WIFI Location System Investigation

Final Report for COMP420Y

Sam Bartels

Supervisor: Dr Alan Holt

Department of Computer Science

Hamilton, New Zealand

October 28, 2005

Abstract

This project aims to investigate the accuracy of distance measured using the time-of-flight of 802.11 packets; distances obtained could then be used to determine location. By undertaking preliminary analysis of standard ICMP pings, it was discovered that software timing was too imprecise and subjected to various non-deterministic delays to be useful for measuring distance. Using custom built 802.11 hardware, a methodology of extracting accurate time-of-flight data by getting closer to the radio was designed and implemented, and an accuracy of 3 metres was obtained.

Acknowledgements

The author would like to thank the following people for their contributions to this project.

Dr Alan Holt - for supervising this project, providing direction and contributing whole heartedly to the project.

Dean Armstrong - for assisting in the author's understanding of the WAG cards, and for help with VHDL and C.

Ilze Ziedins - for help understanding statistical techniques and how they apply to the project.

WAND Group - for providing help in all manners and forms, such as help with LaTeX, feedback on reports/presentations and assistance with Pings.

And many thanks to friends and family who provided support and kept the author sane.

Contents

List of Figures

List of Tables

List of Acronyms

ACF ..... Autocorrelation Function

CCA ..... Clear Channel Assessment

CRC ..... Cyclic Redundancy Checksum

CVS ..... Concurrent Version Control System

FCC ..... Federal Communications Commission

FPGA ..... Field Programmable Gate Array

GIS ..... Geographical Information System

GPS ..... Global Positioning System

HDL ..... Hardware Description Language

ICMP ..... Internet Control Message Protocol

IP ..... Internet Protocol

LAN ..... Local Area Network

MAC ..... Media Access Layer

PACF ..... Partial Autocorrelation Function

PCI ..... Peripheral Component Interconnect

TCP ..... Transmission Control Protocol

VHDL ..... Very High Speed Integrated Circuit Hardware Description Language

WAG ..... Wireless Analysis and Generation

Chapter 1

Introduction

This project is to develop a WIFI location determination system based on the time-of-flight of 802.11 packets using customised wireless hardware developed by the WAND group. Determining location is simple as long as accurate measurement of packet propagation delays between wireless devices can be taken. Distance can then be computed from the propagation delay. Hence it is accurate distance measurement that is the vital component of this project and therefore the focus of this report.

1.1 Motivation

Extensive research is currently underway in this area. Cellphone companies in the United States have to include location determination capabilities in their products by law, so they are currently investigating their options. A recent investigation by Microsoft produced what they call RADAR, which is an in-building RF-based User Location and Tracking System. However many of these systems have poor accuracy and hence this project aims to improve this accuracy using alternative techniques.

The beauty of wireless is its portability. Wireless networks can be built almost anywhere whether it be in a building or spanning a mountain range. Thus

there are numerous applications of a wireless location system that can cater to the environment of the wireless network. Identifying location and measuring distance would be useful to many people and groups such as the WAND group at Waikato University.

1.2 Report Structure

This report has five chapters; the first of which is this introduction. The second chapter is about the initial investigation which provides background information and preliminary analysis conducted by the author. The third chapter explains the development phase including the design and implementation. Chapter four focuses on results and provides some analysis also. Finally chapter five concludes the report and looks at future work.

Chapter 2
Initial Investigation

This chapter focuses on important background information and some preliminary analysis that was carried out as part of this project. Fundamental distance determination techniques are discussed as a lead-in to location techniques. Existing location determination systems such as GPS and Microsoft RADAR are then examined to give some background on what is trying to be accomplished. Applications of location determination are then discussed, followed by a brief explanation of the WAG cards, which are the basis for this project. Though vital to the project, little information on them is needed. Some explanation of the software tools used is then given, and finally the chapter ends with some ICMP ping analysis that was undertaken by the author.

2.1 Distance Techniques

Distance is fundamental to determining location, hence a suitable distance methodology must be selected. There are two general methods; Time-of-Flight and Signal Attenuation.

Time-of-Flight measures the time it takes for a radio wave to travel from point A to point B. Since we know that radio waves travel at the speed of light, we can easily calculate a distance from this. In digital electronics time

is measured in discrete units, which is limited by the frequency of the clock. A high-resolution clock is required to measure Time-of-Flight accurately due to the high velocity of light and a suitable clock may not be available in some situations for various reasons. Reflections of radio waves can cause problems as it is impossible to tell the difference between direct and reflected radio waves. However by taking multiple measurements and taking the minimum it should be possible to eliminate them. Time synchronisation can also be an issue depending if the transmitter and the receiver are ultimately the same object or not.

Signal attenuation is a measure of how strong the signal strength is at a particular distance. Signal strength is inversely proportional to the distance squared and hence has an exponential decay relationship. This decay is due to path loss; path loss is any reduction in signal strength due to transmission and consists of two components. One component is loss due to obstacles and other environmental factors. The other component is free-space path loss, which is the loss encountered in free space ie. no obstacles or other loss-causing factors are present.

$$Loss(dB) = 32.4 + log_{10}f + log_{10}d \tag{2.1}$$

In free space, the path loss is defined as expression 2.1 where f is the frequency in MHz and d is the distance in km. Figure 2.1 shows this relationship. It is possible to get an estimation of distance, however to achieve any kind of accuracy, the environment must be taken into consideration. Once again reflection causes problems and different types of objects have different effects on signal attenuation. Time-of-Flight is not as susceptible to environmental changes and hence is usually more accurate for determining distance than signal attenuation.

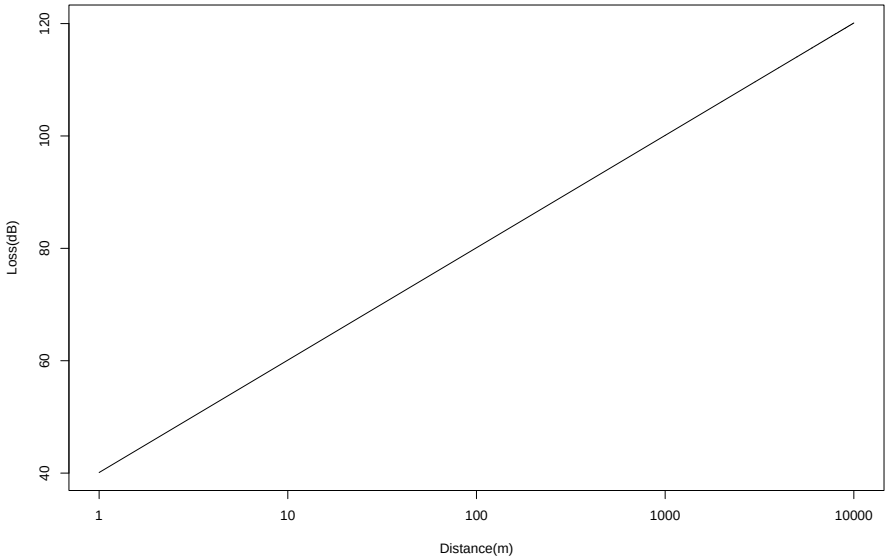

Figure 2.1: Free-Space Loss - Distance Relationship

2.2 Location Techniques

Triangulation uses the geometric properties of triangles to determine location. Triangulation consists of two parts; lateration, which uses distance measurements, and angulation, which makes use of angles as well as distances. Lateration can compute a location using distances from multiple reference points. An objects position can be calculated in 2D with the distances to three non-collinear points, and in 3D, four points would be required. An example of lateration in 2D is shown in figure 2.2. As can be seen, with distances from three known points, it is a simple task of finding the intersection point of all three circles. However the intersection of these three circles will not always meet at one exact point and some further analysis may be required.

Angulation in 2D requires two angle measurements and one distance measurement, in 3D the same is required but in addition an azimuth measurement is

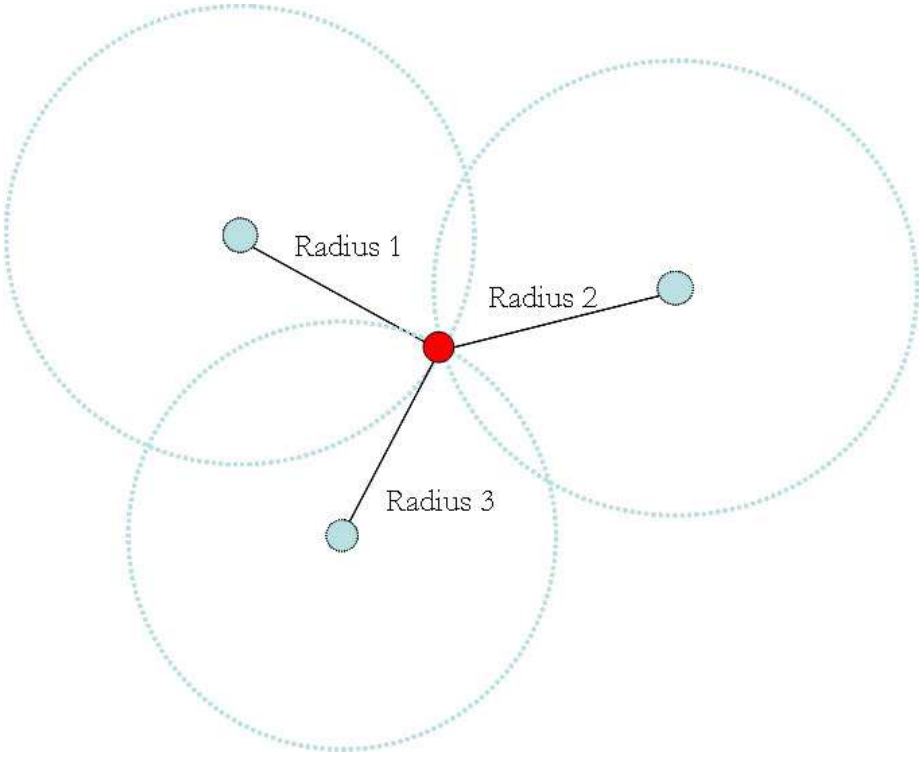

Figure 2.2: Lateration example

needed. However to measure an angle, multiple antenna spread certain distances apart are required. Each antenna knows the time of arrival for the packet and given the differences between these timings it is possible to compute the angle at which the signal originated. Angulation can only be calculated if there are enough antennas to give an accurate angle, hence lateration is a better choice as most devices only have one antenna.

2.3 Location Determination Systems

2.3.1 GPS

GPS is a US government satellite navigation system that consists of 29 satellites, each with an orbit of 12 hours. Hence 8-12 satellites should be visible from any one point on Earth. "Each satellite broadcasts spread spectrum signals at 1575.42 and 1227.6 MHz, also known as L1 and L2, respectively. Currently the civil signal is broadcast only on L1. The signal contains two components: a time code and a navigation message. By differencing the received time code with an internal time code, the receiver can determine the distance, or range, that the signal has travelled." [16]

There are four unknown variables that need to be determined in order to calculate a location: three position co-ordinates (longitude, latitude, height) plus the offset of the receiver clock. The position co-ordinates can be calculated relatively easily from the determined distances, however they are unlikely to be correct. This is because the GPS signal has travelled through the Earth's atmosphere which delays the signal in various ways. The ionosphere is the top layer of the Earth's atmosphere and is electrically charged, which introduces delay into the signal. This can be compensated for because the delay is frequency-dependant, and hence if data is available on both L1 and L2, the delay can be computed.

Multi-path reflections also occur as well as other more complicated errors, some of which can be approximately compensated for, and some of which cannot. Hence the determined distances will be slightly inaccurate and result in an inaccurate location, giving an accuracy of about ten metres. This accuracy can be improved by using differential GPS. A nearby receiver is used as a reference, then by differencing received measurements to the reference receiver the errors common to them both can be eliminated. GPS works well as a global location system, however since it requires clear line-of-sight, GPS is unusable in dense

environments such as indoors and urban areas with many tall buildings.

2.3.2 Active Bat

Active Bat is a location system developed by AT&T and uses ultrasound time-of-flight lateration. Designed for indoor use, several receivers are mounted in a grid-like fashion on the ceiling. Active Bat tags attached to objects emit an ultrasonic pulse to the receivers in response to a request sent from the local controller (controls the receivers and can communicate with the tags). At the same time as the controller was issuing the request it also sent a synchronised reset to all the receivers. Each receiver determines the time delay between the reset and the arrival of the ultrasound pulse. These timings can then be passed on to a central controller for lateration calculation. The tags, called Bats, can be located within 9cm of their true location for 95 percent of the measurements. However because the receivers form a large rigid structure, implementation is costly, scaling is poor and deployment is difficult.

2.3.3 Microsoft RADAR

RADAR is a radio-frequency based location system developed by Microsoft. It operates by measuring signal strength at multiple base stations; these signal strengths are then compared to a signal strength database to determine a distance and then location can be computed. "RADAR operates by recording and processing signal strength information at multiple base stations positioned to provide overlapping coverage in the area of interest."[3]. This has two advantages; only three or four base stations are required, and the infrastructure can easily be put in place as wireless LANs can be set up very easily. However the use of signal strength also causes problems as it is very susceptible to environmental interference. RADAR has a 4.3 metre accuracy with a 50 percent probability.

2.4 Location Applications

GPS is a widespread location determination system in use today, however it is possible with the discovery of more accurate techniques, some of the applications for location determination discussed below could change to use the newer technique.

2.4.1 Military Applications

Determining location is a vital tool for the military. It is commonly known that the military use GPS for their missile guidance systems however there are many other benefits of knowing location. GPS receivers are fast replacing soldiers' compasses so troops can determine their position and avoid straying in to enemy territory. Special forces can use GPS to draw air and artillery fire on enemy targets with deadly precision. Potential targets can be tracked as proof they are hostile and this tracking data can be supplied to the missile guidance system for target termination. GPS is invaluable for rescue missions; it reduces response time and thus saves lives. GPS is also useful in updating maps with the location of various military units and their targets and destinations.

2.4.2 Civilian Applications

Location determination also has important civilian applications, such as the introduction of FCC Docket 94-102 in the US. This docket has been adopted as an official report and order, which officially mandates that 911 cellphone callers receive the same level of emergency service as that of wired callers (those using a standard telephone as opposed to a cellphone). The location of a wired 911 caller can be found using a reverse telephone directory. This allows the callers phone number to be used as the search term and the address will be returned. This is known as Enhanced 911 in the US. FCC 94-102 specifies that cellular, personal communications services and certain mobile radio service providers must provide a means of locating 911 callers to within 125 metres in 67 percent

of all measurements.

Location is already in use in many civilian situations. GPS receivers are used by hikers to avoid getting lost, in the same way people in cars or boats use a GPS navigation system. Civilian aircraft are primary users of GPS for navigation, as are large ships such as ocean liners and oil tankers. Some companies have even developed implants for family pets so that the owner can track where the pet is. GPS is also of great use when carrying out any form of mapping, whether it be a whole organisation managing a geographical information system (GIS) or a hiker planning their route.

2.5 WAG V2 Network Cards

The WAG V2 network card is the backbone of this project. Dean Armstrong is a Ph.D student at Waikato University and has recently produced version 2 of his Wireless Analysis and Generation card, otherwise known as WAG. The aim of his Ph.D is to develop a mini-PCI network card that can be used for MAC layer research. The card has a PCI bus and has a programmable FPGA that provides the ability to customise the firmware. As the WAG card is still a work in progress, there were a couple of issues; the main one of relevance to this project was that the WAG cards have poor range capability, hence distance measurements were of a limited range. Figure 2.3 shows the block layout of the card.

WAG Toplevel is the WAG card itself. As can be seen in figure 2.3 WAG core contains four of the components. The functionality of the block components that make up WAG Toplevel will now be briefly explained.

WAG Core contains all the vital components for communication, these components are heavily reliant on each other.

Serialise is responsible for serialising(sending) packets.

Deserialise is responsible for deserialising(receiving) packets.

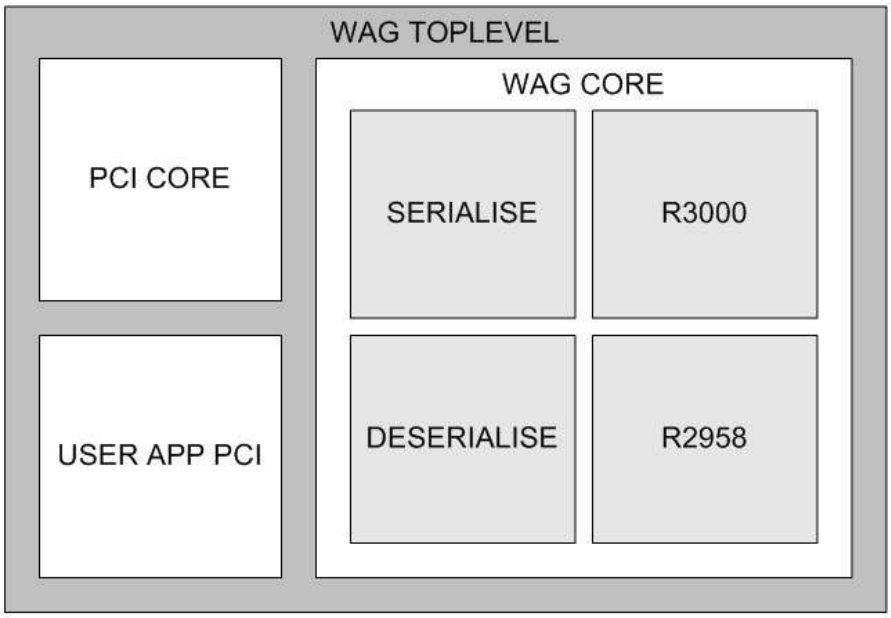

Figure 2.3: WAG Block Layout

R3000 is a CMOS base band processor for use in 802.11b and provides all functions to convert a spread-spectrum radio signal to a bit stream and vice versa.

R2958 is a single-chip transceiver that is specifically designed for 802.11b.

PCI Core is the component that communicates directly with the PCI bus.

User App PCI communicates with the PCI bus via the PCI core. This means that user applications only need to deal with what they require rather than all the PCI functionality available.

2.6 Software Tools

Several software tools were used as part of this project and a significant portion of time was spent learning to use them. Each of these tools will briefly be

described below.

2.6.1 VHDL and C

VHDL is a hardware-description language (HDL) that was used to implement the timestamp design. VHDL is relatively easy to write, however can be difficult to debug. C is a programming language that was used to write the Sounder and Distance programs. Some difficulty was experienced here as the author had a limited knowledge of C, however this knowledge has now been enhanced.

2.6.2 Riviera

Riviera is a HDL simulator that provides an easy to use editor and provides waveform diagrams as output. Riviera was easy to use and provided an advanced debugging environment which made VHDL much easier to debug.

2.6.3 Synplify and Xilinx

Synplify and Xilinx are tools used for synthesising HDL designs and placing/routing the synthesised design on an FPGA. They were straight-forward to use and are discussed in more detail in the 3.4 section.

2.6.4 CVS

CVS is a concurrent version control system that allows the user to record the history of source files. This is useful for a number of reasons; if something goes wrong, then the user can simply go back to the previous working version. Also CVS is helpful when a number of people are using the same files. CVS only stores the differences between file versions and thus is quite efficient with respect to disk space. CVS was confusing at first but once familiar with CVS, it is easy to use.

2.6.5 Awk

Awk is a clever pattern scanning and processing language that is simple to use. Awk was very handy for extracting numeric results from result files for analysis.

2.6.6 J and R

J is a programming language that has very powerful mathematical, statistical and logical analysis support. However J is also a bit difficult to use and hence was only used for simple mathematics. R is a powerful statistical software environment that is easy to use and makes graphing very simple. R was used to analyse results and draw all the graphs seen in this report. Both R and J are freely available.

2.7 Ping Analysis

Ping is a program that is used to determine if a destination on a TCP/IP network is reachable by sending an ICMP echo request and timing how long it takes to get a reply. Pings were examined to verify suspicions that timing in software would be very imprecise. The wireless network used for these tests was CRCnet.

CRCnet is a project that the WAND group from the University of Waikato has been working on for about the past four years. The aim of the project is to connect rural and remote communities using wireless networks. CRCnet consists of several nodes linked together in a tree-like structure. Most of these nodes have GPS co-ordinates and therefore the distance of most links is known. Since the link distances are known it was then a simple task of finding some links of different lengths and 'pinging' them. Using these ping times, the distances can be calculated and then compared to the actual distances to determine whether pings can estimate distance accurately.

Each link was 'pinged' one hundred times and then the minimum was selected to eliminate unwanted queueing delay. Propagation delay (d_{prop}) is the vital component to determining distance, hence a delay model was constructed as shown in equation 2.2. Ping returns a round trip time (d_{rtt}), which is the time that it takes for a packet to be sent and a corresponding reply be received.

$$d_{prop} = \frac{d_{rtt} - d_{proc} - d_{trans}}{2} \tag{2.2}$$

The processing delay (d_{proc}) was empirically determined by putting two Soekris boards back-to-back and 'pinging'. This resulted in a delay of 0.4ms. The serialisation delay (d_{serial}) was calculated using equation 2.3 (S refers to *size of* and β is *bandwidth*). S_P represents the payload size of the packet and the three latter terms refer to the header size of ICMP, IP and the MAC respectively.

$$d_{serial} = \frac{2 \times 8 \times (S_P + S_{ICMP} + S_{IP} + S_{MAC})}{\beta} \tag{2.3}$$

Then by substituting these results into the delay model (expression 2.2), the propagation delays for the pings could be determined and therefore distances could be calculated. As can be seen from the table below the calculated distances have a very weak correlation to the actual distances confirming our suspicions. Delays that pings encounter include scheduling by the operating system, stack handling, clear channel assessment (CCA) as well as those delays expressed above, plus many more. This makes it nearly impossible to get anywhere near a valid result.

Link	Min RTT(ms)	Actual Distance(km)	Calculated Distance(km)
1	3.1	2.19	373.3
2	3.1	5.39	373.3
3	3.5	7.05	432.9
4	3.3	13.7	403.1
5	3.3	17.4	403.1
6	2.3	23.0	254.1

Table 2.1: Ping results

Figure 2.4 shows the difference between the actual distances and those calculated.

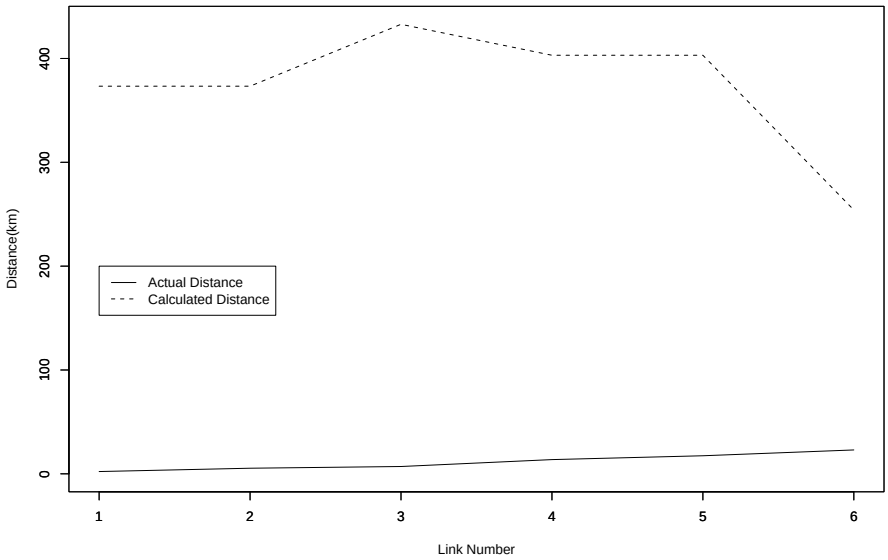

Figure 2.4: Actual and Calculated Distance for CRCnet Links

Chapter 3

Development

This chapter focuses on the design and implementation involved in developing a system to measure Time-of-Flight using WAG cards. The design stage is discussed first, and is followed by the implementation of the firmware and software. The chapter finishes by looking at the configuration process.

3.1 Design

The most important concept for design was that all delays must be identified and correctly compensated for or eliminated. The initial design model (equations 3.1 to 3.4) identified the delays but was not detailed enough to identify the smaller components that made up the delays.

$$d_{total} = d_{serial} + d_{prop} + d_{proc} \tag{3.1}$$

The serialisation delay (d_{serial}) can be defined as equation 3.2 where δ is packet size in bits, ϕ is the bit-rate in bits per second and ϵ_1 is the unknown time delay.

$$d_{serial} = \delta/\phi + \epsilon_1 \tag{3.2}$$

The propagation delay (d_{prop}) can be defined as equation 3.3 where λ is the distance travelled in metres, c is the speed of light in metres per second and ϵ_2 is the unknown time delay.

$$d_{prop} = \lambda/c + \epsilon_2 \tag{3.3}$$

Processing delay (d_{proc}) is unknown and therefore is defined as unknown time delay ϵ_3.

$$d_{proc} = \epsilon_3 \tag{3.4}$$

However when expanding this design model, the concept of identifying delays became very important. It became apparent that if the delays could be eliminated, there would be no need to try and identify the delays. This is where time stamping became fundamental to the whole project. Correct time stamping will allow most of the delay to be eliminated leaving only the propagation delay and minor other delays that can be measured in the testing process. Figure 3.1 below explains how time stamping will achieve this.

It is important to note that these timestamps are measured in WAG clock cycles. Hence when determining the flight time this must be remembered otherwise results will not make sense. Time stamping at these points then makes calculating the total flight time very simple. We can eliminate nearly all serialisation and processing delay with the following equation(with reference to figure 3.1).

$$TotalFlightTime = (D - A) - (C - B) \tag{3.5}$$

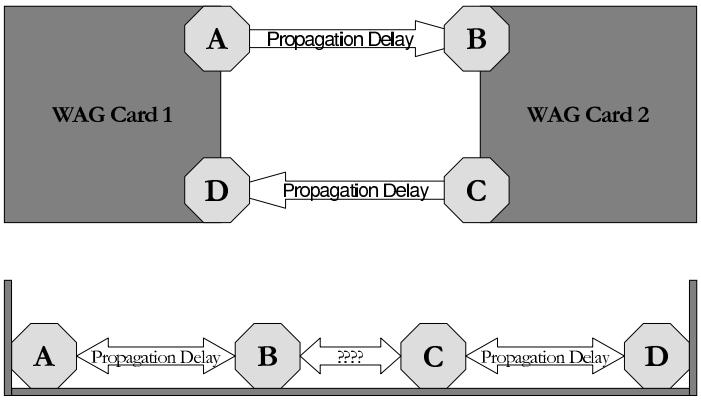

Figure 3.1: Timestamp Model

The total delay time can be determined by subtracting the start time(A) from the end time(D) which is (D - A). The turn-around delay is given by (C - B), so this can then be subtracted from the total delay time to give total flight time (as shown in expression 3.5). Most of the delay calculated from this equation will actually be due to delays in the radio circuitry itself, however this can be eliminated by timing over a distance as close as possible to zero (for example 1cm). The resulting delay can then be attributed to the radio circuitry.

3.2 Firmware

Time stamp capability was implemented in to the existing VHDL code that Dean Armstrong had written. This involved establishing a counter register which would increment every clock cycle, located in what is identified as the wag core. The wag core is the interface between the PCI bus and the MAC

layer. The actual timestamps need to be taken at the beginning and end of both the serialisation (transmission) and deserialisation (reception) of a packet so registers were created to do this.

To verify that these time stamps were working, simulations were run in Riviera, which is a VHDL simulation tool. This was invaluable because without simulation, error checking would have to be done in hardware and would be difficult and time-consuming to test. Simulations made it very simple to see the desired values and made checking for bugs very easy.

Time stamp values cannot be read directly from the registers themselves. Essentially the values to be read need to be passed on to the PCI bus and can then be accessed by the WAG driver. This is just a case of routing the timestamp registers in serialise and deserialise to the User App PCI(see figure 3.2). However simulations do not model real-life conditions, rather everything works almost perfectly. Some conditions could not be picked up in the simulation but were later noticed in the testing stage.

Figure 3.2 gives a block-level diagram of how the WAG card fits with the registers and counters. The operation of the timestamp process can be explained as follows. The Clock Cycle Counter increments for every pulse of the WAG clock.

1. A packet is passed from the software to the PCI.
2. The packet passes from the PCI in to the Serialise unit. A timestamp is taken such that the current value of the clock cycle counter is stored in the TX Start register.
3. The packet is then passed to the radio as the timestamp is extracted by the software via the PCI.
4. The packet is sent by the radio.

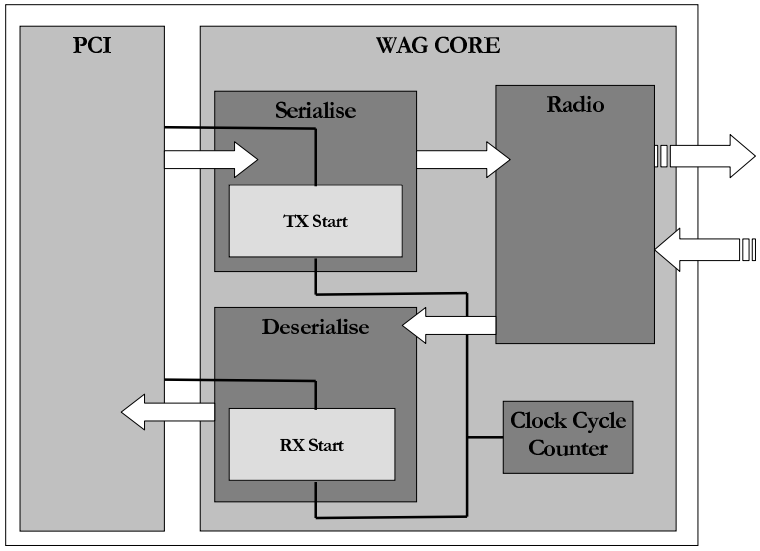

Figure 3.2: Timestamp Firmware

5. Some time later the radio will receive a packet.

6. This packet is passed into the Deserialise unit. A timestamp is taken such that the current value of the clock cycle counter is stored in the RX Start register.

7. The packet is then passed to the PCI as the timestamp is extracted by the software also via the PCI.

8. The packet is passed from the PCI to the software.

3.3 Software

With the firmware in place, software is needed to extract the timestamps. A driver is required to interface between the firmware and the user applications. The driver was also written by Dean Armstrong as it is a slight modification on

the original WAG driver. The modification just gives it the ability to read time stamps from the PCI bus and pass them to the user applications that require them. Two user applications were written to use the time stamps and produce some results.

Sounder is simply a program that listens for packets. When a packet is received it will be time stamped and then a time stamped reply will be sent back. It was assumed that these timings would be constant however they proved to vary quite significantly. This resulted in a change in design as since the timestamps had to be sent in a packet, error checking would be required. Flow control was also needed as if a packet timed out the system would crash. Distance is the program that does all the work and operates in conjunction with the Sounder in the following manner.

1. Distance sends a probe packet to the Sounder (time stamped on serialisation)
2. The Sounder timestamps when it receives the probe packet
3. The Sounder sends a empty reply packet(time stamped on serialisation) back to Distance which timestamps when the packet is received.
4. A short time later the Sounder sends another packet which contains the timestamps and a CRC for error checking from the two previous packets that were sent and received.
5. Distance then receives this packet, checks the CRC and if the CRC is correct then the arithmetic can be performed to extract the flight time.

A timeout or error will restart this process so everything correlates correctly. Timings fluctuated more than expected so this why the sounder must send its timestamps to Distance. The timestamps cannot be sent in the reply packet from the Sounder because the timestamps are not valid until one clock cycle after that packet has finished being sent. The packet structure can be seen in

figure 3.3. Full code for Distance and Sounder can be found in appendices B and C.

Figure 3.3: Packet Structure

Some problems were encountered while writing these programs that were not previously observed. Initially the packets were too small for the radio One such problem had to do with the counters on the WAG cards; these would automatically wrap to zero when they reached their maximum count. Initially Distance and Sounder did not take this in to account, and every so often a very large time-of-flight would be calculated. This was fixed by checking the two timestamps and if the first one was larger than the second one then a wrap had occurred. Another problem was packet sequencing; the sequencing worked correctly until a packet was lost. This was because the sequencing design was too complicated; sequencing was been done at both ends when it only needed to be done at the Distance end. This is how Sounder got its name; it has no sense of state, it just 'reflects' any packets it receives. This allows multiple tests to be done without having to restart Sounder. Timeouts were optimised to make the testing process less tedious; this was done by testing with different timeouts and ensuring that the results were still valid.

3.4 Configuration

Three configuration tools were used to configure the WAG FPGA's. Synplify is a logic synthesis tool developed by Synplicity. Using their "proprietary Behaviour Extracting Synthesis Technology (B.E.S.T.) the tool converts the HDL into small,high-performance, design netlists that are optimised for popular technology vendors"[13]. Xilinx is one of these technology vendors and provides two tools to complete the process. Xilinx ISE takes the output from Synplify and

decides the best way to place and route components on the FPGA. A programming file is then produced containing all this information.

The final and simplest part of the process is using Impact, the other Xilinx tool. Impact simply uses the produced programming-file and uses it to program the FPGA via a special Xilinx Platform cable that plugs into the WAG card.

The WAG cards are not stand-alone (though this is planned) and therefore are mounted in what are called Soekris computers. This project uses the Soekris NET4526 which is a low-power, low cost advanced communication computer based on a 486 processor. "It has one 10/100 Mbit ethernet ports, up to 128 Mbyte SDRAM main memory and uses a CompactFlash circuit soldered on-board for program and data storage. It can be expanded using up to two MiniPCI type III boards."[2]

Two Soekris boards with a WAG card in each were setup using Impact. The boards then had to be reset to force a re-scan of the PCI-bus with the newly programmed FPGA. As yet none of the required software is on the Soekris's so the user must remotely login and load the driver module. The Soekris's mount a local machine on the network as their root file system and this is where the user applications are located.

Chapter 4

Results and Analysis

This chapter is all about the results that were obtained and how to validate them. Firstly a model was derived so the results could be compared. This in turn identified that some unknown delay occurs in the radio, which is looked at in the next section. Finally some actual results are examined and then analysed to identify the underlying statistical model.

4.1 Results Model

It is important that some kind of model be constructed so that it can be seen how the results originated. Taking into account that radio waves travel at the speed of light c, it should be possible to estimate the distance d measured as long as some time metric is used. Timestamping is done on the edge of the WAG clock, which has a frequency f of 44MHz, hence using $p = 1/f$ it can be determined that the period p of one clock cycle is about 22 nanoseconds. With this in mind, the distance travelled in one clock cycle can be determined.

$$d = c \times \frac{1}{f} = 2.998 \times 10^8 \times \frac{1}{44 \times 10^6} \times 10^8 = 6.8m \tag{4.1}$$

Hence it has been determined that a radio wave can travel 6.8 metres in one clock cycle. A model can now be derived using this distance that will compute

the measured number of clock cycles for a given distance. Since it is actually a round trip being measured, the distance is doubled. The number of clock cycles (t_{cc}) can now be determined as $2d/6.8$ however this does not take into account the unknown delay(d_{radio}) that occurs in the analog part of the WAG card, hence the model becomes:

$$t_{cc} = \frac{2d}{6.8} + d_{radio} = \frac{d}{3.4} + d_{radio} \tag{4.2}$$

This model can also be reversed to give a distance for a certain number of clock cycles:

$$d = (t_{cc} - d_{radio}) \times 3.4 \tag{4.3}$$

4.2 Determining Radio Delay

Even though timestamping has eliminated the majority of delays from the time of flight, there is still some delay left. This delay occurs because timestamping can only be done in the digital part of the WAG circuit, hence the delay through the analog radio cannot be measured in this way. However if two WAG cards are place back-to-back, a time-of-flight test can be done; since no distance is involved, the resulting number of clock cycles can be attributed to the radio delay d_{radio}. To determine d_{radio} nearly 20,000 tests were done.

Clock Cycles	410	411	412	413	Median	Mean
Frequency	36	14367	5566	14	411	411.3

Table 4.1: Back-to-back Distribution

As can be seen from these results, the back-to-back delay is about 411 clock cycles. However due to no clock synchronisation, this delay has also been observed as 412 clock cycles. This variation seems to occur because the two clocks can be out of phase by up to one clock cycle. Since four time stamps are taken, the delay can vary by up to four clock cycles, which is why values between 410

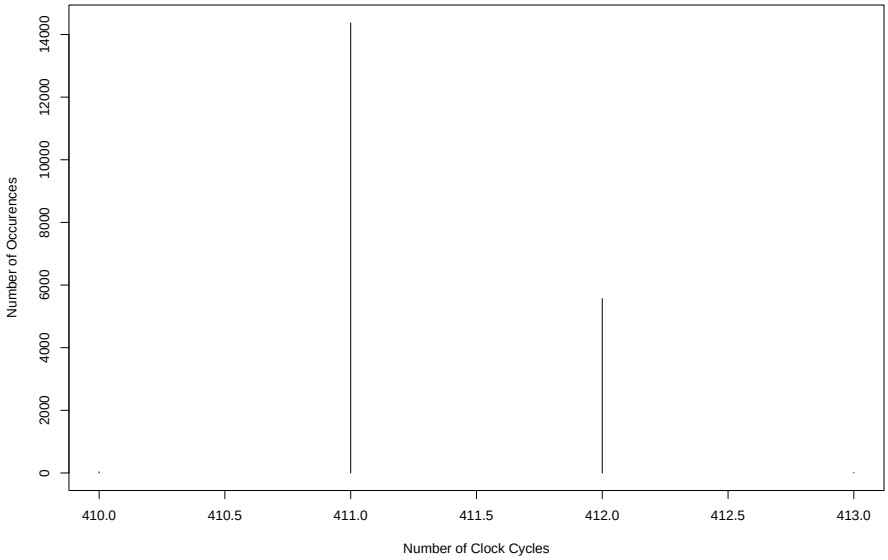

Figure 4.1: Distribution of radio delay

and 413 are observed.

4.3 Results

Given the results below it is clear that there is a strong correlation between distance and the time-of-flight. There is some variation because the clocks on the two WAG cards are not synchronised with each other. The medians are nearly unique for each distance measured and the means are quite distinguishable.

Clock Cycles	412	413	414	415	416	417	418	Median	Mean
3 Metres	171	783	45	0	0	0	0	413	412.9
6 metres	0	108	842	50	0	0	0	414	413.9
9 metres	0	8	299	633	59	1	0	415	414.7
12 metres	0	0	1	163	775	61	0	416	415.9
15 metres	0	0	0	2	387	603	8	417	416.6
18 metres	0	0	0	0	31	698	270	417	417.2

Table 4.2: Results Distribution

The cumulative frequency graph (figure 4.2) below depicts the distributions well. In general, it is easy to pick a distance by its median value, however each distance clearly has its own distribution. The graphs that make up this distribution are located in appendix A.

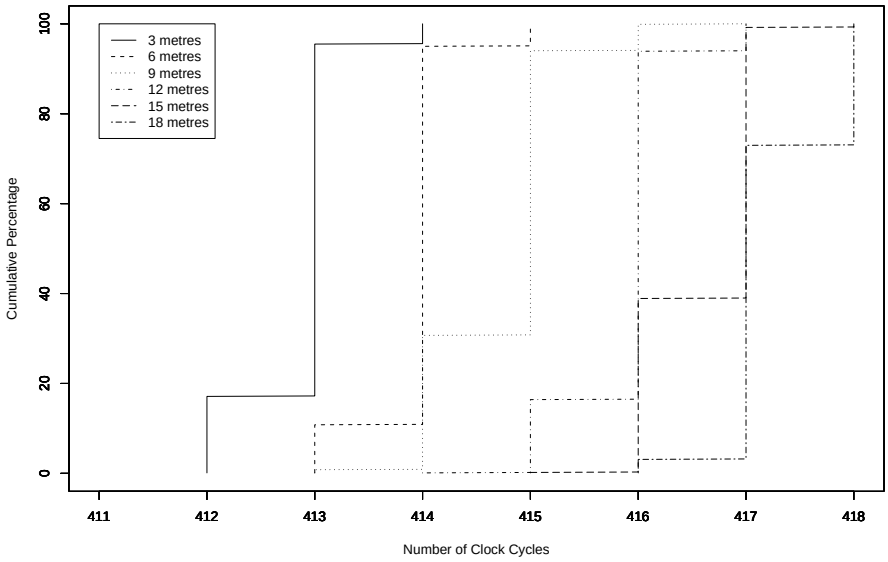

Figure 4.2: Distribution of Results

4.4 Analysis

This section looks at some analysis of the captured data. Firstly the concept of time series is discussed, which is then followed by some time series analysis techniques.

4.4.1 Time-series

A time series is defined as a sequence of data points measured over time such that the data points are taken at fixed time intervals and hence are uniformally

spaced apart. The results obtained from this project form a time series and hence time series analysis can be performed.
Time series analysis has two distinct parts: understanding the underlying model, and making forecasts/predictions. At present the underlying model is of more interest and hence will be focus of this analysis.

The underlying model looks at where the data came from and how it was generated. One common model for time series data is the Moving Average model which will now be discussed.

4.4.2 Moving Average

A moving average infers that the value of a time series at time t is influenced by previous error terms in the time series. A mathematical way of explaining this is

$$Y_t = e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \ldots - \theta_n e_{t-n} \tag{4.4}$$

A moving average is defined by expression 4.4 where Y_t is the current value, e_t is an independent Gaussian random variable at time t and $e_{t-1} \ldots e_{t-n}$ are past e values to which the coefficients $\theta_1 \ldots \theta_n$ are applied. The order of a moving average determines how many of the past terms Y_t is dependant on. For example, an order of one means that only the previous e term has an effect on the current value (expression 4.5 is an order one moving average).

$$Y_t = e_t - \theta e_{t-1} \tag{4.5}$$

A moving average can be observed by performing an autocorrelation on the time series data. The autocorrelation function is used to identify dependencies over time in a time series and also to determine if the data is stationary; stationary data has a mean and variance that are invariant over time.

Autocorrelation can be explained statistically as follows. Covariance is a measure of how two values relate to each other; expression 4.6 gives the covariance of the time-series against a lagged version of itself, known as the autocovariance $\gamma(k)$.

$$\gamma(k) = E[(X_i - \mu)(X_{i+k} - \mu)] \tag{4.6}$$

where E is the expectation operator, X_i is the time-series, μ is the mean and k is the lag. The autocorrelation is the normalised covariance which is $\gamma(k)$ divided by the variance σ^2 as shown in expression 4.7.

$$R(k) = \frac{\gamma(k)}{\sigma^2} \tag{4.7}$$

A partial autocorrelation function identifies the relationship between a current value and previous values, which can be calculated by regressing Y_t against past values of itself; this can be seen in expression 4.8 which has order n.

$$Y_t = b_0 + b_1 Y_{t-1} + b_1 Y_{t-1} + \ldots + b_n Y_{t-n} \tag{4.8}$$

The results obtained from this project gave no indication of a moving average, however when the data was differenced, a moving average model was clearly visible. Differencing is a simple statistical technique that removes trend from a time series; this is done by taking the difference between all adjacent data values and using those differences as a new data set. The ACF and PACF for this differenced data set can be seen in figures 4.3 and 4.4 respectively. The large downwards spike in the ACF graph shows that the model for this data is a moving average of order one. This also shows that the data has some short-range dependencies.

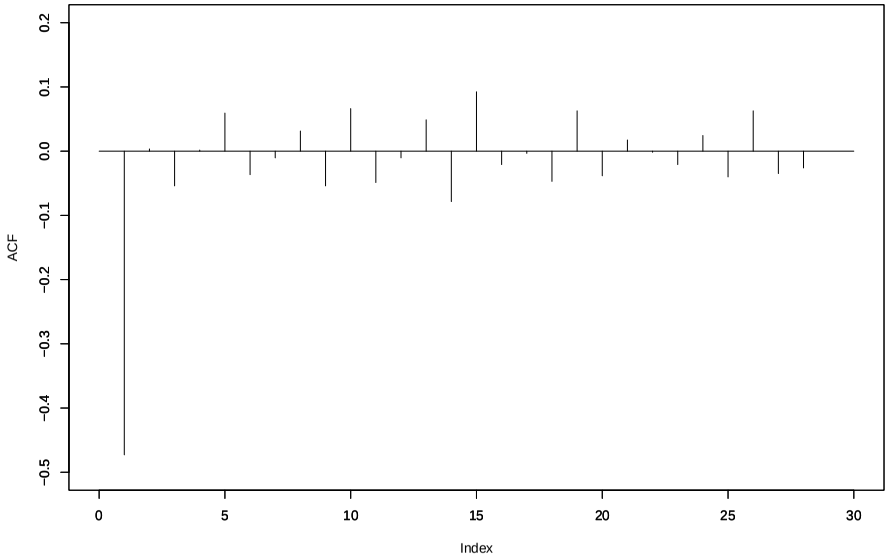

Figure 4.3: ACF of Data

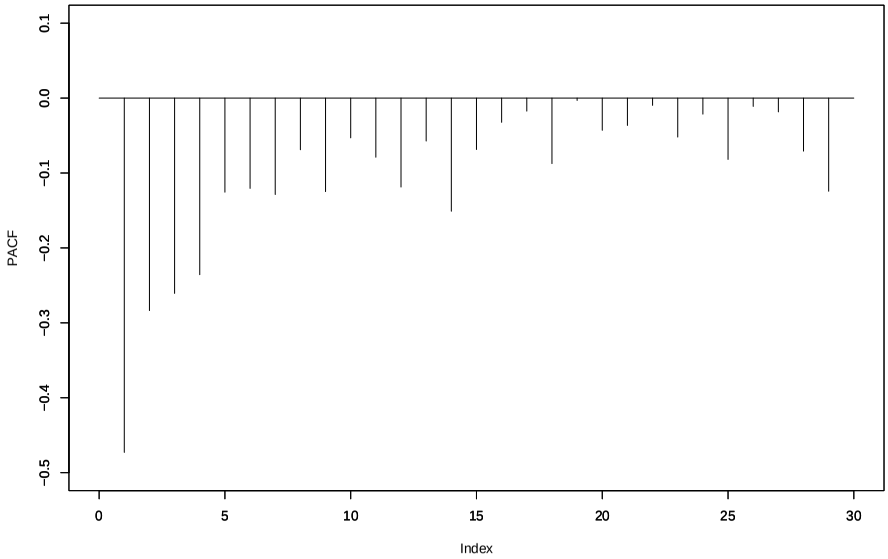

Figure 4.4: PACF of Data

Chapter 5

Conclusions

The results of this project have shown that accurate distance can be measuring using 802.11 time-of-flight. This chapter will summarise the findings of this project and then briefly look at what future work lies ahead.

5.1 Findings

This project used custom 802.11 hardware to measure time-of-flight and determine distance with a 3 metre accuracy. Preliminary ping analysis proved that software timing was too imprecise for measuring distance. The focus was then shifted to timing in hardware, allowing us to get closer to the radio circuitry hence minimising delay. Delays other than time-of-flight could then be identified and compensated for.

However it was soon realised that if these delays could be eliminated then there would be no need to identify them; thus how the timestamping design originated. The hardware is use were WAG V2 network cards, developed by Ph.D student Dean Armstrong. The WAG card has a 44MHz clock and it was determined that it should be possible to achieve a 3.4 metre accuracy. The timestamp design was implemented using VHDL and with some applications written in C it was possible to do some testing.

Testing yielded excellent results, achieving an accuracy of 3 metres by simply looking at the median or mean. The data was analysed and a moving average model of order one was discovered.

5.2 Future Work

There are two major areas of future work: increasing accuracy and usability.

5.2.1 Increasing accuracy

Accuracy can be increased by improving the hardware and/or by using statistical methods. In terms of hardware improvement, using a faster clock with an adequate triggering methodology will increase accuracy. There are several statistical methods/techniques that could be of assistance to increasing accuracy such maximum likelihood and Bayesian statistics.

5.2.2 Usability

Currently this project uses customised packets and the user must look at the raw results to determine distance. Ideally the project would make an extension on the 802.11 framework so that timestamps could be included in a standard 802.11 packet. A user would then want to see the distance and/or location easily, hence some sort of GUI would be required that conducted all the analysis in the background and simply presented the user with the computed distance/location.

This project has already proved that accurate distance can be determined using 802.11 time-of-flight. With hardware improvements and statistical techniques, it should be possible to increase accuracy in to the sub-metre range.

Glossary

RX Recieve

TX Transmit

WAND University of Waikato's Network Research Group

WIFI Another name for 802.11, which is a wireless networking standard

Bibliography

[1] GPS: A military perspective. Web. http://www.gisdevelopment.net/technology/gps/techgp0048a.htm.

[2] Soekris Engineering net4526. Web. http://www.soekris.com/net4526.htm.

[3] Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. Technical report, Microsoft Research.

[4] Yu-Chung Cheng, Yatin Chawathe, Anthony LaMarca, and John Krumm. Accuracy Characterization for Metropolitan-scale Wi-Fi Localization. Technical report, University of San Diego, Intel Research Seattle, Microsoft Corporation, January 2005.

[5] Douglas E. Comer. *Internetworking with TCP/IP Principles, Protocols, and Architectures.* Prentice Hall, 2000.

[6] Li Cong and Weihua Zhuang. Non-Line-of-Sight Error Mitigation in Mobile Location. *IEEE INFOCOM*, 2004.

[7] Borko Furht and Mohammad Ilyas. *Wireless Internet Handbook - Technologies, Standards, and Applications.* CRC PRESS, 2003.

[8] Jeffrey Hightower and Gaetano Borriello. Location Systems for Ubiquitous Computing. Technical report, University of Washington, August 2001.

[9] Jeffrey Hightower and Gaetano Borriello. Location Sensing Techniques. Technical report, University of Washington, July 2001.

[10] A. Holt. Theoretical performance analysis of mirroring World Wide Web sites. *IEE Proc.-Commun., Vol. 150, No. 4*, August 2003.

[11] Timothy Tan Guang Rong and Ang Chee Wee. Close Range Positioning System using Wireless Networks. Technical report, Victoria Junior College, Republic of Singapore Navy, 2003.

[12] Kanoksri Sarinnapakorn. IEEE 802.11b "High Rate" Wireless Local Area Networks. Web, March 2001. http://umsis.miami.edu/ ksarinna/IEEE80211b.html.

[13] Sunnyvale, CA. United States of America: Synplicity, Inc. *Synplicity FPGA Synthesis, User Guide*, April 2005.

[14] Ping Tao, Algis Rudys, Andrew M. Ladd, and Dan S. Wallach. Wireless LAN Location-Sensing for Security Applications. Technical report, Rice University, 2003.

[15] Dr. Steven Case(Faculty Mentor) Travis Calvert. Wireless Location Determination: Using Existing 802.11 Wireless Networks to Determine a User's Location. Technical report, Minnesota State University.

[16] Brian W.Tolman and Ben Harris. The GPS Toolkit. *Linux Journal*, September 2004.

[17] Moustafa Youssef and Ashok Agrawala. On the Optimality of WLAN Location Determination Systems. Technical report, University of Maryland.

[18] James M. Zagami, Steen A. Parl, Julian J. Bussgang, and Karen Devereaux Melillo. Providing Universal Location Services Using a Wireless E911 Location Network. *IEEE Communications Magazine*, 1998.

Appendix A
Graphs

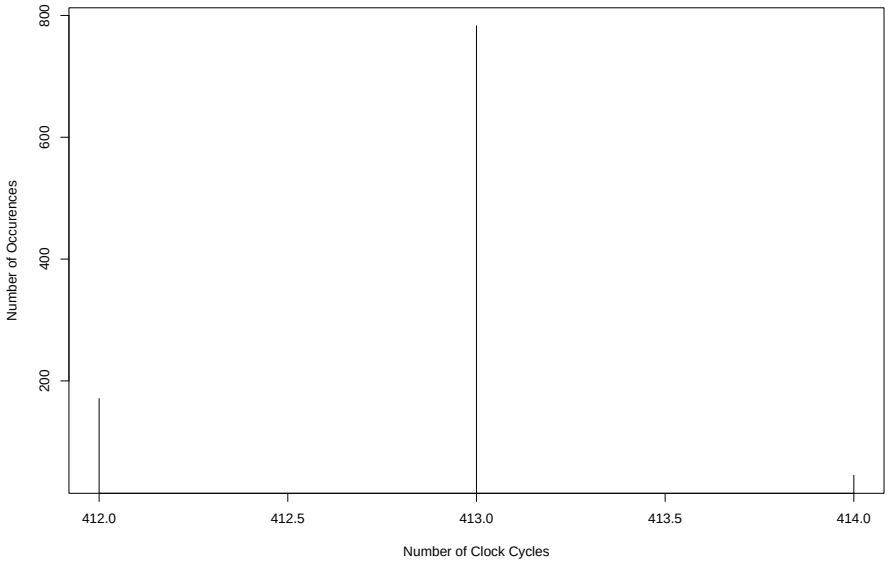

Figure A.1: 3 metres with 1000 tests

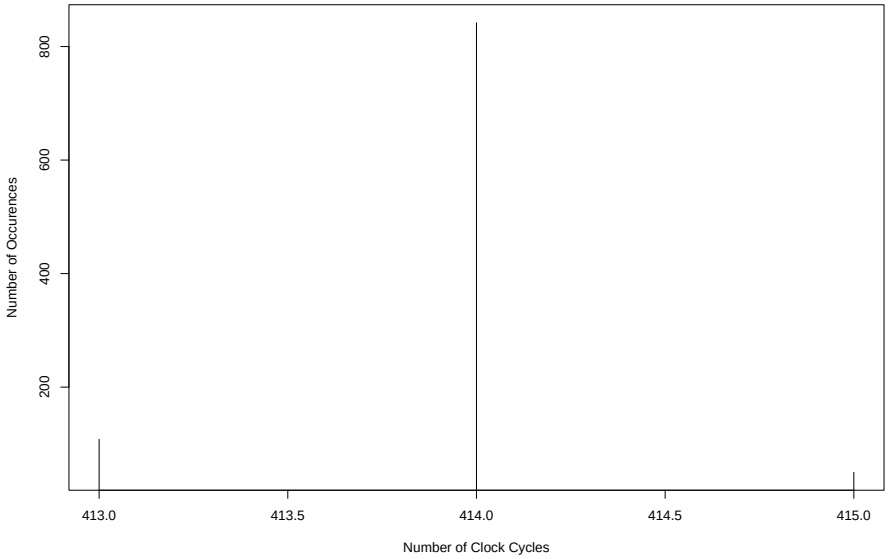

Figure A.2: 6 metres with 1000 tests

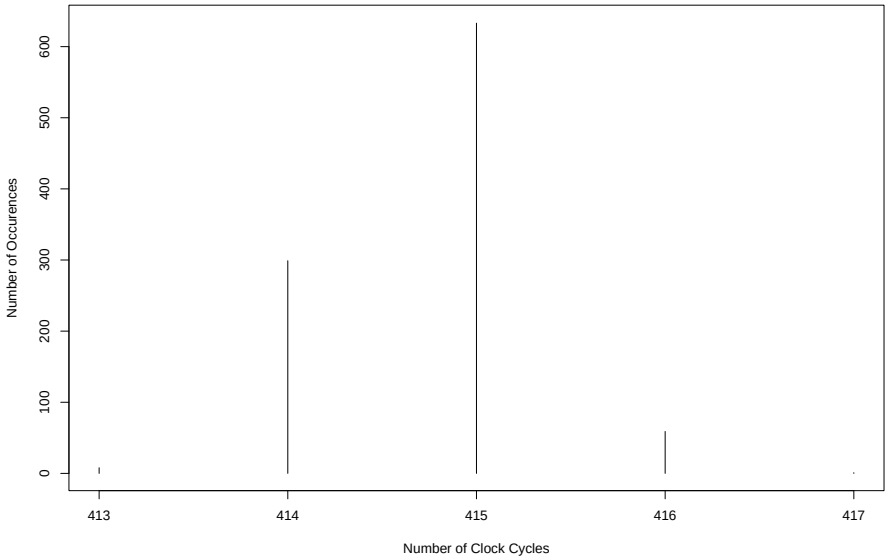

Figure A.3: 9 metres with 1000 tests

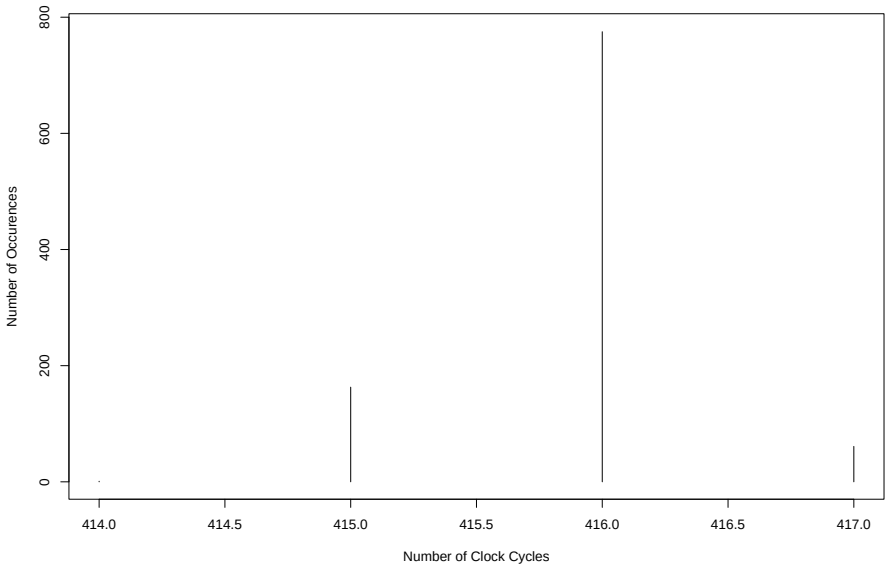

Figure A.4: 12 metres with 1000 tests

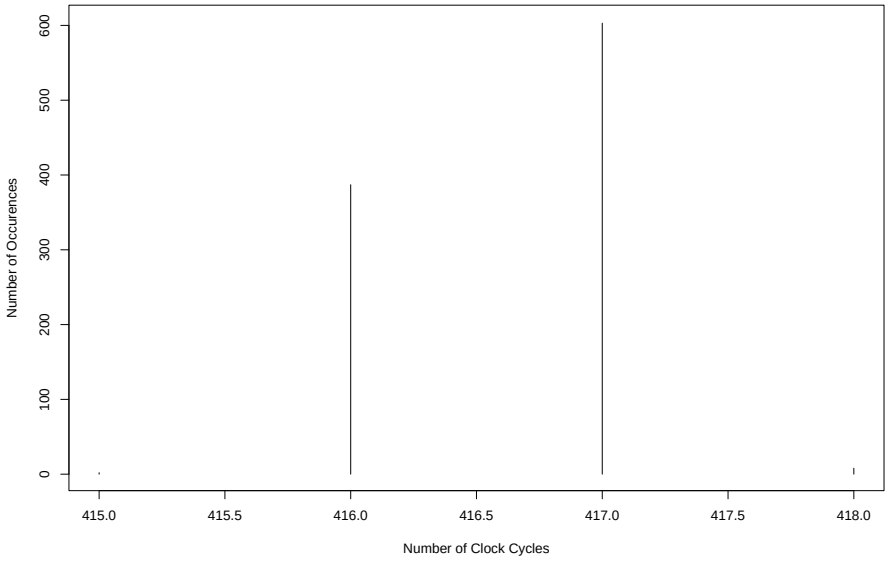

Figure A.5: 15 metres with 1000 tests

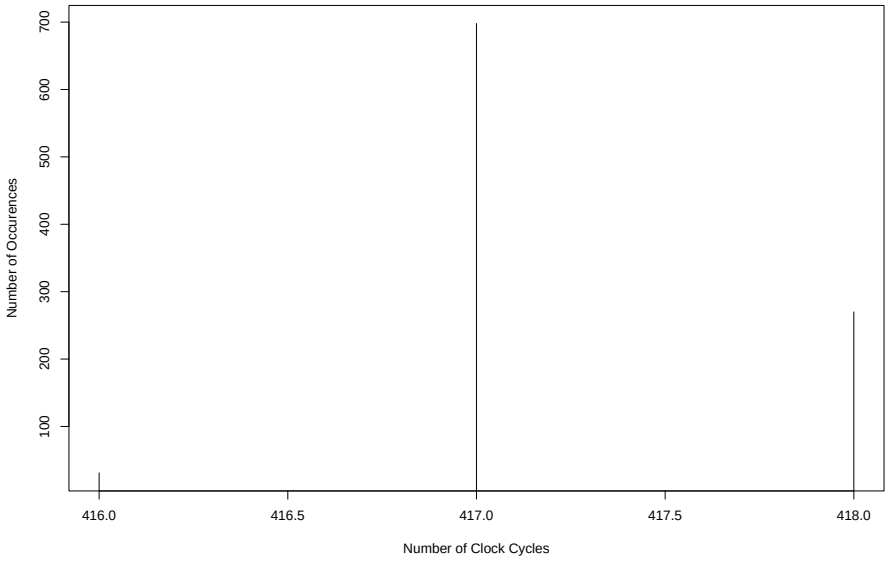

Figure A.6: 18 metres with 1000 tests

Appendix B
Distance

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <time.h>
#include <sys/time.h>
#include <sys/types.h>
#include <string.h>
#include <math.h>
#include <stdint.h>
#include "wag.h"
#include "chksum.h"

/* Function definition */
void timeout();

/* Variables */
uint32_t MAX = 4294967295;
int wagfd;
int count = 0;
int ant;
char *file;
FILE *f;
time_t rawtime;

/* Packet frame */
struct data_frame {
  struct {
    long seq_num;
    uint32_t rxstart_hi;
    uint32_t rxstart_lo;
    uint32_t txstart_hi;
    uint32_t txstart_lo;
  } data;
  char bufdata[1024];
  unsigned short crc;
};

/* Sequence numbers */
```

```
long seq_send = 0;
long seq_recv[2] = {0, 0};

/* TX and RX buffers */
char tx_buffer[WAG_RECV_SIZE];
struct data_frame *tx_frame = (struct data_frame *)tx_buffer;

char rx_buffer[WAG_RECV_SIZE];
struct data_frame *rx_frame = (struct data_frame *)rx_buffer;

/* Timestamp variable */
struct timestamps ts;

/* Timeout variables */
static int TIMEOUT_VALUE = 100000;
int TIMED_OUT = 0;
int main(int argc, char *argv[]){
  printf("FDistance\n");
  printf("(c) Dean Armstrong, Sam Bartels, 2005\n\n");

  if ((wagfd = open("/dev/wag", O_RDWR)) == -1){
    printf("ERROR: Could not open /dev/wag\n");
    exit(-1);
  }

  if(argc < 4){
    printf("USAGE: fdistance [channel] [#packets] [file] [-a antenna] [-t timeout]\n");
    printf("Channel, #packets and file are required, antenna and timeout optional\n");
    printf("Short Omni:0 Long Omni:1 Directional:2\n");
    exit(-1);
  }

  ioctl(wagfd, WAG_SETTXMODE, MODE_1MBPS);
  ioctl(wagfd, WAG_SETTXGAIN, WAG_MAXGAIN);

  if(atoi(argv[1]) < 1 || atoi(argv[1]) > 14){
    printf("channel must be from 1 to 14\n");
    exit(-1);
  }

  ioctl(wagfd, WAG_SETCHANNEL, atoi(argv[1]));
  count = atoi(argv[2]);
  ant = 1;

  if(argc == 6){
    char check = (char)argv[4][1];
    if(check == 'a')
      ant = atoi(argv[5]);
    else if(check == 't')
      TIMEOUT_VALUE = atoi(argv[5]);
  }

  if(argc == 8){
    ant = atoi(argv[5]);
    TIMEOUT_VALUE = atoi(argv[7]);
  }

  file = (char*) malloc(strlen(argv[3])+6);
  strcpy(file,"/var/");
  strcat(file,argv[3]);
  f = fopen(file, "w");
```

```
    if(f <= 0){
      printf("Error opening file\n");
      exit(-1);
    }

    printf("Opened file successfully\n");
    fprintf(f, "Time Stamp Test File by Sam Bartels\n");
    time (&rawtime );
    fprintf (f, "%s", ctime(&rawtime) );

    switch(ant){
      case 0: fprintf(f, "Antenna = Short Omni\n"); break;
      case 1: fprintf(f, "Antenna = Long Omni\n"); break;
      case 2: fprintf(f, "Antenna = Directional\n"); break;
    }

    while(seq_send < count){
      TIMED_OUT = 0;

      /* Send our initial frame */
      tx_frame->data.seq_num = seq_send;
      write(wagfd, tx_buffer, sizeof(struct data_frame));

      /* Wait for the response */
      timeout();

      /* If we haven't timed out then carry on */
      if(TIMED_OUT == 0){
        /* Get the timestamps */
ioctl(wagfd, WAG_GETTIMESTAMPS, (unsigned int)&ts);

/* Save received sequence number */
seq_recv[0] = rx_frame->data.seq_num;

/* Wait for the other end timestamps... */
timeout();

/* If we still haven't timed out then carry on */
if(TIMED_OUT == 0){
  /* Save received sequence number */
  seq_recv[1] = rx_frame->data.seq_num;

  /* If the sequence numbers match and the crc checks out then the data is valid */
  if(seq_recv[0] == seq_recv[1] && seq_recv[1] == seq_send && in_cksum((short *)&rx_frame->data, sizeof(rx_frame->data)) == rx_frame->crc){
    uint32_t rtt, loopback_delay;
    if(ts.rxstart < ts.txstart)
      rtt = (MAX - ts.txstart) + ts.rxstart;
    else
      rtt = ts.rxstart - ts.txstart;

    if(rx_frame->data.txstart_lo <  rx_frame->data.rxstart_lo)
      loopback_delay = (MAX - rx_frame->data.rxstart_lo) +  rx_frame->data.txstart_lo;
    else
      loopback_delay =  rx_frame->data.txstart_lo -  rx_frame->data.rxstart_lo;

    fflush(stdout);
    fprintf(f,"%d s %d %08x %08x %08x %08x \n", seq_send, rtt-loopback_delay, ts.rxstart, ts.txstart, rx_frame->data.txstart_lo,rx_frame->data.rxstart_lo);
    printf("Success: seq = %d\n", seq_send);
    seq_send++;
  }
```

```
  else
    /* Corrupt packet */
    fprintf(f, "%d c\n", seq_send);
}
      }
      fflush(f);
    }
    close(wagfd);

    return 0;
  }

  void timeout(){
    struct timeval timeout;
    fd_set readfds;
    int retval;
    int recv_packet_size = 0;

    do{
      /* Add wagfd to readfds */
      FD_ZERO(&readfds);
      FD_SET(wagfd, &readfds);

      /* Wait up to TIMEOUT_VALUE usecs */
      timeout.tv_sec = 0;
      timeout.tv_usec = TIMEOUT_VALUE;

      /* Select */
      retval = select(wagfd + 1, &readfds, NULL, NULL, &timeout);

      if (retval < 0)
        printf("select err: %m\n");

      /* Check readfds */
      if (FD_ISSET(wagfd, &readfds))
      {
/* Get a packet */
recv_packet_size = read(wagfd, rx_buffer, WAG_RECV_SIZE);
      }
      else
      {
printf("Timeout occurred\n");
fprintf(f,"%d t\n", seq_send);
TIMED_OUT = 1;
break;
      }
      /* check for valid sized packet */
    } while (recv_packet_size != sizeof(struct data_frame));
  }
```

Appendix C

Sounder

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/types.h>
#include <stdint.h>
#include "wag.h"
#include "chksum.h"

int wagfd;
int time_to_wait = 2000;

struct data_frame {
  struct {
    long seq_num;
    uint32_t rxstart_hi;
    uint32_t rxstart_lo;
    uint32_t txstart_hi;
    uint32_t txstart_lo;
  } data;
  char bufdata[1024];
  unsigned short crc;
};

long seq_recv = 0;
char tx_buffer[WAG_RECV_SIZE];
struct data_frame *tx_frame = (struct data_frame *)tx_buffer;
char rx_buffer[WAG_RECV_SIZE];
struct data_frame *rx_frame = (struct data_frame *)rx_buffer;
struct timestamps ts;
int frame_size;

int main(int argc, char *argv[]){
  int i;
  printf("Sounder\n");
  printf("(c) Dean Armstrong, Sam Bartels, 2005\n\n");

  if ((wagfd = open("/dev/wag", O_RDWR)) == -1){
```

```
      printf("ERROR: Could not open /dev/wag\n");
      exit(-1);
    }

    if(argc < 2){
      printf("USAGE: sounder [channel] [timeout(us)]\n");
      printf("Timeout is optional, will default to 2ms\n");
      exit(-1);
    }

    ioctl(wagfd, WAG_SETTXMODE, MODE_1MBPS);
    ioctl(wagfd, WAG_SETTXGAIN, WAG_MAXGAIN);

    if(atoi(argv[1]) < 1 || atoi(argv[1]) > 14){
      printf("channel must be from 1 to 14\n");
      exit(-1);
    }

    ioctl(wagfd, WAG_SETCHANNEL, atoi(argv[1]));

    if(argc == 3)
      time_to_wait = atoi(argv[2]);

    while (1) {
      /* Get a frame of the right size */
      do {
frame_size = read(wagfd, rx_buffer, WAG_RECV_SIZE);
      } while (frame_size != sizeof(struct data_frame));

      seq_recv = rx_frame->data.seq_num;
      printf("Recv %d\n", seq_recv);
      fflush(stdout);
      tx_frame->data.seq_num = seq_recv;

      /* Send one right back */
      write(wagfd, tx_buffer, sizeof(struct data_frame));

      /* Wait for for timestamps to be ready */
      usleep(time_to_wait);

      /* Get the timestamps */
      ioctl(wagfd, WAG_GETTIMESTAMPS, (unsigned int)&ts);

      /* Put them in a packet and do a crc */
      tx_frame->data.seq_num = seq_recv;
      tx_frame->data.rxstart_hi = ts.rxstart_hi;
      tx_frame->data.rxstart_lo = ts.rxstart;
      tx_frame->data.txstart_hi = ts.txstart_hi;
      tx_frame->data.txstart_lo = ts.txstart;
      tx_frame->crc = in_cksum((short *)&tx_frame->data, sizeof(tx_frame->data));

      /* And send the timestamps... */
      write(wagfd, tx_buffer, sizeof(struct data_frame));
    }

    close(wagfd);

    return 0;
  }
```