## Abstract

With the widespread use of smartphones and other mobile networking devices, context-aware applications have become part of our lives. Nowadays, location based services have attracted much attention and, considering that the only accurate location system built into smartphones is GPS which does not function indoors, a suitable indoor positioning system is becoming a necessity.

The purpose of this project is to build an indoor positioning system which uses wireless signal strength to track the location a wireless device. The system should have the following characteristics: easy to install and use by non IT savvy people and offer a very good balance between increased accuracy and number of sensors used.

The system was built using 8 WRT-54GL Linksys routers which were re-flashed to run a custom firmware called DD-WRT. The routers were configure to run in WDS mode which enabled the positioning application to gather wireless signal values from the routers. Three positioning algorithms were explored: Lateration, MinMax and ROCRSSI.

Small and large scale tests were conducted to verify how well the system behaves. Data resulted from small scale tests showed that signal strength varies over time and that collecting a certain number of RSSI readings from the routers leads to a better distance estimation. Experimental data is presented to show that a greater number of routers used as well as a strategic positioning yield better results. The large scale tests provided insight into what it would take to get a good accuracy in a very complex indoor environment.

The achievements and contributions to this project are as follows:

- A highly flexible positioning system was built (see Chapter 3).
- The system does not use time consuming training phases nor expensive, specialised equipment.
- Own versions of the positioning algorithms were implemented.
- A very high accuracy was achieved: 2.47 metres on the X axis and 2.29 metres on the Y axis.

## Acknowledgements

I would like to express my gratitude to my supervisor Kirsten Cater for her support throughout the duration of the project. I am also grateful for always welcoming me with a big smile every time we met to discuss the project at 10 in the morning when people should be counting sheep in their sleep.

Special thanks to Tom Melamed for all the brainstorming sessions which resulted in getting past some serious obstacles encountered while doing this project.

I am forever in debt to my parents, Alin and Elena, for the countless sacrifices they have made to raise me and to offer me a good life and education.

I would also like to thank my girlfriend, Dana, for her endless support and love, not to mention her constantly nagging me to finish writing my dissertation.

Very special thanks to sharkall, loctarar, y4nk and tibos for their efforts to pull me away from writing this dissertation and play a midnight game.

Lastly, I would like to express my gratitude for the Computer Science Department staff for sharing their knowledge during lectures.

.

# Table of Contents


1 Introduction ..... 3

1.1 Indoor positioning system ..... 3

1.2 Motivation ..... 3

1.3 Objectives ..... 4

1.4 Dissertation overview ..... 5

2 Background and related work ..... 6

2.1 Received signal strength ..... 6

2.1.1 Overview ..... 6

2.1.2 Using the RSSI ..... 6

2.1.3 Reliability ..... 7

2.2 Sensor nodes ..... 9

2.2.1 Wireless routers ..... 10

2.2.2 Packet sniffers ..... 11

2.3 RSSI based techniques for location-tracking ..... 13

2.3.1 Range-based localization ..... 13

2.3.2 Range-free localization ..... 17

2.4 Previous work on IPS using RSSI ..... 18

2.4.1 Experimenting with IPS ..... 18

2.4.2 IPS improvements ..... 23

3 Building the indoor positioning system ..... 24

3.1 Installing DD-WRT ..... 24

3.2 Interconnecting the routers ..... 25

3.2.1 Wireless bridge ..... 25

3.2.2 Wireless Distribution System ..... 26

3.2.3 Router configuration ..... 26

3.3 Connectivity testing ..... 28

4 Software design ..... 29

4.1 Initial ideas and obstacles ..... 29

4.2 Software implementation ..... 32
4.2.1 Application design ..... 32
4.2.2 Set-up module ..... 35
4.2.3 Data collector ..... 37
4.2.4 Positioning ..... 39
4.2.5 GUI ..... 41
5 Testing and analysis ..... 43
5.1 Methodology ..... 43
5.2 Small scale testing ..... 44
5.3 Large scale testing ..... 48
6 Conclusions ..... 52
6.1 Objectives and results ..... 52
6.2 Future work ..... 53
7 Bibliography ..... 55
Appendix A – Merchant Venturers Building floor plan ..... 58
Appendix B – getData() method in Collector class ..... 59
Appendix C – getData() method in Collector2 class ..... 61

# 1 Introduction

## 1.1 Indoor positioning system

An indoor positioning system (IPS) is either a device or a network of devices used to wirelessly sense the positions of people or objects inside a building.

Instead of using satellites (GPS), IPS relies on sensors which spread throughout the building and have a known position. These sensors, also called anchors, provide environmental information which is used to sensing. Due to its localized nature, several technologies are being used to create an IPS: ultrasonic, radio frequency and infrared. These can also be enhanced by using inertial systems such as an accelerometer which are generally used to predict future positions.

## 1.2 Motivation

The globally pervasive computing environments of the future will emerge a series of completely new challenges but will also bring numerous and unprecedented opportunities. In environments such as these, people will carry several small devices which will communicate at all times with other similar omnipresent nodes in the environment. Knowing where the user is (location sensing) is probably one of the most important aspects in ubiquitous computing and has attracted a huge amount of attention from researchers over the last few years. Location-tracking of objects and even people both indoor and in outdoor environments has made considerable progress recently. Methods for tracking user position such as Cricket [18], Mote Track [12] or GPS [20] have been well researched and documented and have generate great success. In response, the benefit of these new mobile device technologies has been leveraged by Location Based Services (LBS). These are applications that, depending on the user's position in space, change their behaviour to adapt to the location or give location-dependent information to the user.

Since its appearance, GPS has been the predominant positioning system for outdoor location-tracking. Although it can be successfully used in many scenarios, the biggest drawback of the GPS is that it is subject to interference from large obstacles such as trees and buildings which create a shield that blocks radio signals. Therefore, accurate positioning in a dense urban area with numerous large buildings such as skyscrapers is impossible. Needless to say, indoor location-sensing is a feature that the GPS will never have.

The other technologies currently used in IPS (ultra-sonic, infrared, RF) have also proved inefficient for location sensing. Ultra-sonic systems are vulnerable to signal interference due to obstruction, fading effects and high-frequency sounds. Infrared requires explicit actions to be taken by the users in order to track their locations and RF often requires sophisticated and expensive equipment.

A solution to indoor location tracking may come from the indoor equivalent of the GPS, the 802.11-based positioning systems. Recently, 802.11 compliant hardware has been installed almost everywhere people work and live. All universities, shopping malls, libraries, airports and similar locations now have Wi-Fi hotspots where users can connect and access the internet.

Another important aspect is that most modern communication devices (PDAs, laptops, portable gaming consoles, smartphones) have wireless capabilities. Using Wi-Fi access points as anchored sensor nodes(similar to the satellites for the GPS) it is possible to determine the position of a person using a hand-held Wi-Fi enabled device connected to the wireless network in an indoor environment.

Such a system has a large area of applicability. It could be used to track the movements of firemen inside a burning building or patients in hospitals. It could also be used to bring information on the screen of a phone about a painting the user is standing in front of in an art gallery or give people a thrilling adventure sending messages to their mobile devices and directing them through the route used by Frank Morris to escape from Alcatraz.

## 1.3 Objectives

In this project, the following objectives have been set:

- Build a WiFi signal strength based indoor positioning system using off-the-shelf wireless routers which are relatively cheap.
- Make the system easy to use by people who do not have extensive IT knowledge.
- Minimize the set-up costs and time of the system. It should not require hiring a specialised person to install the system nor having the client to spend a lot of time on doing preliminary measurements.
- Implement range-based and range-free algorithms and determine which give the best results.

- Test in a complex environment and experiment with different router placements in order to determine if the system can be deployed in any building regardless of its geometry.
- Achieving an overall accuracy of less than 7 metres on all dimensional axes.

## 1.4 Dissertation overview

The structure of the project comes from the objectives listed above and the dissertation is organised in chapters as follows:

**Chapter 2** – The next chapter describes the concepts and technologies needed to build an indoor positioning system based on wireless signal strength. It covers the characteristics of signal strength, the hardware and software used to gather data as well as the positioning techniques used in location sensing. The chapter ends with a brief mention of previous work in this field.

**Chapter 3** – A thorough description of how the hardware part of the IPS was built. It explains what obstacles were encountered at this stage and what decisions were made to overcome them.

**Chapter 4** – The software component of the system and its modules is presented together with initial design ideas and flaws that lead to changes. The implementations of the positioning algorithms are presented together with an analysis on which algorithm is better.

**Chapter 5** – The system is tested and observations are made on how the accuracy is affected by changing parameters such as number of data samples collected from each router or number of routers used. Several router placements are explored in order to determine the ones that yield the best results.

**Chapter 6** – The conclusion of the dissertation where the progress made during the project is summarised and assessed.

# 2 Background and related work

## 2.1 Received signal strength

### 2.1.1 Overview

The Received Signal Strength Indicator (RSSI) is basically a circuit built to measure the strength of an incoming signal. The receiver picks up radio signals and then generates and output which has the exact same strength. Devices which come with a built in RSSI are: cell phones, wireless network adapters and even remote controls. Considering that IEEE 802.11 uses RSS for collision avoidance, the radio signal strength is considered by most a very appealing way for range determination in wireless networks mainly because the RSS value can be easily obtained by inspecting the messages sent and received by the network nodes[4,21].

The main drawback when dealing with RSS is that radio signals are incredibly unpredictable. Having a complex indoor environment with lots of rooms and obstacles, both stationary and mobile, means that the radio signals can be reflected and/or attenuated. This makes it difficult to get accurate distance values from the RSS readings without a detailed model of the environment in which the wireless network is set-up. Therefore, the literature points out that two methods of RSS-based positioning are preferred: estimating distance directly from the RSS values [21] and fingerprint matching [19]. Both these methods will be covered in the following chapters.

### 2.1.2 Using the RSSI

The signal strength values can be given in four units of measurement: miliwatts (mW), db-milliwatts (dBm), Receive Signal Strength Indicator (RSSI) and a percentage. The units in which RSS is measured depend on the equipment used. Converting from one unit to another is possible with varying degrees of accuracy.

#### 2.1.2.1 The mW and dBm units of measure

Energy is measured in mW. Therefore, a mW measured signal will indicate the amount of energy used at that time. However, measuring signal strength in this manner is not always convenient mainly because the signal strength is not a linear function. It is inversely proportional to the square of the distance travelled by the signal from the emitting node to the receiver.

Therefore, a logarithmic scale for measuring signal strength was developed as an alternative.

The dBm is a logarithmical measuring unit for signal strength. These values can be easily converted to and from mW values. As indicated in [3] , the formula used for conversion is:

$$dBm = log(mW) * 10$$

#### 2.1.2.2 The Received Signal Strength Indicator

The IEEE 802.11 sets the standard for signal strength measuring. The RSSI is a 1 byte numeric value which can exist in the range of 0-255. No vendors choose to actually measure 256 different signal levels [3]. Therefore, each piece of equipment which has a wireless network card will have a specific maximum RSSI value. Although the RSSI parameter is specified as being optional in the IEEE 802.11 RFC, all network card manufacturers implement it. Also, the RSSI reading doesn't have a specified accuracy, without there being a direct relationship between the RSSI value and the energy level as in the case of mW and dBm measurements. Therefore, each vendor provides its own interpretation of the RSSI value and conversion to and from dBm and mW.

#### 2.1.2.3 RSS as percentage

It is very common to see the signal strength displayed in percentage form. This is done to overcome the complexities of using RSSI as the standard unit of measure. The percentage value is calculated by dividing the RSSI value of a packet by the default maximum RSSI value of the equipment used and then multiplied by 100.

Using percentage for signal strength can provide an easy-to-understand metric for all users. Detecting a signal strength of over 80% indicates that the connection is strong and stable. On the other side, having a signal strength of 30% or lower shows that the connection "is bad". This method relieves the user of the burden of knowing what a signal strength of 15dBm is and what impact it has on the wireless connection.

### 2.1.3 Reliability

Even though signal strength is widely used as a means of determining location in a wireless network, the study of the literature shows that RSSI is not a suitable for location-sensing. Parameswaran et. al. [15] presents a study about the reliability of RSSI as a parameter in sensor localization algorithms. The

study describes practical experiments which were created to determine if RSSI could be used to determine distance between wireless sensors.

The experiments conducted used a development kit called Crossbow Imote2 from Intel which integrates a 802.15.4 radio which runs in the 2.4GHz range, the same as 802.11. In order to ensure that no factors could interfere with the experiments, several precautions were taken: the surface on which the experiments took place had to be level, there had to be no obstacles between the communicating sensors and all other equipment that could cause interference was not present in the experiment area.

The results of the experiments show in fig. 2.1 that the measured RSSI values follow a linear relationship. It can also be inferred that as the distance between nodes increases, the error in RSSI measurement increases which results in a decrease of reliability at the limits of transmission range. And, as stated above, RSSI is an integer value, therefore it cannot have decimals. Upon conversion to dBm, the resulting value does not offer enough accuracy to distinguish small changes in distances. However, it proves to be enough to determine distances with an acceptable margin of error.

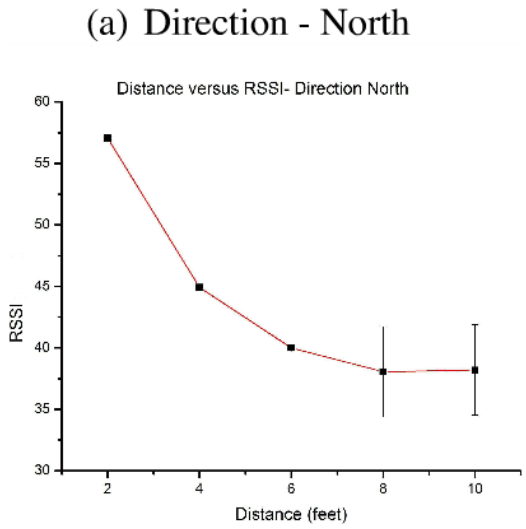



**Fig. 2.1 – RSSI values in relation to distance (source: [15])**

Further experimenting revealed that changing the direction of the sensors had a negative impact on the results. The following two graphs show that very different RSSI values were obtained by varying direction. At times, the data showed the inverse square relation between the distance and RSSI value and sometimes it didn't which led to the conclusion that using signal strength to determine distance is not a reliable method.

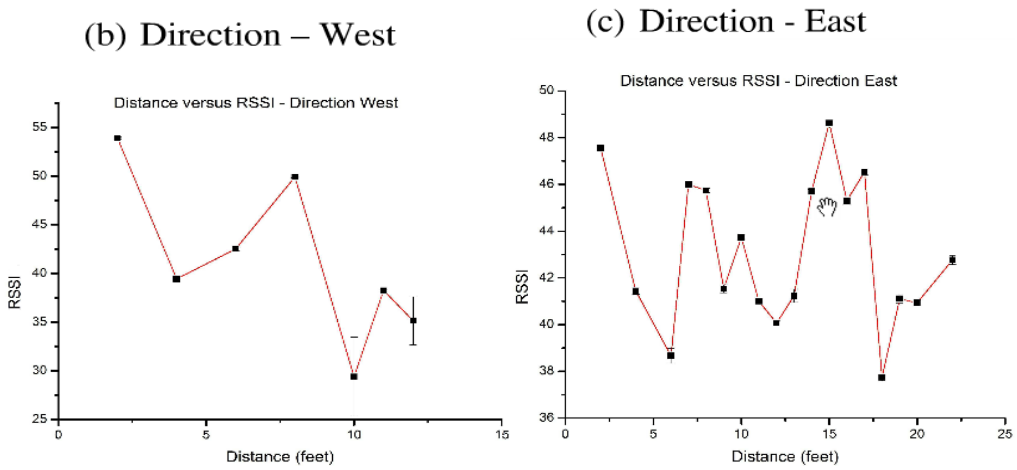



**Fig. 2.2 – RSS values in relation to distance (source: [15])**

These experiments clearly show that in a near-perfect environment, with a flat surface, no obstacles that can cause attenuation of the signals and no other electronic devices that can interfere with the sensors, RSSI cannot be used as a metric to determine distance between wireless sensor nodes. However, most research papers on the subject of indoor location-tracking present RSSI as the main parameter in the algorithms used. The reason is quite simple: RSSI is easy to measure and use. Improving the accuracy of RSSI measurements can be done by creating a virtual map of RSSI readings in the testing environment using a method known as RF profiling [22]. The RSSI values recorded at each position in the area are collected from all anchored nodes. At a later point, a node with an undetermined location can be localized by matching its RSSI value to the RSSI values recorded on the map. This method will be discussed in detail in a following chapter.

## 2.2 Sensor nodes

Having discussed about what kind of metric is relevant to the project, it is time to present the equipment used to record the signal strength values. All wireless devices have four things in common:

- power supply
- sensor which measures something (relevant to us is the signal strength)
- transmitter/receiver which enables communication between devices
- internal CPU

Pertinent to our project are mobile devices (PDAs, smartphones, laptops) and wireless routers and access-points (APs). The routers and APs have the role of anchored nodes. They are responsible for recording the signal strength coming from mobile nodes. Then, all the measured data is sent to a central server which does the computations and reveals the location of the mobile devices.

### 2.2.1 Wireless routers

For this project, Linksys wireless routers will be used in the role of anchored nodes. They are relatively cheap and can run the DD-WRT operating system by Cisco. All routers are able to convert the quantity of interest (RSSI in our case) into an electric signal that can be processed. The power consumption of the router is directly related to its hardware specifications.


**Fig. 2.3 – Linksys wireless router (source: www.cisco.com)**

The measured signal strength values must be processed at a central point in the network which usually is the server. Therefore, specialized software is needed to record and send data to the server. As mentioned above, DD-WRT can be installed on these routers. This enables the use of several pieces of software that can record and send information (including RSSI) about messages sent between nodes in a wireless network. These programmes are called packet sniffers and will be discussed in the following section of the chapter.

### 2.2.2 Packet sniffers

#### 2.2.2.1 Overview

A packet sniffer, also known as a network analyser, is a computer programme or a piece of hardware that has the ability to intercept and log traffic passing through a network. The sniffer can capture a packet and decode it in order to reveal the values of different fields in that packet. Packet sniffers can be used for a large array of tasks: detecting network intrusion attempts, analysing network problems, isolating exploited systems, filtering out suspect content from network traffic, etc.

The research done shows two programmes which are best suited for the project. Both are open source which ensures that they can be altered to meet certain requirements and both are compatible with the DD-WRT operating system.

#### 2.2.2.2 Wi-viz

Wi-viz is an open source packet sniffer which runs on Linksys routers using the DD-WRT operating system. Wi-viz places the router in a special monitoring mode which, apparently, does not interfere with the device's normal routing functions. This programme is made out of two parts:

- Back-end packet analyser
- Front-end DHTML Javascript display

The packet analyser works by first putting the router into the monitoring state. All packets which are meant for the router are processed normally. Other packets which should be dropped are tagged and sent to a virtual network interface. The programme uses the libpcap library to capture these packets and then runs an analysis on them, inspecting only the relevant fields such as MAC source/destination addresses and SSIDs.

The front-end display is somewhat simple and does not have many uses. The display shows the network topology and updates itself at a regular time interval.

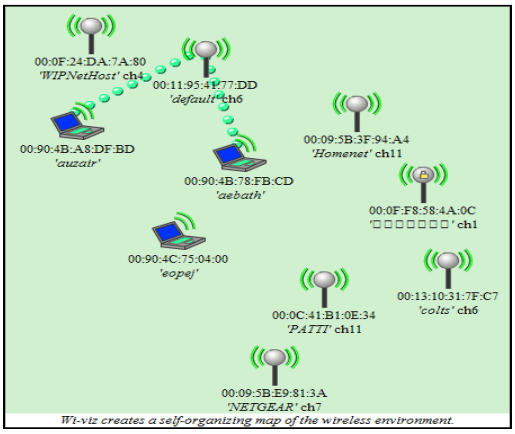



**Fig. 2.4 – GUI for Wi-viz (source: http://devices.natetrue.com/wiviz/)**

In order to view the RSSI values which are collected from the sniffed packets, the mouse pointer must be positioned on top of an end-device (laptop, phone).

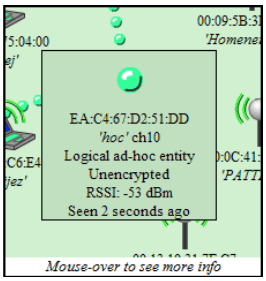



**Fig. 2.5 – RSSI indication in Wi-viz (source: http://devices.natetrue.com/wiviz/)**

Although Wi-viz is very easy to use, the manner in which it displays the information may lead to possible alterations of the software.

### 2.2.2.3 Kismet

Kismet is an open-source wireless network detector, sniffer and intrusion detection system. It can be used on any router running the DD-WRT operating system. It identifies networks by passively collecting packets and analysing them. Kismet can either work in server mode or drone mode. Drones support all of the capture methods that Kismet normally supports. The drones capture wireless data and forward it to a Kismet server for the decoding of the packets and data analysis.

## 2.3 RSSI based techniques for location-tracking

### 2.3.1 Range-based localization

Range-based localization requires the anchored nodes(wireless routers, access points) to determine the distance to connected mobile nodes. Systems that use this technique include: RADAR [1], PinPoint [23], SpinLoc [2]. The distance between the anchored nodes and mobile nodes can be determined in two possible ways: direct calculation based on signal strength or by fingerprint matching.

#### 2.3.1.1 Calculating distance based on RSSI measurements

Being an innovation in the field of RSSI-based location-tracking, SpotOn [6] shows that determining the position of mobile sensor nodes is possible by using measured signal strength values. In [15], the RSS is seen as a function of the distance (d) between the anchored node and the mobile node:

$$RSS(D) = 0.0236 * d^2 - 0.629 * d + 4.871$$

Due to its early emergence, SpotOn has several drawbacks: depends on a large number of anchored nodes which has a negative impact on cost-effectiveness, it is highly susceptible to errors caused by signals being reflected and attenuated. In [15,16], a new equation for signal strength is presented. Unlike the previous one which is deterministic in nature, [17] describes signal propagation using a statistic model:

$$\tilde{P}(d) = \pi_0 - 10 * n_P * log_{10}({}^{d}/_{\Delta_0})$$

where $\tilde{P}$ is the RSS, $d$ is the distance between nodes, $n_P$ is the attenuation factor which is usually a value between 2 and 6 and $\pi_0$ is the received power measured in dBm at a reference distance of $\Delta_0$. Based on this equation, Patwari et al [17] come up with a distance estimator:

$$\delta_{i,j}^{BC} = \frac{\Delta_0}{C} * 10^{\frac{n_0 - P_{i,j}}{10 - n_P}} \quad , \qquad C = \exp\left(\frac{1}{2 * \left(\frac{10 - n_P}{\sigma_{dB} * log10}\right)^2}\right)$$

where $P_{i,j}$ is the measured RSS with zero mean Gaussian noise of variance ${\sigma_{dB}}^2$.

Furthermore, SISR [9] proposes a similar method of calculating RSSI as a function of distance between network nodes. However, just as before, equipment calibration is required in order to be able to correctly convert

signal strength values to distance. Given an unknown attenuation factor *a* and bias *b* , the equation is:

$$RSSI(d) = 10 * log_{10}(d^a) + b$$

#### 2.3.1.2 RSS fingerprint matching

Seeing as how trying to determine the distance between wireless nodes using just the RSSI measured values is quite inaccurate, most range-based location-tracking systems use a method called profiling. This involves creating a map of RSSI values throughout the entire wireless network area. RSSI values are recorded for each anchored node in the network. All the measured values are stored in tables. The record for a particular position in the environment is called the RF fingerprint of that position. From this point on, determining the position of a node is reduced to matching its RF fingerprint to the recorded ones. This method is also prone to give false results. Radio signals are very unpredictable. It is highly likely that a location's RF fingerprint will be different depending on the time of the day, changes in room temperature and humidity, mobile obstacles and equipment failure(one or more anchored nodes may fail).

RADAR [1] uses this technique quite successfully. The location is determined by using the *k*-nearest neighbour training based algorithm. Given the mobile node's RSSI values read by the anchored nodes, the system searches the data set to find the nearest *k* matching records. Proximity is defined in Euclidian distance and is calculated using the equation:

$$D_j = \sqrt{\sum_{i=1}^{N} \left(RSS_i^j - RSS'_i\right)^2}$$

where $RSS_i^j$ is the signal profile for position *j* on the map and $RSS'_i$ is the value measured from the anchored nodes. Using this method, RADAR obtained a good performance with an error of just 2-3 metres.

This technique is costly to implement mainly because the map must be reconstructed each time there is a change in network topology or a change in the environment.

#### 2.3.1.3 Range-based positioning algorithms

Having calculated the approximate distance between the mobile node and each of the anchored nodes, the actual position in space can be

determined using either one of two basic algorithms: lateration (better known as trilateration) and min-max.

#### 2.3.1.4 Lateration

The principle behind lateration is quite simple. Knowing the distances between a target node and 2 anchored nodes will determine the target node's 2D location. Adding an extra anchored node, thus having 3 reference points, will result in determining the 3D location of a target node.

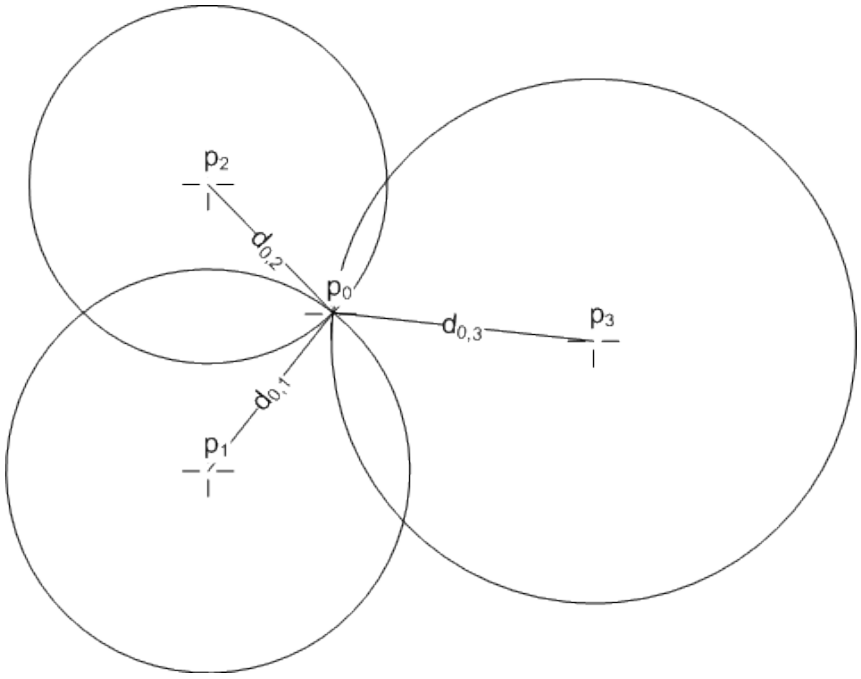



**Fig. 2.6 – Example of lateration**

In the figure above, $p_1$, $p_2$ and $p_3$ are anchored nodes. $p_0$ is a mobile node whose location is unknown and $d_{0,1}$, $d_{0,2}$ and $d_{0,3}$ are the determined distances between $p_0$ and $p_1$, $p_2$ and $p_3$ respectively. Then, for each anchor node, a circle is drawn around it having the radius equal to the distance from it to the mobile node. Thus, each anchor node may assume that the mobile device can be found somewhere along the circumference of its respective circle. By overlapping the circles, the position of the mobile node can be determined with a reasonable margin of error. In the best case scenario, all three circles intersect in exactly one point but, in general, the overlapping will result in a region rather than one point. Of course, the accuracy can be improved by adding more anchor nodes. For each node added, the resulting region will get smaller and smaller.

Although it may be simple and straightforward, this algorithm has several drawbacks. In ideal conditions, the lateration method gives great results, however, in the presence of signal attenuation and shadowing, this method may not give any results. It is possible that the errors in distance estimation may cause the circles to not overlap. A solution would be to add more reference nodes therefore increasing the implementation cost.

#### 2.3.1.5 MinMax

In the Min-Max algorithm, each anchor node uses the calculated distance $d$ to the mobile node to draw a square of width equal $2d$ around itself. The mobile node should then be found in the overlapping area of all the squares drawn around all the anchor nodes. The output of the result should be the centre of the overlapping area. Based on a simple idea, this algorithm proves to be quite robust in the presence of noisy channels.

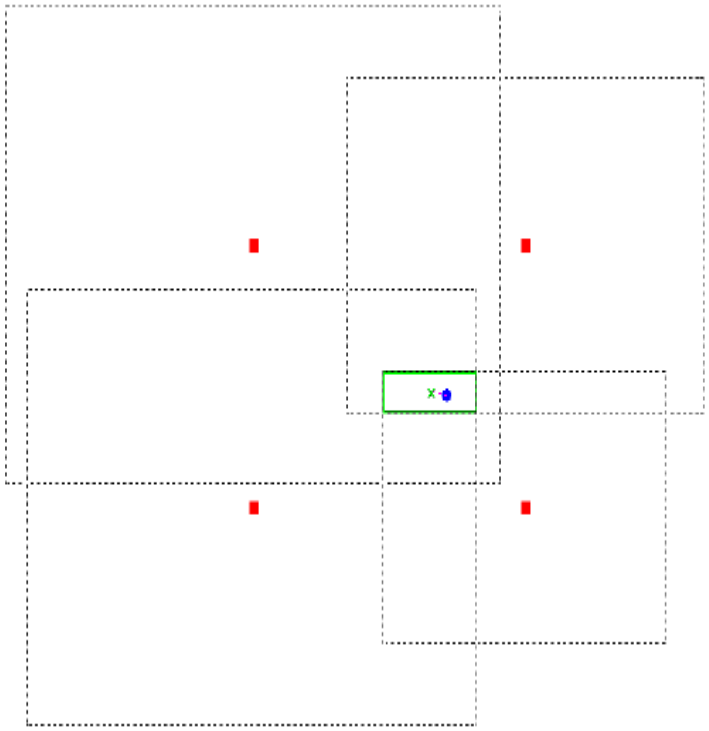


**Fig. 2.7 – Min-max example**

The figure above clearly shows how the min-max algorithm works. Similar to the lateration method, the mobile node can be found in the area resulted from overlapping the squares (highlighted in green). Furthermore, the positioning accuracy can be increased by adding more anchor nodes.

The min-max algorithm is relatively easy to implement. However, its results may lack in accuracy because it uses a bounding box instead of a circle to determine the position.

### 2.3.2 Range-free localization

Range-free localization methods do not use costly distance calculations nor do they assume there is any prior knowledge of RSSI measurements and values. According to [15], early range-free solutions made use of the proximity information. Later on, investigations were made concerning connectivity-based localization. Recently, studies describe event-driven localization methods in which artificially generated events are used to track location by analysing event detections. However, there is only one method which uses RSSI as a parameter in range-free localization.

The Ring Overlapping Circles RSSI (ROCRSSI) algorithm [11] uses the concept of signal loss to estimate a target's location. In order for this method to be used, all anchor nodes in a wireless network must have known locations. Each of these nodes reads not only the RSSI value from its peers but also from the target node whose location is unknown. As explained in [13], the RSSI values recorded by each anchor node will be split into two sets. One set will contain the values which are lower than the RSSI value received from the target node and the other set will hold the remaining values which are greater.

This algorithm also treats the RSSI value as a function of distance. Therefore, by sorting the two sets of signal strength values, two numbers will stand out. From the first set, the greatest RSSI value will be closest to the RSSI measured from the target, and from the second set, the lowest value will be the one closest to the target's RSSI. These two values are used to draw circles around the anchor nodes, similar to the lateration method. The figure below illustrates this.

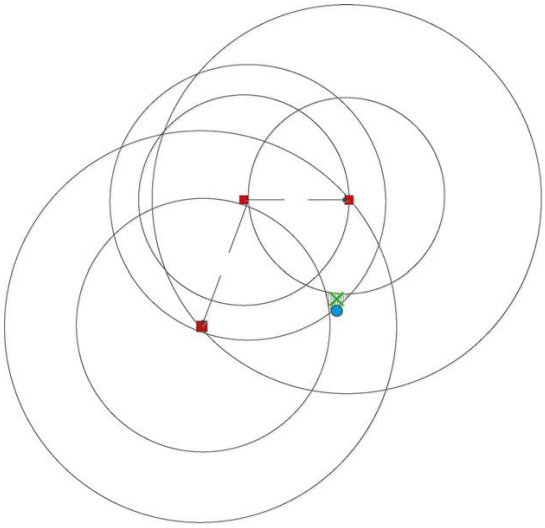


**Fig. 2.8 – ROCRSSI example (source: [13])**

The rings drawn for every anchor node define a lower and upper bound for locating the mobile node. Overlapping the areas between each set of two rings will result in a smaller region in which the target node may be. The output of the algorithm is the centre of that region.

The beauty of this algorithm comes from the fact that it is range-free. It does not require doing costly computations to determine distances between the nodes. Therefore it is well suited for wireless routers and sensors which have very limited computational power.

## 2.4 Previous work on IPS using RSSI

### 2.4.1 Experimenting with IPS

EVENNOU et al [5] present their experiments with indoor location-tracking using wireless networks and signal strength indicators. They propose the use of a Kalman filter and a particle filter in order to handle the variations of the signal strength measurements. The aim of their study is to obtain correct trajectories without wall-crossings. Their investigations revealed that simply using distance calculation and lateration algorithms did not provide a good positioning of the target node.

In order to get better results, EVENNOU et al propose using two range-based positioning methods: RSSI distance calculation and fingerprinting. For the distance estimation they used the Motley-Keenan model [14]. For the RF fingerprint matching they used the *k*-nearest neighbour algorithm. However,

their method also has limitations. The fluctuations in the RSS values that appear over time have an impact on both distance estimation and fingerprint matching. They explain how, in largely populated database of RSS values, the *k*-nearest algorithm may give different results even if the target node is not changing its position.

In order to overcome the signal fluctuations, a Kalman filter is used for distance estimation and a particle filter for the RF fingerprint database. Their proposed particle filter is rather complex, and comprises of several steps: prediction, correction, particle update and re-sampling. Therefore, this particle filter is inherently costly to implement. The study concludes by stating the overall success of using the particle filter combined with fingerprint matching. This method led to a position accuracy of 1.8m.

King et al [8] also study the use of the fingerprinting method based on signal strength signatures measured throughout the test environment. Their research, however, is done in a much more controlled environment. Testing was done in an area of approximately 312 square metres. Reference RSS samples were measured in 612 points evenly arranged in a grid. During the set-up stage, 110 RSS values were collected at each of the reference points. There were 83 target nodes for which the position had to be calculated. For each of these nodes, 110 RSS measurements were conducted.

While running the experiments, several parameters were under investigation:

- Number of access points – between 1 and 14
- Number of training set samples – varied from 1 to 110
- Grid spacing – varied from 0.5m to 4m in 0.5m increments
- Number of online samples – varied from 1 to 110

As expected, increasing the number of access points decreased the positioning error. For example, the error drops from 7.2 meters using one access point to 4.8 meters using two access points. Further adding access points decreased the error by less and less amounts. This experiment is of particular interest to this research because it shows there is a clear relationship between number of anchor nodes and position accuracy.

Varying the number of training set samples also proved to have a direct effect on the position accuracy. The difference between using five samples instead of on is a decrease in error by almost 2 meters. The decrease is consistent until reaching 20 samples. At that point, the error decrease can be

expressed in centimetres. As a conclusion, using more than 20 training samples is not optimal. The exact conclusion can be drawn from varying the number of online samples. The error using 20 samples is 2.4 metres whereas using 110 samples results in an error of 2.37 metres. In this respect, the authors suggest that a trade-off must be made. Using less online samples may decrease accuracy but will provide quicker updates.

Varying the grid spacing showed that increasing the distance between the reference points also increased the positioning error. A grid spacing of 0.5 metres resulted in an error of 2.33 metres. Increasing the spacing to 4 metres also increased the error to 2.97 metres.

The study results show that 20 training samples at each reference point are enough to create a usable database for fingerprint matching. Also, for determining the position, at least 3 online samples must be taken. Furthermore, a spacing between 1 and 2.5 metres is suggested, depending on how much time can be invested in setting up the system and creating the training sample sets. King et al showed through their experiments that an error less than 2 metres could not be achieved.

A more daring and innovative research study is [24]. Zaruba et al try to determine indoor location using RSSI readings from a single Wi-Fi access point. All other studies showed that at least 3 access points are needed in order to get a relative location. The method behind this experiment is the creation of RSS maps. These maps are created using a two-step parametric and measurement driven ray-tracing approach which accounts for absorption and reflection characteristics of obstacles found in the testing environment. The study describes that RSSI based measuring techniques can be divided into deterministic and probabilistic techniques. The deterministic ones need the area to be subdivided into smaller cells. Then, RSSI samples are measured in each cell from several known anchored nodes which constitutes the training phase. The probabilistic positioning techniques define a probability distribution of the user's location.

The experiment conducted by Zaruba et al is set-up in two stages. The first stage represents the creation of the virtual map of RSS signatures measured throughout the testing environment. The floor plan of the testing area is defined as seen in the image below. The next step in this

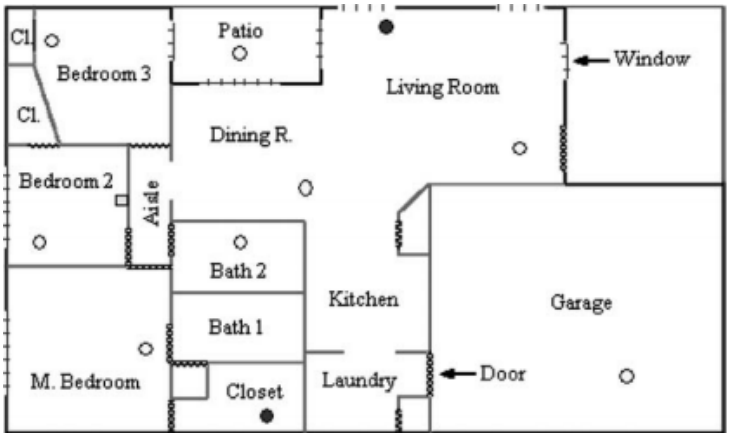



**Fig. 2.9 – Floor plan (source: [24])**

stage is to set measurement points. These can be seen as discs scattered around the floor plan in the image above. The number of measurement points is related to the number of obstacles found in the environment. In this case, 10 points were used. As seen in previous studies, precision increases with the number of measurement points. After doing several measurements in each point, a parametric ray-tracing algorithm is used to express the RSS values measured in all reference points as a function of transmission and reflection parameters of the obstacles. The algorithm works in a similar way to the classic computer graphics ray-tracing algorithm. The antenna of the access point sends out radio signals in all directions. Each signal is seen as a ray. Once the ray hits an obstacle, it separates into two separate rays: one that is reflected back and one that gets passed the obstacle. Both the newly created rays inherit properties from the parent. When a ray crosses the area of a measurement point its current parameters such as distance travelled are stored for that respective point. The final step is the obstacle parameter determination. At this point, a similar ray-tracing is done. The difference is that rays leave a scalar power level footprint in all areas they travel to, thus populating the entire map with RSS values. A complete wireless signal map can be seen in the figure below.

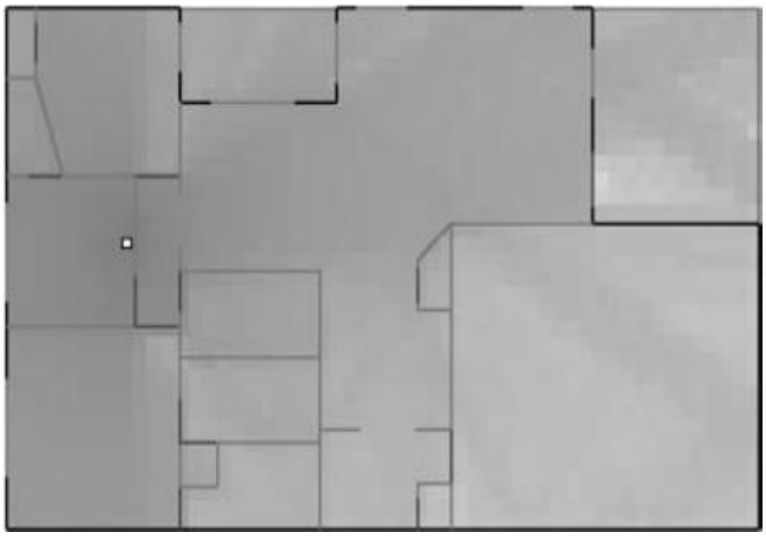


**Fig. 2.10 – Wireless Signal Map (source: [24])**

The second stage of the experiment consists of using Monte Carlo sampling-based Bayesian-filtering to accurately determine location in the newly created wireless map. The authors introduce four estimation models that have been implemented and tested.

The simple sequential Monte Carlo filter is the simplest localization algorithm used in this study. It assumes that at any point in time, the target node moves with a random velocity. This model does not use any information about the environment. Simple sequential Monte Carlo filter with boundary information uses information about the location of walls to limit available choices to velocity that do not lead to collisions with permanent obstacles. Sequential Monte Carlo filter with internal motion model assumes that the mobile nodes move at a constant velocity. Therefore, the model relies on previous velocity with current random acceleration. The sequential Monte Carlo filter with internal motion and rotation model, unlike the previous ones, does not assume that the mobile node is facing the access-point.

The results of the experiment are quite surprising. Testing showed that the filters described above were successful at estimating the mobile node's location despite the ambiguity of RSSI data collected from a single Wi-Fi access-point. The study demonstrates that an average precision of around 1 metre can be easily obtained.

### 2.4.2 IPS improvements

Parthornratt et al [16] indicate that improvements to the accuracy of Wi-Fi positioning systems can be done by using Geographical Information System (GIS). Their methods take into account measurements of paths and obstacles and together with a Wall Attenuation Factor model provide a very accurate positioning system. Testing revealed an error of 8.95 metres. However, after applying the geographical information of the testing environment, the error decreased to 5.05 metres. Although the accuracy is not as good as in other studies, the conclusion is that coupling geographical information with other mathematical models will improve indoor positioning system accuracy.

In another study [10], Li et al present their own version of a RSSI quantization based algorithm (SRangeQ) for indoor localization. Their approach to a RSSI based positioning algorithm can be categorised as indirect mapping. All other studies presented in this report use direct mapping methods. The proposed algorithm uses RSSI to obtain estimated distances and then the mapping of signals is done by approximation using a quantization model. Testing the effectiveness of the SRangeQ algorithm was done using MATLAB simulations. These showed that SRangeQ is suitable for range estimation and localization. The performances of SRangeQ are summarized below:

- It is insensitive to RSSI fluctuations when estimating range
- It is 10% to 50% more accurate than a direct RSSI quantization method
- It is a distributed algorithm and can be used together with other connectivity-based localization algorithms

# 3 Building the indoor positioning system

## 3.1 Installing DD-WRT

One of the most important parts in the project is represented by re-flashing the routers which ran stock Linksys firmware. This process is not without hazards because even the slightest mistake can cause the routers to "brick" rendering them useless. After a thorough research and careful reading of the DD-WRT forums, several steps for a safe and correct firmware installing were identified:

- Check whether the router is DD-WRT compatible
- Find out the Flash memory size in order to select the proper DD-WRT build to be installed
- Learn how to do a "hard reset" and a "power cycle"
- Install DD-WRT and test the router

Having taken all these into consideration, re-flashing became a pretty straight forward process.

1. The WRT54GL routers used in the project had 4MB single chip NAND Flash memory. This prompted choosing the NEWD_MINI Generic 12548 Eko build which came with WiViz2 already installed.
2. Perform a "hard reset". This involves holding down the RESET button for 30 seconds, unplugging the router for 30 seconds while still pressing the RESET button and finally plugging the router back in and pushing the RESET button for another 30 seconds. After the reset, allow the router up to 1 minute to get back into the ready state which is usually signalled by the Wireless LED turning on.
3. Power cycle the router by unplugging and plugging it back in. Again, about 1 minute is needed for the router to get into the ready state.
4. Opening the Web interface, load the DD-WRT build and press the install button. This process lasts about 3 minutes and during this time the web interface must remain open and connectivity between router and PC must be kept alive. This is the most critical stage in the install process and any sort of interruption will lead to a bricked router.
5. Test the new build by opening the web interface after the router has reached the ready state.

6. Power cycle.
7. Hard reset.

## 3.2 Interconnecting the routers

Another important stage in the project was figuring out how to get the routers connected in the same network. The obvious solution would be to use Ethernet cables and link the routers to one another. However, this is impractical and expensive considering that the devices can be spread across large areas, the distance between them is at least 10 meters and that the purpose of the project is to test the system in different locations with the routers being set up in different places each time. Therefore, a more flexible and cost effective solution is required.

Keeping in mind that the routers have wireless capabilities, two solutions that made use of this attribute were investigated.

### 3.2.1 Wireless bridge

The design for a wireless bridge is rather simple. It involves having two or more groups of routers that are configured to be on the same LAN being connected in the same way bridges connect islands. The figure below better illustrates this concept.

**Fig. 3.1 – Wireless bridge connection**

Using a wireless bridge in the network topology seemed to solve only part of the initial problem. As seen in the above figure, several routers still have to be connected through a physical medium. This solution still doesn't allow for the flexibility required by the system.

### 3.2.2 Wireless Distribution System

The second solution found proved to be the best and the one used for this project. As was previously mentioned, the wireless bridge solution could not replace all physical links with wireless ones. The answer to this problem is called a *Wireless Distribution System* (WDS).

WDS enables the interconnection of wireless routers and access points in an 802.11 network. Furthermore, it allows for the network to be expanded easily without having the need for a traditional wired backbone link. Having said that, this system offers high degree of flexibility when choosing the router locations and it does not require the use of wired connections making it perfect to be used for the indoor positioning system.

WDS does have a few drawbacks. The maximum wireless throughput is cut in half after the data goes passed the first hop (router). For example, a laptop wirelessly connected to router A communicates with a PC connected via Ethernet cable to router B (A and B are part of a WDS). The throughput that router A can achieve is cut in half because it has to transmit the data to both the laptop and router B. Furthermore, if more than one router is linked to A in a WDS then the throughput continues to decrease. The second disadvantage is that dynamically assigned and rotated encryption keys are not a standard for the WDS connections. As a result, dynamic Wireless Protection Access (WPA) and dynamic key assignment are not available for most routers.

### 3.2.3 Router configuration

As mentioned in the previous section, the WDS is ideal for the project mainly because it allows almost unrestricted router placement without having any sort of financial impact. In order to create a proper wireless distribution system, several steps are taken:

- Design the network topology. For this stage it is important to keep in mind that the more adjacencies a router has the lower the throughput is. For this project, a simple tree topology was chosen with at most two adjacencies for each router. This sort of configuration is suitable for setting up the routers either on one or two floors.

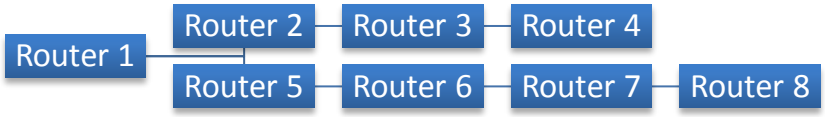



**Fig. 3.2 – Tree topology**

- Change the WAN connection type to "Disabled" on all devices except for the gateway.
- Assign static IP addresses to all routers and disable the DHCP server service on all except the gateway (Router 1). The address space used is 192.168.1.11 – 192.168.1.18 with a subnet mask of 255.255.255.0. Also, set the gateway to the root router's IP address. The next figure shows the basic IP configuration for Router 17.

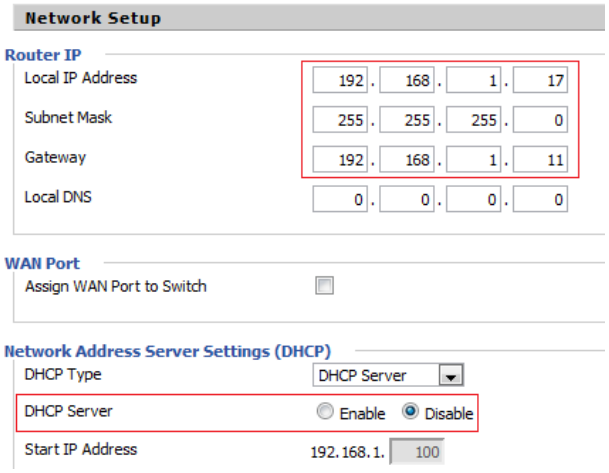



**Fig. 3.3 – Basic IP configuration**

- Disable wireless security on the **wl0** interface (DD-WRT does not support any form of security for WDS).
- In the **Setup -> Advanced Routing** tab set the operating mode to **Gateway**.
- In the **Wireless -> Basic Setup** tab set the routers to the **AP** wireless mode. Set the **SSID** to the same value on all routers and enable the **Bridged** connection type as seen in the following figure.

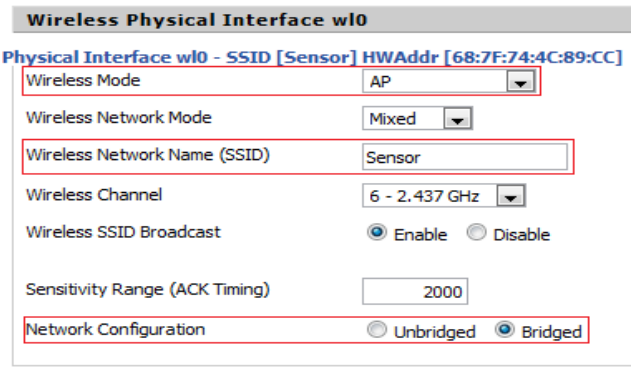



**Fig. 3.4 – Wireless interface configuration**

- In the **Wireless -> WDS** tab set the MAC addresses of the adjacent routers. Although DD-WRT supports up to 10 entries into the WDS

physical address table, for reasons stated in the previous section, it is best to have at most 2 neighbouring routers. This configuration needs to be done on all WDS nodes.

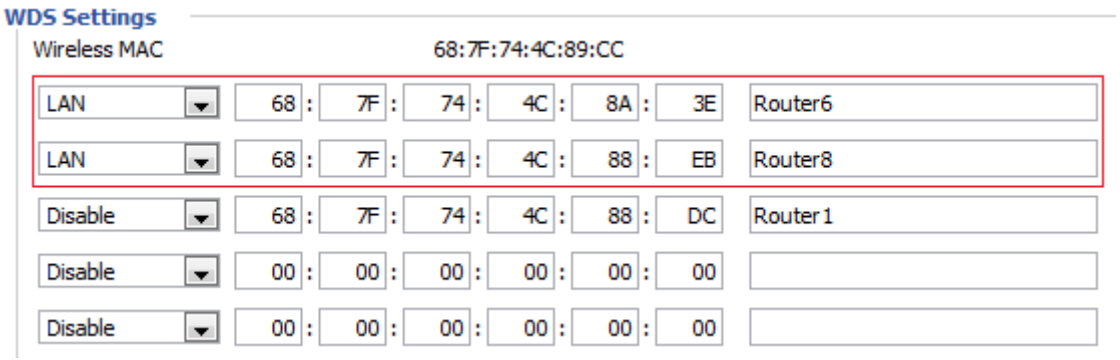



**Fig. 3.5 – WDS adjacencies**

As a result of configuring the routers to run in WDS mode, the final step in the creating the hardware part of the system is achieved. All the nodes are interconnected and can be placed throughout any indoor environment without any constraint other than the presence of a power outlet.

## 3.3 Connectivity testing

Checking whether there is communication among the routers inside the WDS was done using the **ping** tool and a wirelessly connected laptop to the gateway router. As expected, the packets were sent and received successfully to all the routers in the network. There was, however, a connectivity problem when trying to have more than 2 direct neighbours on a router. Although the wireless distribution system on the DD-WRT allows having up to 10 adjacent routers, the ping packets could not be sent/received, not even when testing without using extra hops (router to adjacent neighbour).

# 4 Software design

## 4.1 Initial ideas and obstacles

Once all routers had been re-flashed with the custom firmware DD-WRT, a suitable programming language was needed to write the software meant to extract the data (RSSI values) from the WiViz output. Since Linux is running on the routers, the obvious choice would be C/C++. The initial design was to have the software added on all devices and use the server – client programming paradigm. Each router would parse its own WiViz output to find the MAC address and RSSI value of the target device and then send the data to a central server (laptop) via TCP sockets. The central server would then convert the stored RSSI readings which are in the dBm unit of measurement into metric values using the equation resulted from applying the *Log-Normal Shadow Path model*. The newly computed distance values would be used in three different positioning algorithms (Lateration, MinMax, ROCRSSI) to determine an estimated position for the target node.

There was, however, a critical flaw in this design. The file system which came with the Linux kernel 2.4.37 was **read-only**. The only folder in the file system which had read-write permissions was **/usr/tmp/**. The drawback of using this folder is that it is temporary. Its contents are deleted each time the router is restarted. A possible solution to this predicament would be:

- Create another piece of software which sends the source code that is meant to run on the routers piece by piece to all the nodes. This can also be done using TCP sockets.
- Use the same method to send a series of scripts that compile the code and then run it.

The solution described above has several flaws. The Linux kernel does not have a built in C/C++ compiler so it would be impossible to compile the code directly on the routers. Moreover, there is no automated means of remotely running the scripts that compile and run the code. This process can only be done by logging in via Telnet and manually starting each script which of course is rather time consuming and not at all an elegant and practical solution. One other flaw is that all the code and scripts, being in the temporary folder, are deleted upon router restart and resending the code and manually running the scripts has to be done every time a router crashes.

A more technical approach to the problem at hand is to make the positioning program a part of the operating system and add the scripts that run it to DD-WRT. Further research revealed that adding new packages to the firmware could be done in a simple manner. The requirements for this process are:

- A compatible Linux platform.
- GNU C/C++ compiler (gcc/g++).
- Standard C/C++ runtime library.
- GNU make.
- TAR and GZIP tools.

Adding the scripts to the firmware was done in three stages: extract the firmware, add the packages which contain the scripts and re-build the firmware. The process has been made easy by using Jeremy Collake's *Firmware Modification Kit*. The kit provides the scripts needed for all three stages.

To extract the firmware files, the following command was used:

```
$ ./extract_firmware.sh dd-wrt.v24-12548_NEWD_mini $PATH
```

where *$PATH* is the working directory. Upon executing this command, the firmware file system is extracted in the supplied directory. The subfolders created are: **rootfs** (the root directory of the file system), **image_parts** (contains the kernel image), **installed_packages** (all custom packages reside here).

Installing the new packages which contain the scripts is done using a similar command:

```
$ ./ipkg_install.sh package.ipk $PATH
```

Finally, re-building the new firmware image is done using the *build_firmware* script by issuing the following command from the directory the script resides in:

```
$ ./build_firmware.sh output_directory $PATH
```

In order to assemble the scripts and the compiled source code into a *ipkg* package, another kit made by Collake was used. The *IPK Creation Kit* puts specific emphasis on using the packages for embedded Linux platforms such as OPEN-WRT and DD-WRT.

IPK files can be seen as a sort of archives which contain the following file structure:

- **data.tar.gz**: contains data.tar
  - **data.tar**: contains the files that will either be installed or removed from the file system. *data.tar* should have the same structure as the file system in which the package will be installed (i.e. ./usr/sbin/; ./usr/temp)
- **control.tar.gz**: contains control.tar
  - **control.tar**: contains files that give information about the package such as its name, version and possible dependencies
- **debian_binary**: it is a text file that indicates the platform and forma types. It should be set by default to 2.0 in order to be used with DD-WRT.

The kit provides an easy to use template for the package structure. The first step in creating the .ipk package is editing the **control** and **conffiles** which can be found in **control.tar**. The *control* file has the following fields:

*Package*: name of the package
*Priority*: set by default to *optional*
*Depends*: list of all dependencies (i.e. libpcap libncurses)
*Section*: set by default to *net*
*Description*: a short description of the package
*Maintainer*: name of the package creator
*Source*: set by default to *N/A* ( can be an URL where to get the *data.tar* from)
*Version*: package version
*Architecture*: router architecture (*MIPsel* for the WRT54-GL routers)

The **conffiles** file contains a list of all the files inside the package that are used to store the configuration. The next step in making the package is to copy all the scripts and executable files into the same directories they will be installed on the file system.

The final step is to actually create the package and install it using the firmware modification kit. Creating the .ipk is done by executing the following command:

```
$ ./make_ipk.sh output_package_name base_directory
```

where *output_package_name* is the IPK file that will be created (i.e. package.ipk) and *base_directory* is the location of the package files.

## 4.2 Software implementation

The final solution described in the previous section seemed to perfectly fit all the project requirements. It allowed adding useful scripts and programs to the firmware which in turn made it easier to extract the RSSI data from WiViz. However, like most software projects, frequent testing of the code was necessary. Since the code could neither be edited nor compiled on the routers, updating the packages and firmware had to be done after every new few lines of code. Taking into consideration that compiling the software and re-flashing the routers with the updated firmware image took as long as 10 minutes, progress was being achieved at an extremely slow pace. Therefore, a new design was thought of and used for this project.

Instead of having the same program run on each router, the new design consists of running the positioning program on a computer which is connected to the wireless network formed by the routers. The program connects to each router, collects the data, applies the algorithms and determines the location position of a target node.

The strengths of this solution are the greatly reduced application testing times resulted from not having to re-compile and re-install the firmware after every change in the source code, increased control over the whole positioning process and not burdening the routers' processing capabilities by running computationally expensive programs on them. There are, however, some issues with this new design, issues which will be covered in the following sections.

### 4.2.1 Application design

The IPS application was implemented in Java. The reasons are that there are many classes which can be used to communicate with the routers and the fact that, being object oriented, it is suitable to describe the devices which have several attributes such as SSID, IP address and MAC address.

The following diagram shows the application classes which can be grouped into several modules which will be described thoroughly in the subsequent sections:

- **Set-up**: starts WiViz on all routers
- **Data collector**: retrieves the RSSI values from all routers
- **Positioning**: applies three algorithms to determine a target's position
- **GUI**: simple interface between user and application

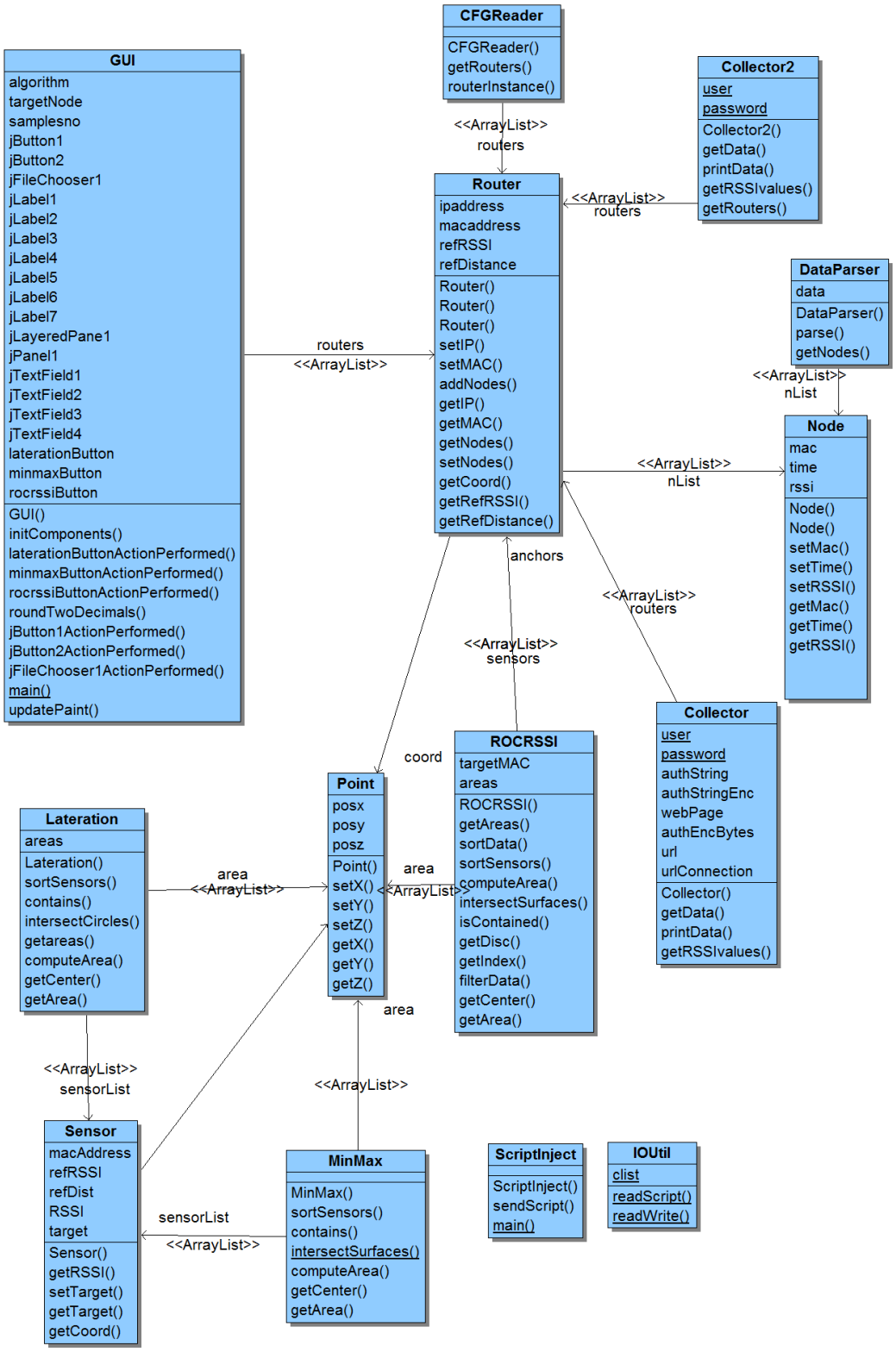



**Fig. 4.1 – Application class diagram**

As seen in the above UML diagram, three classes are used to describe the wireless devices involved in the positioning process. The *Router* class is

used to create objects that represent the routers used to detect devices in the network and measure their signal strengths. Each router has several attributes:

- **ipaddress**: logical address used to identify a device inside the network
- **macaddress**: physical address used to identify a device in the WiViz output
- **nList**: is a list of Node objects which is created after gathering the WiViz data
- **coord**: the (x,y,z) special coordinates of the router
- **refRSSI**: a signal strength value measured at a short reference distance
- **refDistance**: reference distance at which calibration measurements are done

The *Sensor* class is similar to the *Router* class and the instances of this class are used in the positioning algorithms. The main difference between the two is that a sensor object does not require an IP address field but does have a field which stores the signal strength to the target node and the means to convert that value into distance.

The *Node* class is used to create objects that store information about the wireless devices which WiViz collects data about. These devices can be either the routers (sensors) or target nodes.

A diagram showing how the positioning of a target wireless device is done can be seen in the following figure:

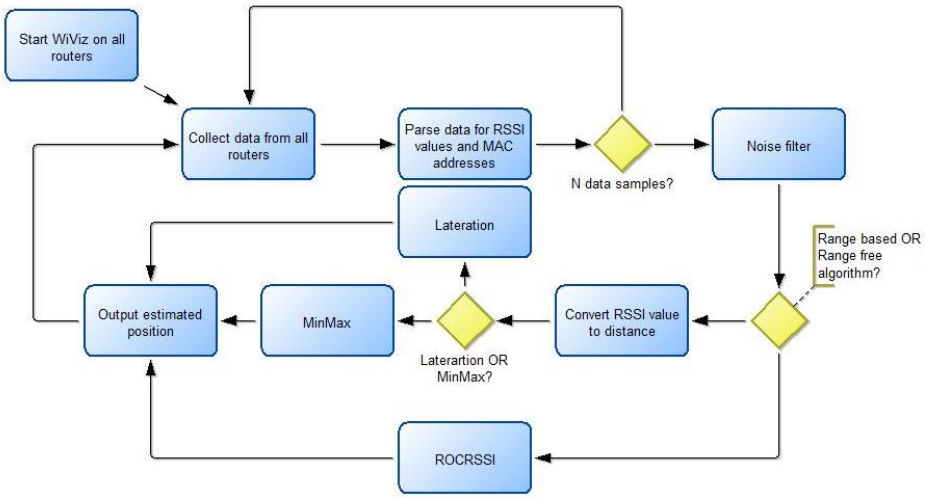



**Fig. 4.2 – Application diagram**

The following subchapters describe in detail the modules, the rest of the classes as well as the mathematical concepts behind the conversion of signal strength (dBm) to distance (meters).

### 4.2.2 Set-up module

The function of this module is to run WiViz on every router in the wireless distribution system. WiViz was designed to be started using DD-WRT's web interface and shut down whenever the web browser would be closed. This proved to be quite an obstacle in the long run because having an open web browser with several tabs running WiViz for every router can lead to a heavy load being put on the processor and network communication abilities of the PC running the positioning software. It wasn't the case for this project considering that the number of routers used is fairly low but in the future this number can increase and having to manually start the packet sniffer for a large number of devices is neither elegant nor optimal.

The solution found involves executing a script on each router that starts WiViz and keeps it running until a *quit* command is issued or the routers are restarted. Figure X presents this script which is called *wiviz_run.sh*:

```
#!/bin/sh
WIVIZ_PATH=/usr/sbin/wiviz
while sleep 5; do
killall -USR1 wiviz >/dev/null 2>&1
if [ 0 -ne $? ]
then
    if [ \! -x $WIVIZ_PATH ]
        then
        chmod 755 $WIVIZ_PATH
        fi
    $WIVIZ_PATH >/dev/null </dev/null 2>&1 &
    killall -USR1 wiviz > /dev/null
 fi
done
```

*WIVIZ_PATH* is a variable that stores the location of the main WiViz start script which runs the sniffer for 5 seconds and then stops it. *wiviz_run.sh* will keep it running by creating a loop and starting it every 5 seconds. This ensures that WiViz will behave the same way as it would do if it were being run inside a web browser.

As discussed in the previous sections, the DD-WRT file system is read-only. Therefore an automated method was needed for creating the script on each router and executing it as opposed to manually adding and running it in the temporary folder on the file system. The best solution which also has a good degree of flexibility was to use Telnet to communicate with the routers.

The *ScriptInject* and *IOUtil* classes handle the process of creating and starting the script shown in figure X. In order to write to and read from the Telnet connection two threads are created. Synchronization between these two threads is simply done by making one wait for a very short time (100 ms) while the other sends or receives a command. Because Telnet does not support file transfers, the script was created on the file system of the router by issuing the same commands one would execute directly from a console. The steps in this process, after connecting via Telnet using authentication are:

- Write $cd\ /usr/tmp\backslash ntouch\ wiviz\_run.sh\backslash n$ to the Telnet output stream – change directory to the *tmp* folder which has read/write permissions and create an empty script file using the $touch$ command
- Read the script file and store each line in a String array
- For each element of the array write $echo\ "SCRIPT\_LINE" >> wiviz\_run.sh\backslash n$ to the output stream – appends the line in the script file
- Write $sh\ wiviz\_run.sh\ \&\backslash n$ to the output stream – runs the script in background

The output stream practically types the commands described above character by character. The $\backslash n$ character represents pressing the Enter key to execute the command it follows. *SCRIPT_LINE* consists of one line of the script that is to be appended in the file. The $\&$ parameter found in the $sh$ command signals that the script is to be run in the background. This prevents the script from being stopped when the Telnet connection closes.

Although this method doesn't seem to be quite elegant, it provides very good flexibility. Whenever there is a need to make changes in the script, these can be done in only one file and then sent to all the routers by executing the *StringInject* class. This also enables the project to scale well – adding more routers has no effect whatsoever on the system other than adding a few seconds to the start-up time. The module can be further upgraded to account for a router crashing or being accidentally restarted/reset. This can be done by using the method described above to send a *ps* command which writes all the processes that are running at a given time to the Telnet input stream. Then, a simple

search through the received data can reveal whether the script is still running or if it needs to be resent or restarted.

### 4.2.3 Data collector

This module is responsible for gathering the WiViz output from every router and parsing the data for RSSI values and MAC addresses. As seen in the class diagram in figure X, there are two data collector classes.

The *Collector* class uses an URLConnection (based on the HTTP protocol) with basic authentication to establish communication with the routers. As opposed to the Telnet authentication process where the username and password can be sent in clear text via the socket, these have to be encrypted in order to be used in a HTTP connection. In this respect, a base64 encryption class from the *org.apache.commons.codec.binary* package was used. The following code illustrates the usage of the base64 encryption:

```
authString = user+ ":" + password;
authEncBytes = Base64.encodeBase64(authString.getBytes());
authStringEnc = new String(authEncBytes);
url = new URL(("http://" +router.getIP() + "/Wiviz.live.asp"));
urlConnection = url.openConnection();
urlConnection.setRequestProperty("Authorization", "Basic " + authStringEnc);
```

WiViz stores the latest scan in an admin page called *Wiviz.live.asp*. The content of this page is read using the URL connection's input stream and then passed on to a *DataParser* object which extracts the RSSI values and MAC addresses of the scanned devices. It then compiles a list of nodes which is stored in the data collector object. The WiViz output page stores the data in the following format:

```
h = new Object();
h.mac = '68:7F:74:4C:88:EB';
h.rssi = -44;
h.type = 'sta';
h.self = false;
h.sta_state = 'unassoc';
h.sta_laststsid = '';
h.age = 40;
```

The fields that are relevant to the positioning system are $h.mac, h.rssi$ and $h.age$. The latter indicates the time (in seconds) that has passed since the device was last seen in a scan. This is used to filter out data that can corrupt the positioning process. For example, if the target node has an age of 20 seconds or above it means that it has either been turned off or has been moved to a location where it has a very poor signal (-100 or below). In either case the RSSI value scanned cannot be used in the positioning algorithms.

Using an HTTP connection to retrieve the data from the routers worked well from small scale testing and debugging. However, when all the routers were deployed, it was noticed that the time it took to gather just one data sample from each router was not suitable for conducting hundreds of measurements in dozens of locations. Establishing an HTTP connection to each router takes about 8 seconds which meant that collecting the WiViz output from the entire system takes over 1 minute which is not optimal for an application which should calculate the target node's position every few seconds. Another issue with using this type of connection is that the Java class does not have the methods to close a connection. This process is handled by the routers which use HTTP version 1.1 with persistent connections. Therefore, using either the already opened connections or creating new ones to collect more data samples resulted in receiving blank output pages.

The *Collector2* class uses a Telnet connection to overcome the obstacles faced by using an HTTP connection. It is based on the same idea used to send and run the WiViz start script on the routers. Because the *Wiviz.live.asp* page cannot be accessed through a Telnet connection, a substitute was found in another output file created in the $/usr/tmp/$ folder which stores the latest scan in the same format.

The $cat\ ../wiviz2 - dump\backslash n$ command is used to display the contents of the output file in the Telnet console. Then the content is read by the *Collector2* class through the Telnet input stream and passed on to be parsed by a *DataParser* object. Switching from HTTP to Telnet significantly improved the time taken to retrieve the scan data – from 8 seconds less than 1 second – and, having the means to close each connection, several data samples can be collected as required.

### 4.2.4 Positioning

This module is responsible for using the RSSI values collected from all the routers to calculate an estimated position of a target wireless device. The algorithms used can be split into two groups: ranged based and range free. The range based algorithms convert the RSSI values which are in represented in dBm to distance. The technique used for the conversion is called *log-normal shadow path* and will be explained in the following subchapter. One important and common aspect of both types of algorithms is that the RSSI values are filtered before being processed. Filtering is done by ignoring signal strength values which are below a certain threshold. It is recommended to use a higher value if the routers are separated by physical obstacles such as walls. In this project the threshold was set to -65 dbm but it can be easily changed to any value.

#### 4.2.4.1 Ranged based positioning

The algorithms used for ranged based positioning are *Lateration* and *MinMax* as described in chapter 2. However, the literature does not provide specific details on how to actually implement them in any programming language.

The difference between the two algorithms is the method used to determine the areas in which the target node is. Such an area is calculated by checking every point between $(x, y, z) - distance$ and $(x, y, z) + distance$ where $(x, y, z)$ are the spatial coordinates of the routers and *distance* is the distance between the target node and the routers. Since the lateration algorithm considers these areas to be circles for two-dimensional positioning and spheres for three-dimensional positioning, a point $(x, y, z)$ is inside the coverage area if

$$(R_X - x)^2 + (R_Y - y)^2 + (R_Z - z)^2 \leq r^2$$

where $(R_X, R_Y, R_Z)$ are the router coordinates and $r$ is the distance between the router and the target node. The MinMax algorithm considers all points between $(x, y, z) - distance$ and $(x, y, z) + distance$.

Apart from the different methods of calculating the coverage areas, the two algorithms are implemented in the same way:

1. Convert RSSI values to distance using the log-normal shadow path model.

2. Sort the routers by distance in ascending order – the closest router to the target node is first.
3. For each router create a list of all points in the coverage area.
4. Iterate through the lists created in step 3.
   4.1. Intersect current list with the next list.
   4.2. If the intersection is null, keep the first list, otherwise proceed.
5. Calculate the centre of the area that resulted from intersecting all lists.

The conversion from dbm to metres is done using the log-normal shadow path model which is a general model suitable for both indoor and outdoor environments. The model provides several parameters which can be configured to suit different types of environments. The formula is as follows:

$$PL(d)(dB) = PL(d_0) - 10\eta \log_{10}(\frac{d}{d_0})$$

$PL(d)$ represents the RSSI value measured by the routers. The $PL(d_0)$ parameter is a reference RSSI value measured at a reference distance of $d_0$. Lastly, $\eta$ is an environment complexity coefficient which can be any value between 1 and 7 and indicates the average number of obstacles between a router and the target node. Using the above formula, the distance can be easily calculated:

$$d(m) = d_0 10^{\frac{PL(d_0)-PL(d)}{10\eta}}$$

For this project, the reference RSSI values were measured at a distance of 1 metre and $\eta$ was set to 3. Through testing it was observed that the RSSI values measured in the same point change over time. Therefore it is necessary to adjust the parameters of the formula according to the time of day.

#### 4.2.4.2 Range free positioning

The ROCRSSI algorithm was used for range free positioning. As presented in chapter 2, this algorithm draws its strength from the fact that it does not depend on the dBm to metre conversion which does not provide accurate results.

Unlike the range based algorithms which only need the target's RSSI values measured by each router, ROCRSSI also uses the RSSI values of the other sensors in the network. The algorithm is implemented as follows:

1. Sort the routers by target node RSSI value in descending order – the closest router to the target node is first.
2. For each router (anchor):
    2.1. Create a list containing the RSSI values of the target node and the other routers.
    2.2. Sort the list by RSSI values in ascending order.
    2.3. Eliminate all RSSI values except the target node's and the routers' which are immediately below and above the target node.
    2.4. Calculate the areas that have the current anchor as the centre and a radius equal to the distance between the current anchor and the ones that still exist in the list.
    2.5. Calculate the difference of the two areas resulted from step 1.4.
3. For each anchor:
    3.1. Intersect the area resulted from step 2.5 with the next anchor's area.
    3.2. If the intersection is null, keep the first area, otherwise proceed.
4. Calculate centre of the final area.

Although the ROCRSSI algorithm seems to be very effective in theory, when tested in a real life situation it did not work as expected. It relies heavily on correct RSSI measurements which are always under the influence of fading and shadowing effects. For example, a router, after sorting its list of RSSI values, can end up not having the target node's value between other routers'. Therefore a correct application of the algorithm becomes impossible, conclusion which was reached after thorough tests.

### 4.2.5 GUI

For testing purposes, a graphical user interface was created for the indoor positioning application. It allows choosing different configuration files for router placement and parameters, choosing the positioning algorithm and displays the calculated position of the target node and the area in which it may be. The GUI can be seen in fig. 4.3.

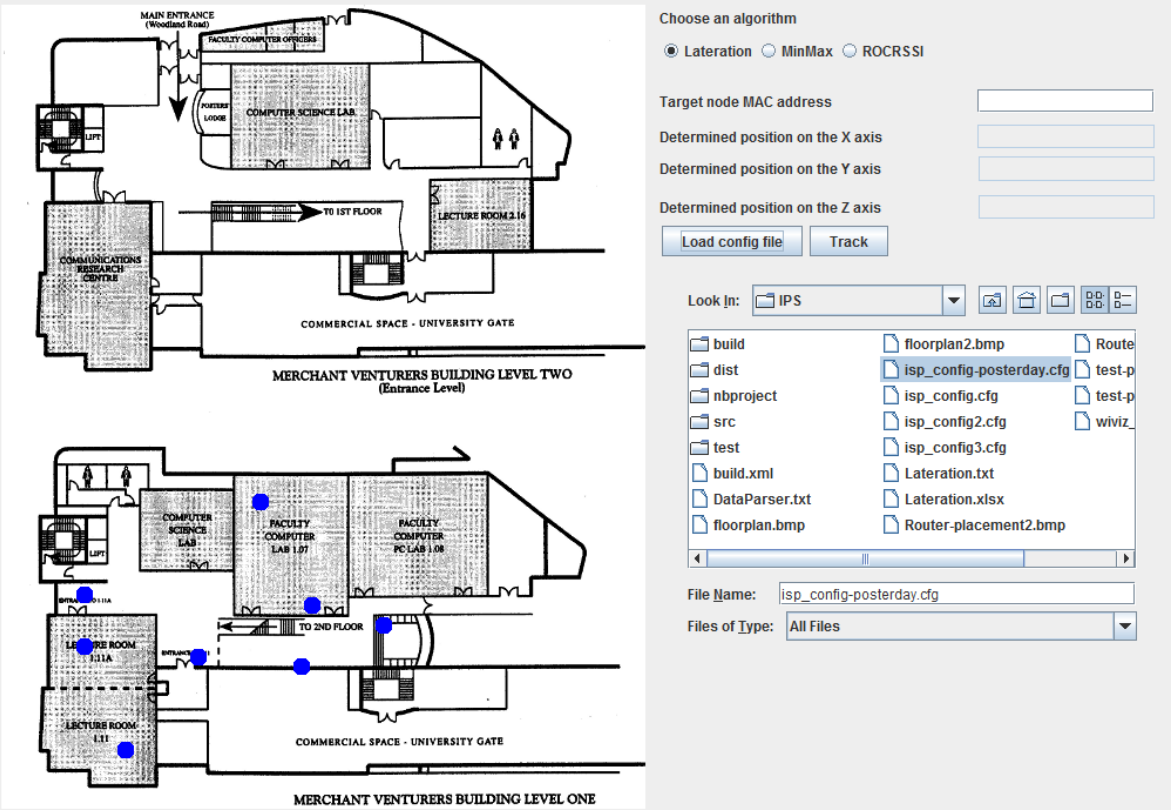



**Fig. 4.3 – Application graphical user interface**

On the left hand side of the GUI there is a floor plan of the indoor environment in which the positioning system runs. The routers are displayed as blue dots and the calculated area which contains the target node is displayed in red. The GUI allows tracking any wireless device by typing its MAC address in a text box.

The router configuration file can be any text file as long as it has the following format:

*R1_MAC_ADDRESS R1_IP_ADDRESS X1 Y1 Z1 PL1(d1) d1*
*R2_MAC_ADDRESS R2_IP_ADDRESS X2 Y2 Z2 PL2(d2) d2*
.
.
.
*Rn_MAC_ADDRESS Rn_IP_ADDRESS Xn Yn Zn PLn(dn) dn*

where $PLn(dn)$ is the RSSI value measured at a distance of $dn$ and $(Xn, Yn, Zn)$ represent the router's coordinates in space.

# 5 Testing and analysis

An evaluation of the indoor positioning system is needed in order to determine the average accuracy that can be achieved in an indoor environment. This chapter covers tests done to analyse the variation of signal strength over time and analyse the impact different parameters (number of routers, number of data samples used, router placement) have on accuracy.

The testing was done in the Merchant Venturers Building which is a rather complex building with three floors, many rooms and large unobstructed areas as well. These features made this particular building a suitable testing environment for the indoor positioning system.

## 5.1 Methodology

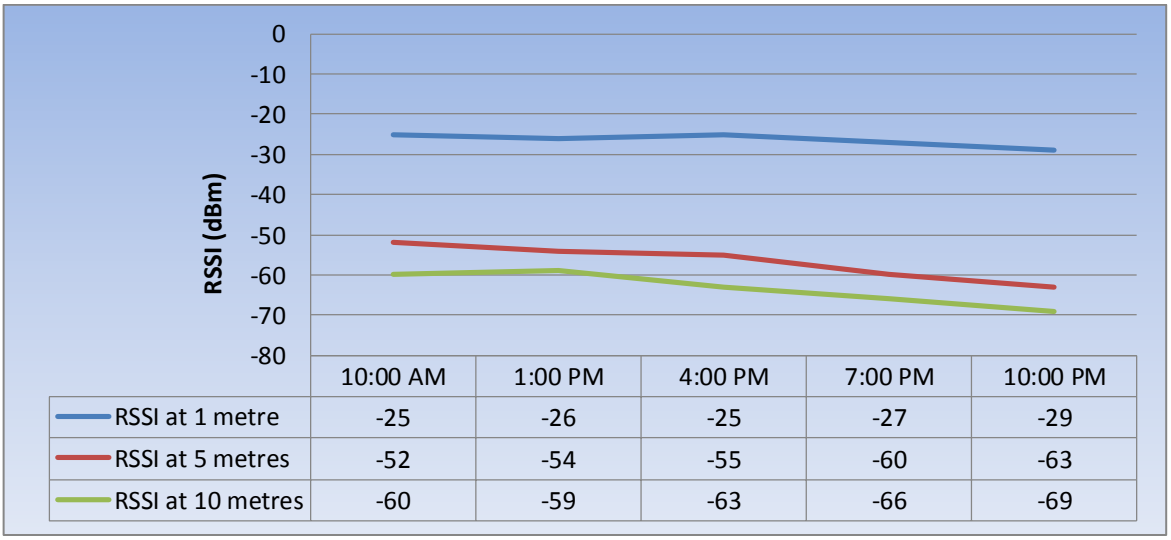
Testing was done on both floors of MVB. Several router placement configurations were explored. In order to determine the spatial coordinates of the routers, a floor plan of the building was used. However, it did not include measurements for floor height, width and length. Using a tape measure to determine the building's characteristics proved very inefficient and error prone. Therefore an improvised method was used to calculate the position of each router.

An image editing application called Gimp was used to load the floor plan and determine the length, width and height of the building and its various parts. Using a tape measure it was found out that the entrance to room 1.11 (see Appendix A) was 1.8 metres wide. The entrance was also measured in Gimp using the application's built in tool and the width was 51 pixels. Therefore a distance of 1 metre is 28 pixels long in the floor plan image. This information was used to accurately calculate the position of the routers.

The target node used in all the tests is an HTC Desire S smartphone which has wireless capabilities. A wireless network analyser was installed on the phone in order to ensure that it generated continuous network traffic throughout the duration of the tests.

The reference RSSI values were measured at a distance of 1 metre for every router. These values varied between -25 dBm and -30 dBm depending on the orientation of the router antennae and remained constant over time. However, it was observed that values measured at greater distances (over 5

metres) changed as time passed. Fig. 5.1 shows this variation over 3 hour intervals.




| | 10:00 AM | 1:00 PM | 4:00 PM | 7:00 PM | 10:00 PM |
|---|---|---|---|---|---|
| RSSI at 1 metre | -25 | -26 | -25 | -27 | -29 |
| RSSI at 5 metres | -52 | -54 | -55 | -60 | -63 |
| RSSI at 10 metres | -60 | -59 | -63 | -66 | -69 |

**Fig. 5.1 – Variation in time of RSSI values**

All tests were done using the Lateration algorithm because it has been proven to be more accurate than MinMax and ROCRSSI, due to reasons presented in the previous chapter, proved to be highly unsuitable for indoor environments where wireless signals are subject to interference, fading and shadowing effects.

## 5.2 Small scale testing

The small scale tests were done in order to get an overview of how the positioning system behaves in a controlled environment. The computer laboratory situated on the second floor was used in this respect because it provides an area free of large physical obstacles such as walls. There are several router placements according to the number of such devices used.

The first set of tests focused on the impact of the number of data samples taken from a single router on the distance calculation. This parameter presents great importance in the positioning process because wireless signals are unpredictable. Different RSSI values can be obtained by taking several measurements in the same place. In order to get an accurate signal strength, several samples must be collected and the final result is the average of the data set. Carefully deciding how many data samples are necessary for a good result is important for the positioning system because collecting too much data leads to delayed position estimation.

For this type of test only one router was used. Measurements were taken at different distances from the router and the number of data samples collected varied between 1 and 100. Table 5.2 contains the averaged RSSI values converted to metres for every test point and number of samples collected.

| | 1 | 3 | 5 | 7 | 10 | 15 | 20 | 50 | 100 |
|---|---|---|---|---|---|---|---|---|---|
| **1 metre** | 1.21 | 1.14 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| **3 metres** | 2.51 | 2.68 | 3.26 | 3.05 | 3.05 | 3.05 | 2.86 | 3.05 | 3.05 |
| **5 metres** | 5.53 | 5.17 | 5.17 | 4.54 | 4.84 | 5.17 | 5.17 | 4.84 | 4.84 |
| **7 metres** | 8.76 | 8.76 | 7.68 | 8.20 | 7.68 | 7.19 | 7.68 | 7.68 | 7.19 |
| **10 metres** | 10.68 | 11.406 | 10.68 | 9.3632 | 10.68 | 10.0 | 10.68 | 10.0 | 10.68 |

**Table 5.2 – Distance calculation in relation to number of data samples collected**

The data shows that the accuracy of determining the distance between the router and the target node improves when collecting a larger number of data samples. Fig. 5.3 provides a visualisation of this observation.

Although this test was performed in an environment without massive obstacles between the router and the target, the information provided is also relevant for a more complex setting. This is true due to the filtering stage in the positioning process in which all RSSI readings below a certain threshold are not taken into consideration. Therefore, collecting 5 data samples from each router gives a very good accuracy without having a negative impact on the running time of the positioning process.

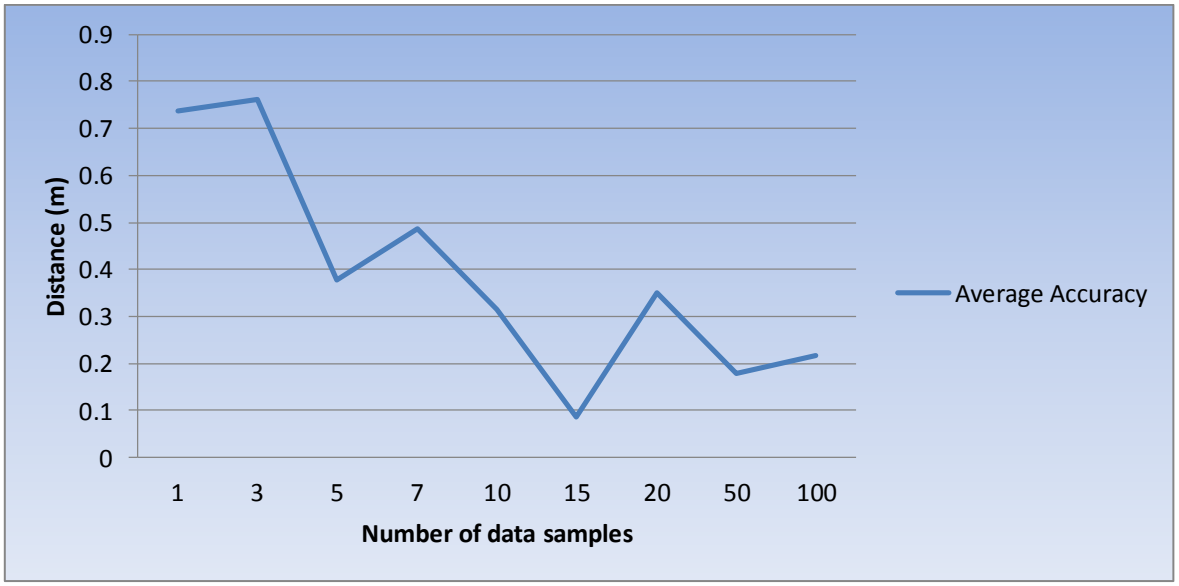



**Fig. 5.3 – Average accuracy in relation to number of data samples**

The next set of tests is aimed at discovering the minimum number of routers needed to achieve a good accuracy. This aspect is very important because one of the aims of the project is to create a positioning system that has a very good accuracy for the least possible number of routers. This prevents possible clients to spend more than it is necessary on equipment.

Testing was done by varying the number of routers from 2 to 8. These tests were also used to determine the best possible router placement for each scenario. It was observed that, in a large room, most routers should be placed near the walls and the others somewhere in the middle of the room. The space between them is also important. Having two routers too close to each other would mean a waste of resources because they would share a large portion of the coverage area. The same can be said if they are too far apart in which case there would be a gap between their coverage areas and a target node would not be located if it were in that gap.

The first test was done using only 2 routers. Fig. 5.4 shows the placement of the devices.

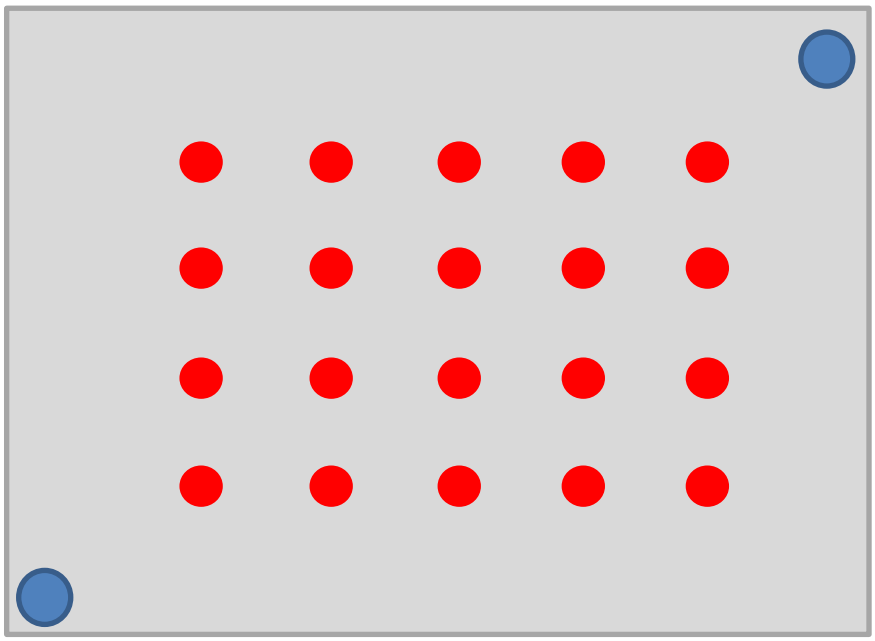


**Fig. 5.4 – Router positioning (2 routers)**

The manner in which the routers (represented by blue circles) are positioned enables a full coverage of the entire room. Testing was done in 20 points (represented by red circles). These are approximately 2 metres apart.

For the points that form a line between the two routers the average accuracy was under 2 metres. The RSSI values measured in the other points

were very close and sometimes identical to the values measured on the diagonal. This lead to a drop in accuracy from 2 metres to an average of 7 metres.

Using 3 routers improved the average accuracy as expected. Fig. 5.5 shows the optimal router placement for this scenario. As in the case of using 2 routers, the best accuracy was achieved for the points in the centre of the grid and the ones directly between the routers.

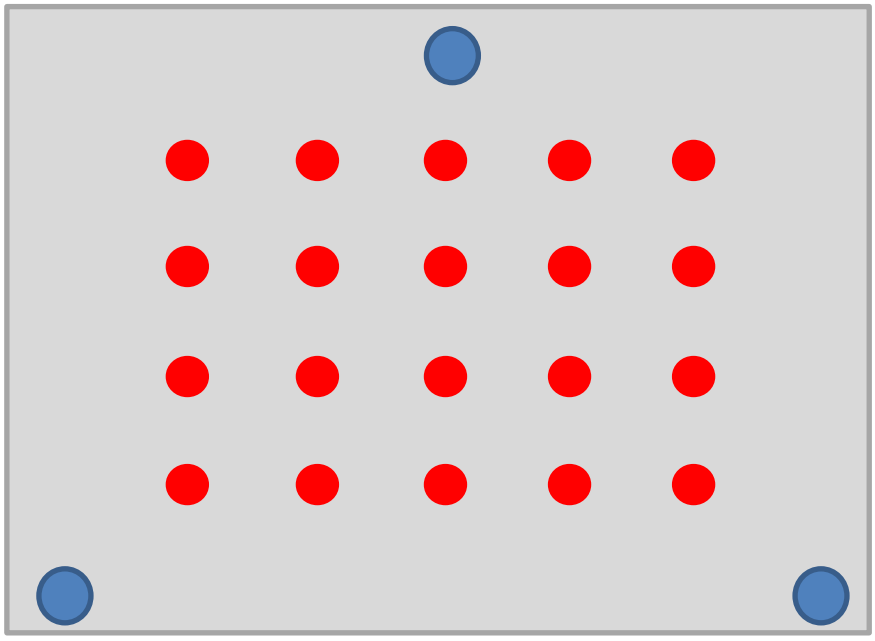


**Fig. 5.5 – Router positioning (3 routers)**

Further adding more routers lead to an increasingly better overall accuracy. An important observation is that the accuracy remains the roughly the same (approx. 2.5 metres) for the 4 and 5 router scenarios and increases only slightly when 6, 7 and 8 routers are used. Fig. 5.5 shows a graph describing the impact of the number of routers on the positioning accuracy.

The best result was achieved when 8 routers were used. The overall accuracy in that scenario was 2.1 metres. In theory, the Lateration algorithm allow for even better accuracies especially when using as many as 8 routers. However, the limitations of the wireless signal strength play a major role in the positioning process. The fact that it is very unpredictable make it difficult to get good readings from the sensors. Furthermore, the log-normal shadow path model used to convert the RSSI values into distance can never provide a good enough result that would lead to accuracies below the 2 metre mark.

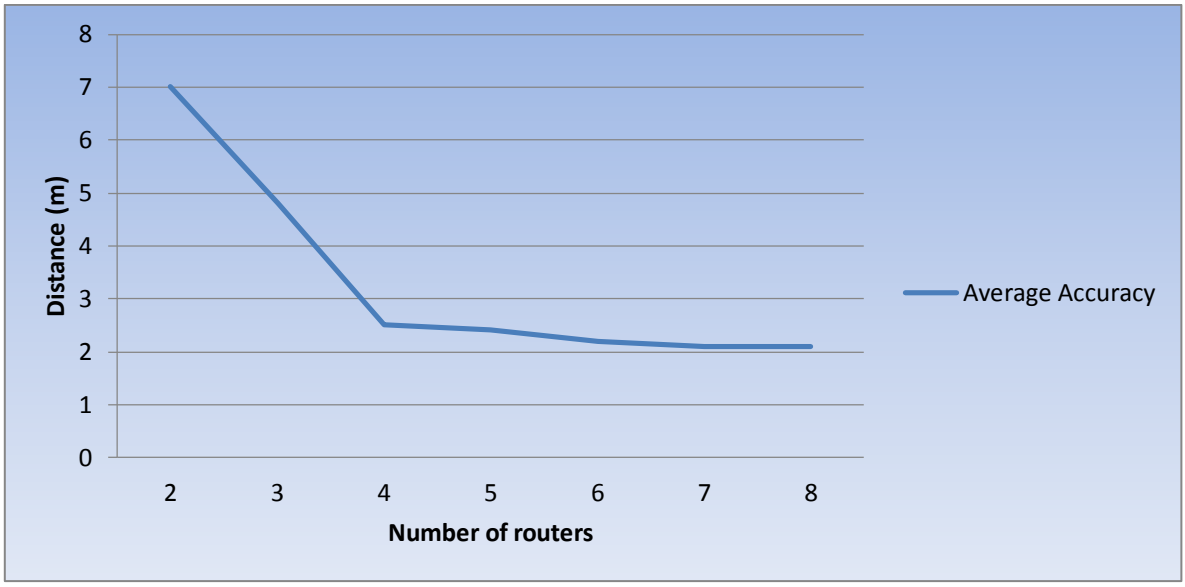



**Fig. 5.5 – Average accuracy in relation to number of routers**

As a result of the small scale testing, the following observations were made:

- Collecting 5 data samples is sufficient to overcome the wireless signal's vulnerability to interference.
- The parameters used in the log-normal shadow model formula should be changed according to the time of day.
- Router placement plays an important role in getting the most out of the positioning algorithms.
- Having 4 routers in every large room yields the best results and it is more cost-effective as opposed to having 5 or more.

## 5.3 Large scale testing

With the information provided by the small scale tests, the positioning system was deployed throughout the major part of the Merchant Venturers Building. Because of the relatively small number of routers available for this project, it was impossible to cover the entire building. Also, the optimal number of routers could not be placed in the rooms that were a part of the testing environment.

The first test was done on the first floor of the building using all 8 routers. This proved to be the perfect environment for a large scale test because it has several rooms as well as a large open area. Fig. 5.6 shows the router positions (blue circles) for this scenario.

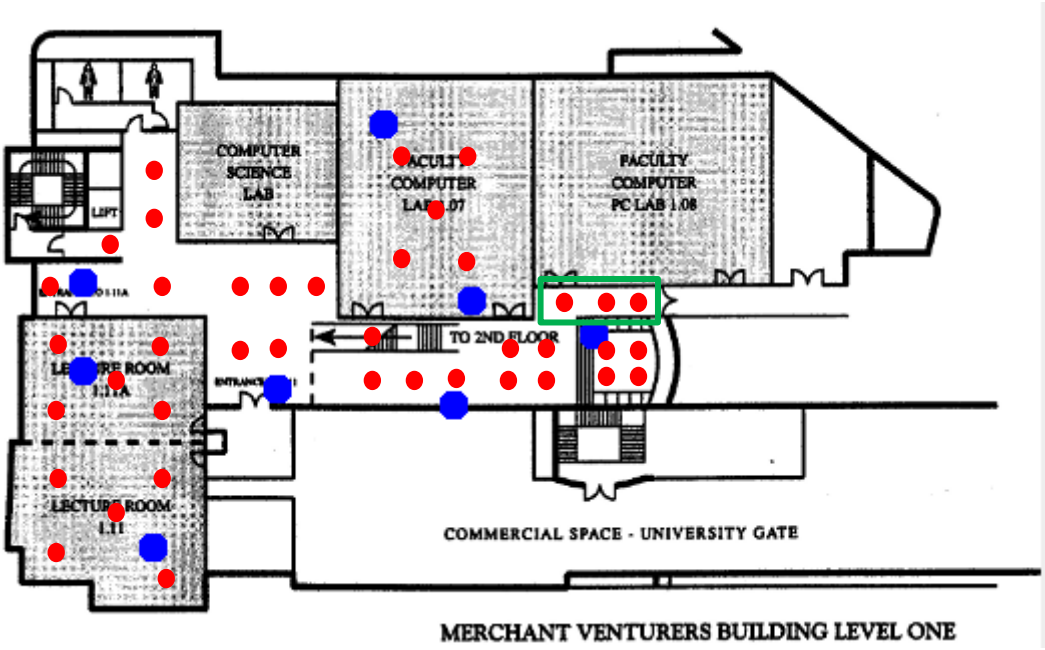



**Fig. 5.6 – Router positioning for large scale testing on one floor**

As seen in the above figure, it would have been ideally to have 4 routers in LAB 1.07 as well as rooms 1.11 and 1.11a. Due to this impediment, it was expected to get lesser accuracies in these rooms. As in the case of the small scale tests, 5 data samples were taken at every test point (red circle). The area highlighted with green proved to be the most problematic of all. The RSSI values measured there where almost always very low for the closest router and higher for the router situated near the stairs. This was caused by the small enclosed space in which measuring was done and the calculated positions in that area were off by 7 to 10 metres.

The average accuracy resulted from this test was 2.47 metres on the X axis and 2.29 metres on the Y axis. Fig. 5.7 shows the measured position in each test node as well as the calculated position. The graphs clearly illustrate the places where, due to insufficient coverage, the accuracy is very low. In some of these places, adding one or more routers can lead to better results. This is not the case of the area highlighted in green where the geometry of the building is responsible for incorrect RSSI readings.

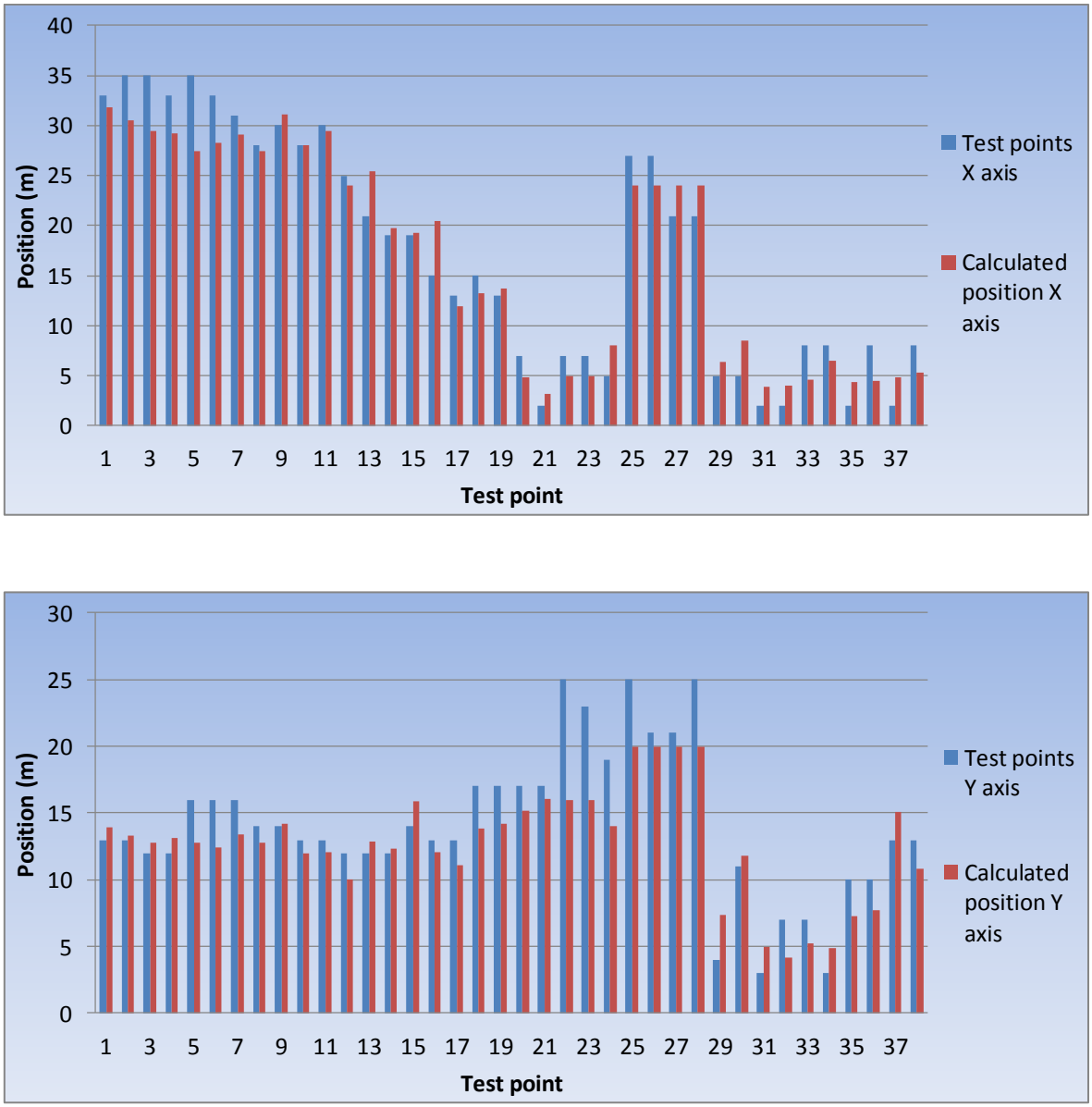



**Fig. 5.7 – Calculated position in comparison to actual position**

The location tracking system created for this project can also determine the position of a target node on the Z axis. Another test was done in this respect. The routers are placed four on each floor as seen in fig. 5.8. In order to get as much coverage as possible only one router was placed in the rooms that were a part of the testing environment.

As expected, the average accuracy did drop to 4.30 metres on the X axis and 2.83 metres on the Y axis. However, the position on the Z axis was determined with an error of 0.76 metres. Considering that the routers were spread so far apart, the results are still impressive. The overall good accuracy is a consequence of the fact that there is a large gap between the 1$^{st}$ and 2$^{nd}$

floors. Therefore, data from almost all the routers could be used, especially to determine the positions in the points that are in the immediate vicinity of the stairs.

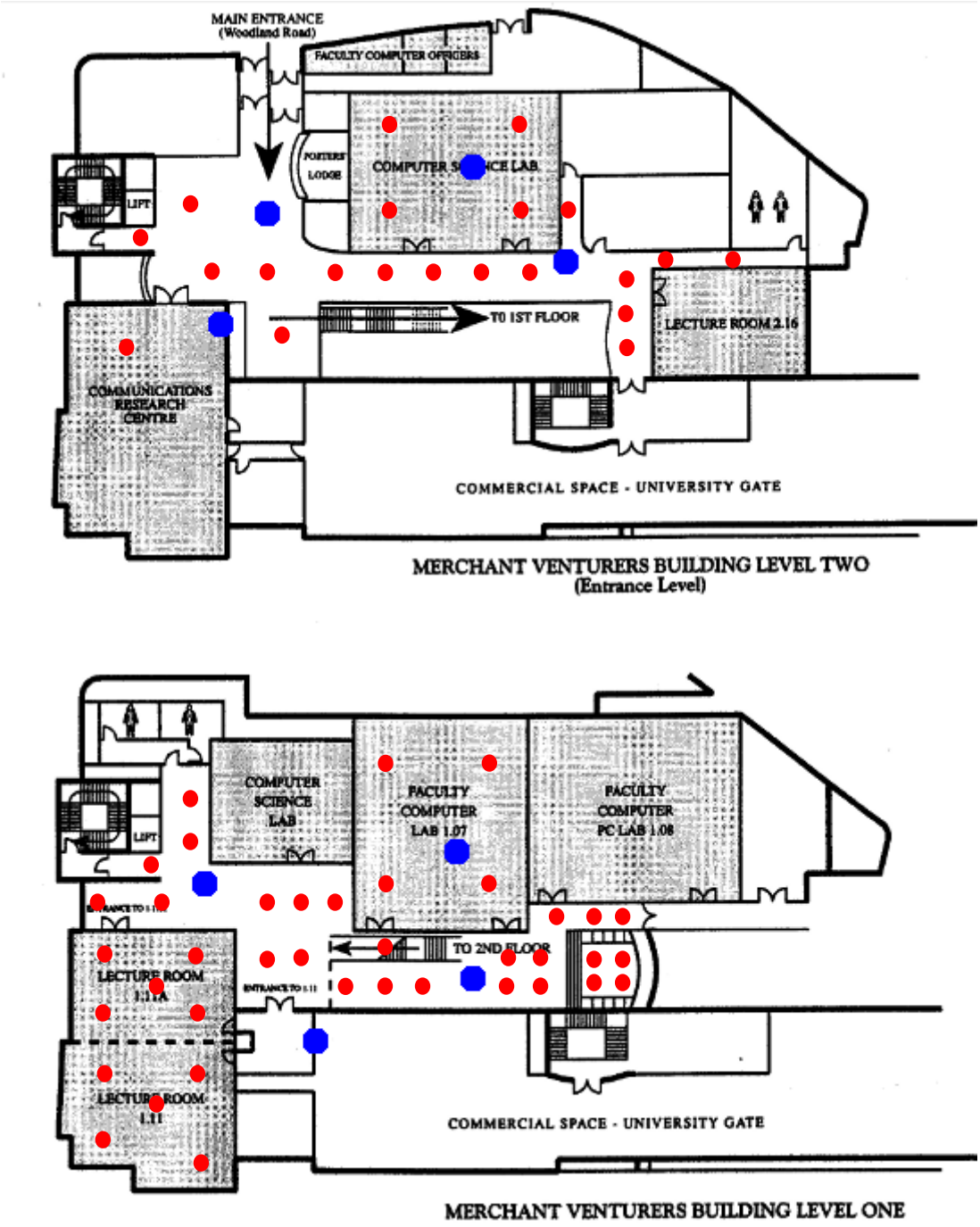



**Fig. 5.8 – Router positioning for large scale testing on two floors**

# 6 Conclusions

This chapter summarises the work conducted in this project and makes further comments on the results presented in the previous sections. It mentions justifications for the research tracks that were followed and how they relate to the initial goals. Having presented all the relevant results in Chapter 5, only a few key notes and observations will be made here. In the end, appropriate future work will be presented.

## 6.1 Objectives and results

As discussed in Chapter 1, this project was started with six goals in sight and during this project all six goals have been successfully addressed.

The indoor positioning system was built using WRT-54GL Linksys wireless routers. These devices are very common and can be easily acquired either at the local IT&C store or from an online retailer. The WRT-54GL is relatively cheap, one costing around £40.

The system can be easily used by almost any person who knows what a computer is. The graphical user interface presented in Chapter 4 is very intuitive and, the graphical feedback it provides, gives the user a better understanding of where the target node is located in the building as opposed to just seeing numbers.

Because the routers used cannot be bought with either DD-WRT or WiViz already installed, the set-up stage may prove to be a bit difficult. Depending on the number of routers needed, installing DD-WRT on all of them could take as long as a few hours. Moreover, as seen in Chapter 3, the routers need to be configure to run in WDS mode which in turn adds more to the set-up time. This part of the set-up stage could be eliminated if a vendor sold a complete IPS with pre-configured routers that a have DD-WRT installed. Lastly, the RSSI reference values have to be measured for each router at different times of day in order to overcome the wireless signal's unpredictability. These measurements can easily be done by a non-specialised person and should not take too much time. Gathering the reference values at 3 hour intervals is sufficient.

Three positioning algorithms were implemented and used. Lateration and MinMax are range-based and rely on an accurate conversion of the RSSI value to distance. ROCRSSI is a range-free algorithm which proved to be very

inefficient when used in environments with irregular RSSI readings. Out of all three, the best suited for conducting the tests was Lateration.

The tests designed for the system are small scale and large scale tests. The small scale tests focus on determining the right combination of parameters and router positions that leads to a very good accuracy. It was shown that RSSI values measured at great distances can vary by as much as 11 dBm over a 12 hour period. Collecting many data samples proved to improve accuracy in the distance estimation. It was found that 5 samples were sufficient to get a good accuracy and still keep a low running time of the positioning process. The same applies for the number of routers used. Having 4 routers in a large room resulted in an accuracy of 2.5 metres whereas using 5 or more only improved this number by 0.3 metres. Therefore, it is not cost-effective to purchase more than 4 routers to improve accuracy by such a slight value.

The large scale tests were done in a larger, more complex environment. When testing with all 8 routers on the second floor, the average accuracy was 2.47 metres on the X axis and 2.29 metres on the Y axis. The second large scale test presented was done using two floors with 4 routers on each one. The results were, as expected, not as good: 4.30 metres on the X axis, 2.83 metres on the Y axis and 0.76 metres on the Z axis. Having said that, the correct floor was determined in every test point. As a result of testing the system, it can be concluded that achieving an accuracy of below 7 metres is possible even when using a low number of routers (3 or 4).

The system can be successfully deployed in any indoor environment keeping in mind that the routers should be placed 5 to 15 metres apart (depending on the geometry of the rooms) and that the optimal number of routers that should be in a large room (lecture room, laboratory) is 4.

## 6.2 Future work

A number of improvements related to both software implementation and positioning methods may be added to the existing system in order achieve better running times and possibly a better accuracy.

Even though using Telnet instead of HTTP for the application to communicate with the routers significantly improved running time, there is a possibility that switching back to HTTP would improve it further. A thread pool could be used to open HTTP connections to the routers. The threads would make sure that the connections are not automatically closed by the routers and that they are always ready for data collection. Opening all connections would

still be slow but it would only be done once and data retrieving should be done much quicker.

One technique which was not explored in this project is the fingerprint matching method which is based on creating a large database of RSSI values measured throughout the building. This database also called a training set, would then be processed by a *k-nearest neighbour* algorithm in order to determine the position. This approach was deemed unsuitable for this project mainly because the training phase is very time consuming

Another possible improvement could be found in using additional devices such as an accelerometer and compass. Most smartphone have these built in. They could be used to determine direction of movement and speed and possibly predict the location of a target.

# 7 Bibliography

[1] Bahl, P., Padmanabhan, V.N. "RADAR: an in-building RF-based user location and tracking system". *In Proceedings of the 9th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*, Tel-Aviv, Israel, Mar. 26-30, 2000, pp. 775-784.

[2] Chang, H., Tian, J., Lai, T., Chu, H., and Huang, P. "Spinning beacons for precise indoor localization". *In Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys '08)*, Raleigh, NC, USA, Nov. 05-07, 2008, pp. 127-140.

[3] Converting Signal Strength Percentage to dBm Values. Available at*: http://www.wildpackets.com/elements/whitepapers/Converting_Signal_Strength.pdf.*

[4] Elnahrawy, E., Li, X., Martin, R. P. "The limits of localization using signal strength: a comparative study". *In Proceedings of 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON '04),* Santa Clara, CA, USA, Oct. 4-7, 2004, pp. 406-414.

[5] Evennou, F. and Marx, F. "Improving Positioning capabilities for indoor environments with WiFi", *IST Summit 2005*.

[6] Hightower, J., Borriello, G., and Want, R. "SpotON: an indoor 3D localization sensing technology based on RF signal strength", *Technical Report #2000-02-02, University of Washington, Computer Science and Engineering*, Feb. 18, 2000.

[7] Kalman R.E., A New Approach to Linear Filtering and Prediction Problems, Transaction of the ASME-Journal of Basic Engineering, pp. 35-45, Research Instutute for Advanced Study, Baltimore, Md., USA, March 1960.The Univerisity of Michigan, USA, 2005.

[8] King, T., Haenselmann, T., Effelsberg, W.: Deployment, Calibration, and Measurement Factors for Position Errors in 802.11-based Indoor Positioning Systems. *In: Hightower, J., Schiele, B., Strang, T. (eds.) LoCA 2007*. LNCS, vol. 4718, pp. 17–34. Springer, Heidelberg (2007)

[9] Kung, H. T., Lin, C., Lin, T., and Vlah, D. "Localization with snap-inducing shaped residuals (SISR): coping with errors in measurement". *In Proceedings of the 15th Annual international Conference on Mobile Computing and Networking (MobiCom '09),* Beijing, China, Sep. 20-25, 2009, pp. 333-344.

[10] LI X., SHI H., SHANG Y.: 'A sorted RSSI quantization based algorithm for sensor network localization'. *Proc. 11th Int. Conf. Parallel and Distributed Systems (ICPADS'05)*, Fuduoka, Japan, 2005, pp. 557–563

[11] Liu, C., Scott, T., Wu, K., Hoffman, D., Range free sensor localization with ring overlapping based on comparison of received signal strength indicator, *International Journal of Sensor Networks 2* (2007) 399–413.

[12] Lorincz, K. and Welsh, M. , "MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking", *In Proceedings of the International Workshop on Location- and ContextAwareness (LoCA 2005) at Pervasive 2005*, Oberpfaffenhofen, Germany, May 2005.

[13] Luo X. et al., Comparative evaluation of Received Signal-Strength Index (RSSI) based indoor localization techniques for construction jobsites, Adv. Eng. Informat. (2010), doi:10.1016/j.aei.2010.09.003

[14] Motley, A. and J. Keenan, "Personal communication radio coverage in buildings at 900 MHz and 1700 MHz," *Electronics Letter*, vol. 24, June 1988.

[15] Parameswaran, A. , Mohammad Iftekhar Husain and Shambhu Upadhyaya, "Is RSSI a Reliable Parameter in Sensor Localization Algorithms - An Experimental Study", *Field Failure Data Analysis Workshop (F2DA'09)*, New York, 2009.

[16] Parthornratt , T. and Techakittiroj , K., "Improving Accuracy of WiFi Positioning System by Using Geographical Information System (GIS)," *2006 WTS '06 Wireless Telecommunications Symposium* , pp.1-6, April 2006.

[17] Patwari, N., Hero, A. O., Costa, J. A. "Learning Sensor Location from Signal Strength and Connectivity", *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*. Advances in Information Security Series, vol. 30,Springer, 2006.

[18] Priyantha N., Chakaborty A., Balakrishnan H., "The Cricket Location-support System", *ACM Mobicom Conference*, Boston, MA, August 2000.

[19] Roos, T., Myllymaki, P., Tirri, H. "A statistical modeling approach to location estimation". *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, Mar. 2002, pp. 59-69.

[20] Stoleru, R.; Tian He; Stankovic, J.A., "Walking GPS: a practical solution for localization in manually deployed wireless sensor networks," *29th Annual IEEE International Conference on Local Computer Networks*, vol., no.pp. 480- 489, 16-18 Nov. 2004.

[21] Whitehouse, K., Karlof, C., Woo, A., Jiang, F., and Culler, D. "The effects of ranging noise on multihop localization: an empirical study". *In Proceedings of the 4th inter-national Symposium on information Processing in Sensor Networks (IPSN '05),* LosAngeles, California,USA, Apr. 24-27, 2005, pp. 73-80.

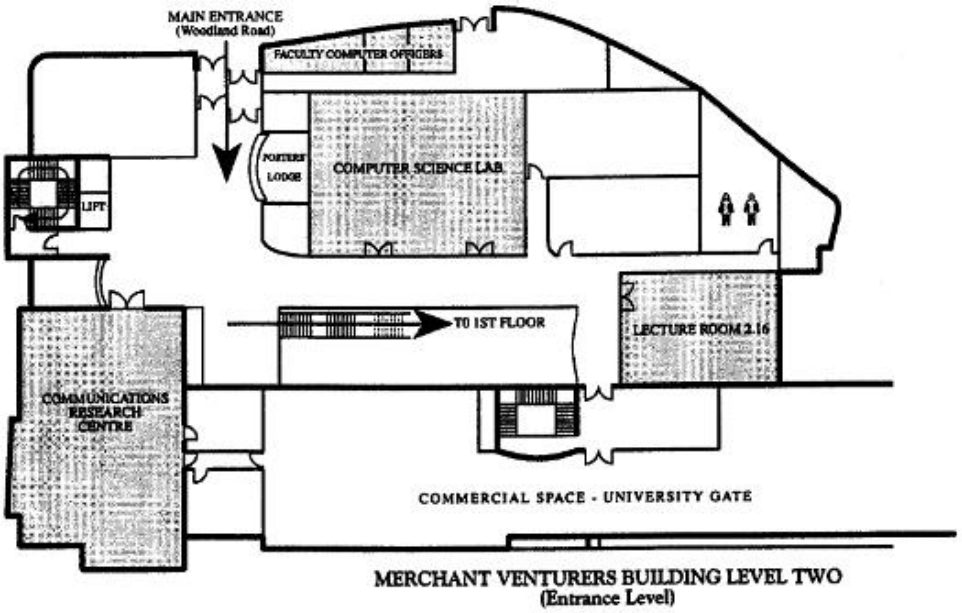
[22]Yedavalli, K. and Krishnamachari, B. "Sequence-Based Localization in Wireless Sensor Networks*". IEEE Transactions on Mobile Computing*, vol. 7, no. 1, Jan. 2008, pp. 81-94.

[23] Youssef, M., Youssef, A., Rieger, C., Shankar, U., and Agrawala, A. "PinPoint: an asynchronous time-Based location determination system". *In Proceedings of the 4th international Conference on Mobile Systems, Applications and Services (MobiSys '06),* Uppsala, Sweden, Jun. 19-22, 2006, pp. 165-176.

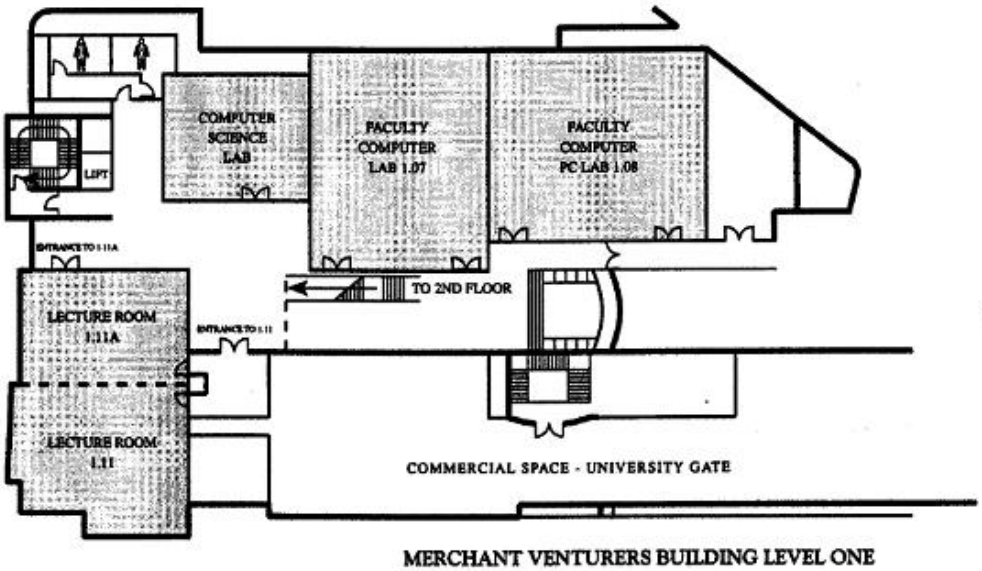
[24] Zaruba, G. et al., "Indoor location tracking using RSSI readings from a single Wi-Fi access point," *Wireless Networks*, vol. 13, no. 2, pp. 221–235, 2007.

[25] Zhong, Z., Range-Free Localization And Tracking in Wireless Sensor Networks, Available at: *http://www-users.cs.umn.edu/~zhong/papers/dissertation.pdf*

## Appendix A – Merchant Venturers Building floor plan




MERCHANT VENTURERS BUILDING LEVEL TWO
(Entrance Level)




MERCHANT VENTURERS BUILDING LEVEL ONE

## Appendix B – getData() method in Collector class

```
public void getData()
{
   for(int i=0;i<routers.size();i++)
   {
      try {
          Router router = routers.get(i);
           url = new URL(("http://" +router.getIP() + "/Wiviz.live.asp"));
           urlConnection = url.openConnection();
           urlConnection.setRequestProperty("Authorization", "Basic " +
authStringEnc);
          InputStream is = urlConnection.getInputStream();
               InputStreamReader isr = new InputStreamReader(is);

          int numCharsRead;
               char[] charArray = new char[1024];
               StringBuilder sb = new StringBuilder();
               while ((numCharsRead = isr.read(charArray)) > 0)
          {
                              sb.append(charArray, 0, numCharsRead);
               }

               webPage = sb.toString();
          isr.close();
          is.close();
          DataParser parse = new DataParser(webPage);
```

```
            parse.parse();
            ArrayList<Node> nodes = parse.getNodes();
            router.addNodes(nodes);
                routers.set(i, router);


        }   catch (MalformedURLException e) {
                } catch (IOException e) {
                }
    }
}
```

## Appendix C – getData() method in Collector2 class

```
public void getData() throws SocketException, IOException, NullPointerException
  {
     for(int i=0;i<routers.size();i++)
     {
         Router router = routers.get(i);
         Thread reader, writer;
         final TelnetClient telnet = new TelnetClient();
         telnet.connect(routers.get(i).getIP(), 23);
         int ch;
         final StringBuilder sb = new StringBuilder();
         reader = new Thread()
         {
           @Override
           public void run()
           {
              try {
                 sleep(300);
              } catch (InterruptedException ex) {
Logger.getLogger(Collector2.class.getName()).log(Level.SEVERE, null, ex);
              }
              try {
                 final OutputStream out;
```

```
            int ch;
            StringReader user = new StringReader("root\n");
            StringReader pass = new StringReader("admin\n");
            StringReader readfile = new StringReader("cat ../wiviz2-
dump\n");
            out = telnet.getOutputStream();
            while (!interrupted() && (ch = user.read()) != -1) {
               out.write(ch);
               out.flush();
            }
            try {
               sleep(300);
            } catch (InterruptedException ex) {
Logger.getLogger(Collector2.class.getName()).log(Level.SEVERE, null, ex);
            }
            while (!interrupted() && (ch = pass.read()) != -1) {
               out.write(ch);
               out.flush();
            }
            try {
               sleep(300);
            } catch (InterruptedException ex) {
Logger.getLogger(Collector2.class.getName()).log(Level.SEVERE, null, ex);
            }
```

```
                while (!interrupted() && (ch = readfile.read()) != -1) {
                    out.write(ch);
                    out.flush();
                }
            } catch (IOException ex) {
Logger.getLogger(Collector2.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    };
    writer = new Thread()
        {
            @Override
            public void run()
            {
                final InputStream in = telnet.getInputStream();
                InputStreamReader isr = new InputStreamReader(in);
                int numCharsRead;
                char[] charArray = new char[1024];
                //StringBuilder sb = new StringBuilder();
                try
                {
                    while ((numCharsRead = isr.read(charArray)) > 0)
                    {
                                sb.append(charArray, 0, numCharsRead);
```

```
                }
            }
            catch (IOException e)
            { }
        }
    };
    writer.setPriority(Thread.currentThread().getPriority() + 1);
    writer.start();
    reader.setDaemon(true);
    reader.start();
    try
    {
      writer.join(1500);
      reader.interrupt();
    }
    catch (InterruptedException e)
    {}
    telnet.disconnect();
    //System.out.println(sb.toString());
    DataParser parse = new DataParser(sb.toString());
    parse.parse();
    ArrayList<Node> nodes = parse.getNodes();
    router.addNodes(nodes);
    routers.set(i, router);
}
```