

# THE UNIVERSITY OF NEW SOUTH WALES



SYDNEY • AUSTRALIA

## SCHOOL OF COMPUTER SCIENCE & ENGINEERING

### CHARACTERISING ERRORS IN WIRELESS LAN

(THESIS REPORT)

**PARTICIPANTS** : **CHIEN HUNG CHEN (3007916), BE Comp Eng**  
**SIEW CHEIT HIN KELVIN (3013826), BSc CompSci**

**SUPERVISOR** : **Dr. Tim Moors,**  
**Senior Lecturer,**  
**School of Electrical Engineering &**  
**Telecommunications,**  
**UNSW**

**ASSESSOR** : **Dr. Mahbub Hassan**  
**Associate Professor,**  
**School of Computer Science and Engineering,**  
**UNSW**

## **ACKNOWLEDGEMENTS**

## **ABSTRACT**

This project will characterize the transmission errors that occur in IEEE 802.11 wireless LANs operating in the 2.4GHz and 5GHz bands. These errors are important for their effect on network capacity (achievable throughput, delays experienced by real-time media, and association of stations to access points), protocol design (level of error coding and approaches to retransmission) and in determining the position of stations.

In this report, we evaluated the effects of interfering radiation sources, and of attenuation due to distance and obstacles, and on the resulting packet loss and bit errors from these impairments. This project involves packet-level and bit-level collection of data and analysis of the received data. The results of this project will be used to improve protocols that are sensitive to these characteristics (e.g. transport protocol design, the level of coding for error control, and the ability to meet time constraints).

## CONTENTS

<b>1. Introduction</b>	<b>1</b>
<b>1.1 Objectives of this Thesis</b>	3
<b>1.2 Project deliverables</b>	3
<b>1.3 Structure of the report</b>	3
<b>2. Past Works</b>	<b>4</b>
<b>2.1 Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network</b>	4
<b>2.2 Wireless LAN Measurements Project</b>	5
<b>3. Source of Wireless Impairments</b>	<b>7</b>
<b>3.1 Impairments we take into account</b>	7
<b>3.1.1 Attenuation</b>	7
<b>3.1.2 Front End Overload</b>	7
<b>3.1.3 Spread-spectrum Interference</b>	7
<b>3.1.3 Path Loss (Dispersion)</b>	7
<b>3.1.4 Multipath Interference</b>	7
<b>3.2 Impairments we didn't take into account</b>	8
<b>3.2.1 Natural Background Noise</b>	8
<b>3.2.2 Motion</b>	8
<b>3.2.3 Data Dependent Effects</b>	8
<b>4. Methodology/Development</b>	<b>9</b>
<b>4.1 Equipment and Drivers</b>	10
<b>4.1.1 Equipment Used</b>	10
<b>4.1.1.1 Receiver</b>	10
<b>4.1.1.2 Sender</b>	10
<b>4.1.2 Sender Equipment Set Up</b>	10
<b>4.1.3 802.11 Drivers Used</b>	11
<b>4.1.3.1 MADWIFI IEEE 802.11a v20030802</b>	11
<b>4.1.3.2 AVS IEEE 802.11b linux-wlan™ v0.1.15</b>	11
<b>4.2 Packet-level Data Collection</b>	11

4.2.1 Files	11
4.2.2 Sequence of File Execution	13
4.3 Bit-level Data Collection	
13	
4.3.1 802.11b Driver Modification	14
4.3.2 Sequence on File Execution used to collect and analyze Bit-Level Data	15
4.3.3 Promiscuous Receive	15
4.3.4 Specially Formatted UDP packet	16
4.3.5 Data Collection	17
4.3.6 <i>Bitcompare.c</i> Program features	17
4.3.7 Filtering	18
4.3.8 Analysis of a Corrupted Packet	19
4.3.2 802.11a Driver Modification Attempt	22
<b>5. Experimental Results for packet-level analysis</b>	<b>23</b>
<b>5.1 Indoor</b>	23
<b>5.1.1 Single Wall</b>	23
5.1.1.1 802.11a (Perfect)	24
5.1.1.2 802.11a (Uniwide)	25
5.1.1.3 802.11b (Perfect)	25
5.1.1.2 802.11b (Uniwide)	26
<b>5.1.2 Two walls</b>	27
5.1.2.1 802.11a (Perfect)	28
5.1.2.2 802.11a (Uniwide)	28
5.1.2.3 802.11b (Perfect)	29
5.1.2.2 802.11b (Uniwide)	30
<b>5.1.3 Microwave</b>	31
5.1.3.1 802.11a	31
5.1.3.2 802.11b	32

<b>5.2 Outdoor</b>	<b>33</b>
<b>5.2.1 Water Effects</b>	<b>33</b>
<b>5.2.1.1 802.11a</b>	<b>34</b>
<b>5.2.1.2 802.11b</b>	<b>35</b>
<b>5.2.2 Distance Effects</b>	<b>36</b>
<b>5.2.2.1 50m</b>	<b>36</b>
<b>5.2.2.1.1 802.11a</b>	<b>36</b>
<b>5.2.2.1.2 802.11b</b>	<b>37</b>
<b>5.2.2.2 100m</b>	<b>38</b>
<b>5.2.2.2.1 802.11a</b>	<b>38</b>
<b>5.2.2.2.2 802.11b</b>	<b>39</b>
<b>6. Experimental Results for Bit-level Analysis</b>	<b>40</b>
<b>6.1 Line of sight, Within 1m</b>	<b>41</b>
<b>6.1.1 Results of Line of Sight, 1m</b>	<b>41</b>
<b>6.1.2 Observations</b>	<b>41</b>
<b>6.2 Passive Obstacles</b>	<b>42</b>
<b>6.2.1 1 Wall</b>	<b>42</b>
<b>6.2.1.1 Results of Passive Obstacle, 1 Wall</b>	<b>42</b>
<b>6.2.1.2 Observations</b>	<b>43</b>
<b>6.2.1.3 Conclusion: Passive Obstacle, 1 Wall</b>	<b>44</b>
<b>6.2.2 2 Walls</b>	<b>44</b>
<b>6.2.2.1 Results of Passive Obstacles, 2 Walls</b>	<b>45</b>
<b>6.2.2.2 Observations</b>	<b>45</b>
<b>6.2.2.2 Conclusion: Passive Obstacles, 2 Walls</b>	<b>47</b>
<b>6.3 Competing sources</b>	<b>47</b>
<b>6.3.1 Microwave Oven</b>	<b>47</b>
<b>6.3.1.1 Results of Competing Sources, Microwave Oven</b>	<b>48</b>
<b>6.3.1.2 Observations</b>	<b>48</b>
<b>6.3.1.3 Conclusion: Competing Sources, Microwave Oven</b>	<b>49</b>
<b>6.3.2 1 Access Point (Uniwide)</b>	<b>50</b>
<b>6.3.2.1 Results for Competing sources, 1 AP</b>	<b>50</b>
<b>6.3.2.2 Observations</b>	<b>50</b>
<b>6.3.2.3 Conclusion: Competing Sources, 1 AP</b>	<b>51</b>

<b>6.4 Line of Sight</b>	<b>52</b>
<b>6.4.1 50 meters</b>	<b>52</b>
<b>6.4.1.1 Results for Line of Sight, 50m</b>	<b>52</b>
<b>6.4.1.2 Observations</b>	<b>52</b>
<b>6.4.1.3 Conclusion: Distance, 50m</b>	<b>53</b>
<b>6.4.2 100 meters</b>	<b>54</b>
<b>6.4.2.1 Results for Distance, 100m</b>	<b>54</b>
<b>6.4.2.2 Observations</b>	<b>54</b>
<b>6.4.2.3 Conclusion: Line of Sight, 100m</b>	<b>56</b>
<b>7. Future Work</b>	<b>57</b>
<b>References</b>	<b>i</b>
<b>Appendix A - Driver Installation and Configuration for 802.11b</b>	<b>ii</b>
<b>Appendix B - Driver Installation and Configuration for 802.11a</b>	<b>v</b>
<b>Appendix C - IEEE 802.11 Channels – Frequency Table</b>	<b>vi</b>
<b>Appendix D - ASCII Table</b>	<b>vi</b>
<b>Appendix E - Kernel buffer output of 1 Frame</b>	<b>vii</b>
<b>Appendix F - Error Correlation Program</b>	<b>viii</b>
<b>Appendix G - Correlation Program Output</b>	<b>x</b>
<b>Appendix H - Output file (filename_results) of <i>bitcompare.c</i></b>	<b>xi</b>

## **1. Introduction**

The use of wireless technology is quickly becoming an established part of many societies today. There are millions of people around the globe who are using wireless technology. Wireless technology has greatly improved our ability to work and communicate at home or at work in our local and global communities. More and more people are using wireless technology not only for work, but also for the convenience.

Wireless local area network or WLAN, provides an excellent way to extend the reach of local area networks, through a wireless connection. Wide area networks provide the largest coverage and which requires infrastructure investment in terms of wired base stations.

However, when signals propagate through space, as compared to wired transmissions, many more impairments can influence the signal quality. Thus, characterizing the environment is an important step in providing reliable communication service to applications which rely on WLAN technology. This might result in new transport layer protocols that typically isolate wireless network segments from wired segments, to changes in the MAC and/or physical layer, e.g. retransmission or FEC.

WLAN is a technology that enables connecting computers to a network wirelessly with high bit rates, compared to IR and Bluetooth. The purpose of WLAN was originally to enable in office working without the hassle of network cables, but it has since evolved and is still currently evolving very rapidly towards offering fast connection capabilities within large area.

The first WLAN standard, IEEE 802.11 is based on radio technology operating in the 2.4 GHz frequency and has a maximum throughput of 1 to 2 Mbits per second. The general idea of WLAN was basically just to provide a wireless network infrastructure comparable to the wired Ethernet networks in use. Currently, the most spread and deployed standard is the IEEE 802.11b.

Standards like the 802.11a and 802.11b have been published and these standards improve on data transfer rates. The 802.11b operates in the 2.4 GHz band, data rates can be as high as 11Mbps.

The 802.11a standard was published as a supplement to the 802.11. It operates in the 5GHz band instead of the traditional 2.4GHz that the earlier WLAN standards used, thus being subjected to less interference and supports data rates up to 54Mbps.

802.11a is not compatible with 802.11b and therefore its emergence has been quite slow. The disadvantage with the 5GHz frequency is the reduced working distance.

An important characteristic of IEEE 802.11 wireless LANs is that transmission errors occur regularly because of noise and interference. These transmission characteristics affect:

- a. The range of wireless devices (e.g. the positioning of access points, association of stations amongst them).
- b. The level of coding needed to protect frames from error.
- c. The ability of wireless networks to meet the delay requirements of media such as voice.
- d. The design of transport layer protocols (e.g. variants of TCP and whether reliable multicast protocols should multicast retransmissions or unicast them).

Existing models of wireless transmission errors are generally simple two-state Markov models, e.g. the Gilbert-Elliott model developed in 1960s; these models have questionable relevance to modern networks. In particular, the parameters for these models have largely persisted substantiated by actual measurements from real modern networks. These models also often fail to accurately characterize the distribution tail, which is critical when attempting to establish network coverage across whole environments.

Wireless Technologies	Standard	Data Rate	Frequency Band	Range
Personal Area Networks	Bluetooth	1 – 22.4 Mbps	5 GHz	10 m
	IR	1 – 15 Mbps	0.76 – 1.33 MHz	5 m
Local Area Networks	WLAN	11/54 Mbps	2.4/5 GHz	50 - 150m
	Hiperlan	20 Mbps		
Wide Area Networks	GSM	9.6 Kbps		30 Km
	GPRS	170 Kbps		

Table 1-1: Wireless Technologies Summary

### 1.1 Objectives of this Thesis

In order to confidently characterize wireless LAN behavior, a comprehensive set of data recording the behavior of operational wireless networks are required. This thesis focuses on:

- a. Packet-level data collection and analysis of errors (e.g. monitoring which packets broadcast using UDP are successfully received).
- b. Bit-level data collection and analysis of errors (e.g. modifying NIC drivers to retain errored frames for a comparison at the bit level to what is known to have been transmitted).

### 1.2 Project deliverables

1. Packet-level data collection and analysis of errors in both 802.11a and 802.11b.
2. Bit-level data collection and analysis of errors in 802.11b.

### 1.3 Structure of the report

This thesis report is organized as follows. In Chapter 2, we first give a brief description of wireless technologies and zoom into IEEE 802.11 WLAN. In Chapter 3, we present a summary of the sources of wireless impairments. We then present our approach for data collection and analysis in Chapter 4. This is followed by an analysis of the data collected, packet-level and bit-level, in Chapters 5 and 6 respectively. Possible improvements and suggestions for future research are mentioned in Chapter 7.

## **2. Past Works**

In this section we cover work that has been done in the past, their methods and conclusions, related to this project report.

### **2.1 Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network[1]**

In this paper, they reported the results of a study characterizing the error environment provided by the AT&T WaveLAN, which operates at 2.4 GHz ISM band with a 2Mb/s link speed.

For different tests, they placed the PCs in different environments or added competing radiation sources. They first ascertained the possible sources of wireless errors in order to define the types of experiments they were going to carry out. They also noted that there were many independent variables which were beyond their control (which can be investigated for effects on loss and error rates). Examples include: noise emitted by different models of laptop computers, relative position of transmitter and receiver, orientation of antenna, temperature, humidity, and position of human bodies, different transmit and receive capabilities of individual units, different materials of walls, ceilings, floors and furniture, and effects of very distant competing transmitters. Therefore they decided to focus on those with the most obvious effect, i.e. attenuation, obstacles and active radiation sources.

They collected bursts of packets at the maximum possible transmission rate (approximately 1.4 Mb/s for the machine and protocol stack), aggregating multiple bursts to form a long trial.

The data transfers consisted of specially-formatted UDP datagrams. On the receiver, the kernel device was modified to place both the Ethernet controller and the modem control unit into “promiscuous” mode and to log, for each incoming packet, every bit and all available status information, even if the packet failed the Ethernet CRC check. Using these specially formatted UDP datagrams and with the

help of some heuristics, they were able to filter out packets that they were only interested in even if the header was very badly corrupted.

From their experiments with the AT&T WaveLAN they concluded the following:

1. Distance alone seemed to have little effect in a fairly large area (indoors) and WaveLAN can frequently penetrate obstacles, although it eventually succumbs to attenuation.
2. They observed very few bit errors.
3. WaveLAN proved resistant to several sources of radio interference and self-interference was substantial enough to impede building a robust cellular architecture.

## **2.2 Wireless LAN Measurements Project[2]**

In this report, they reported on the throughput and measured the error characteristics of the IEEE 802.11b wireless LAN, traced the behaviour of IEEE 802.11b wireless LAN access points at the MAC layer which allowed them to observe how stations implement the 802.11 protocol.

They first developed two sets of UDP programs, for unicasting and multicasting of data packets, which enabled them to measure the throughput and measure the error characteristics of the wireless network. Their experiments were carried out at different locations and different times of day, by changing the parameters like, unicast or multicast, ad-hoc or infrastructure mode, wired or mobile, payload transmission rate, range and environmental factors which causes interference. They also carried out bit-level experiments which obtained statistics about various kinds of 802.11b frames.

At the packet level analysis, they tried to explain the causes of errors occurring in WLANs and relate those errors to the theoretical concepts and models. Testing and analysis of test data with different test conditions helped to predict the error characteristics that will be experienced in those conditions.

At the bit level, they were able to capture and calculate various types of frames being transmitted. They showed that the access point transmits Beacon frames very frequently for the duration of their frame capturing experiment. This would show that even if an access point is idle, it still could cause significant interference.

### **3. Source of Wireless Impairments**

An important characteristic of WLAN is that transmission errors can occur from impairments. WLAN error characteristics can vary widely, to confidently characterize WLAN behavior; we would first need to identify possible wireless impairments.

#### **3.1 Impairments we took into account**

##### **3.1.1 Attenuation**

When electromagnetic energy encounters matter, some of it is lost in the form of heat. Examples are: People and walls in direct line-of-sight path. The signals fades when there is an obstruction in the line-of-sight[1].

##### **3.1.2 Front End Overload**

If a very powerful transmitter of one frequency band is near a receiver of another band, the transmitter may overwhelm filters in the receiver and inject substantial noise. We expect wireless computer networks to be employed in close proximity to microwave ovens and cellular phone. Microwave ovens are powerful sources of potential interference which are common in an office environment. Though we expect them to be well shielded, even a small leakage percentage could be significant. Since most microwave ovens operate in the 2GHz range, it is possible that 2.4GHz 802.11b WLAN unit could receive more interference[1].

##### **3.1.3 Spread-spectrum Interference**

This is due to an unfriendly transmitter either switching between narrowband frequencies or spreading its energy simultaneously across a wide frequency band. We have investigated interference between competing WLAN transmitters[1].

##### **3.1.3 Path Loss (Dispersion)**

The intensity of electromagnetic energy reaching a receiver is decreased by distance even in free space[1].

##### **3.1.4 Multipath Interference**

The propagation phenomenon (electromagnetic radiation reflects off of or diffracts around objects) that results in radio signals reaching the receiver by two or more

paths. Since these paths are typically of different lengths, there will be destructive interference, which can greatly reduce signal strength[1].

## **3.2 Impairments we didn't take into account**

### **3.2.1 Natural Background Noise**

For example, infrared wireless networks may perform poorly if are near sources of direct sunlight. We did not attempt to measure or control for background noise as we couldn't possibly control the back ground noise and they might not have effect on the measurements[1].

### **3.2.2 Motion**

If two communicating objects are moving with respect to each other, the frequency of the electromagnetic energy changes according to the Doppler effect. While this effect may be significant in some radio environment, the Doppler shift due to moving a 802.11 WLAN unit at the speed of sound would be substantially less than the inaccuracy of the clock crystals employed by Wifi. Hence we have not investigated errors due to motion[1].

### **3.2.3 Data Dependent Effects**

Some modulation schemes can lose clock synchronization in the face of certain long bit patterns. While we have transmitted fix data packets in our investigation, we have not particularly examined 802.11 WLAN for data dependent error patterns[1].

#### **4. Methodology/Development**

To characterize the errors in WLAN, we monitored the quality of data (packet-level and bit-level) transfers between two laptops running Linux and an access point (see Section 4.1 for the equipments used in the experiments). Experiments were carried out at various environments with or without competing radiation sources.

Packets are aggregated multiple bursts to form a long trial. For each packet, the data words were incremented and the data value was identical between packets to facilitate identification.

The MAC in 802.11b was developed to work seamlessly with standard Ethernet to ensure that wireless and wired nodes on an enterprise LAN are the same logically. 802.11b defines both a frame format and MAC scheme that differs from standard Ethernet. This frame format enables number of features such as fast acknowledge, handling hidden stations, power management, and data security. Under the 802.11b standard, the MAC layer must also handle acknowledgement and resending of lost frames, which results in faster acknowledgement and more efficient bandwidth usage. The resending of lost frames is not desired in our experiments and which only occurs during unicast transmission; therefore, our experiments are based on multicasting.

Many independent parameters could be investigated for effects of impairments. Many of these parameters have too many values to investigate and may vary greatly from place to place and time to time. Therefore, the approach for this thesis was to select certain parameters, manipulate them focus on those with the most obvious effect.

## 4.1 Equipment and Drivers

### 4.1.1 Equipment Used

#### 4.1.1.1 Receiver

1. 1 Toshiba TE2000 Pentium III 1Ghz Laptop
2. 1 D-Link DWL-650 (Prism II) 802.11b PCMCIA card
3. 1 Netgear Cardbus HA501 (AR5000) 802.11a PCMCIA card

#### 4.1.1.2 Sender

1. 1 Acer TravelMate 213TX Intel Celeron 850Mhz Laptop
2. 1 3Com Megahertz 3CCFE574BT 10/100LAN PC Card
3. 1 Linksys BEFSR41 Router/Switch
4. 1 Avaya AP-3 Access point (802.11a/b capable)

### 4.1.2 Sender Equipment Set Up

The following figure depicts how the sender is set up for all experiments which were carried out in infrastructure mode. The cables used to link all three equipment are UTP Category 5 *straight* cables. The experiment uses a switch is to connect the laptop to the access point and only the switching capability Linksys BEFSR41 Router/Switch was used. All the cables are to be plugged into the normal ports of the switch (i.e. none should be plugged into the uplink port). Although it was suggested that using a cross cable to connect directly to the AP should work, we could not get the AP to work with either cross or straight cables.

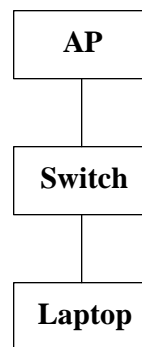


Figure 4-1: Sender Equipment Set Up

### **4.1.3 802.11 Drivers Used**

#### **4.1.3.1 MADWIFI IEEE 802.11a v20030802**

This is the only driver available at the moment for the Netgear HA501 (AR5000 family chipset) PCMCIA card that is able to operate in RedHat Linux. Instructions to compile and install this driver can be found in Appendix B.

#### **4.1.3.2 AVS IEEE 802.11b linux-wlan™ v0.1.15**

This driver was used to operate the D-Link DWL-650 (Prism II chipset) PCMCIA card in RedHat 7.3. Instructions to compile and install this driver can be found in Appendix A.

At the receiver end, the same driver was used to collect both packet-level and bit-level data, however, the original (unmodified) driver was used to collect packet-level data and another modified driver was used to collect bit-level data. The reason for this is that with the modified driver, it prints packets out to the kernel buffer (and to screen for text-mode Linux). The printing to kernel buffer causes a delay at the receiver-end in clearing the card receive buffer which might result in packets loss due to buffer overrun instead of noise impairments.

Driver modifications are explained below in section 4.2.

## **4.2 Packet-level Data Collection**

The experiments were carried out in multicast mode. The transmitter sends each UDP packet with a body of size 725 octets. In each experiment, a total of 5000 packets were transmitted from the sender in a burst of 100 (802.11a) or 150 (802.11b). There is not delay in between each packet but a delay of 550ms in between each burst; this is to allow the receiver to clear its buffer before receiving another burst.

### **4.2.1 Files**

This packet-level analysis software packages consists of 5 files, one as sender, one as receiver, one to convert the receiver output file suitable to calculate the correlation and the error burst duration, one to calculate the average correlation and the last one is used to calculate the error burst duration. The main reason that the mechanism to determine the correlation and error burst duration is not build in the

receiver is because we want to minimize the system related activities as they contribute to higher packet lost rate.

**mcastServer.c:**

It is used to receive the test packets received over its network interface and output statistics to the screen. The output to the screen is at minimum in order to avoid unnecessary delays in receiving test packets. It will then output the received test packets into two files; one for packets received while the other is missing test packets numbers.

**mcastClient.c:**

It is used to send bursts of fix pattern test packets at a fix interval. In order to capture the various interference signatures over a period of time, we are using the longest possible burst size with shortest delay in-between each burst while without causing buffer overrun. It can be used for both packet level and a bit level experiment depends on the input parameters.

**result.cpp:**

This program reads in output files from mcaseServer.c. Base on the missing packet sequence number, it will output a sequence of '0' if the receiver received and '1' if the packet lost to a file. Every input file will have a correspondent output file in the format of '0's and '1's. There will be another file which contains the joined result from all the input file(s), which is used to calculate the error burst duration.

**cor-average.cpp:**

This program will take in the file(s) output from result.cpp, which is in '0' and '1' format. It then calculate the individual correlation and output the average correlation into a file with '\_cor' as part of the file name.

**err-interval.cpp:**

This program will read in the join result file output from the result.cpp and based on the input factor calculate the error burst duration.

## **4.2.2 Sequence of File Execution**

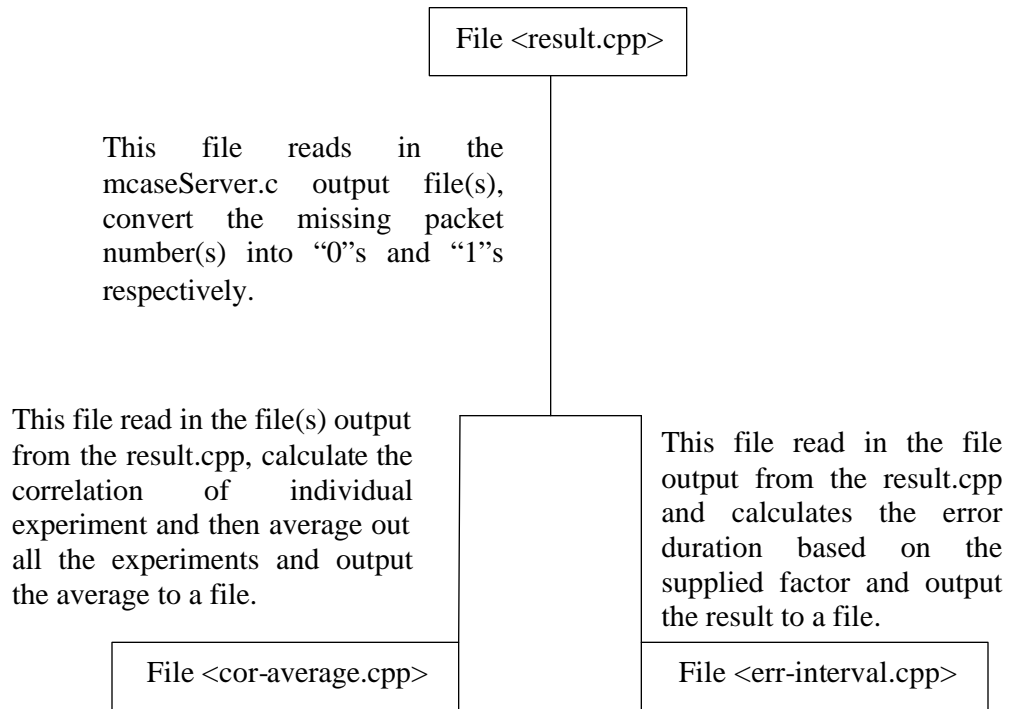


Figure 4-2: Sequence of File Execution

### 4.3 Bit-level Data Collection

There are many reasons why a transmitted test packet might not be received, such as unrelated system activities which might cause I/O operations on the hard disk which can cause the card buffer to overrun. Errors in the packet headers and trailers can also lead to the Ethernet or IP layers to discard the packet due to damage or miss-addressing. Packets might arrive truncated and/or with bit errors.

The experiments were carried out in multicast mode. The transmitter sends each specially formatted UDP packet with a body of size 725 octets. The format can be seen in Appendix E. In all, 500 packets were transmitted from the sender, in between each packet is a delay of 1 second (so total length of one experiment is approximately 8 minutes), this is to allow the receiver to clear its buffer before receiving another packet as printing from the driver to kernel buffer significantly slows down the performance.

An unsuccessful attempt was also made to modify the 802.11a driver; this is explained further in Section 4.3.2.

#### 4.3.1 802.11b Driver Modification

Dhanani and Tuli[2] modified the AVS IEEE 802.11b linux-wlan™ v0.1.15 driver to print out the headers of the 802.11b MAC frame in promiscuous receive (monitor mode). In this part of the project, we further modified the driver to print out the full data body as well. They pointed out that it would be better if an alternative could be found to print out the received frame via the kernel buffer (i.e. using printk). No other viable alternatives could be found that would not require an overhaul of the driver, but a more efficient way was found to reap the data from the kernel buffer, which is explained in Section 4.3.5.

However, there were certain fields in the MAC frame which we had no control over, namely, Sequence Control and Address 4. Sequence Control is decided by the driver at the transmitter and is incremented every time a new packet is sent. Address 4 is ignored as the To DS and From DS bits are set to 0 and 1 respectively for a multicast transmission (see Table 4.1); therefore, the value in Address 4 is not fixed. As such, the analysis portion of this project did not take into account the header of the MAC frame. Further explanations, as to the exclusion of the header, are mentioned in Section 4.3.8.

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	DA	SA	BSSID	N/A
0	1	DA	BSSID	SA	N/A
1	0	BSSID	SA	DA	N/A
1	1	RA	TA	DA	SA

Table 4-1: Usage of different Addresses according to To DS and From DS bit settings[3]

### 4.3.2 Sequence on File Execution used to collect and analyze Bit-Level Data

The sequence of programs in which we used to collect data is the following:  
(These sequences assume that the driver is already up and running, please refer to Appendix A on how to set up the 802.11b driver)

1. Set the card to promiscuous mode and ignore CRC.

```
$ wlanct1-ng wlan0 lnxreg_wlansniff enable=true channel=7  
prismheader=true stripfcs=true
```

2. Start recording bit level data.

```
$ cat /proc/kmsg >> dir/filename
```

3. Run the bitcompare program which filters and runs the error correlation program.

```
$ ./bitcompare dir/filename
```

Detailed explanations of what each step does and how they are derived are explained in the sections below.

### 4.3.3 Promiscuous Receive

The aim of this part of the project was to collect the corrupted packets and analyse them at the bit-level. Frames which had a specific pattern in their data body that matches a certain pattern specified had to be collected regardless of their headers, as the packet headers could be corrupted. This implies that even frames which claim to be management frames but carry a payload which matches a certain pattern would have to be considered.

Thus the device had to be set to promiscuous receive (monitor mode) to receive all frames regardless and the automatic CRC filtering was disabled in the driver. The utility that does this is provided with the PCMCIA support package in the Card Services for Linux, i.e. pcmcia-cs v3.2.1 (installation instructions can be found in Appendix A). The following command line was executed in order to achieve this:

```
wlanctl-ng wlan0 lnxreg_wlansniff enable=true channel=7 prismheader=true stripfcs=true
```

Important Note:  
The channel changes depending on access point settings, so it is best to first check the channel by running iwconfig wlan0 at command line and comparing the frequency with the channel-frequency table available in Appendix A. This is important if the experiment is carried out with a competing access point operating at another channel, if the channel is set to that of the competing access point, the receiver will \*NOT\* be able to receive any of the test packets.

Figure 4-3: Command to activate promiscuous receive

#### 4.3.4 Specially Formatted UDP packet

A specially formatted UDP packet was used to facilitate easy identification of a packet which is corrupted. A sample of the data body of the specially formatted UDP packet is given below:

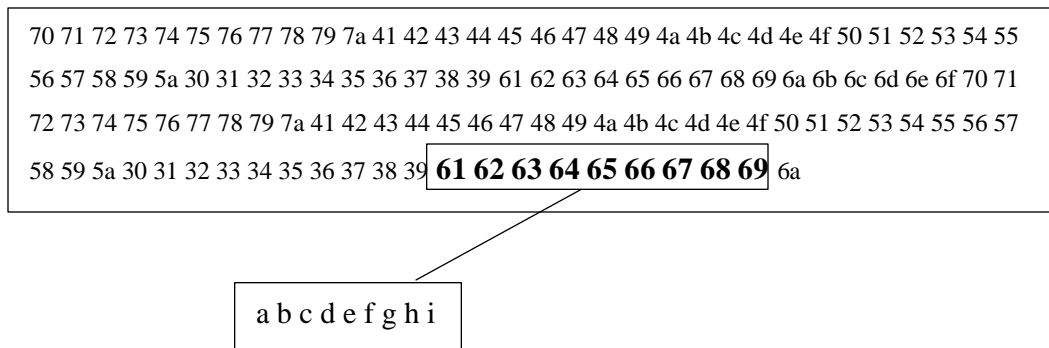


Figure 4.4: Sample portion of the specially formatted UDP packet

The numbers represented are in hex and must be translated using the ASCII table available in Appendix D. The entire data packet printed out from the buffer can be found in Appendix E.

### 4.3.5 Data Collection

Tuli and Dhanani previously collected the MAC frame headers from the kernel buffer via `dmesg` at user specified time intervals and dumping the contents of the `dmesg` out to hard disk. If the interval is too short, this will result in duplicate frames and, if the interval is too long, missing frames[2].

An alternative was to clear the buffer via `dmesg -c` at some user specified interval and dumping out the kernel buffer to disk. The `-c` flag clears the kernel buffer when run together with the `dmesg`.

A second alternative was used for this project, which did not require any timing intervals but instead printed out kernel buffer as they come. This was achieved by reading kernel messages in `/proc/kmsg`. This file is used to hold messages generated by the kernel, and essentially, it clears by default. The following command was used to collect received MAC frames printed from the kernel buffer:

```
cat /proc/kmsg >> filename
```

Figure 4-5: Command to output received frames to disk from kernel buffer

A SIGINT (Ctrl-C) is needed to stop the program.

### 4.3.6 *Bitcompare.c* Program features

The *bitcompare.c* program takes the raw data received from the above and will result in 3 files below:

- filename\_filter* : This file contains the raw data but filtered for only packets which matches the pattern specified.
- filename\_result*: This file contains, for each frame its error correlation as well as details on the error bursts (see Appendix H).
- filename\_finalavg*: This file contains an average of the error correlations of all the packets in the target file.

The steps to which the program derives these 3 files are explained in the sections below.

#### 4.3.7 Filtering

The ‘grep’ command was used filter out the specially formatted packet and by pattern matching part of the packet. For example, the pattern ‘30 31 32 33’ was used to identify the transmitted packets in the experiments. As the pattern ‘30 31 32 33’ is repeated 11 times in the packet (see Appendix E), and the possibility of all of the 11 sets of these 4 bits corrupting is very low (0.5%), we can safely say that the filtered packets which ‘looks like’ the test packets which were transmitted.

Note that the ‘grep’ command and the pattern used to extract the packets are incorporated into the analysis program *bitcompare.c*, as explained in Section 4.3.5; the *filename\_filter* will result from this.

#### 4.3.8 Analysis of a Corrupted Packet

After the filtering of the packets which belong, the filtered frames received are firstly compared to a non-corrupted packet (full packet available at Appendix E) at the hex level. The box at the top of Figure 4.4 represents what is expected in a sample portion of the frame body, the box below represents a sample portion of a corrupted frame body. The bold hex pairs represent bytes which are corrupted.

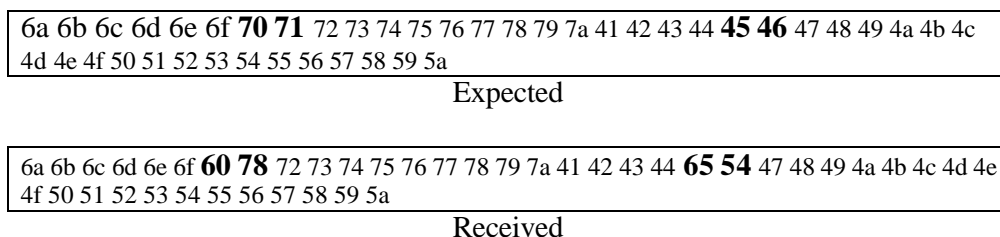


Figure 4-6: Sample portion of expected frame and received frame

The hex pairs in Figure 4.4, which is in *bold* (errored), are then converted to binary/bits. The first two bits which are corrupted in the above figure are used as an example in the figure below. *802.11 specifies little endian format for everything except CRCs*, so the bits are transmitted from *Least Significant Bit (LSB)* first[3]. Therefore 70 hex in binary is 01110000 has to be inverted to be read as 00001110. The table below shows a sample conversion.

<b>Expected</b>		<b>Received</b>	
70	71	60	78
00001110	10001110	00000110	00011110

Table 4-2: Comparison of expected and received at the bit level (in binary)

The expected bits and received bits are then compared to produce a string of 1s and 0s where the 1s represent bits which have been errored and 0s are the ok bits.

Expected	: 00001110 10001110
Received	: 00000110 00011110
Result	: 00001000 10010000
Note: Result is *NOT* in binary, it is a just a representation of bits which are errored (1s) and bits which are ok (0s).	

Figure 4-7: Representation of bits which have errored and are ok

Each hex pair received which matches the expected hex pair will be converted to 8 0s to represent all 8 bits are ok. These will be concatenated to another 8 0s if they are ok or if not, will be concatenated to, using Figure 4.5's results as an example, 00001000 10010000, if the following 16 bits (8 x 2) have error bits in them.

The *bitcompare.c* program will also output error bursts and accumulate all the bits which have not errored (see Appendix H). The `0 x <number>` specifies the number of bits that are ok and the `1010000101111101` indicates the error burst. The number of 0s in between an error burst is decided by a threshold set by the program, in our case it is set at *0.5%*.

The program will also output a probability on the correlation between the value of the bits transmitted and the types of errors that they incur, e.g. are 0s less likely to be turned into 1s than 1s being turned into 0s.

The data is then put through a C++ program designed to detect correlations. A sample fragment of this program is available in Appendix F, which has been integrated into *bitcompare.c*, this program has been separated for easy perusal of the algorithm used for data correlation. The statistics produced from this program will be helpful in comparison between the error results and determining error correlations. This is one of the algorithms that help to do that by calculating the burstiness of errors over a given range. It calculates relative probability between the occurrences of errors. Such kind of probability distribution can be used to predict error burstiness over a given range. The value of RANGE for the correlation results in the analysis is 100. It is the maximum distance over which the correlation is being measured.

Range	P(match match)	P(match mismatch)
1	0.416444	0.000205
2	0.430317	0.000428
3	0.439166	0.000630
4	0.434155	0.000848
5	0.399823	0.001048
6	0.390719	0.001249
7	0.420516	0.001442
8	0.374960	0.001617
9	0.373362	0.001896
10	0.365071	0.002127
11	0.359082	0.002358
12	0.354619	0.002598
13	0.350790	0.002816
14	0.347459	0.002983
15	0.344128	0.003176

Figure 4-8: Sample error correlation

The output has rows that indicate the distance over which the correlation is being measured. A range of “1” means that it is testing the correlation between adjacent characters in the file. A range of “3” means that it is testing characters that have 2 characters inserted between them. The second column indicates the probability that one character matching the target “1” (char target = ‘1’ see Appendix F) given that the character RANGE characters back matched. The

third column indicates the probability that the character matched the target, given that an earlier character did not. So in Figure 4.6, the value of 0.416444 for  $P(\text{match}|\text{match})$  with Range = 1 indicates that the error (1s) are moderately bursty, and has a probability of 41.6% that it is followed by another error bit.

As the data correlation program takes in consecutive samples of bits, gaps in between packets and gaps in the header, which we could not control, was a problem.

Frame Control	Duration ID	Address1 (source)	Address2 (destination)	Address3 (rx node)	Sequence Control	Address4 (tx node)	Data	FCS
2	2	6	6	6	2	6	0- 2,312	4

Figure 4-9: 802.11 MAC Header

As mentioned above in Section 4.3.1, the header was ignored in the data as we had no control over Sequence Control and Address 4 fields in the MAC frame (see Figure 4.7). Having no control over these two fields means that the driver on the transmitter will decide what to put into these two fields, and thus preventing us from producing an expected frame to compare with the received frame. One solution to this problem would be to modify the driver on sender side to print to the kernel buffer what it is transmitting, however this would require modifying the PCMCIA driver for the LAN card, and comparing with whatever was transmitted to whatever was received. Due to time constraints we were unable to do this.

The problem with the gap caused by the two fields is similar to the gap between two frames which are ours. Similar in the sense that the gap between two frames that are ours, could contain another AP's frame or an ICMP frame which the wireless driver sends out from time to time (this can be seen from the raw data collected for bit-level and is available in the CD-ROM disc provided with the report). In the case of the gap caused by the fields which we were unable to control, it is similar because we were unable to dictate what the driver puts into those two fields on the transmitter side. The experiments also do not take into account frames which have their headers so very badly corrupted that the driver/hardware ignores it completely, therefore we cannot capture such frames.

### **4.3.2 802.11a Driver Modification Attempt**

We modified the MADWIFI IEEE 802.11a driver to print out frame header and body to the kernel buffer as they were received; however, we could not print the corrupted frames as we were unable to prevent the driver from dropping the corrupted frames.

The MADWIFI driver is still in beta (released in 01 August 2003), therefore support for this driver was extremely limited. Attempts were made to locate a utility, which is similar to 802.11b's (i.e. wlanctl-ng that came with the 802.11b driver) which allowed the stripping of the frame CRC, but was unsuccessful.

## 5. Experimental Results for packet-level analysis

The group members carried out a lot of experiments at different environment settings in order to confidently characterize wireless LAN behavior. Given below are a comprehensive set of data recording the behavior of a wireless networks. (we do not present the repeated results of experiments i.e. confidence testing, which is available in the CD-ROM).

Before we commenced the experiments, we have gone through a series of long-running trials designed to find out the best combination between the error burst size and the delay in between each burst in order to avoid the buffer over run problem. Even though we are in a nearly perfect environment (not other interferences, i.e. front end narrowband interference, spread-spectrum interference, any form of transmitters), some system related process is causing packets to be lost, though at a rate of one per 5000 packets.

### 5.1 Indoor

#### 5.1.1 Single Wall

In this experiment, a transmitter and receiver are separated by approximately 5m (Figure 5-1), and then further separated by approximately 30cm of wall. We use brick wall(s) through all the experiments. The transmitter is broadcasting via an Access Point (AP).

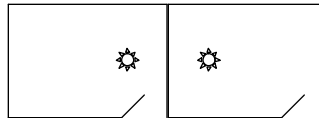


Figure 5-1: Layout of One Wall Experiment

	A		B	
	802.11a (Perfect)	802.11a (Uniwide)	802.11b (Perfect)	802.11b (Uniwide)
	1-wall		1-wall	
Total Pk Tx	25000	50000	25500	51000
Total Pk Lost	81	98	1	2
Avg Pk Lost	16.2	9.80	0.2	0.20
Max Pk Lost	31	24	1	1
Min Pk Lost	0	0	0	0
Avg Signal Strength	80%	75%	81%	70%
Link Speed (Mbps)	54	54	11	11
STDEV	12.35718415	1.67332	0.447213595	0.447214
% Pk Lost	0.32400%	0.20%	0.00392%	0.00%

Table 5-1: Result of 1-wall experiment

#### 5.1.1.1 802.11a (Perfect)

From table 5-1, we can see that the wall affects the signal level as it dropped by 20% compare to the signal level when both the transmitter and receiver are within one meter to each other.

Series 1 in the figure 5-2 below shows that the probability of an error packet follows by another error packet or packets is basically zero. Which implies that the probability for an error burst to happen is basically zero. Series 2 shows a 0.31 – 0.33% probability that the following packet(s) will error when previous packet is ok.

From this we can conclude that in a perfect environment with a brick wall in between, the probability to cause error burst is very low.

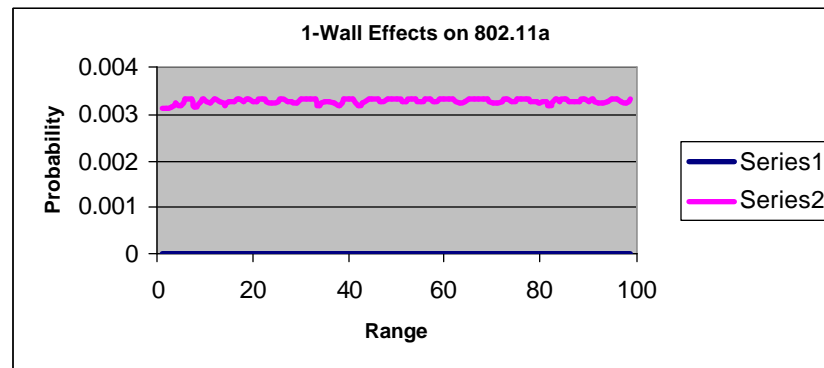


Figure5-2: 1-Wall effects on 802.11a

### 5.1.1.2 802.11a (Uniwide)

From table 5-1, we can see that the effect of an adjacent 802.11b AP to the 802.11a wireless signal is not as much as comparing to the brick wall. The signal level reduced by 5 % compare to the perfect environment experiment.

From series 1 in figure 5-3 below, we can see that an additional furniture and 802.11b AP which didn't exist in the 1-wall experiment did increase the probability of error burst. They seem to be a hindrance for the wireless signals as the probability that the following packet(s) will error when previous packet is ok. Probability to have a range 1 error is 14.29%. It then decreases gradually to an average of 0.858%. Series 2 shows a 1.2471% probability that the following packet(s) will error when previous packet is ok.

From this we can conclude that with additional 802.11b AP and furniture presents in the environment, it might cause multipath interference and which will increase the probability of error burst.

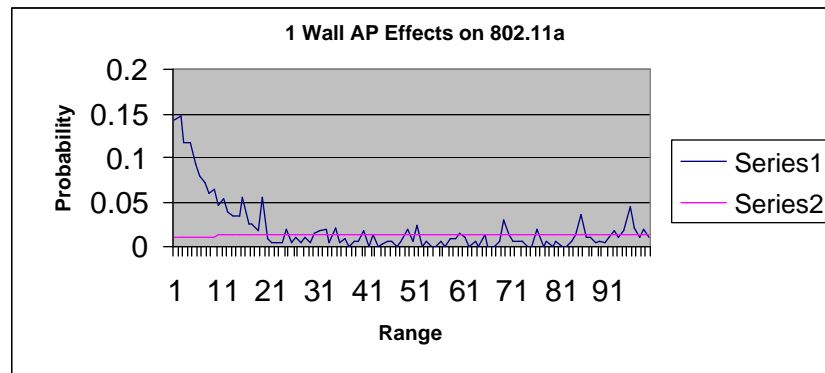


Figure 5-3: 1-wall AP effects on 802.11a

### 5.1.1.3 802.11 B (Perfect)

From table 5-1 we can tell that a single brick wall is having a similar effect on both the 802.11a and 802.11b wireless signal as their signal level is all reduced by about 20%.

Again, from figure 5-4 below, series 1 tell us that the probability to have error burst is basically zero, which implies that the effect of a single brick wall on

802.11b is basically none. Series 2 shows a 0.004% probability that the following packet(s) will error when previous packet is ok, which is much lower compare to 802.11a under the same environment. This implies that 802.11b wireless signal is having a better penetration power compare to 802.11a.

From this we can conclude that in a perfect environment with a brick wall in between, the probability to cause packet(s) to lose is very low or nearly zero.

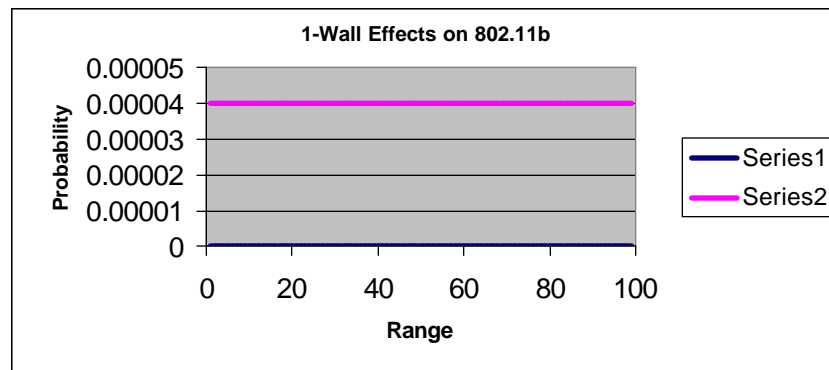


Figure 5-4: 1-Wall effect on 802.11b

#### 5.1.1.2 802.11b (Uniwide)

From table 5-1, we can see that the effect of an adjacent 802.11b AP to the 802.11b wireless signal is quite significant as it further reduce the signal level by another 10 %, which drops to 70%.

From series 1 in the figure 5-5 below, we can see that the probability to have an error follow error(s) is basically 0%. Series 2 shows a 0.018% probability that the following packet(s) will error when previous packet is ok, which is approximately 4.5 times higher compare to the same environment without the presents of any 802.11b AP or additional furniture.

From this we can conclude that even with additional furniture and 802.11b AP presents in the environment, it does not really contribute to the error burst rate as the effect of multipath interference on 802.11b is lesser than on 802.11a.

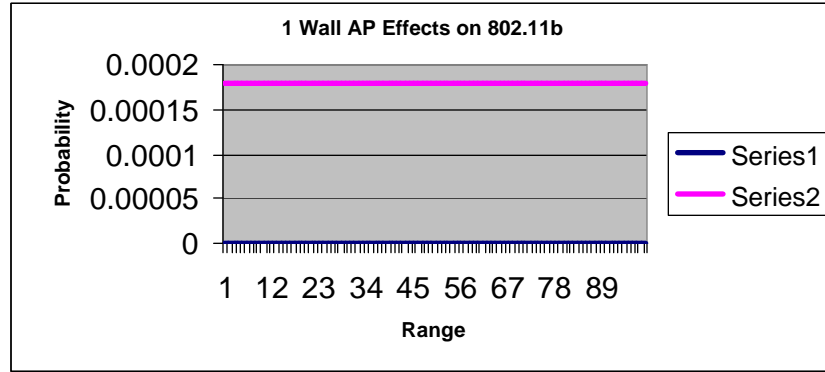


Figure 5-5: 1-Wall AP effects on 802.11b

### 5.1.2 Two walls

In this scenario, we use a similar setup to the previous experiment but with 2 brick walls in between (Figure 5-6). In this experiment, the transmitter and receiver are separated by approximately 12m, and then further separated by approximately 2 walls of approximately 30cm width each.

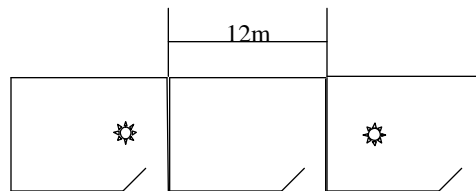


Figure 5-6: Layout of Two Walls Experiment

	A		B	
	802.11a (Perfect)	802.11a (Uniwide)	802.11b (Perfect)	802.11b (Uniwide)
	2-wall		2-wall	
Total Pk Tx	50000	50000	51000	51000
Total Pk Lost	44	220	46	2
Avg Pk Lost	4.80	22.00	4.60	0.20
Max Pk Lost	13	99	45	1
Min Pk Lost	1	0	0	0
Avg Signal Strength	67.00%	64.00%	46.00%	41.00%
Link Speed (Mbps)	48	48	11	11
STDEV	4.969909	11.52389	20.12461	0.547723
% Pk Lost	0.09%	0.44%	0.09%	0.00%

Table 5-2: Result of 2-wall experiment

### 5.1.2.1 802.11a (Perfect)

From table 5-2, we can see that the walls effects the signal level as it dropped by as much as 33%, which is 13% more compare to the 1-wall experiment. The link speed dropped from 54Mbps to 48Mbps.

Series 1 in the figure 5-7 below a different result as the 1-wall experiment, the probability of an error burst at range 1 is 36.29%. It then drop sharply and reach 0% at range 6. Series 2 shows a 0.5763% that probability that the following packet(s) will error when previous packet is ok. We can also see some major spikes between ranges 7 – 9, 34 – 45 and 57 – 66.

From this we can conclude that in a perfect environment with brick walls in between, the probability to have a range 1 error burst is quite high.

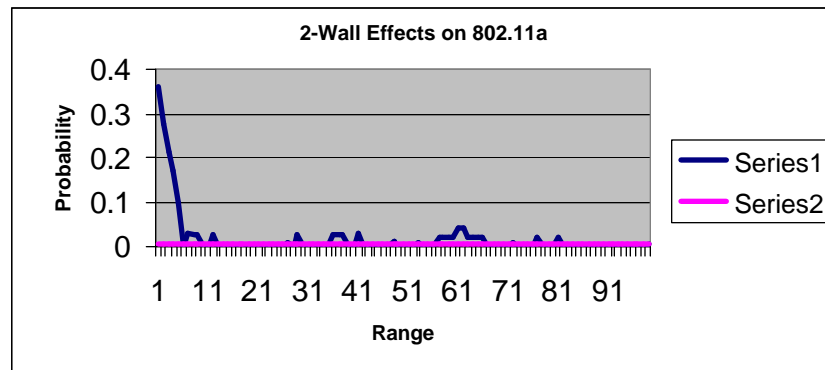


Figure 5-7: 2-Wall effects on 802.11a

### 5.1.2.2 802.11a (Uniwide)

From table 5-2, we can see that the signal level didn't affect too much by two walls as it drops by only 3%. The packet lost rate slightly increase to 0.35% and the link speed drop from 54Mbps to 48Mbps.

Series 1 in the figure 5-8 below shows that the probability for an error burst to happen is basically zero. Any packet lost might due to the system related activities. Series 2 shows an increase in the probability that the following packet(s) will error when previous packet is ok. It increases from 0.22% to 0.70%.

From this experiment, we can see that although there is an increase in the distance between the transmitter and receiver, there is an additional wall in between, but the performance of 802.11a didn't degraded too much. It might be due to the decrease in transmission rate.

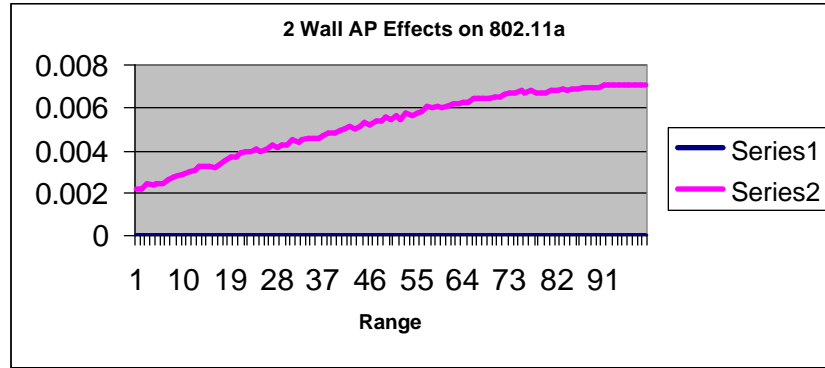


Figure 5-8: 2-Wall AP effects on 802.11a

### 5.1.2.3 802.11 B (Perfect)

From table 5-2, we can see that although the signal level dropped by about 50% when comparing to 1-wall experiment. The probability of error burst rate increased but the link speed remain s the same, 11Mbps.

From figure 5-9 below, we can see that series 1 starts at the probability of error burst 97.14% and 2 are showing a probability of 0. This implies that the probability to have error burst is zero while series 2 suggest that the probability that the following packet(s) will error when previous packet is ok is 0. This could be due to that the increasing in the passive obstacles and distance between causes the transmission rate to drop, which in term having a better performance.

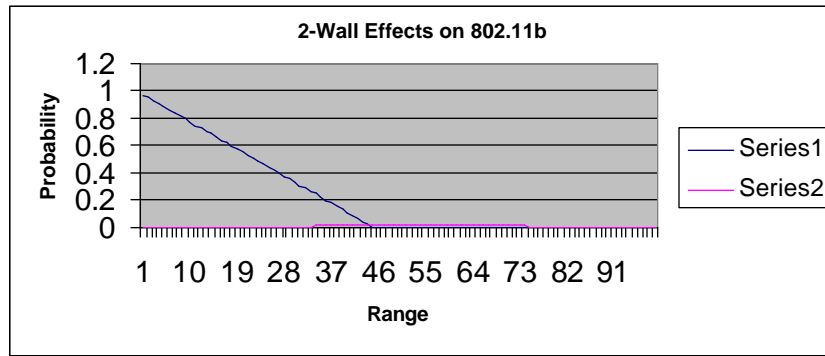


Figure 5-9: 2-Wall effects on 802.11b

#### 5.1.2.4 802.11b (Uniwide)

From table 5-10 above, we can see that same as 802.11a, the signal level didn't drop drastically with the increasing distance in between and the additional brick wall. The packet lost rate is slightly increased to 0.02%. But the over all performance is better than 802.11a under the same setting.

Again, from series 1 in figure 5-10 below, we can see that the probability to have error burst is basically 0 while series 2 suggest that the probability that the following packet(s) will error when previous packet is ok is about 0.02%.

From this experiment, we can see that 802.11b is having better performance when passive obstacles like brick walls are involved.

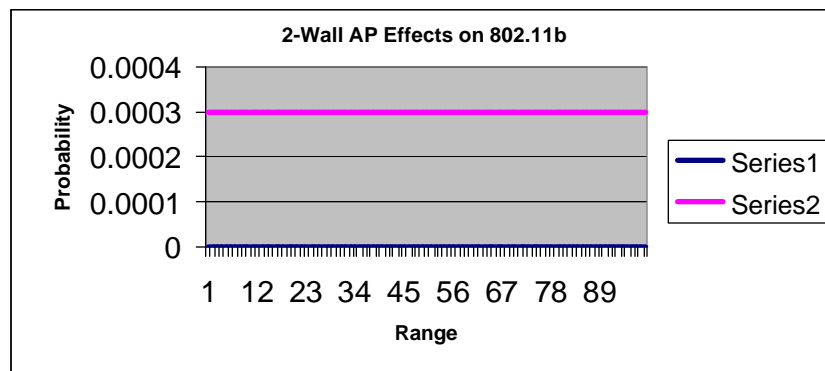


Figure 5-10: 2-Wall AP effects on 802.11b

### 5.1.3 Microwave

In this experiment, a transmitter (AP) is placed directly on top of the microwave and receiver is approximately 3m away (Figure 5-11). There are not other sources of interferes.

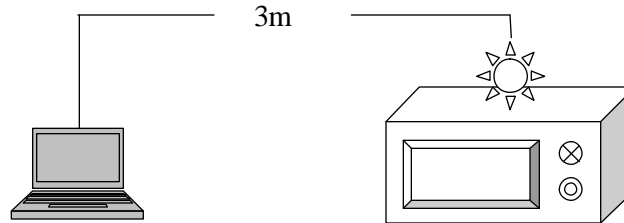


Figure 5-11: Layout of Microwave experiment

	A	B
	802.11a	802.11b
	Microwave	
Total Pk Tx	50000	25500
Total Pk Lost	335	5507
Avg Pk Lost	33.50	1101.4
Max Pk Lost	213	1849
Min Pk Lost	0	380
Avg Signal Strength	93.00%	64%
Link Speed (Mbps)	54	11
STDEV	6.363961	565.0268135
% Pk Lost	0.67%	21.59608%

Table 5-3: Result of Microwave experiment

#### 5.1.3.1 802.11a

From table 5-3, we can see that the microwave effect on the signal is not as much as brick walls; it only drops by 7%, which is 13% better than the single wall experiment and 30% better than the two walls experiment. The link speed was at maximum, 54Mbps.

Series 1 in the figure 5-12 below, we can see a small spike at ranges 19 - 20. There is a gentle decreasing in error burst rate between ranges 1 to 99. During the experiment, we observed that the microwave is not “heating up” all the time, there is a fix period of time where the microwave will heat up and stop and heat up again. Since microwave is a very powerful transmitter, the early

filter stages of the receiver, which are designed to reject out-of-bound signals, may be overwhelmed[1], this could be the possible reason cause the spike. Major error burst rate is between ranges 1 - 20. However, the probability for error burst is mainly below 4%. Series 2 shows an average of 1.4822% that probability that the following packet(s) will error when previous packet is ok.

From this we can conclude that although the microwave and 802.11a is operating at different frequency (2.4GHz and 5GHz respectively), but microwave ovens are a powerful sources of potential interference to 802.11a wireless signal.

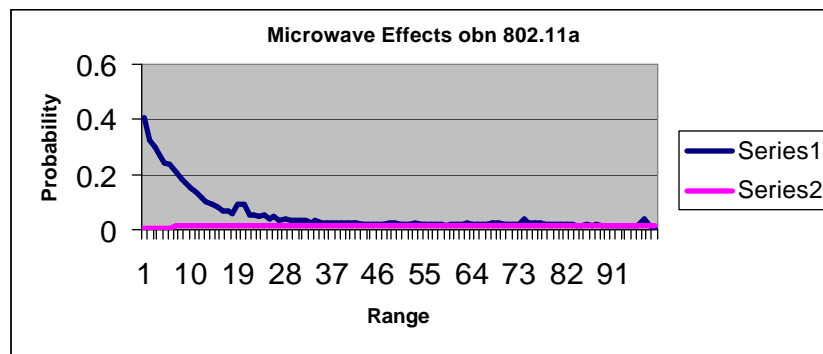


Figure 5-12: Microwave Effects on 802.11a

### 5.1.3.2 802.11b

From table 5-3, we can see that the microwave has significant effects on the 802.11b wireless signal. The signal level dropped by as much as 46% within such a close distance. The packet lost rate is 21.59%, much higher than any of the 1-wall and 2-wall experiments.

Instead of spikes like series 1 in figure 5-13 above, we see a wave like series 1 and 2. The error burst happens approximately every 5 packets away and is having an average probability of 50%, which is about 10 times more than 802.11a. One cycle is approximately 25 seconds (7 seconds 'on' and 18 seconds 'off') The probability of range 1 error burst is occurring most frequent. The wave like series 1 could be due to the heating up cycles of the

microwave. Series 2 shows an average of 48% that probability that the following packet(s) will error when previous packet is ok.

From this we can conclude that because the microwave and 802.11a is operating at same frequency (2.4GHz), therefore, microwave is a powerful source of interference to 802.11b wireless signal.

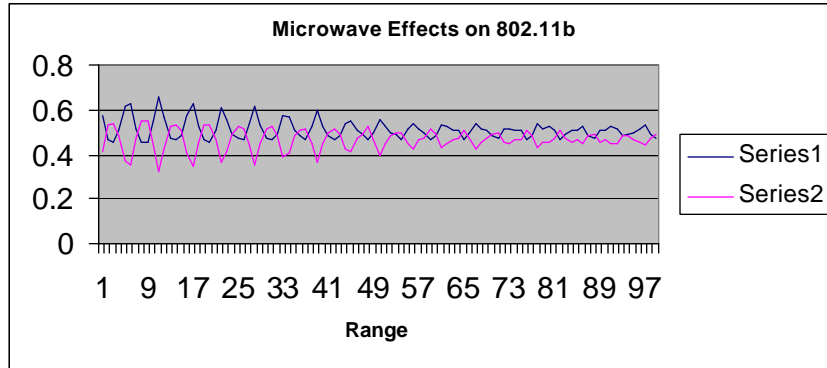


Figure 5-13: Microwave effects on 802.11b

## 5.2 Outdoor

### 5.2.1 Water Effects

In this experiment, a transmitter and receiver are separated by approximately 10m (Figure 5-14), and there are rain waters in between.

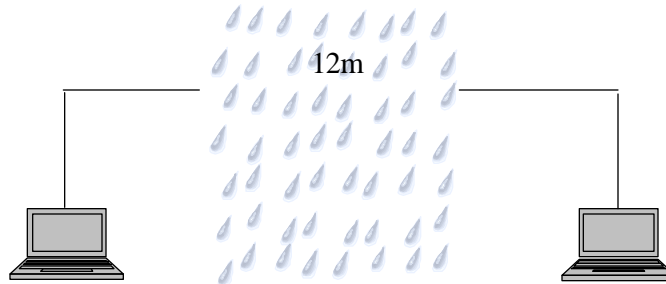


Figure 5-14: Layout of Water experiment

	A	B
	802.11a	802.11b
	Water	
Total Pk Tx	25000	25500
Total Pk Lost	24	47
Avg Pk Lost	4.8	9.4
Max Pk Lost	8	47
Min Pk Lost	1	0
Avg Signal Strength	84%	64%
Link Speed (Mbps)		
STDEV	2.774887385	21.01903899
% Pk Lost	0.09600%	0.18431%

Table 5-4: Result of Water experiment

### 5.2.1.1 802.11a

From table 5-4, we can see that the water effect on the wireless signals isn't as much as brick walls. The distance in this experiment is the same as the 2-wall experiment, but the signal level only drops by 16% instead of 33%. But the packet lost rate is double of the 2-wall without AP experiment.

In series 1 from figure 5-15 below, we can see there are few spikes. There is a sharp decrease of error burst rate between ranges 1 – 7, it drops from 11.96% to 0.87%. Significant increase in error burst occurring approximately between ranges 16-18, 23, 35 – 36, 55 and 83. Majority of the error burst rate is between 1 to 7 packets. Series 2 shows an average of 0.31% that probability that the following packet(s) will error when previous packet is ok.

From this experiment, we can see that water does have effect on the 802.11a wireless signals.

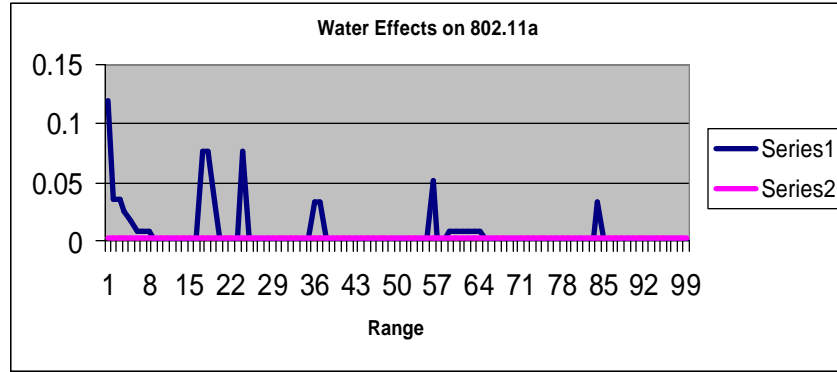


Figure 5-15: Water effects on 802.11a

### 5.2.1.2 802.11b

From table 5-4, we can see that the water has more effect on the 802.11b than 802.11a. The distance in this experiment is the same as the 2-wall experiment, but the signal level only drops by 36% instead of 54%. But the packet lost rate is higher than both of the 2-wall experiment.

From series 1 in figure 5-16 below, we can see there is a smooth decreasing of error burst rate from ranges 1 to 45. This implies that the maximum burst size with the water interference is about 45 packets. The probability to have a burst error rate of 2 packets is about 97%. Series 2 shows an average of 0.59% that probability that the following packet(s) will error when previous packet is ok.

From this experiment, we can see that water does have a greater effect on the 802.11b than 802.11a.

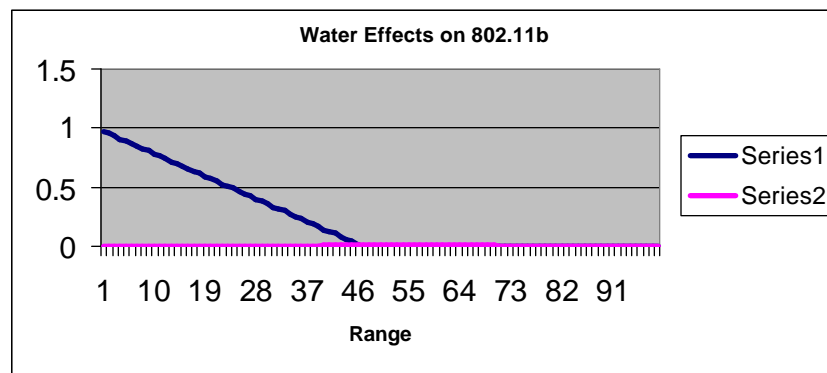


Figure 5-16: Water effects on 802.11b

## 5.2.2 Distance Effects

In this experiment, the only parameter change is the distance between the transmitter and the receiver.

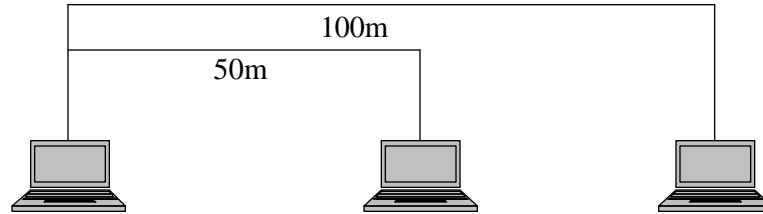


Figure 5-17: Layout of Distance experiment

	A		B	
	802.11a (50m)	802.11a (100m)	802.11b (50m)	802.11b (100m)
Total Pk Tx	50000	25000	25500	25500
Total Pk Lost	1707	8724	191	11355
Avg Pk Lost	170.70	1744.8	38.2	2271.00
Max Pk Lost	787	3473	187	2556
Min Pk Lost	0	880	0	2102
Avg Signal Strength	67%	56%	33%	2.00%
Link Speed (Mbps)	54	48	11	11
STDEV	341.6947	1109.177037	83.19074468	171.4730299
% Pk Lost	3.41%	34.89%	0.75%	44.53%

Table 5-5: Result of Distance experiment

### 5.2.2.1 50m

#### 5.2.2.1.1 802.11a

Although there isn't any passive obstacles or interferences in between the transmitter and receiver, and even though the transmitter and receiver are in the line of sight, from table 5-5, we can see that at 50m, the signal level drops to 67%, this demonstrate the path loss effects, the signal level changes as a function of distance. The packet lost rate is about 10 times higher comparing to the 1-wall experiment.

In series 1 from figure 5-18 below, we can see that the probability to have a range 1 error burst is 85.89%. There is a gentle and smooth decrease in the probability of the error burst rate between ranges 1 – 99, it drops from 85.89% to 44.32%. Series 2 shows a slight increment in the probability that the following packet(s) will error when previous packet is ok, it starts at the probability of 2.751% to 11.125%.

From this experiment, we can see that the distance has an great effect on the 802.11a as it weaker the signal. When the signals getting too weak, it does contribute to the error burst rate.

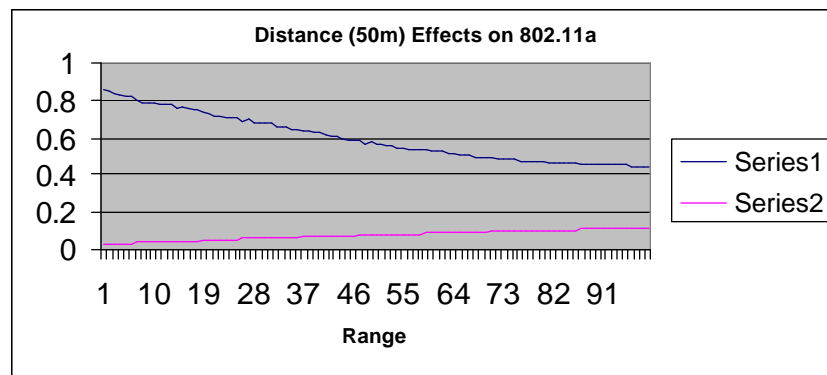


Figure 5-18: Distance (50m) effects on 802.11a

### 5.2.2.1.2 802.11b

From table 5-5 we can see that the signal level is dropped by more than 50%. The distance has a greater effect on 802.11b compare to 802.11a. Although the packet lost rate is not as bad as the microwave experiment, but is worse than the 2-wall AP experiment.

From series 1 in figure 5-19 below, we can see that the probability of error bursts are decreasing in nature. The probability of an error packet follow by an error packet is as high as 48.78%. It shows that there is a possibility to have an error burst rate of 100 packets. Series 2 shows an average of 2.821% that probability that the following packet(s) will error when previous packet is ok.

From this experiment, we can see that the distance has a greater effect on the 802.11b compare to 802.11a.

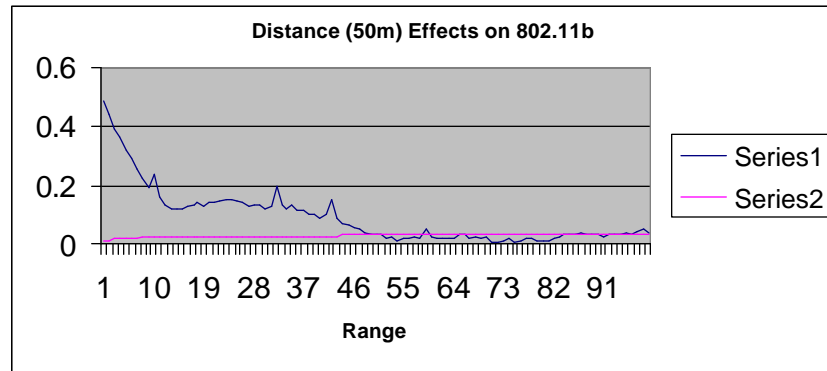


Figure 5-19: Distance (50m) effects on 802.11b

### 5.2.2.2 100m

#### 5.2.2.2.1 802.11a

From table 5-5, the signal level at 100m drop further to 56%, which is not really proportional to the distance increased. But the packet lost rate increase dramatically from 0.056% to 34.89%, which is much higher than the microwave experiment. Even 802.11b, who operates at the same frequency as microwave, when under the effect of microwave, the packet lost rate is not as high as this.

From series 1 in figure 5-20 below, we can see that the probability of error bursts are decreasing in nature, it decrease gently from 77.78% to 64.79%. It suggests that the probability to have an error burst of 100 packets is more than 50%. Series 2 shows a gradually increasing probability that the following packet(s) will error when previous packet is ok; it increases from 33.15% to 53.89%.

From this experiment, we can see that when the 802.11a up to its operating range limit, its performance degraded severely.

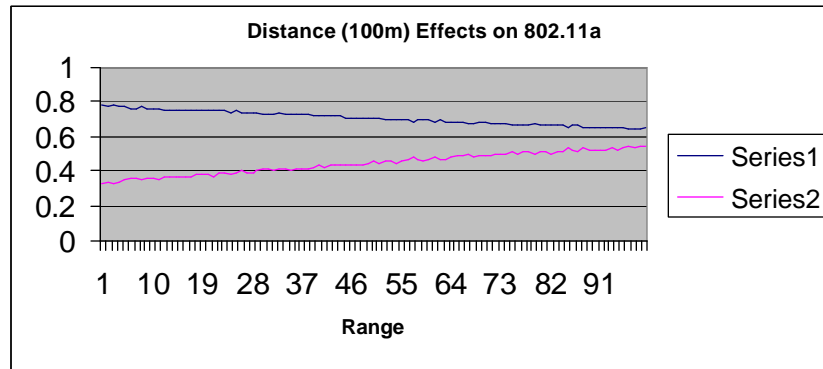


Figure 5-20: Distance (100m) effects on 802.11a

#### 5.2.2.2.2 802.11b

From table 5-5, we can see that the signal level at 100m is approaching zero while the error burst rate further increase 0.75% to 44.53, which is double of the packet lost rate in the microwave experiment.

The series 1 and 2 in figure 5-21 below shows similar waveform from the microwave experiment. The difference between the two is that series 1 over here is having a higher probability of error burst rate. The peak to peak variation is smaller compare to the microwave experiment. The peak is average every 4 packets away. Series 1 suggest that the probability to have an error burst of 100 packets is more than 83%. Series 2 shows a gradually increasing probability that the following packet(s) will error when previous packet is ok; it varies between 56.29% to 74.68%.

From this experiment, we can say that 802.11b is more prompt to environment than 802.11a as there are more electronic equipments operating at 2.4GHz.

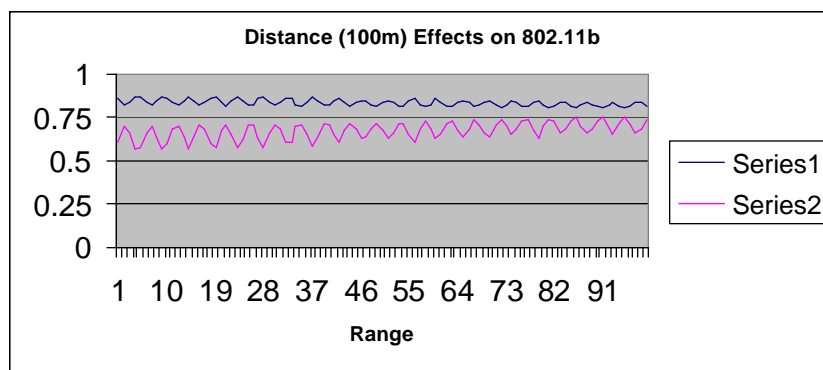


Figure 5-21: Distance (100m) effects on 802.11b

## 6. Experimental Results for Bit-level Analysis

This section shows the results of our experiments. All the experiments have been carried using multicast UDP frames.

Although we do not show the repeated results of experiments (i.e. confidence testing, which is available in the CD-ROM), however, we note similar patterns among the graphs for error correlation across repeated experiments.

Experiment 6.1 is carried out in a so called “perfect” environment where there are no foreseeable wireless impairments. It is used as a comparison to most of the other experiments below as new variables (possible wireless impairments) are added. And since this experiment is used as a “benchmark”, it is separated from the other line of sight experiments (Experiment 6.4).

An interesting observed behaviour to note, is that throughout all the bit-level experiments, the first significant spike in the Series 1 graph for error correlation (i.e. probability of one character matching target ‘1’ given that the character RANGE characters back matched), always occurs at Range 7. This means a higher probability that the 7<sup>th</sup> bit will be errored if 6 bits before are already in error, this is followed by a significant drop in the probability of the next bit being errored, thus most errors are 8 bits long regardless of the impairments.

## 6.1 Line of sight, within 1m

Other Possible Interference: None

Experimental Set Up:

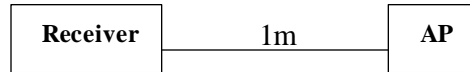


Figure 6-1: Line of Sight, within 1m

### 6.1.1 Results for Line of Sight, 1m

Average number of bits corrupted per frame: 37

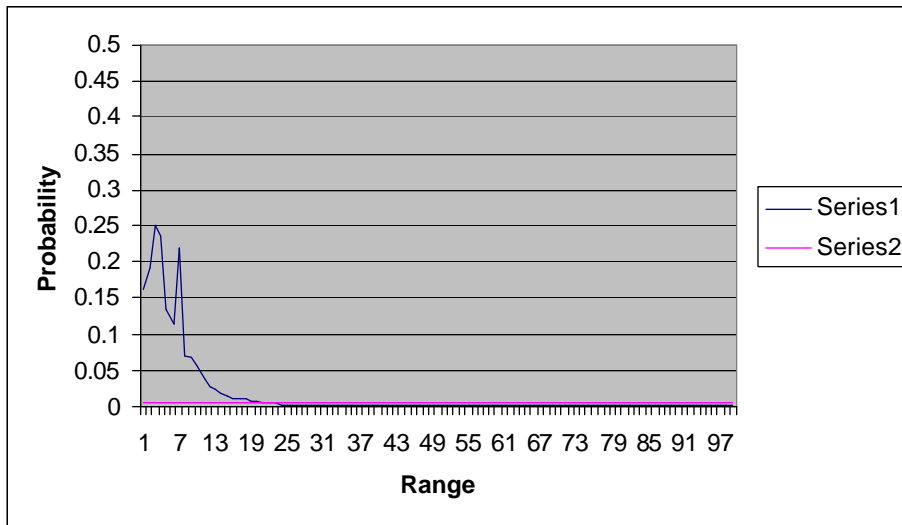


Figure 6-2: Error correlation for Line of sight, within 1m

### 6.1.2 Observations

1. Probability 0->1 is approximately the same as 1->0.
2. Significant error correlation in the Ranges of 7 and 4.

## 6.2 Passive Obstacles

### 6.2.1 1 Wall

Other Possible Interference: None

Experimental Set Up:

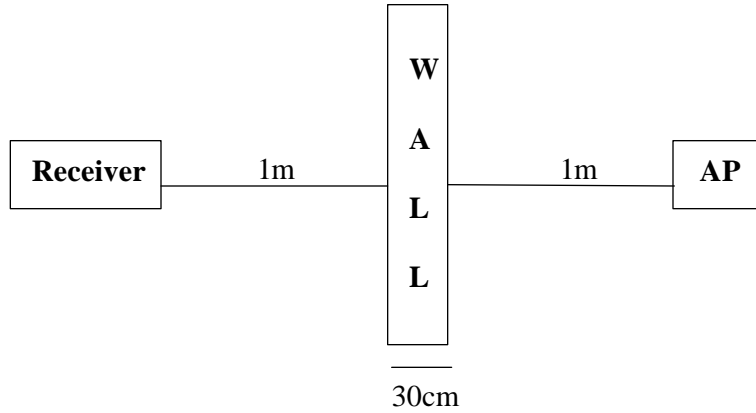


Figure 6-3: Passive Obstacle, 1 Wall

#### 6.2.1.1 Results of Passive Obstacle, 1 Wall

Average number of bits corrupted per frame: 156

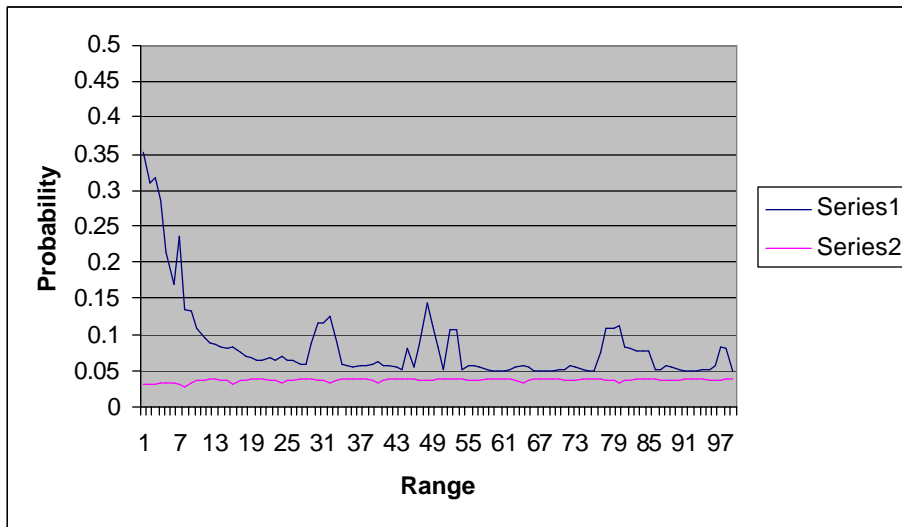


Figure 6-4: Error correlation for Passive Obstacle, 1 Wall

### 6.2.1.2 Observations

#### i. Similarities across similar 1 wall experiments:

1. Probability of 0->1 is approximately 10% more than 1->0.
2. Significant increase in error correlations (spikes in Series 1) occurring approximately between Ranges: 7, 29 – 31, 45 – 49, and 77 – 81.
3. Series 2 shows a 1 - 3% probability that the following bit will error when previous bits are ok.

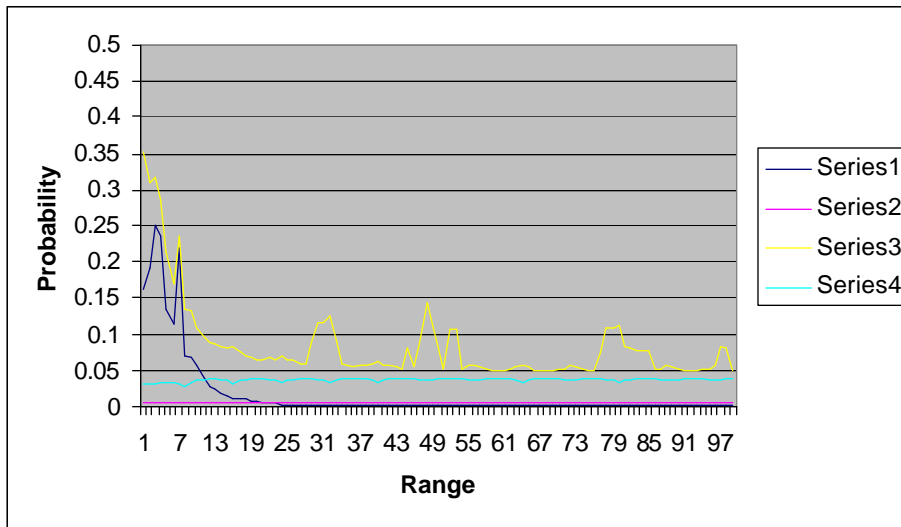


Figure 6-5: Comparison between Experiment 6.1 and Passive Obstacle, 1 Wall

#### ii. Similarities and differences compared to experiment 6.1:

1. Both graphs show an increased probability at Range 7, thus regardless of a 1 wall impairment this spike will occur.
2. An increase of approximately 1 - 5% across Series 1 (excluding the spikes), therefore a higher probability of correlated errors.
3. An increase of 1% on Series 2, therefore a small increase in the probability that the next bit will error when the previous bits are ok.
4. Differences between the two graphs are the spikes at Ranges 29-31, 45-49, and 77 – 81 with a probability of 7 – 14%.

### 6.2.1.3 Conclusion: Passive Obstacle, 1 Wall

1. A higher probability (approx. 1 - 5% increase, spikes excepted) as compared to experiment 6.1 at which errors correlate, is caused by the 1 wall impairment.

### 6.2.2 2 Walls

Other Possible Interference: None

Experimental Set Up:

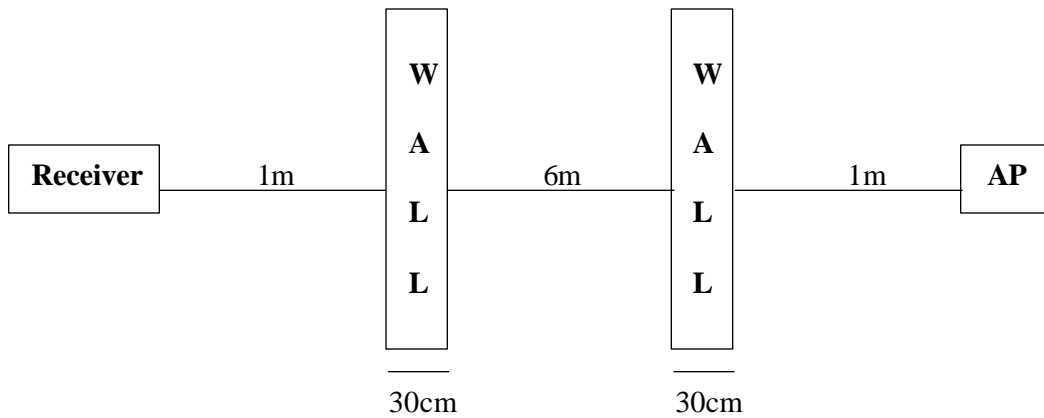


Figure 6-6: Passive Obstacles, 2 Walls

### 6.2.2.1 Results of Passive Obstacles, 2 Walls

Average number of bits corrupted per frame: 209

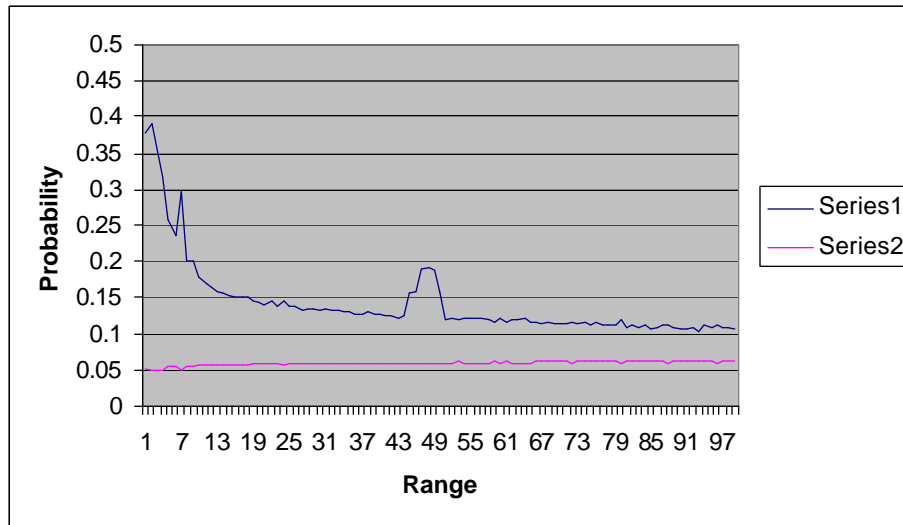


Figure 6-7: Error correlation for Passive Obstacles, 2 Walls

### 6.2.2.2 Observations

#### i. Similarities across similar 2 wall experiments:

1. 1->0 and 0->1 vary, sometimes 1->0 more than 0->1 and vice versa, however difference between 1->0 and 0->1 is approximately 10%.
2. Significant increase in error correlations (spikes in Series 1) occurring approximately between Ranges: 7, 44 - 48.
3. Series 2 shows a 2 – 6% probability that the following bit will error when previous bits are ok.

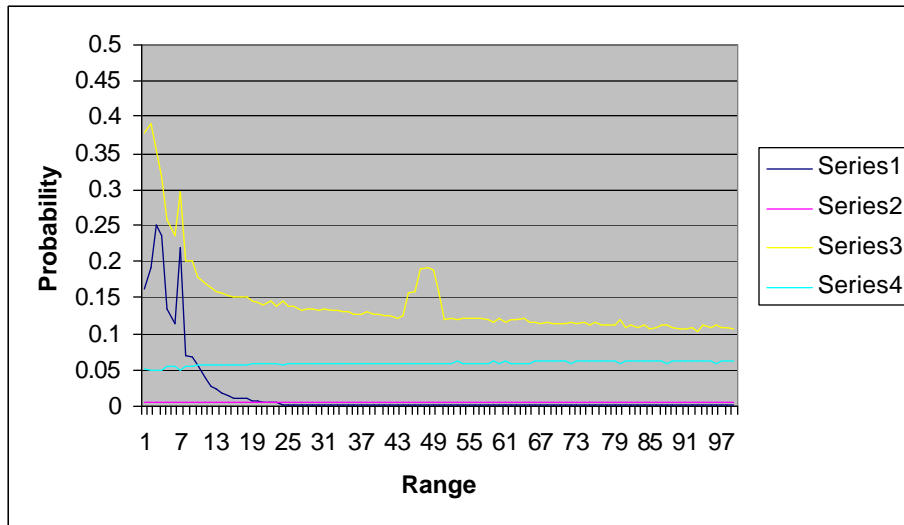


Figure 6-8: Comparison between Experiment 6.1 and Passive Obstacles, 2 Walls

**ii. Similarities and differences compared to experiment 6.1:**

1. Both graphs show an increased probability at Range 7, thus regardless of the 2 wall impairment this spike will occur.
2. Differences between the two graphs are the spikes at Range 44 – 48.
3. Increased error correlation of 5 -14% (Series 1 and 3) and 5 – 6% (Series 2 and 4) for the next bit will be errored.

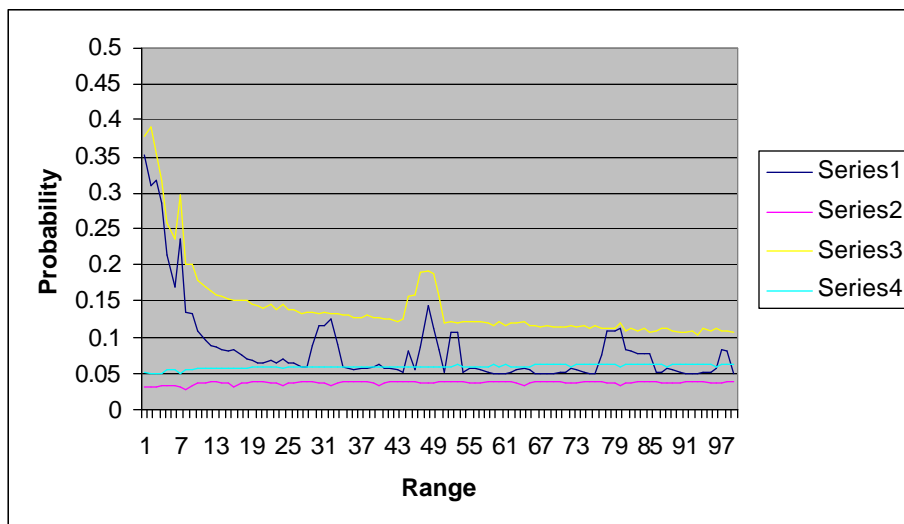


Figure 6-9: Comparison between Passive Obstacles 1 Wall and 2 Walls

### iii. Similarities and differences compared to 1 wall:

1. Both graphs show an increased probability (a spike) at Range 7, however, 2 walls have a higher probability of approximately 25 – 30% (i.e. 7-10% more).
2. A similar spike can also be observed at Range 44 – 48 with 2 walls as compared to Range 45 – 49 with 1 wall.
3. Differences are the spikes at Ranges 29-31 and 77 – 81 for 1 wall whereas 2 walls do not have increased probability of correlated errors at these Ranges.
4. Increased probability of correlated errors of approximately 10 – 13% (i.e. approximately 5% more).

#### 6.2.2.2 Conclusion: Passive Obstacles, 2 Walls

1. A higher probability at which errors correlate across the entire Range, as compared to Experiment 6.1, is caused by the 2 wall impairment.

## 6.3 Competing sources

### 6.3.1 Microwave Oven

Other Factors: None

Experimental Set Up:

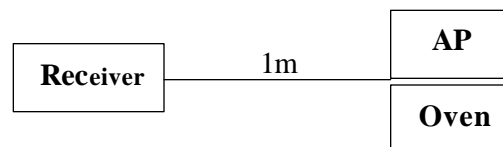


Figure 6-10: Competing Sources, Microwave Oven

### 6.3.1.1 Results of Competing Sources, Microwave Oven

Average number of bits corrupted per frame: 190

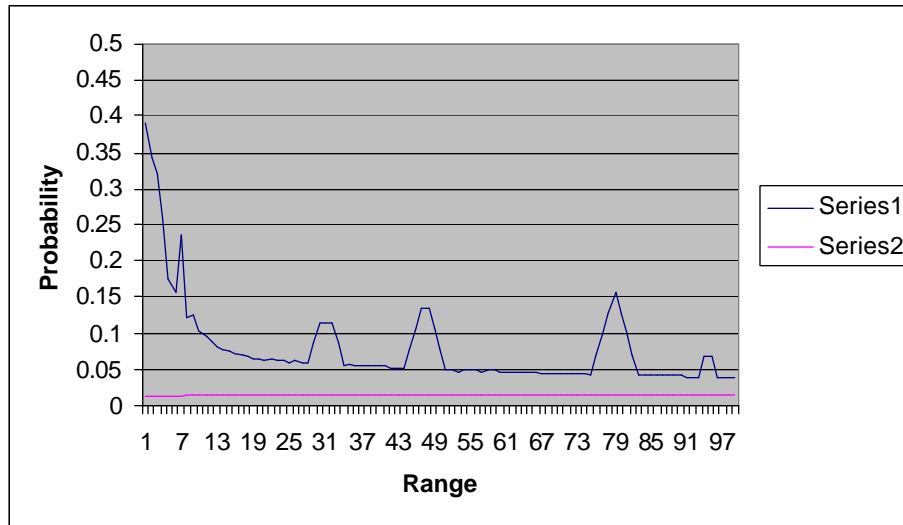


Figure 6-11: Error correlation for Competing Sources, Microwave Oven

### 6.3.1.2 Observations

#### i. Similarities across similar microwave oven experiments:

1. Probability of 0->1 is approximately 7-20% more than 1->0.
2. Significant increase in error correlations (spikes in Series 1) occurring approximately between Ranges 29 – 31, 45 – 48 and 76 – 79.
3. Series 2 shows a 1 – 6% probability that the following bit will error when previous bits are ok.

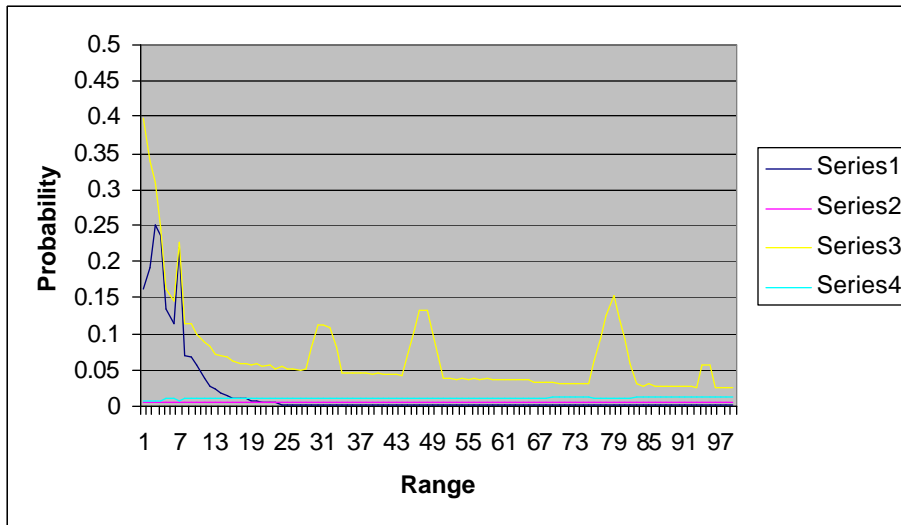


Figure 6-12: Comparison between Experiment 6.1 and Competing Sources, Microwave Oven

**ii. Similarities and differences compared to experiment 6.1:**

1. Both graphs show an increased probability at Range 7, thus regardless of the microwave impairment this spike will occur.
2. Differences between the two graphs are the spikes at Ranges 29 – 31, 45 – 48 and 76 – 79.

**6.3.1.3 Conclusion: Competing Sources, Microwave Oven**

1. The spikes in the observations of this experiment are caused by the microwave oven which operates at the same frequency as the 802.11b card.
2. The increased probability of errors correlating is also caused by the microwave interference.

### 6.3.2.1 Access Point (Uniwide)

Other factors: Unspecified number of clients using the Uniwide Access Point

Experimental Set Up:

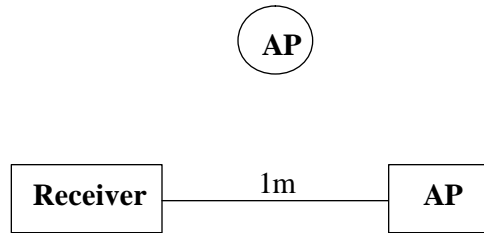


Figure 6-13: Competing Sources, 1 AP

#### 6.3.2.1 Results for Competing sources, 1 AP

Average number of bits corrupted per frame: 155

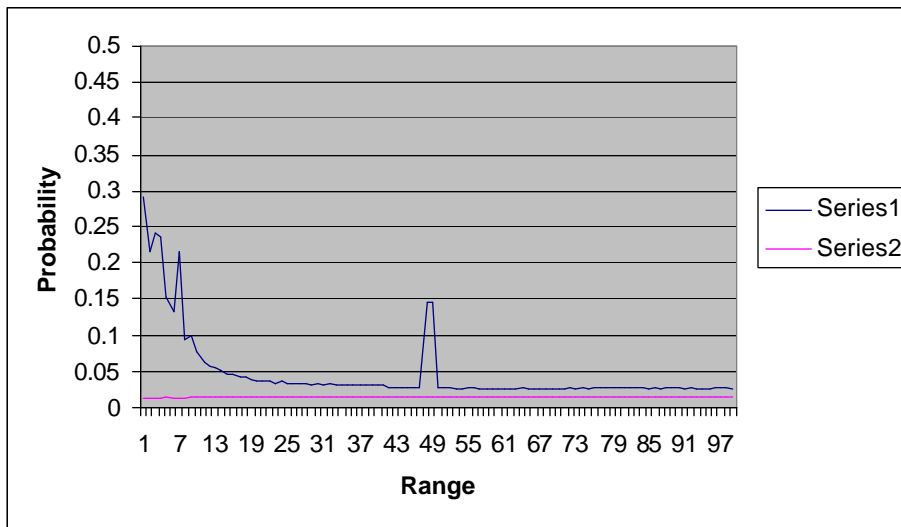


Figure 6-14: Error correlation for Competing Sources, 1 AP

#### 6.3.2.2 Observations

##### i. Similarities across similar 1 competing Access Point experiments:

1. 1->0 and 0->1 vary, sometimes 1->0 more than 0->1 and vice versa.
2. Significant increase in error correlations (spikes in Series 1) occurring approximately between Ranges: 7, 48 - 49.

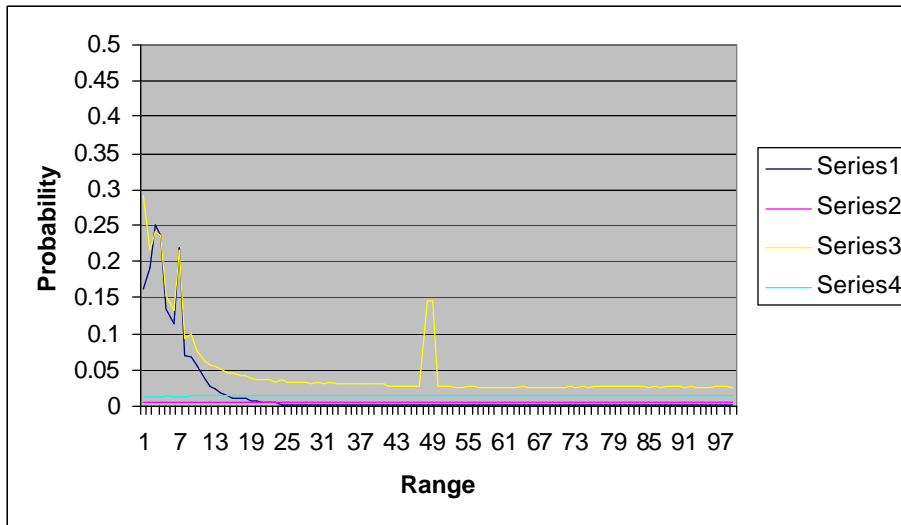


Figure 6-15: Comparison between Experiment 6.1 and Competing Sources, 1 AP

**ii. Similarities and differences compared to experiment 6.1:**

1. Both graphs show an increased probability at Range 7, thus regardless of the microwave impairment this spike will occur.
2. Differences between the two graphs are the spikes at Range 48 - 49 with a probability of approximately 15%.

**6.3.2.3 Conclusion: Competing Sources, 1 AP**

1. The increased probability of correlated errors is marginal (approximately 1 – 2%) more than in a perfect environment (i.e. experiment 6.1).

## 6.4 Line of Sight

The distance experiments were carried out in an empty lecture theatre (Clancy auditorium); so as to simulate as close as possible the distance variable, this was necessary as electrical power points were needed for both the receiver and the transmitter. However, beacon frames sent out by the Uniwide Access Point in the lecture could pose cause significant interference, with reference to Tuli and Dhanani's experiment which characterized the behaviour of an access point[2], it shows that beacon frames are transmitted very frequently in order to show their presence in the network.

### 6.4.1 50m

Other factors: Beacon frames sent out by Uniwide Access Point

Experimental Set Up:

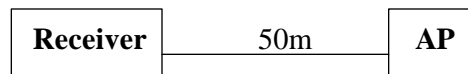


Figure 6-16: Line of Sight, 50m

#### 6.4.1.1 Results for Line of Sight, 50m

Average number of bits corrupted per frame: 162

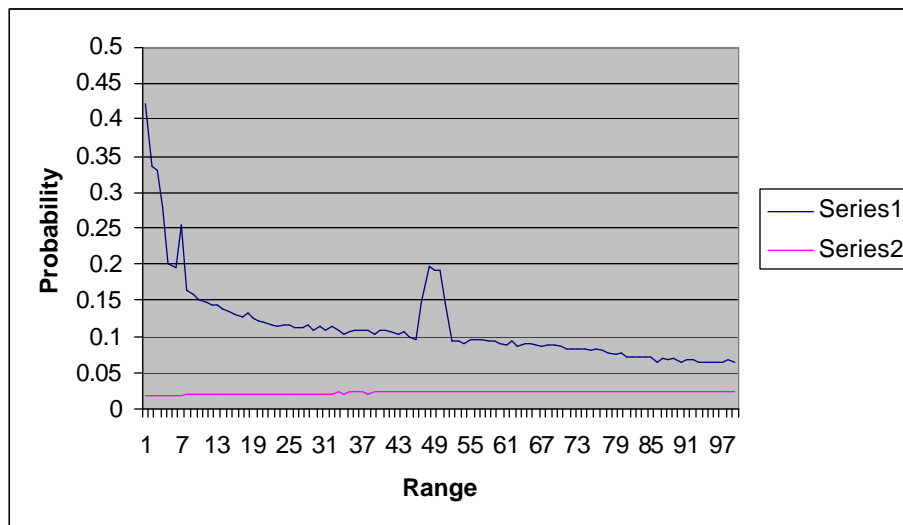


Figure 6-17: Error correlation for Line of Sight, 50m

#### 6.4.1.2 Observations

i. Similarities across similar distance 50 meters experiments:

1. Probability of 0->1 is approximately 13% more than 1->0.
2. Significant increase in error correlations (spikes in Series 1) occurring approximately between Ranges: 7, 47 - 49.
3. Series 2 shows a 3 – 4% probability that the following bit will error when previous bits are ok.

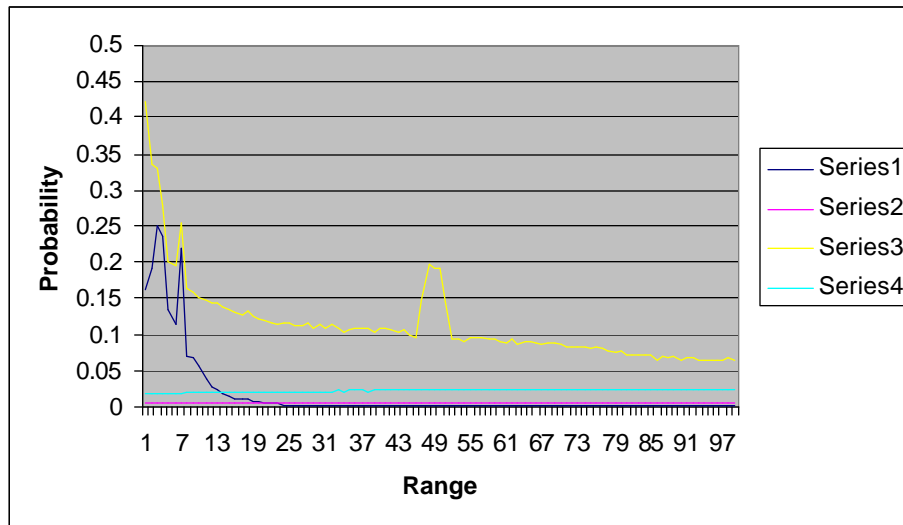


Figure 6-18: Comparison between Experiment 6.1 and Line of Sight, 50m

**ii. Similarities and differences compared to Experiment 6.1:**

1. Both graphs show an increased probability at Range 7, thus regardless of the 50 meter distance this spike will occur.
2. Differences between the two graphs are the spikes at Range 47 - 49 with a probability of approximately 20%.

**6.4.1.3 Conclusion: Distance, 50m**

1. The increased probability of correlated errors is significantly higher (approximately 7 - 10%) than in a perfect environment (i.e. experiment 6.1), this is most probably caused by the 50 meter distance factor.
2. The spike at Range 47 – 49 *could* possibly be caused by the Access Point present in the lecture theatre and not the distance.

## 6.4.2 100 meters

Other factors: Beacon frames sent out by Uniwide Access Point

Experimental Set Up:

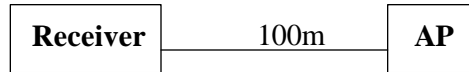


Figure 6-19: Experimental Set Up for Distance, 100m

### 6.4.2.1 Results for Distance, 100m

Average number of bits corrupted per frame: 470

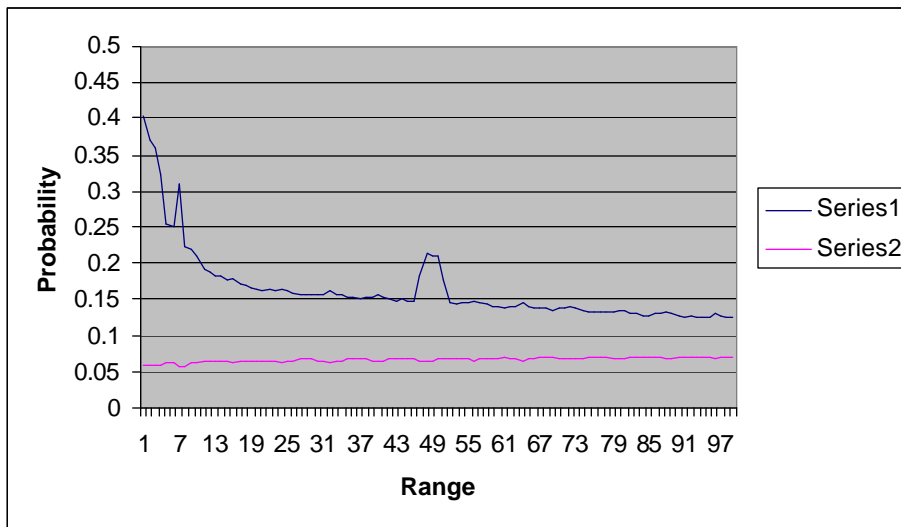


Figure 6-20: Error correlation for Distance, 100m

### 6.4.2.2 Observations

#### i. Similarities across similar distance 100 meter experiments:

1. Probability of 0->1 is approximately 10% more than 1->0.
2. Significant increase in error correlations (spikes in Series 1) occurring approximately between Ranges: 7, 45 - 49.
3. Series 2 shows a 5 – 6% probability that the following bit will error when previous bits are ok.

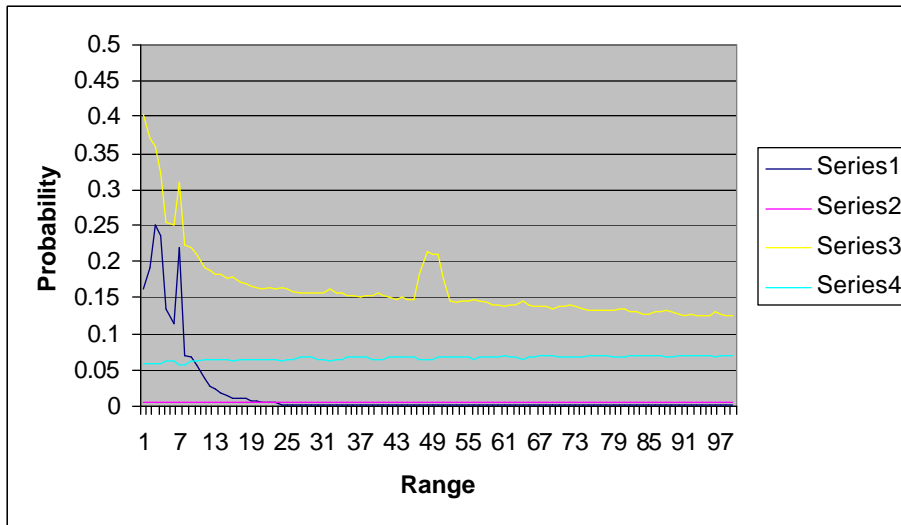


Figure 6-21: Comparison between Experiment 6.1 and Line of Sight, 100m

**ii. Similarities and differences compared to experiment 6.1:**

1. Both graphs show an increased probability at Range 7, thus regardless of the 100 meter distance this spike will occur.
2. Differences between the two graphs are the spikes at Range 48 - 49 with a probability of approximately 21% at the peak.

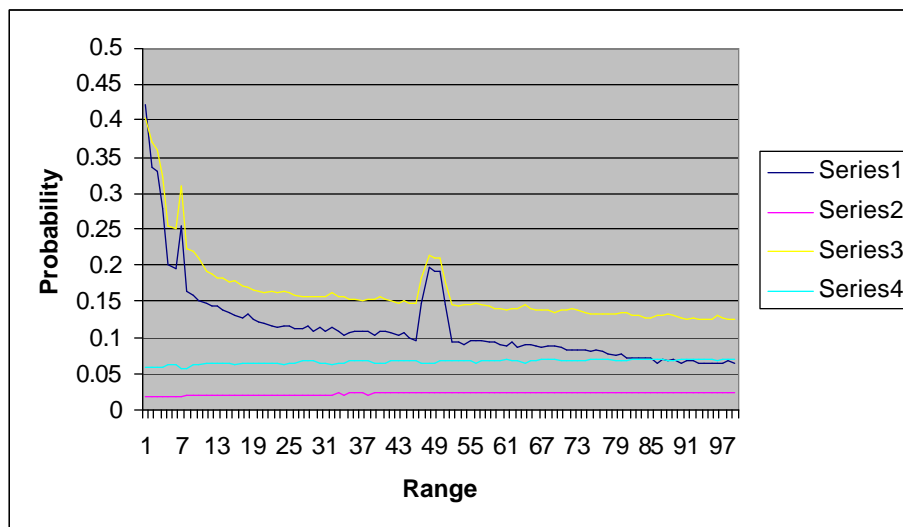


Figure 6-22: Comparison between Line of Sight, 50m and 100m

### **iii. Similarities and differences compared to 50 meters:**

1. Error correlation rate for 100 meters shows an increased probability of about 5%.
2. A similar spike can also be observed at Range 48 – 49 with distance 100 meters as compared to Range 47 – 49 with 1 wall.
3. The probability of the following bits corrupting when the previous bits are ok (i.e. Series 2 and 4) is higher at 100 meters than at 50 meters.

### **6.4.2.3 Conclusion: Line of Sight, 100m**

1. The increased probability in error correlations is most probably caused by the increase of the distance between receiver and transmitter.
2. The probability of a higher probability bits corrupting when the previous bits are ok is also caused by an increased distance between receiver and transmitter.

## **7. Future Work**

As we were unable to strip the CRC for the 802.11a driver, this prevented us from comparing the differences of transmission errors at the bit level between 802.11a and 802.11b. In the future, the 802.11a driver (which is still in beta test), could provide utilities which would help facilitate the stripping of the CRC in promiscuous mode similar to that of the AVS IEEE 802.11b's wlanctl-ng utility.

## References

- [1] D. Eckhardt, P. Steenkiste. *Measurement and Analysis of the Error Characteristics of an In-Building Wireless Network*, Computer Communication Review 26:4, pp. 243-254, October 1996, ACM SIGCOMM, ISSN #0146-4833 (CCR 26:4 is the proceedings of [SIGCOMM '96](#); [presentation slides are also available](#)).
  
- [2] N. Tuli and A. Dhanani. *Wireless LAN Measurements Project*, Computer Science and Engineering, UNSW, Sydney, Australia, November 2002.
  
- [3] [IEEE 802.11, 1999 Edition](#) (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology -- Telecommunications and Information Exchange between Systems -- Local and Metropolitan Area Network -- Specific Requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

## Appendix A – Driver Installation and Configuration for 802.11b

In order to install and configure the D-LINK card it is required to have following packages:

1. Linux Kernel Source Code.
2. Pcmcia-cs drivers.
3. Linux-wlan.ng drivers

The above packages must be stored in /usr/src directory.

Following commands have to be typed after downloading the drivers

1. `cd /usr/src`
2. `tar -xvzf pcmcia-cs.3.2.1.gz`
3. `tar -xvzf linux-wlan.ng-0.1.16.tar.gz`
4. `cd linux-2.4`
5. `cd configs/`
6. `cp kernel-2.4.18-i686.config ../config`
7. `cp pcmcia-cs-3.2.1`
8. `./Configure`

When the `./Configure` command is run series of question will be asked. The configuration instruction shown below should be followed exactly in order to make the card functional.

```
----- Linux PCMCIA Configuration Script -----
```

```
The default responses for each question are correct for most users. Consult the PCMCIA-HOWTO for additional info about each option/
```

```
Linux kernel source directory [/usr/src/linux]: /usr/src/linux-2.4.18-3
```

```
The kernel source tree is version 2.4.18-3custom
```

```
WARNING: the current kernel is sublevel 2.4.18-3
```

```
The current kernel build data is Thu Apr 18 07:37:53 2002
```

```
Build 'trusting' versions of card utilities (y/n) [n] :
```

```
Include 32-bit (CardBus) card support (y/n) [y] :
```

```
Include PnP BIOS resource checking (y/n) [n] :
```

The PCMCIA drivers need to be compiled to match the kernel they will be used with, or some or all of the modules may fail to load. If you are not sure what to do, please consult the PCMCIA-HOWTO.

How would you like to set kernel-specific options?

1. – Read from the currently running kernel
2. – Read from the linux source tree

Ether option (1-2) [1] :

Module install directory [/lib/modules/2.4.18-3] :

Kernel configuration options:

- Kernel-tree PCMCIA support is enabled.
- Symmetric multiprocessing support is disabled.
- PCI BIOS support is enabled.
- Power management (APM) support is enabled.
- SCSI support is enabled.
- IEEE 1394 (FireWire) support is disabled.
- Networking support is enabled.
  - Radio network interface support is enabled.
  - Token Ring service support is enabled.
  - Fast switching is disabled.
  - Frame Diverter is disabled.
- Module version checking is enabled.
- Kernel debugging support is disabled.
- Preemptive kernel patch is disabled.
- /proc filesystem support is enabled.

It doesn't look like you are using 'lilo'.

It looks like you have a System V init file setup.

X Window System include files found.

Forms library not installed

If you wish to build the 'cardinfo' control panel, you need the Forms.

Library and the X Windows System include files, See the HOWTO form details.

Configuration successful.

Your kernel is configured with PCMCIA driver support. Therefore, 'make all' will compile the PCMCIA utilities but not the drivers.

Once the configuration is successful the below command can be typed at shell.

- Make all
- Make install

This would complete the installation and configuration of PCMCIA slot.

Similar instruction needs to be followed to install and configure the drivers.

- Cd ../linux-wlan-ng-0.1.15
- ./Configure

When the `./Configure` command is run series of question will be asked. The configuration instruction shown below should be followed exactly in order to make the card functional.

```
----- Linux WLAN Configuration Script -----

The default responses are correct for most users.

Build Prism2.x PCMCIA Card Services (_cs) driver? (y/n) [y] :

Build Prism2 PLX9052 based PCI (_plx) adapter driver? (y/n) [n] :

Build Prism2.5 native PCI (_pci) driver? (y/n) [n] :

Build Prism2.5 USB (_usb) driver? (y/n) [n] :

Linux source directory [/usr/src/linux]: /usr/src/linux-2.4.18-3

The kernel source tree is version 2.4.18-3.
The current kernel build data is Thu Apr 18 07:37:53 2002.

Pcmcia-cs source dir [/usr/src/pcmcia-cs-3.2.1]:

Alternate target install root directory on host []:
PCMCIA script directory [/etc/pcmcia]:
    Module install director [/lib/modules/2.4.18-3]:

It looks like you have a System V init file setup.

Prefix for build host compiler? (rarely needed) []:

Configuration successful.
```

- `make all`
- `make install`

Once the installation is successful it is required to change the SSID in `/etc/pcmcia/wlan-ng.opts` file. The SSID will be the SSID of the nearest access point (AP),

After this copy the file `/etc/sysconfig/network-scripts/ifcfg-eth0` to `/etc/sysconfig/network-scripts/ifcfg-wlan0`

- `reboot`

This is the complete installation and configuration of the pcmcia slot and drivers for any card which support prism2 chipset.

## Appendix B – Driver Installation and Configuration for 802.11a

In order to install and configure the Netgear card, it is required to have following package:

1. Linux Kernel Source Code.
2. sharutils-4.2.1-14.i386.rpm
3. madwifi-0.8.3.2.tar or madwifi-20030802.gz

The above packages can be stored in /usr/src directory.

Following command shave to be typed after downloading the drivers

- cd /usr/src
- rmp -ivh sharutils-4.2.1-14.i386.rpm
- tar -xvzf madwifi-0.8.3.2.tar or tar -xvzf madwifi-20030802.gz
- cd linux-2.4
- cd configs/
- cp kernel-2.4.18-i686.config ../config
- cd ../madwifi-0.8.3.2 or cd ../madwifi-20030802

in order to enable Ad-hoc mode in the current version (20030802), we need to change two lines in the file wlan/if\_ieee80211wireless.c. The changes are as follow:

```
ifr.ifr_media == IFM_IEEE80211_IBSS; to  
ifr.ifr_media == IFM_IEEE80211_ADHOC;  
and  
else if (imr.ifm_active & IFM_IEEE80211_IBSS) to  
else if (imr.ifm_active & IFM_IEEE80211_ADHOC)
```

Once the lines are changed, the below command can be typed at shell.

- make all
- make install
- reboot

## Appendix C – IEEE 802.11 Channel – Frequency Table

Channel	Central Frequency
1	2.412
2	2.417
3	2.422
4	2.427
5	2.432
6	2.437
7	2.442
8	2.447
9	2.452
10	2.457
11	2.462

## Appendix D – ASCII Table

Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex
0	00	NUL	16	10	DLE	32	20	48	30
								0	64
								40	@
								80	50
								P	96
								60	`
								112	70
								p	
1	01	SOH	17	11	DC1	33	21	!	49
								31	1
								65	41
								A	81
								51	Q
								97	61
								a	113
								71	q
2	02	STX	18	12	DC2	34	22	"	50
								32	2
								66	42
								B	82
								52	R
								98	62
								b	114
								72	r
3	03	ETX	19	13	DC3	35	23	#	51
								33	3
								67	43
								C	83
								53	S
								99	63
								c	115
								73	s
4	04	EOT	20	14	DC4	36	24	\$	52
								34	4
								68	44
								D	84
								54	T
								100	64
								d	116
								74	t
5	05	ENQ	21	15	NAK	37	25	%	53
								35	5
								69	45
								E	85
								55	U
								101	65
								e	117
								75	u
6	06	ACK	22	16	SYN	38	26	&	54
								36	6
								70	46
								F	86
								56	V
								102	66
								f	118
								76	v
7	07	BEL	23	17	ETB	39	27	'	55
								37	7
								71	47
								G	87
								57	W
								103	67
								g	119
								77	w
8	08	BS	24	18	CAN	40	28	(	56
								38	8
								72	48
								H	88
								58	X
								104	68
								h	120
								78	x
9	09	HT	25	19	EM	41	29	)	57
								39	9
								73	49
								I	89
								59	Y
								105	69
								i	121
								79	y
10	0A	LF	26	1A	SUB	42	2A	*	58
								3A	:
								74	4A
								J	90
								5A	Z
								106	6A
								j	122
								7A	z
11	0B	VT	27	1B	ESC	43	2B	+	59
								3B	;
								75	4B
								K	91
								5B	[
								107	6B
								k	123
								7B	{
12	0C	FF	28	1C	FS	44	2C	,	60
								3C	<
								76	4C
								L	92
								5C	\
								108	6C
								I	124
								7C	
13	0D	CR	29	1D	GS	45	2D	-	61
								3D	=
								77	4D
								M	93
								5D	]
								109	6D
								m	125
								7D	}
14	0E	SO	30	1E	RS	46	2E	.	62
								3E	>
								78	4E
								N	94
								5E	^
								110	6E
								n	126
								7E	~
15	0F	SI	31	1F	US	47	2F	/	63
								3F	?
								79	4F
								O	95
								5F	_
								111	6F
								o	127
								7F	DEL



## Appendix F – Error Correlation Program

```
#include <stdio.h>
#include <stdlib.h> // exit()
main() {
#define RANGE 100
    int match_match[RANGE];
    int match_mismatch[RANGE];
    int mismatch_match[RANGE];
    int mismatch_mismatch[RANGE];
    char buffer[RANGE];
    int i, j;
    char target = '1';

    // Prefill the buffer & initialise arrays
    for(i=0; i<RANGE && !feof(stdin); ++i) {
        buffer[i] = getchar();
        match_match[i] = match_mismatch[i] = mismatch_mismatch[i] =
mismatch_match[i] = 0;
    }
    if(feof(stdin))
    {
        fprintf(stderr, "Couldn't test for correlations over %d samples, since
only %d samples are available\n", RANGE, i);
        exit(EXIT_FAILURE);
    }
    i=0;
    do {
        // Scan (++) for matched/mismatched within RANGE of current
        // position (i)
        for(j=1; j<RANGE; ++j) {
            if(buffer[(i+j) % RANGE] == target) {
                if(buffer[i]==target)
                    ++match_match[j];
                else
                    ++mismatch_match[j];
            } else {
```

```

    if(buffer[i]==target)
        ++match_mismatch[j];
    else
        ++mismatch_mismatch[j];
    }
}

buffer[i] = getchar();
i = (i+1) % RANGE;

} while (!feof(stdin));

printf("Range\tP(match|match)\tP(match|mismatch)\n");
for(i=1; i<RANGE; ++i)
    printf("%d\t%f\t%f\n",i, (float)match_match[i]/((float)match_match[i] +
match_mismatch[i]),
(float)mismatch_match[i]/((float)mismatch_match[i]+mismatch_mismatch[i]));
}

```

## Appendix G – Correlation Program Output

Range	P(match match)	P(match mismatch)
1	0.416444	0.000205
2	0.430317	0.000428
3	0.439166	0.000630
4	0.434155	0.000848
5	0.399823	0.001048
6	0.390719	0.001249
7	0.420516	0.001442
8	0.374960	0.001617
9	0.373362	0.001896
10	0.365071	0.002127
11	0.359082	0.002358
12	0.354619	0.002598
13	0.350790	0.002816
14	0.347459	0.002983
15	0.344128	0.003176
16	0.341081	0.003347
17	0.337858	0.003598
18	0.334877	0.003828
19	0.331590	0.004076
20	0.329243	0.004286
21	0.326080	0.004445
22	0.324080	0.004600
23	0.321105	0.004779
24	0.319142	0.004898
25	0.316704	0.005125
26	0.313636	0.005368
27	0.311258	0.005593
28	0.308530	0.005802
29	0.305633	0.005972
30	0.302570	0.006125
31	0.299707	0.006293
32	0.297544	0.006398
33	0.294314	0.006658
34	0.290981	0.006901
35	0.287512	0.007120
36	0.284950	0.007337
37	0.280993	0.007489
38	0.277752	0.007640
39	0.274954	0.007782
40	0.272195	0.007887
41	0.269340	0.008115
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.

Average bits corrupted: 99  
Percentage of headers corrupted: 0.75

