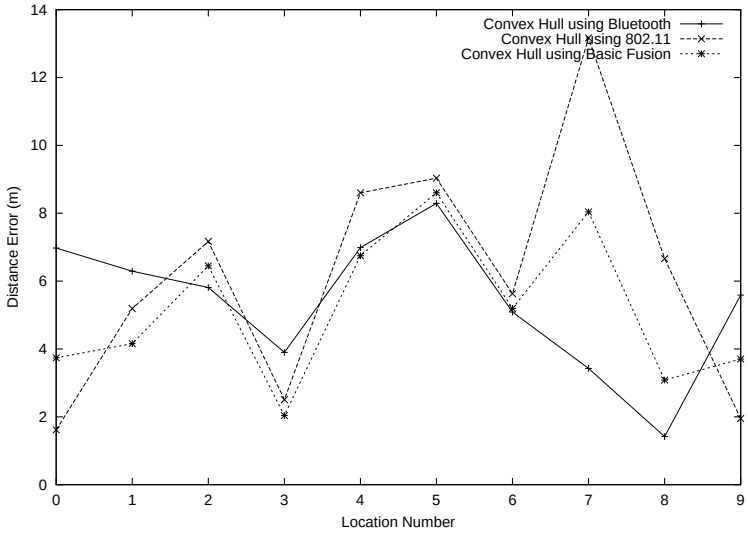

Figure 5.27:

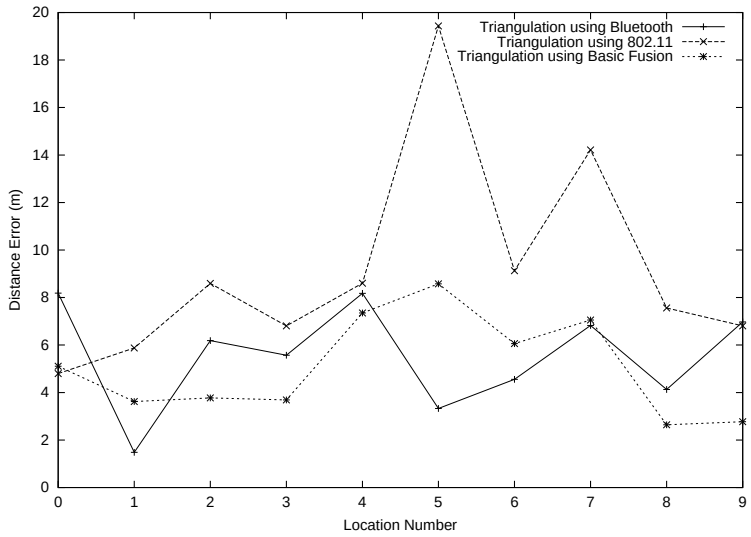

Figure 5.28:

	Centroid (m)	Smallest Polygon (m)	Convex Hull (m)	Triangulation (m)
Bluetooth	5.13	6.09	5.38	5.54
802.11	7.77	6.61	6.15	9.18
Fusion	5.56	5.13	5.17	5.06

Table 5.6: The table summarizes the average distance error involved in Centroid, Smallest Polygon, Convex Hull and Triangulation, for Bluetooth, 802.11 and Basic Fusion.

using the Convex Hull algorithm. At locations 1, 3 and 4 the distance error for Basic Fusion is less than that of Bluetooth and 802.11. For the remaining locations, Basic Fusion helps reduce the error involved in either Bluetooth or 802.11.

- Figure 5.28 compares the distance error involved with Bluetooth, 802.11 and Fusion using the Triangulation algorithm. At locations 2, 3, 4, 8 and 9 the distance error for Basic Fusion is less than that of Bluetooth and 802.11. For the remaining locations, Basic Fusion helps reduce the error involved in either Bluetooth or 802.11.

In all the graphs discussed above, Basic Fusion is observed to either propose a location with distance error less than both Bluetooth and 802.11, or propose a location which improves upon the error of one of the two technologies. Furthermore, certain locations are observed to have a lower error with Bluetooth while others enjoy a reduced distance error with 802.11. Basic Fusion ensures that the error at a location is at least better than one technology. Table 5.6 summarizes the average error involved in Bluetooth, 802.11 and Fusion for all the four algorithms.

Chapter 6

Implementation

6.1 Chapter Summary

Based on the system architecture presented in Chapter 3, the prototype implementation has focussed development on three components:

- The back-end server system.
- The location tracking architectures to collect sensor information.
- A demonstration application.

All implementations are developed in the Java programming language.

6.2 The Server System

The back-end server system consists of three components:

Database An Oracle database housing tables based on the schemas described in Appendix B. The schemas cater for storing technology-specific information, sensor-independent location information and user details, for each of Bluetooth, 802.11 wireless LAN, GPS and Terminal Login.

API The data stored in the relational tables is accessible via a standard Java application programming interface (API) detailed in Appendix C. The API may be used by applications to query the database for real-time user location, or by individual sensor frameworks to feed raw information.

Daemons The API may also be used for location estimation using the algorithms discussed in Chapter 4. The business logic of these algorithms is encapsulated in daemon processes that run periodically to estimate user location.

6.2.1 Location Estimator Daemons

The daemons undertake the task of location estimation using static scene analysis and sensor-independent location estimation algorithms discussed in Chapter 4. In Figure 6.1, the daemon process acquires signal information from the offline and run-time tables, processes it using

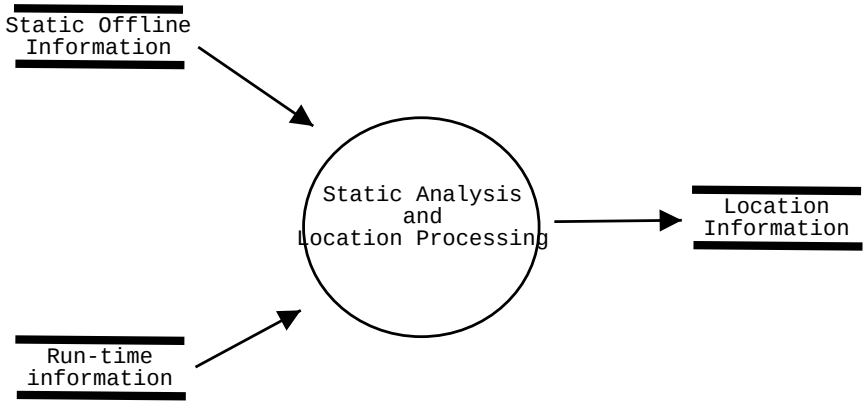

Figure 6.1: The diagram shows a daemon process using the offline and run-time information to produce a location result using static scene analysis and sensor-independent estimation.

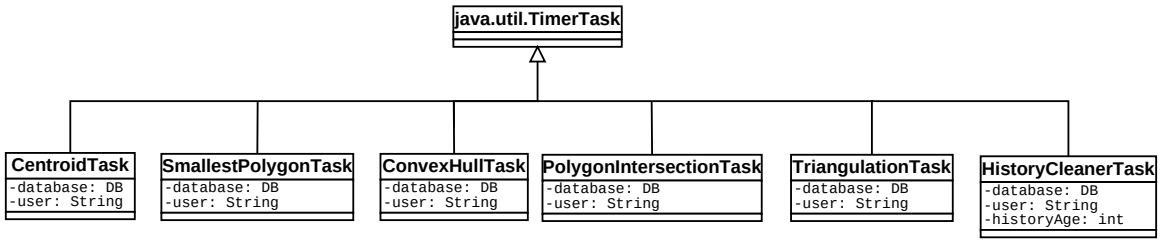

Figure 6.2: The diagram shows the `java.util.TimerTask` is used to provide the daemon functionality.

static scene analysis and sensor-independent location estimation, and stores the resulting location information into the location tables.

The location processing has been implemented as a daemon because this method optimizes periodic execution using minimum system resources. As shown in Figure 6.2, the daemon implementation is achieved by inheriting from the `java.util.TimerTask`. The `HistoryCleaner` is an additional daemon responsible for deleting sensor and location information that is classified invalid or stale based on some time criterion. This is done to prevent the collection of *stale data* over a period of time. For example, in the prototype implementation, data relating to Bluetooth is erased at half-hour intervals. The time criterion may be set to meet the nature of the technology. For example, in contrast to Bluetooth, GPS information may be erased on a weekly or monthly basis.

Daemon initialization is achieved using the `java.util.Timer` class as shown in Figure 6.3. The daemons are initialized with the following parameters:

- The time period between successive executions of the daemon. By default, this has been set to one second.
- Username of the user to be tracked by the daemon.
- Password with which the daemon is authenticated at the database.

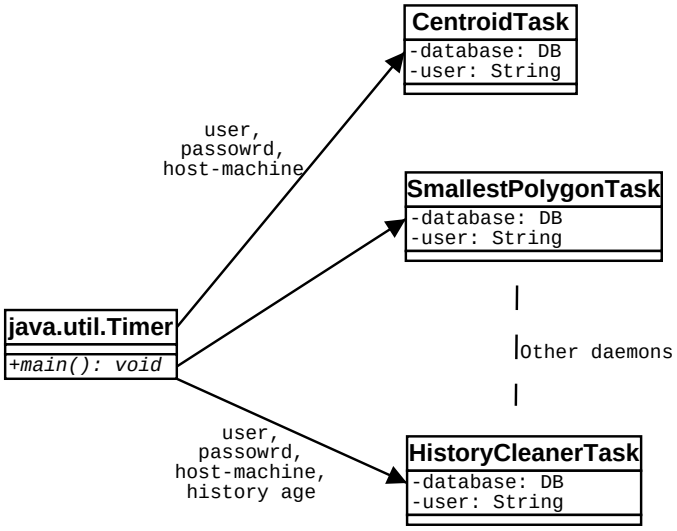

Figure 6.3: The diagram shows that the `java.util.Timer` class is used to initialize the daemon processes. During initialization, parameters are passed to configure daemon execution to suit the task at hand.

- Any further information specific to the task of the daemon. For example, the HistoryCleanerTask needs to be initialized with a time criterion to judge the validity of information.

6.3 Tracking Implementations

The previous section discussed the server-side implementations responsible for computing location information for raw sensor information. This section describes the methods implemented to collect sensor information from each technology. In this sense, a tracking architecture is analogous to a data collection system.

6.3.1 Bluetooth

Section 3.3.1 evaluated two possible Bluetooth tracking architectures based on the experimentation discussed in Appendix A. The evaluation concluded that the architecture with the Active Client was more practical because it allowed signal strength updates to be made in real time. We have developed the Active Client system where the mobile client searches for access points in the area, connects to them sequentially, measures signal information for each connection and updates the database via a database server running on a pre-determined host. A simplified version of the Active Client architecture of Section 3.3.1 is shown in Figure 6.4. The active client implementation uses the Java API built at Telcordia Technologies as part of a student placement. The software architecture is illustrated in Figure 6.5.

Robustness at the active client is maintained by using the event handler feature of the Bluetooth Java API. For any connection or disconnection, the Bluetooth device fires an event. The cause of the event may be determined and if the connection between the active client and access point running the database server is broken, the client is programmed to sever all

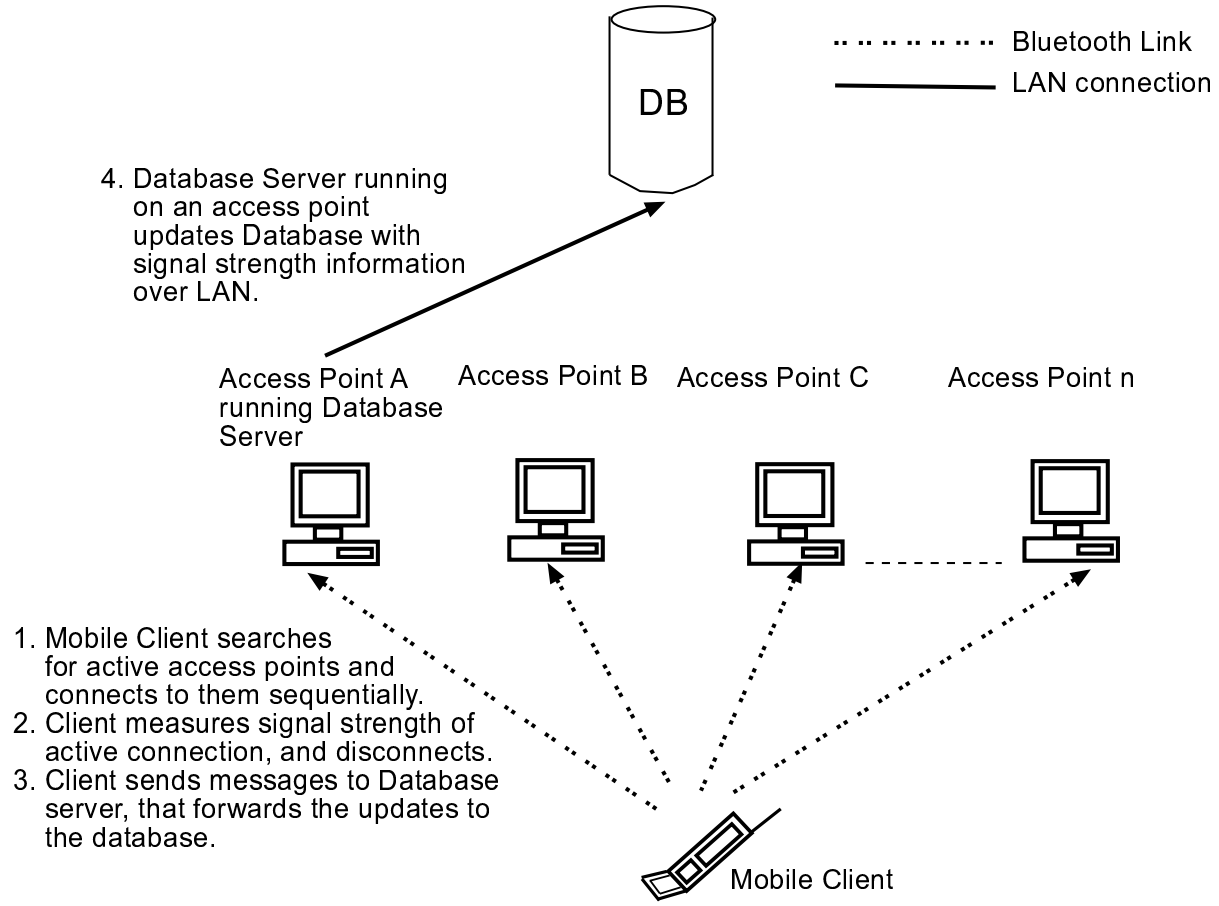

Figure 6.4: A simplified ActiveClient Bluetooth tracking architecture.

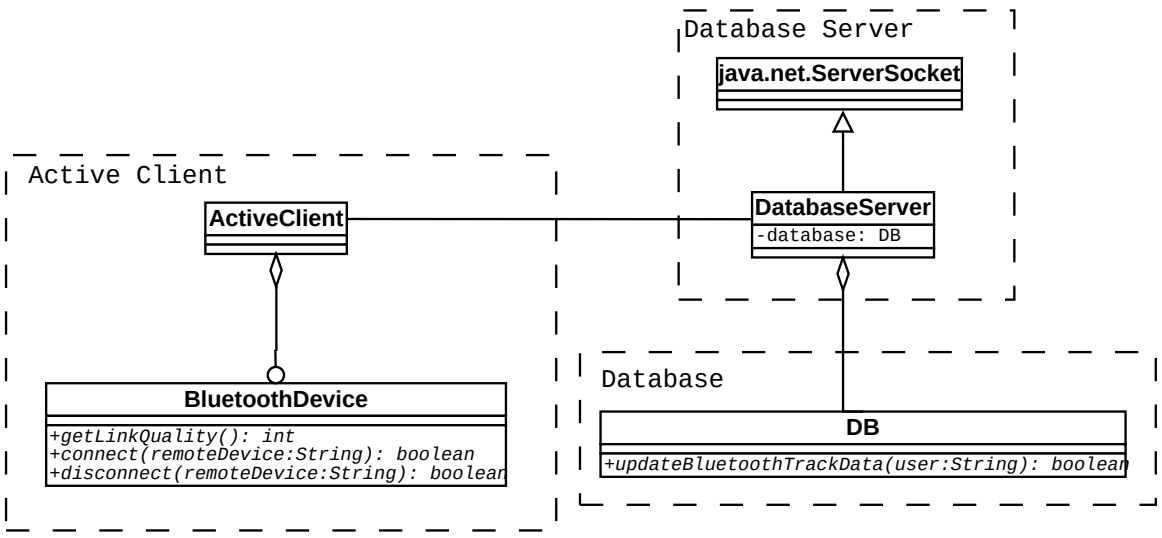

Figure 6.5: The diagram shows the relationship between various the components of the Bluetooth tracking system, and the sub-components of each component. The BluetoothDevice class abstracts away the various initialization procedures required to operate Bluetooth from the Java API of Telcordia. The relationship between ActiveClient and Database Server operates over Bluetooth.

existing connections and attempt to re-establish a connection with the database server access point.

Finally, low pass filtering, described in Section 5.3.1, is carried out at the database server. This design frees the client from computing averages and may assist in saving battery power.

6.3.2 802.11 Wireless LAN

Section 3.3 proposed two systems that allowed signal data to be collected from the base station and the client. Since we did not have permissions to access the base stations in the 802.11 departmental infrastructure, it was not possible to query the base station. Furthermore, local query on the mobile client was challenging because it required the use of proprietary APIs that were not available. However it was possible to configure the Orinoco Client Manager, shown in Figure 6.6, to log signal information into a data file. Since the format of the logs were the similar in structure, a Java parser was implemented that collected the required signal information from the log file. Figure 6.7 illustrates the various components that were involved in this data collection process.

6.3.3 GPS

Section 3.3.3 introduced an architecture where GPS information could be collected from a receiver using a PC connection. However, since we are equipped with a Garmin receiver, pictured in Figure 6.8, whose PC interface was not available, GPS coordinates of the corner of the floor plan were measured empirically, and are shown in Figure 6.9.

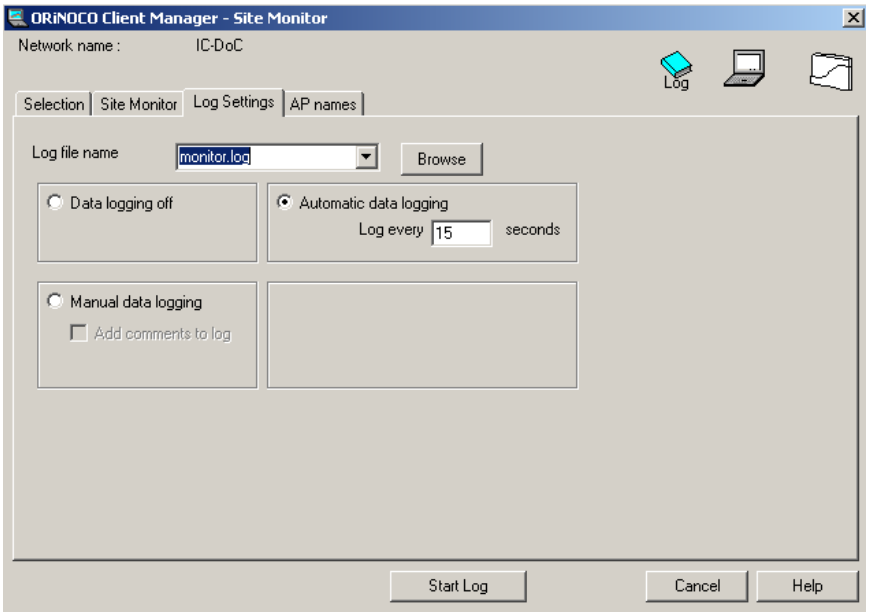

Figure 6.6: The diagram shows a snapshot of the Orinoco Client Manager that was configured to log signal data into a file.

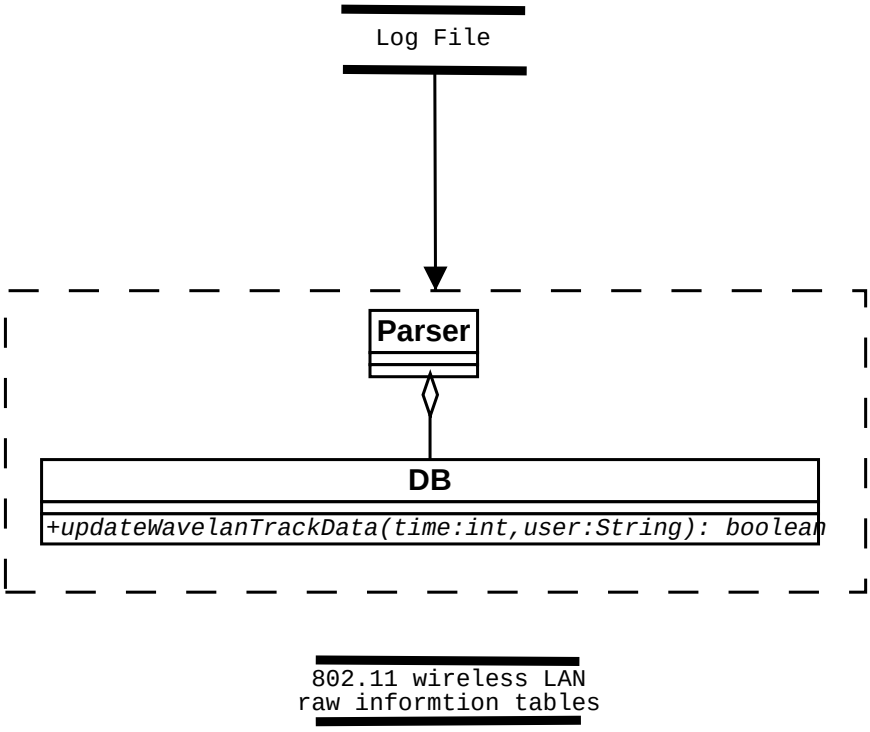

Figure 6.7: The diagram shows a Parser class collecting relevant signal information from a log file, and updating the appropriate tables in the database.

Figure 6.8: The picture shows a Garmin Etrex receiver that was used collect GPS measurements.

6.3.4 Terminal Login

Login information is collected from a Computing Department database that maintains a record of terminals and the users logged on them. Figure 6.10 illustrates the software architecture that collects, filters and updates location information from the departmental database into our location tracking database.

6.4 A Demonstration Application

The demonstration of the location system includes a simple map application that displays a region in which the user is possibly located.

GUI Description Figure 6.11 explains the various panels in the application interface. The larger panel on the right is the called the *Map Panel* as it displays a static map on which user location is traced. The panel at the bottom is the *Status Bar* and indicates log information and error conditions. The panel on the left is termed the *Control Panel*, as it contains the buttons used to interact with the application.

Application Startup User location tracking may be initiated and discontinued by clicking on the "Start" and "Stop" buttons respectively in the Control Panel. Application interaction with the back-end system may be observed in the Status Bar. On successful startup, the application colours a region in the Map Panel to indicate the possible location area of the user.

Running The Application During normal operation, the user may select an estimation algorithm from the list of algorithms in the Control Panel, shown in Figure 6.12, and observe their effect visually at improving location estimation.

Figure 6.13 shows that the application interacts with the database via the `Database` class.

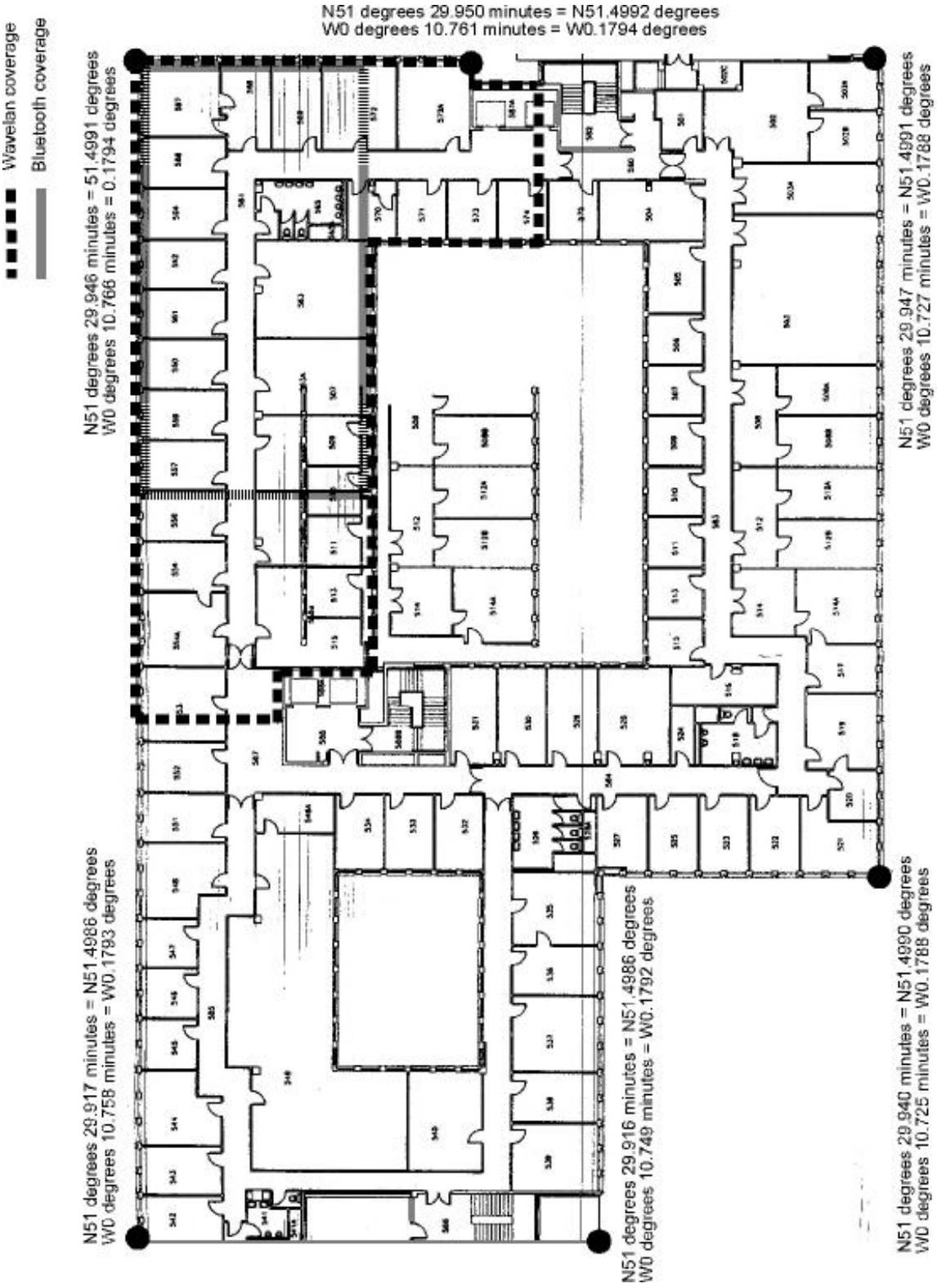

Figure 6.9: The diagram shows the GPS coordinates at the corners of the floor plan.

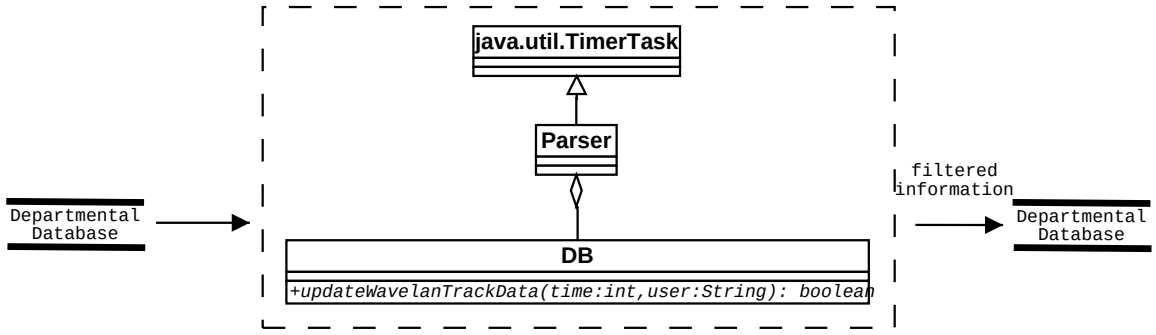

Figure 6.10: The diagram shows the Terminal Login data collection architecture. Data is collected from the Computing Department database, filtered and updated in our location tracking database.

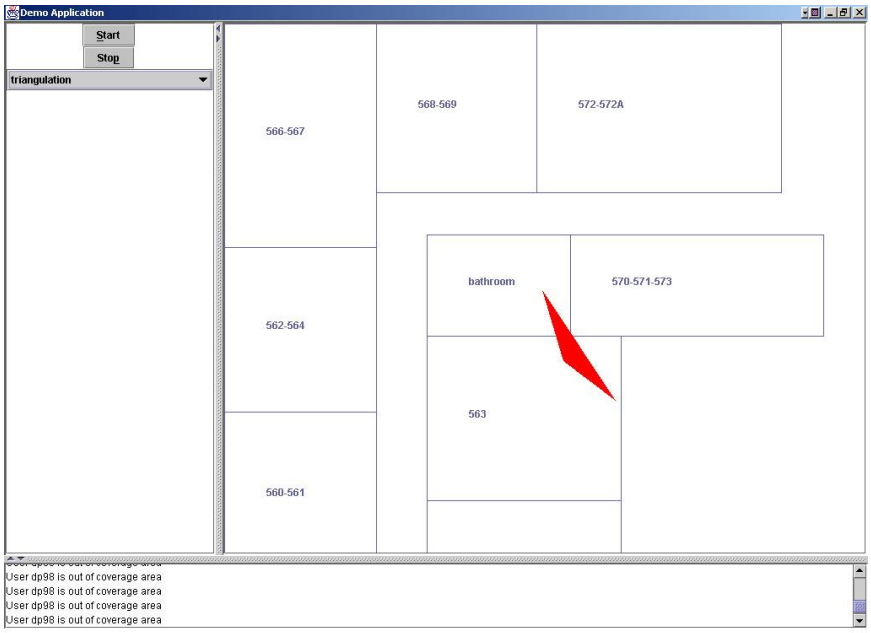

Figure 6.11: The Map Application. The larger panel on the right is the called the *Map Panel* as it displays a static map on which user location is traced. The panel at the bottom is the *Status Bar* and indicates log information and error conditions. The panel on the left is termed the *Control Panel*, as it contains the buttons used to interact with the application. During operation, the application colours a region in the map panel to indicate the possible location area of the user.

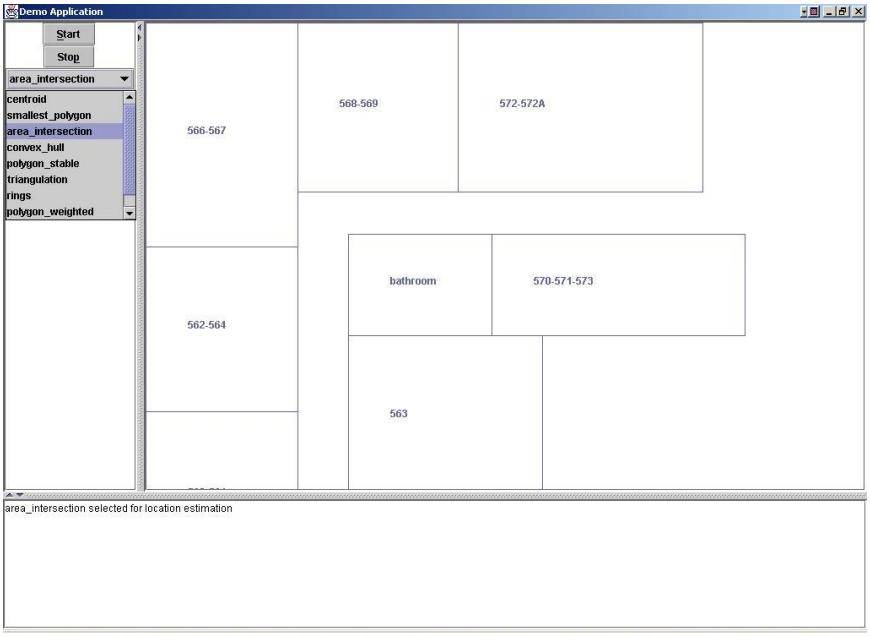

Figure 6.12: This snapshot shows how a user may select an algorithm when running the application.

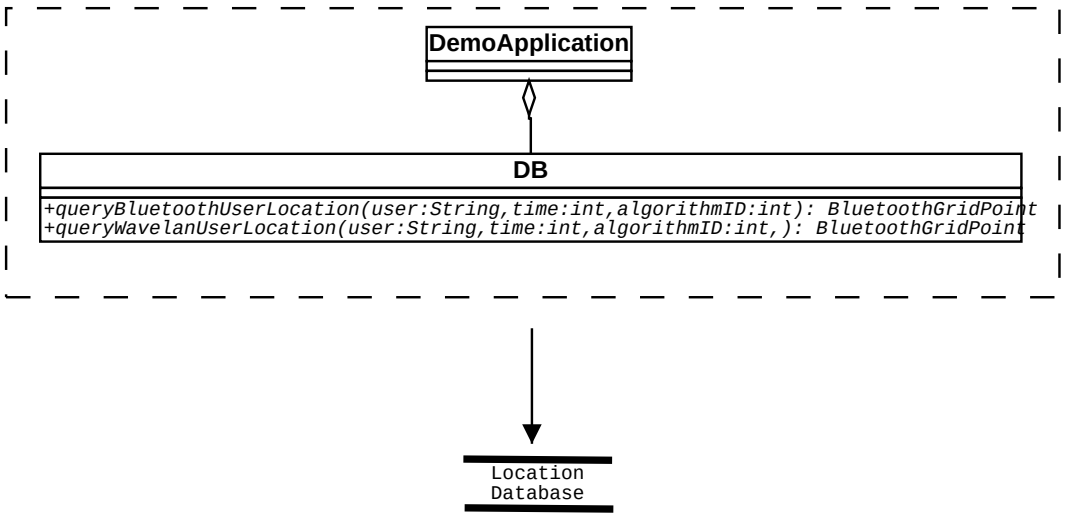

Figure 6.13: The diagram shows the application interacts with the database using the API presented by the `Database` class.

Chapter 7

Conclusion

7.1 Chapter Summary

This chapter includes:

- A review of the aims of the project.
- A summary of achievements and conclusions from experiments.
- A discussion into current limitations and proposals for further work.

7.2 Aims

The proliferation of mobile computing devices and the development of high-speed, low-costs wireless networks, has created an environment of "everywhere connectivity", thus presenting ample opportunity for the creation of ubiquitous systems that respond to changes in user environment in a bid to enrich user experience.

Location systems have gained a large audience, with the aim of enabling applications to respond to changing user location. Like any other environmental feature, location needs to be computed from a set of raw sensor readings, with sensors based on radio-frequency waves, ultrasound and infrared having gained popular use.

Contemporary location systems have shown an increasing trend to develop around the technology being used compute location. This in turn restricts the application set to the limitations of the sensor technology. Furthermore, the method employed to represent location is determined by the sensor and hence very specific to the requirements of the system. For example, the location computations and representation in the Cricket system described in Section 2.4.2, is based on custom-built wireless network of sensors. Hence any applications are limited to the scale of the sensor infrastructure.

Future mobile devices are expected to feature multiple technologies, thus creating the opportunity of collecting sensor information from multiple sources for the same location. Therefore, it may be possible to alternate between the various sources information to overcome the limitations of a technology in particular circumstances. Furthermore, the application is offered the flexibility to select between the various sources of information to suit its particular requirements. This is in contrast to the Cricket approach where the application is constrained by the technology in use.

Based on these observations, the project focusses on the following issues:

- Collect sensor data for Bluetooth, IEEE 802.11 Wireless LAN, GPS and Login.
- Explore sensor-independent location estimation techniques.
- Identify a common model to represent location information.
- Investigate the possible techniques for merging location information, and identify application domains that would effectively make use of any fused information.

7.3 Achievements

Work has been done in the following areas:

Data Collection Architectures Architectures have been proposed and implemented to collect raw sensor data from the Bluetooth, IEEE 802.11 Wireless LAN, GPS and Login technologies. Limitations of the Bluetooth technology were identified, quantified experimentally and duely considered in the architecture design solutions. For the remaining technologies, the prototypes systems are modified from ideal architectures to consider practical limitations.

Sensor-independent Location Estimation Raw sensor information is transformed into location estimates using a common location estimation technique called Static Scene Analysis. The nature of this technique makes it suitable for indoor location tracking and hence has been used to estimate locations using the radio-frequency based sensors , Bluetooth and 802.11 wireless LAN. Since radio-frequency waves are vulnerable to random interference, we have proposed a filtering method and an adaptation to static scene analysis to take interference into account. The estimates from static scene analysis are represented on an imaginary grid system on our floor. These estimates are further processed by estimation algorithms to improve on location accuracy. Since the algorithms operate on coordinates referenced in the imaginary grid system, their business logic in independent of the sensor technology. The performance gain achievable on filtering, scene analysis adaptation and the sensor-independent algorithms has been quantified and evaluated experimentally. The highest average accuracy achievable from the algorithms was 5.10m.

Data Fusion Since the Bluetooth and 802.11 wireless LAN infrastructures overlapped, it was possible to combine estimates from each technology. A basic fusion algorithm is proposed and experimental evaluations show a **maximum improvement in average accuracy of 0.96m and 2.21m**, and **minimum improvement of -0.43m and 0.21m** over Bluetooth and 802.11 wireless LAN respectively. These are encouraging results considering that the algorithm finds the average between the Bluetooth and 802.11 estimates.

7.4 Limitations and Further Work

This section attempts to identify weaknesses in the system and open challenges to active research areas:

Bluetooth Tracking Architecture The ActiveClient architecture outlined in Section 3.3.1 is not scalable to track multiple clients in a closed environment. In contrast, the DormantClient architecture is functionally similar to 802.11 wireless LAN tracking systems and hence can cater for a larger audience of devices. However, the long delays associated with Bluetooth connections make this architecture inefficient in registering signal information at a reasonably high frequency of one sample per second. Current hardware developments with Bluetooth implementations have managed to reduce this delay significantly, and this may make the DormantClient architecture a better candidate for Bluetooth location tracking.

Bluetooth Signal Information The signal information available from our Toshiba Bluetooth hardware is not a standard physical quantity used to measure the intensity of radio signals. It is possible that the peculiar characteristics of Link Quality, discussed in Section 3.2, caused location accuracy to be distorted. The fact that the average accuracy of Bluetooth was 5.10m at full transmitter power is still encouraging, if not satisfactory. Furthermore, the wide acceptance of Bluetooth evident in contemporary mobile devices stands as a fair argument for using Bluetooth as a location sensing technology. Hence, we encourage all effort aimed at using Bluetooth as a location tracking sensor.

Database Bottleneck The database is the bottleneck of the system, and raises significant concerns over scalability. This is aggravated by the fact that that there are multiple technologies feeding information for a large number of users. Furthermore, the flexibility allowed to the applications will make their queries more complex resulting in higher processing requirements at the server. The bottleneck problem may be resolved by the use of distributed database systems. Since a user's location does not change over geographically large distances, less stringent requirements may be placed on data synchronization between peer databases.

User Privacy User Privacy is fundamentally important issue that requires consideration. Distributed systems like Cricket are able to secure user privacy by the nature of their architecture. In such systems, a user application initiates location estimation when required, rather than being explicitly tracked by the network of sensors. In systems that perform explicit tracking, user privacy may be addressed using access control policies being set up at the server. In our prototype implementation, policies may be stored along with user details and used to filter queries depending on their configuration.

Data Fusion Our basic data fusion algorithm is able to achieve a small but significant improvement in location accuracy, and holds promise for more sophisticated algorithms processing data from more than two sensors. Furthermore, improving location accuracy is an application of data fusion. Data fusion may also be used in *location data interconversion*. Location data interconversion involves converting location data from one representation to the other. For example, the GPS coordinates at the corners of our floor plan are determined using an ordinary GPS receiver. Bluetooth or 802.11 wireless LAN may be used to determine the location of a user within the floor plan. If the distance of the user from one of the corners is calculated in terms of metres, and the length of one degree of latitude and longitude is known at the corner GPS coordinates, than it is possible to calculate the GPS coordinates of the user inside the floor plan. In

this way, the limitation that GPS cannot work inside buildings is avoided. Furthermore, the GPS coordinate system may potentially be used to represent any physical location.

Trajectory Prediction Predicting user location presents a challenging task. The use of multiple sensors may provide location with different granularity and may be used in prediction algorithms. For example, location information provided by cellular systems may be used to in combination with user mobility patterns to predict the GPS coordinates of the user's final destination.

Appendix A

Characteristics of the Bluetooth specification

A.1 Timing considerations

Table A.1 shows the minimum, maximum and average connection times for a typical Bluetooth implementation. However, recent development work on the hardware has shortened these times.

Figure A.1 and A.2 show that the Toshiba implementations have a maximum connection and disconnection time of 3s and 0.5s respectively. Figure A.3 shows that time to measure link quality is relatively minute and can be ignored.

Procedure	Minimum Time	Average Time	Maximum Time
Inquiry	0.00125s	3-5s	10.24-30.72s
Paging	0.0025s	1.28s	2.56s
Total	0.00375s	4.28-5.28s	12.8-33.28s

Table A.1: Theocratical Bluetooth connection set up times.

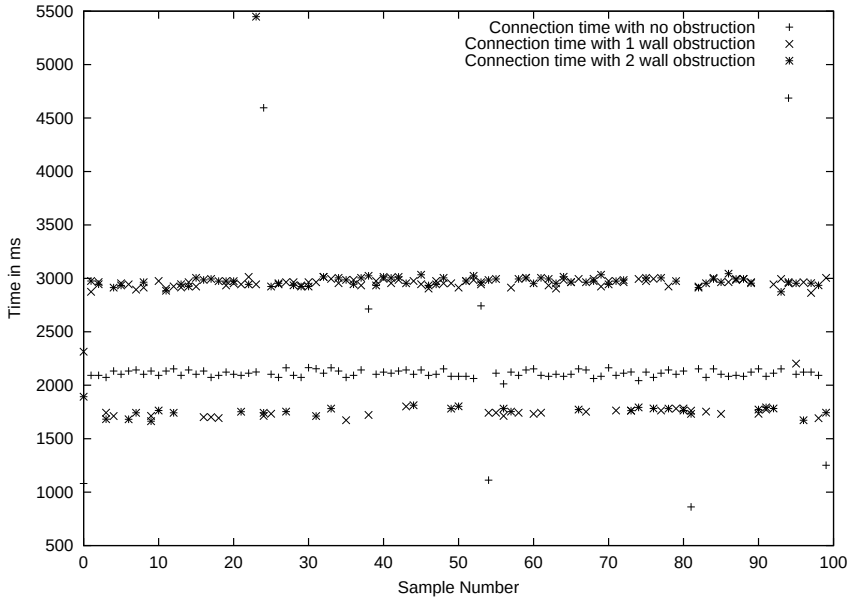

Figure A.1: The graph shows that our hardware takes approximately 2.1 seconds to establish a Bluetooth connection in the absence of an obstruction. This time is increased to 3 seconds with obstructions, however the number of obstructions has no effect on the connection set up time.

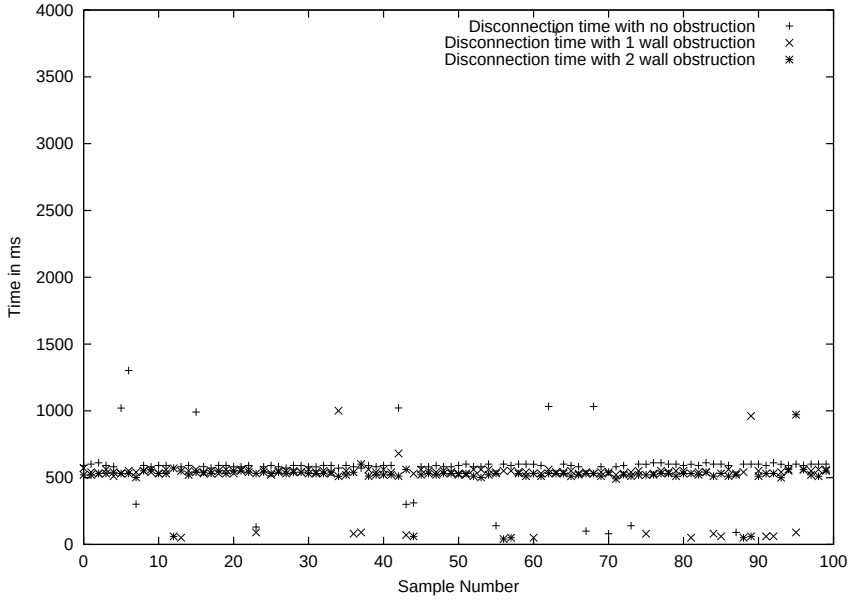

Figure A.2: The graph shows that our hardware takes approximately 0.5 seconds to disconnect an active connection. As expected, the level of obstruction has no effect on the time value.

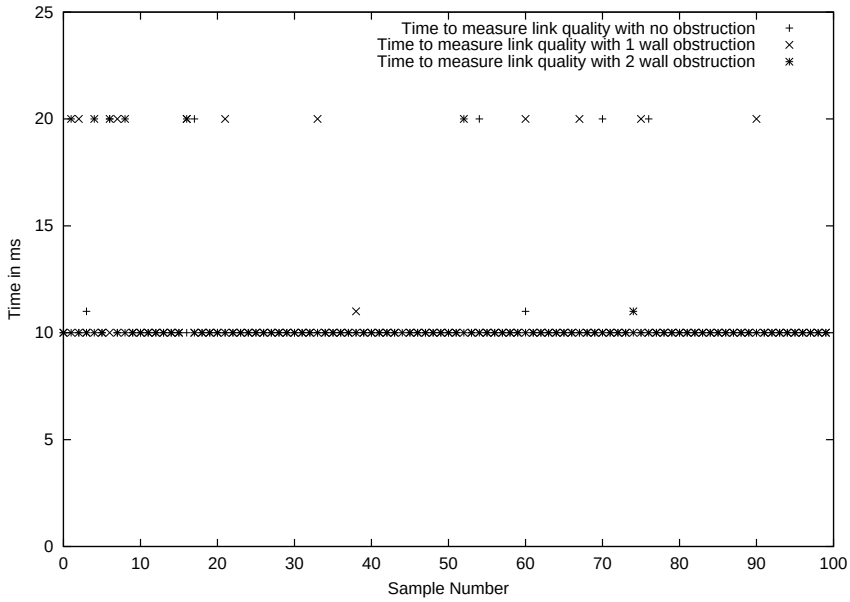

Figure A.3: The graph shows that time to measure one link quality is approximately 10ms, and as can be expected, it is not affected by the any obstructions.

Appendix B

A Location Data Model

This appendix details the API that allows unified database access to applications and tracking systems.

```
/* Bluetooth offline table */
CREATE TABLE bluetooth_offline (
x FLOAT,                        /* grid location, x coordinate */
y FLOAT,                        /* grid location, y coordinate */
access_point VARCHAR(12)        /* mac address of access point */
PRIMARY KEY (x,y,access_point),/*x,y and access_point identify a
                                              unique link_quality*/
link_quality INTEGER NOT NULL  /* signal data */
);

/* Bluetooth run-time table */
CREATE TABLE bluetooth_runtime (
collect_time INTEGER,           /* time in seconds at which data
                                                    is collected */
id VARCHAR (10),                /* user id to whom data belongs */
PRIMARY KEY (collect_time, id),/* collect_time and id identify a
                                           unique link_quality */
link_quality INTEGER NOT NULL  /* run-time signal information */
  );

/* Bluetooth location estimate */
CREATE TABLE bluetooth_location (
time INTEGER,                   /* time in seconds of sighting */
user_name VARCHAR(10),          /* user id */
PRIMARY KEY (time, user_name), /* time and user id identify
                                             a unique location */
x FLOAT,                        /* grid location, x coordinate */
y FLOAT                         /* grid location, y coordinate */
  );
```

Figure B.1: Schemas for holding Bluetooth offline, run-time and location information. `bluetooth_offline` holds offline link quality information. `bluetooth_runtime` holds run-time link quality information. `bluetooth_location` holds location information referenced on the grid system, and an instance exists for each location estimation algorithm.

```
/* 802.11 wireless LAN offline table */
CREATE TABLE wirelesslan_offline (
  x FLOAT,                        /* grid location, x coordinate */
  y FLOAT,                        /* grid location, y coordinate */
  mac_address VARCHAR(12),        /* mac address of card on base station */
  PRIMARY KEY (x,y,mac_address),  /* x,y and mac_address identify unique
                                                          sl,nl,snr */
  sl INTEGER,                     /* signal strength in dBm */
  nl INTEGER,                     /* noise level in dBm */
  snr INTEGER                     /* signal to noise ratio */
  );
/* 802.11 wireless LAN run-time table */
CREATE TABLE wirelesslan_runtime (
  collect_time INTEGER,           /* time in seconds at which data
                                                       is collected */
  id VARCHAR (10),                /* user id to whom data belongs */
  PRIMARY KEY (collect_time, id), /* user id and time identify a unique sl*/
  sl INTEGER NOT NULL             /* run-time signal information */
  );
/* 802.11 wireless LAN location estimate */
CREATE TABLE wirelesslan_location (
  time INTEGER,                   /* time in seconds of sighting */
  user_name VARCHAR(10),          /* user id */
  PRIMARY KEY (time, user_name),  /* time and user id identify a unique
                                                          location */
  x FLOAT,                        /* grid location, x coordinate */
  y FLOAT                         /* grid location, y coordinate */
  );
```

Figure B.2: Schemas for holding 802.11 wireless LAN offline, run-time and location information. `wirelesslan_offline` holds offline signal strength information. `wirelesslan_runtime` holds run-time signal strength information. An offline and run-time table exists for each base station. `wirelesslan_location` holds location information referenced on the grid system. A table exists for each location estimation algorithm.

```
/* GPS offline table */
CREATE TABLE gps_offline (
  latitude FLOAT,                  /* latitude */
  longitude FLOAT,                 /* longitude */
  location VARCHAR(20) PRIMARY KEY /* symbolic location */
  );
/* GPS run-time table */
CREATE TABLE gps_runtime (
time INTEGER,                 /* time in seconds of sighting */
user_name VARCHAR(10),        /* user id */
PRIMARY KEY (time, user_name),/* time and user id identify unique
                                                    gps coordinates */
latitude FLOAT,               /* latitude */
longitude FLOAT               /* longitude */
  );
/* GPS location estimate */
CREATE TABLE gps_location (
  time INTEGER,                 /* time in seconds of sighting */
  user_name VARCHAR(10),        /* user id */
  PRIMARY KEY (time, user_name),/* time and user id identify a unique
                                                          location */
  latitude FLOAT,               /* latitude */
  longitude FLOAT,              /* longitude */
  location VARCHAR(20)          /* symbolic location */
  );
```

Figure B.3: Schemas for holding GPS static, run-time and location information. `gps_offline` holds the translation between symbolic names, for example building name, and latitude-longitude coordinates. `gps_runtime` holds run-time latitude-longitude information. `gps_location` holds name of physical location where the user is positioned.

```
/* Login offline table */
CREATE TABLE login_offline (
  terminal_name VARCHAR(10) PRIMARY KEY,   /* name of the terminal */
  location VARCHAR(10),                    /* location of terminal */
  );
/* Login run-time table */
CREATE TABLE login_runtime (
  time INTEGER,                  /* time in seconds of sighting*/
  user_name VARCHAR(10),         /* user id */
  PRIMARY KEY (time, user_name), /* time and user id identify a unique location */
  terminal_name VARCHAR(10)      /* name of the terminal */
  );
/* Login location estimate */
CREATE TABLE login_location (
  time INTEGER,                  /* time in seconds of sighting*/
  user_name VARCHAR(10),         /* user id */
  PRIMARY KEY (time, user_name), /* time and user id identify a unique location */
  terminal_name VARCHAR(10)      /* name of the terminal */
  location VARCHAR(10),          /* location of terminal */
  );
```

Figure B.4: Schemas for holding Terminal Login static, run-time and location information. `login_offline` holds the symbolic locations where terminals have been placed, `login_runtime` holds the name of the terminal where users are logged, and `login_location` holds the name of the location where the user is logged on and hence located.

Appendix C

A Java API for Location Query

This appendix details the API that allows unified database access to applications and tracking systems.

```
/* Allocates memory for a TerminalInformation object and initializes it
 * with the given parameters
 */
public TerminalInformation (String terminalName, String roomName)

/* returns the name of the terminal where user is logged in */
public float getTerminalName()

/* returns the name of the room where the terminal is located */
public float getRoomName()
```

Figure C.1: TerminalInformation. This container class is used to hold login information about a user.

```
/* Allocates memory for a BluetoothGridPoint object and initializes it
 * with the given parameters
 */
public BluetoothGridPoint(float xCoord, float yCoord)

/* return the x-coordinate of the location */
public float getX()

/* return the y-coordinate of the location */
public float getY()
```

Figure C.2: BluetoothGridPoint. This container class is used to hold location information represented on the imaginary grid system introduced in Section 5.2.3.

```
/* Connects to a database running on a remote
 *
 * parameter host: name of remote host
 * parameter username: name with which user is identified
 * parameter password: to authenticate the user
 */
public DB (String host, String username, String password)


/* closes any connections to the database */
public void close()


/* returns real time Bluetooth tracking information
 *
 * parameter user: name with which user is identified
 * parameter time: time in seconds since "epoch" or January 1, 1970, 00:00:00 GMT
 *
 * returns: a BluetoothTrackData object
 */
public BluetoothTrackData queryBluetoothTrackData (String user, int time)


/* updates the database with Bluetooth link quality information measured at
 * the given time
 *
 * parameter user: name with which user is identified
 * parameter time: time in seconds since "epoch" or January 1, 1970, 00:00:00 GMT
 * at which signal measurements were taken
 * parameter data: signal information
 *
 * returns: true if update successful, false if otherwise
 */
public boolean
        updateBluetoothTrackData (String user, int time, BluetoothTrackData data)


/* returns all the grid locations which exactly match the given link quality
 * for a particular access point
 *
 * parameter time: time in seconds since "epoch" or January 1, 1970, 00:00:00 GMT
 * parameter linkQuality: run-time link quality
 *
 * returns: an array grid locations
 */
public BluetoothGridPoint[] queryBluetoothMap (String accessPoint, int linkQuality)
```

```
/* returns the location estimate for a user at particular time,
   computed by the algorithm identified by algorithmID
 *
 * parameter user: user whose location is queried
 * parameter time: time in seconds since "epoch" or January 1, 1970, 00:00:00 GMT,
 * for which location is queried
 * parameter algorithmID: the location algorithm to use to compute location
 *
 * returns: a BluetoothGridPoint that identifies the user on the grid
 */
public BluetoothGridPoint queryBluetoothUserLocation (String user, int time,
                                                      int algorithmID)

/* updates the database with a location estimate of user and a certain time.
 * The algorithm generating the estimate is identified in the parameter.
 *
 * parameter algorithmID: the location algorithm used to compute the location
 * parameter time: time in seconds since "epoch" or January 1, 1970, 00:00:00 GMT,
 * for which location was computed
 * parameter user: the user whose whose location was computed
 * parameter loc: the location estimate of the user
 *
 * returns: true if the update was successful in the database, false otherwise.
 */
public boolean updateBluetoothLocation (int algorithmID, int time, String user,
                                        BluetoothGridPoint loc)

/* Queries the database for 802.11 wireless LAN signal strength data for a user
 * at a particular time.
 *
 * parameter user: name with which user is identified
 * parameter time: time in seconds since "epoch" or January 1, 1970, 00:00:00 GMT
 * parameter data: signal strength data
 *
 * returns: true if the update was successful in the database, false otherwise.
 */
public WavelanTrackData
            updateWavelanTrackData (String user, int time, WavelanTrackData data)
```

```
/* Updates the database with 802.11 wireless LAN signal strength data for a user at a
 * particular time.
 *
 * parameter user: name with which user is identified
 * parameter time: time in seconds since "epoch" or January 1, 1970, 00:00:00 GMT
 * at which data was measured
 *
 * returns: a WavelanTrackData object
 */
public WavelanTrackData queryWavelanTrackData (String user, int time)


/* returns all the grid locations which exactly match the given signal strength
 * for a particular base station
 *
 * parameter macAddress: identity of the base station
 * parameter sl: signal strength data
 *
 * returns: a WavelanTrackData object
 */
public BluetoothGridPoint[] queryWavelanMap (String macAddress, int sl)


/* updates the database with a location estimate of user and a certain time.
 * The algorithm generating the estimate is identified in the parameter. Wireless
 * LAN signal strength is used to estimate the location.
 *
 * parameter algorithmID: the location algorithm used to compute the location
 * parameter time: time in seconds since "epoch" or January 1, 1970, 00:00:00 GMT,
 * for which location was computed
 * parameter user: the user whose whose location was computed
 * parameter loc: the location estimate of the user
 *
 * returns: true if the update was successful in the database, false otherwise.
 */
public boolean updateWavelanLocation (int algorithmID, int time, String user,
                                                              BluetoothGridPoint loc)
```

```
/* returns the location estimate for a user at particular time,
 * computed by the algorithm identified by algorithmID. The estimates are
 * calculated using wireless LAN signal strength
 *
 * parameter user: user whose location is queried
 * parameter time: time in seconds since "epoch" or January 1, 1970, 00:00:00 GMT,
 * for which location is queried
 * parameter algorithmID: the location algorithm to use to compute location
 *
 * returns: a BluetoothGridPoint that identifies the user on the grid
 */
public BluetoothGridPoint queryWavelanUserLocation (String user, int time,
                                                    int algorithmID)


/* queries the database for the location of a base station with a particular
 * mac address
 *
 * parameter macAddress: the location algorithm used to compute the location
 *
 * returns: a WavelanBaseStationLocation object
 */
public WavelanBaseStationLocation queryWavelanBaseStationLocation (String macAddress)


/* queries the database for information about the terminal where
 * the user is logged on
 *
 * parameter user: name or id of user
 *
 * returns: a TerminalInformation object
 */
public TerminalInformation queryTerminalInformation (String user)


/* updates the tables with the new terminal information for the specified
 * user
 *
 * parameter user: id of user to whom this data belongs
 * parameter info: information about terminal where user is logged on
 *
 * returns: true if update is successful, false otherwise
 */
public boolean updateTerminalInformation (String user, TerminalInformation info)
```

Figure C.3: DB. The DB class encapsulates database access by providing methods to query and update location information. It also provides methods for tracking systems to update individual sensor information.

```
/* Allocates memory for a BluetoothTrackData object
 */
public BluetoothTrackData()


/* Adds an access point and its corresponding link quality
 *
 * parameter accessPoint: mac address of the access point
 * parameter linkQuality: link quality associated with accessPoint
 **/
public void add (String accessPoint, int linkQuality)


/* Returns the link quality associated with the given access point
 *
 * parameter accessPoint: mac address of the access point
 *
 * returns: the link quality associated with the given access point
 **/
public int getLinkQuality (String accessPoint)


/* Returns the mac addresses of the access points contained in this object
public String[] getAllAccessPoints ()


/* Checks if the container is empty or not */
public boolean isEmpty()

/* Removes all the access point information contained in the object */
public void clear()
```

Figure C.4: BluetoothTrackData. This container class is responsible for holding Bluetooth signal data generated during run-time.

```
/* Allocates memory for a WavelanBaseStationLocation object and initializes it with
 * the given parameters
 *
 * parameter mAddress: mac address of the base station
 * parameter xVal: x-coordinate of the location where station is located
 * parameter yVal: y-coordinate of the location where station is located
 * parameter sName: administrator-defined name
 * parameter floorName: name of the floor on which station is located
 * parameter subfloorName: name of the part of a floor where station is located
 */
public WavelanBaseStationLocation (String mAddress, float xVal, float yVal,
                           String sName, String floorName, String subfloorName)

/* returns the mac address of the base station */
public String getStationMACAddress()

/* returns the x-coordinate of the location of the base station */
public float getX()

/* returns the y-coordinate of the location of the base station */
public float getY()

/* returns the administrator-defined name of the base station */
public String getStationName()

/* returns a string indicating the name of the floor on which the base
 * station is located
 */
public String getFloor()

/* returns a string indicating the part of the floor on which the base
 * station is located
 */
public String getSubFloor()
```

Figure C.5: WavelanBaseStationLocation. This container class is a container class that holds both symbolic and geometric location information for a base station.

```
/* Allocates memory for a WavelanTrackData object
 */
public WavelanTrackData()


/* Adds a base station and its corresponding signal strength
 *
 * parameter baseStation: mac address of the base station
 * parameter sl: signal strength associated with base station
 **/
public void add (String baseStation, int sl)


/* Returns the signal strength associated with the given base station
 *
 * parameter base station: mac address of the base station
 *
 * returns: the signal strength associated with the given base station
 **/
public int getSignalStrength (String baseStation)


/* Returns the mac addresses of the base stations contained in this object
public String[] getAllBaseStations ()


/* Checks if the container is empty or not */
public boolean isEmpty()

/* Removes all the base stations and its corresponding information contained
 * in the object
 */
public void clear()
```

Figure C.6: WavelanTrackData. This container class is responsible for holding 802.11 wireless LAN signal data generated during run-time.

Bibliography

[1] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. Implementing a sentient computing system. *IEEE Computer*, 34(8):50–56, August 2001.

[2] Aeronautics and National Research Council Space Engineering Board. *The Global Positioning System: A Shared National Asset.* National Academy Press, 1995.

[3] Paramvir Bahl and Venkata N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *IEEE Infocom 2000*, pages 775–784. Microsoft Research, 2000.

[4] T. E. Boult, R. Michaels, X. Gao, P. Lewis, C. Power, W. Yin, and A. Erkan. Frame-rate omnidirectional surveillance and tracking of camouflaged and occluded targets. In *Second IEEE Workshop on Visual Surveillance*, 1999.

[5] R. T. Collins, A. J. Lipton, and T. Kanade. A system for video surveillance and monitoring. In *American Nuclear Society Eighth International Topical Meeting on Robotics and Remote Systems*, 1999.

[6] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color and pattern detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 601–608, 1998.

[7] Andy Dornan. *The Essential Guide to Wireless Communications Applications, From Cellular Systems to WAP and M-Commerce.*

[8] I. Getting. The global positioning system. *IEEE Spectrum*, pages 36–47, December 1993.

[9] The Bluetooth Special Interest Group. The bluetooth specification. Technical report.

[10] I. Haritaoglu, D. Harwood, and L. Davis. Who? when? where? what? In *Third International Conference on Face and Gesture Recognition*, 1998.

[11] URL: http://content.techweb.com/wire/story/TWB20000907S0001. Ibm preparing wireless "sea change". Web resource.

[12] URL: http://grouper.ieee.org/groups/802/11/main.html#Tutorial. Tutorial of draft standard ieee 802.11. Web resource.

[13] URL: http://java.sun.com/products/jini. Jini homepage. Web resource.

[14] URL: http://stoenworks.com/Tutorials/UnderstandingVors.html. Vhf omnidirectional ranging (vor). Web resource.

[15] URL: http://www.bluetooth.com. The official bluetooth website. Web resource.

[16] URL: http://www.cooltown.hp.com. Cooltown. Web Resource.

[17] URL: http://www.fcc.gov/e911/. E911. Web resource.

[18] URL: http://www.garmin.com/aboutGPS/waas.html. Wide area augmentation system. Web resource.

[19] URL: http://www.locationforum.org. Location interoperatibility forum. Web resource.

[20] URL: http://www.mariner.org/age/lat_long. Longitude-latitude scale. Web resource.

[21] URL: http://www.wireless nets.com/whitepaper_overview_80211.htm. Overview of the ieee 802.11 standard. Web resource.

[22] V. Kettnaker and R. Zabih. Bayesian multi-camera surveillance. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 253–259, 1999.

[23] John Krumm, Steve Harris, Brian Meyers, Barry Brumitt, Michael Hale, and Steve Shafer. Multi-camera multi-person tracking for easy living. In *Third IEEE International Workshop on Visual Surveillance*. Microsoft Research, July 2000.

[24] J. Orwell, S. Massey, P. Remagnino, D. Greenhill, and G. A. Jones. A multi-agent framework for visual surveillance. In *International Conference on Image Analysis and Processing*, pages 1104–1107, 1999.

[25] T. S. Rapport S. Y. Seidel. 914 mhz path loss prediction model for indoor wireless communications in multi-floored buildings. In *IEEE Trans. on Antennas and Propagation*, February 1992.

[26] Roy Want, Andy Hopper, Veronica Falcao, and Jon Gibbons. The active badge location system. In *ACM Transactions on Information Systems*, volume 10, pages 91–102. Olivetti Research Ltd, January 1992.