Honours project 2012

GROUT

Exploring Bluetooth and wireless transfer to aid the construction of adhoc-networks via smartphones

Done by: Tsu-Shiuan Lin

Supervised by: Gary Marsden

	Category	Min	Max	Chosen
1	Requirement Analysis and Design	0	20	17
2	Theoretical Analysis	0	25	0
3	Experimental Design and Evaluation	0	20	18
4	System Development and Implementation	0	15	0
5	Results, Findings and Conclusion	10	20	15
6	Aim Formulation and Background Work	10	15	10
7	Quality of Report Writing and Presentation	10		10
8	Adherence to Project Proposal and Quality of Deliverables	10		10
9	Overall General Project Evaluation	0	10	0
Total marks		**80**		**80**

Department of Computer Science

University of Cape Town

2012

Abstract

The vast and ever growing popularity of smart phones in today's society creates a need for cheap and effective wireless file transfer methods. Despite the availability of internet resources, retrieving information through data networks is expensive. Due to the increased capabilities of smart phones, it is proposed in this paper that alternate methods can be used to send data across multiple smart phones. In particular, two different wireless technologies were explored to facilitate the creation of adhoc networks amongst smart phones.

The two wireless technologies presented in this paper are Bluetooth and Wi-Fi. Each candidate was tested according to the following criteria: Latency, Data transfer rate, Power consumption, Robustness. Each of these criteria was further evaluated according to these specific sub-categories: File size, different Bluetooth versions, buffer size. In particular, different data transfer algorithms were also evaluated for Bluetooth. Lastly, the scalability factor was evaluated whereby more than two smartphones were exchanging data through their multi-tasking capabilities.

Although Windows Phone was the original choice, Android was the chosen mobile platform due to its extensive and rich Bluetooth and Wi-Fi libraries. This research paper proposes a refined data exchange protocol to decrease power consumption, increase data transfer rate whilst providing steady and reliable wireless connections between smart phones.

Acknowledgements

Firstly, I would like to thank Prof. Gary Marsden for all the guidance and feedback he has provided throughout the course of the project. He always made time to go through my work as well as pointing me in the right direction. Furthermore he provided the GROUT team Android devices for us to perform experiments and testing.

Secondly I would like to acknowledge the NRF Research Foundation (NRF) for the financial assistance provided towards this research. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Table of Contents

Chapter 1

1. Introduction

The convenience and effectiveness of smart phones has allowed them to be integrated into our daily lives [11]. Providing an array of functionalities such as alarm clocks and calendars, smart phones have replaced several gadgets common to non-smart phone users [9]. From email notifications, social networking to music and much more, the amount of interaction between the user and smart phones has drastically increased over the years [10]. Downloading and uploading of data and media have become readily accessible due to the constant expansion of the internet. Consequently, this invokes a strain on network data usage particularly in African countries where data usage is capped [12]. This limitation restricts users from being able to effectively and freely exchange data. Bluetooth and Wi-Fi are good alternative tools to enable data transfers between multiple smart phone devices through an adhoc network. An adhoc network allows multiple devices to be connected together and have data shared between them. In order for a feasible wireless data transfer protocol to be implemented, the limitations of smart phones need to be evaluated. These limitations include: limited battery life, mobility issues and wireless data transfer speeds [13]. Optimizing a protocol that takes into account the limitations of smart phones allows users to freely exchange data wirelessly without extra costs.

Due to the increasing costs of mobile internet [12], data transfer to and from smart phones have become restricted. Users only use a small percentage of their smart phone's networking capabilities due to the costs that come along with it. This creates a need for the development of software that can facilitate the exchange of information directly between smart phones. Although this software acts as a tool for smart phone users to share files, it can also serve as a link between people. The aim of this project is to create software that not only allows reliable wireless transfer of information between smart phones, but also has a user interface that supports the interaction between people. As a result, this allows the connecting of smart phones together at a software level but simultaneously connecting people together at a social level. The transfer of information is not limited to only the transfer of media files but also can extend to offering services over the network. Example: A smart phone user might offer file compression services where files are sent to the smart phone and a compressed version is sent back to the user.

Due to the limitations of smart phones such as limited battery life [13], the software needs to be energy aware in its design. This creates a need for an efficient device discovery component where smart phones will be able to know which other phones are in its vicinity. Broadcasting signals will put a lot of strain on the battery life of smart phones and a more efficient method will need to be developed. Thereafter, a reliable and efficient data transfer protocol needs to be developed to aid the exchange of information between the smart phone devices. Once again this protocol needs to take into account the battery limitations of the smart phone and have an energy aware design. The protocol also needs to provide a rapid data transfer speed so users may transfer their files quickly. Lastly, a user interface needs to be

designed that promotes the representation of tiles as people. This transparently allows the software to become the link between people and supports the interaction between them.

The project title “Grout” comes from the paste used for filling up or connecting the gaps between tiles or floors. In particular, the Windows Phone has a unique tiling interface where the use of tiles is prominent [14]. The Grout project will develop software that uses these tiles to represent people and also aim to connect people together. This is done by promoting interaction between the tiles that represent people and having smart phones offer services. The software becomes the link between people and hence the name Grout.

This project was divided into three separate components. Specifically, project partners Sashen Singh and Bryan Davies will be handling the Human Computer Interaction and Device Discovery sections respectively. This paper will focus on the data transfer component of the project.

This paper aims to refine a Bluetooth data transfer protocol that is suitable for smart phones as well as evaluate the protocol under these criteria: Data transfer rate, Power consumption and Latency. Due to hardware and software restrictions, Wi-Fi data transfer will be facilitated using access points rather than being programmed in Android. The success of a data transfer protocol is largely dependent on the rate in which data is transferred from one device to another [4]. Users do not want to wait for files to be transferred and data transfer speeds have increased over the years with technologies such as USB 3.0. Secondly since the smart phone battery is limited in size and therefore also limited in capacity [15], the protocol needs to minimize battery usage. Smart phone users often complain about their limited battery life and software that does not use an energy aware design will make uses reluctant to use it [16]. Latency, the time taken to establish a connection between two devices, will be evaluated to ensure devices take reasonable time (under 4seconds [5]) to connect. Although not as significant as the first two criteria, latency can affect the responsiveness of the system by taking long for data transfer to start.

The two candidates chosen for wireless data transfer are Bluetooth and Wi-Fi. Although the project was originally proposed to be programmed for Windows Phone, their libraries did not fully support Bluetooth or Wi-Fi. This is because Windows Phone is still in development and is new to the market. As a result, Android replaced Windows Phone due to its rich libraries for Bluetooth and Wi-Fi. Several iterative implementations took place to optimize protocols and remove barriers that limited data transfer speeds. Comparisons between the advantages and disadvantages of Wi-Fi and Bluetooth influenced the process of refining the data transfer protocol to determine the optimum solution for a given set of solutions.

Development for Android, like programming for any new language, poses many challenges due to its complexities in setting up such as synchronizing the Android SDK (Software Development Kit) with the Eclipse IDE (Integrated Development Environment). However, Android provides many opportunities to write innovative applications for smart phones due to its rich API libraries. Android was chosen for its rich Bluetooth libraries and together with Android’s multitasking capabilities [17]. These provide the platform for the creation of adhoc networks between smart phones. Furthermore, Android is

programmed in the Java programming language which is a familiar language and together with Android's rich libraries; this allows the data transfer protocol to be programmed with flexibility.

1.1 System Overview

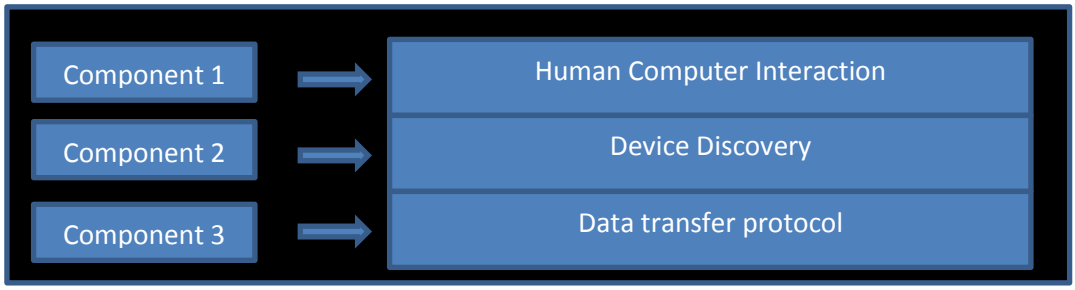


Figure 1.1: Figure showing the different components of the project

The project workload was divided equally into three separate components. The Device Discovery component provides a list of MAC address of all nearby smart phones. The data transfer protocol component thereafter establishes a connection between multiple smart phones in order to exchange data. Both these components require the software to be designed to optimize efficiency in terms of battery usage. Finally, the Human Interaction Component provides a prototype design that is user-friendly in order to use the underlying software.

1.2 Research questions

There are two main research questions that this research project aims to answer.

1. How do Bluetooth and Wi-Fi compare in using network data in terms of data transfer speeds and power consumption?
2. Are connections between multiple devices possible? If so, how does this affect their data transfer efficiencies as the number of devices connected increases?

Firstly, an evaluation needs to be made of the feasibility of using Bluetooth and Wi-Fi for peer-to-peer data transfer despite their limitations on the Android smart phone platform. These limitations include flexibility due to the mobility of smart phones, high power supply demands and data transfer speeds. Thereafter several refinement steps (see Section 1.2) will take place to optimize the data transfer protocol. A choice between the use of Wi-Fi technology or Bluetooth technology will be made after these refinement steps. These refinement steps will focus on the battery limitations of smart phones as well data transfer speeds between devices.

1.3 Refinement steps

There are several factors that determine whether the proposed data transfer protocol will be practical enough to be employed. These factors differ in priority and are given in order of impact by the following table:

Criteria	Impact
Low power consumption	Highly Significant impact
High Data transfer rates	Highly Significant impact
Flexibility and scalability	Significant impact
Latency	In-significant impact

Table 1.1: Table showing data transfer protocol evaluation criteria

Being able to refine a protocol according to the first two criteria above will be the aim of the project since it corresponds to the first research question. A protocol that can both minimize the use of battery power and yet at the same time have high data transfer rates will be comparable to the use of network data [18]. Flexibility and scalability corresponds to the second research question where multiple devices will connect and exchange data. This will test the flexibility of the system and how well it scales as the number of connected devices increase. As mentioned before, latency is not a big factor in the success of the data transfer protocol. This is due to its large dependence on the hardware used to initialize the connection. However, it will still be evaluated and compared and attempts at iterative improvements will be made [19].

1.4 Testing and evaluation

Several graphs will be drawn up for both Bluetooth and Wi-Fi regarding their data transfer speeds and power consumption. In order to evaluate whether it is feasible as an alternative to using network data, the power consumption and data transfer speeds will be compared. During the evaluation process, many tests will be conducted whereby slight changes to the protocol will be made, and their impact on the above criterion will be graphically represented. These changes include:

- Changing of read and write buffer sizes.
- Increase and decrease of file sizes
- Testing different versions of Bluetooth
- Testing for compatibility
- Minor changes in code that may influence data transfer speeds

Once the above experimental data has been collected, it was graphically represented and analyzed to gain an insight into the main factors that affect our criterions.

1.5 Thesis structure

This paper focuses on the Data transfer protocol component shown in Figure 1.1. Chapter 2 outlines current and previous research done in the field of wireless technology. Chapter 3 discusses the design plans for wireless data transfer as well as variations of designs that may influence the project aims. It also provides insight into the development of Bluetooth in Android and programming techniques that allows wireless connections between multiple devices. Furthermore, Chapter 3 dives into the implementation details of Bluetooth and discusses how variations of data transfer protocols may affect the criterions shown in Table 1.1. Each of these variations is tested and experimental data is extracted and presented in Chapter 4. Chapter 5 provides an analysis of the data collected in Chapter 4 in order to refine the data transfer protocol as well as provide an understanding into interesting results. Chapter 6

takes into account the experimental tests and analysis provided to propose a wireless data transfer protocol that is optimized for smart phones. Finally, Chapter 7 presents suggestions for future work for wireless technology.

Chapter 2

2. Background work

The convergence of computing and communications has been inevitable over the past decade [4]. There has been a great deal of growth in both numbers and varieties of new technological gadgets such as laptop computers, personal digital assistants, smart phones digital cameras and so on. Most of these devices come equipped with either wired or wireless networking capabilities. Until recently, facilitating the communication channel between these devices has been "cumbersome" [4] since they often require the use of special wired cables to connect them together. In addition, along with these special cables, each device undoubtedly requires a piece of driver software in order for data to be transferred. This limits the usefulness of these devices.

Bluetooth and Wi-Fi technology allows devices to communicate without the use of wired cables through short-range wireless radio frequency communications [5]. Due to the low costs of Bluetooth chips (5USD)[5], Bluetooth has been predicted to be the preferred solution to wireless adhoc networking.

2.1 Adhoc networking

An ad hoc network is that which allows devices and computers to connect and communicate with each other by being connected to the same network and does not require a base structure [8]. Information is relayed between connected devices in packets through the network and requires no physical connection between devices. Ad hoc networks are used only for the connection and transfer of information between devices and is therefore aptly named, as *ad hoc* in Latin translates directly as '*for this*' [8].

Due to the development of the internet and wireless technology, the idea of an ad hoc network and its subsequent protocols were able to be developed and integrated. Popular peer to peer file sharing sites such as LimeWire and Bit torrent are variants of ad hoc networks as these sites/applications allow users to upload and download files of all types (text, audio, video, et cetera) [8]. In addition to the file sharing capabilities, these sites also allow communication between users by providing a 'chat' facility. This method of both communication and file sharing, when implemented well, becomes highly efficient whilst reducing costs. If one were to consider the instance where data transfer requires the use of an external device such as a USB flash disk, external hard drive et cetera it is easy to see how the process of copying and transferring data would become cumbersome and time consuming. In addition these physical devices would then have to be taken directly to the other device in order to complete the data transfer transaction between two devices.

When considering an ad hoc network no such physical device is required, and multiple devices can share data simultaneously, whilst also being able to communicate with other users. This becomes incredibly useful to users with constrained finances and who also require a need for data transfer. Consider a university student who has the need for data transfer consistently throughout their studying period. Access to and storing of this data would become costly and even in the case of retrieving data from fellow students and friends would become cumbersome. An ad hoc network provides an efficient

solution to this. If the student has access to the same ad hoc network as other students then the student would be able to freely exchange data with other and have an optional chat facility to have discussions.

The use of an ad hoc network already showed the benefits and its popularity in society. An ad hoc network can be implemented readily and quite simply in many different fields that require data transfer. This accessibility and ease of use coupled with the communication tool makes ad hoc networking a rather appealing branch of computer science and its implications on the data transfer world are of utmost importance. There are currently several constraints towards what can be achieved through the use of an ad hoc network. However the research and development of ad hoc networks is still continuing and will be improved and built upon. Specifically, this paper will focus on wireless data transfers between mobile devices that will be used to facilitate the creation of mobile ad hoc networks.

2.2 Smart phones

In recent years, electronic mobile devices have been equipped with functionalities that have PC-like capabilities and as a result are now commonly known as smart phones [30]. Modern smart phones integrate diverse functionalities such as voice communication, web browsing and more. Among these functionalities are rich wireless networking capabilities which, aided by the increasingly more powerful smart phone devices, can establish reliable and coherent wireless connections. Unfortunately, although these devices are capable of establishing robust connections via the use of Bluetooth and Wi-Fi, they are completely dependent on the energy derived from batteries. These are limited in size and therefore capacity [30]. As a result, much research has been done to gain an understanding into where and how energy is used in a smart phone. It has been shown that the wireless communication subsystem accounts for a major component of the total power consumption due to continuous usage [31].

2.2.1 GSM vs. Wi-Fi

Previous works by Carrol [32] suggests that the GSM (global system for mobile communications) module constitutes a significant fraction of the total power usage of a smart phone. Using different energy models and directly measuring the power consumed by individual components, an energy distribution analysis was completed. In particular, the power consumption of Wi-Fi vs. the power consumption of the GSM module was evaluated under two conditions of interest; the power consumption of sending and receiving emails and the power consumption of web browsing. In the first case of sending and receiving emails, the GSM module consumed three times more energy than Wi-Fi. In the latter case of web browsing, the GSM module consumed two and a half times more energy. Despite presenting identical workloads to the wireless components, GSM consistently many times more power than Wi-Fi.

On top of the superior power efficiency, Wi-Fi also provides higher throughput. It was shown that downloading a file using Wi-Fi showed a throughput of approximately 660KB/s which far exceeds the throughput of using the GSM module [32]. However, evaluations done on the power consumption of the two candidates while transferring data over the same time interval showed an even amount of power usage. However, since Wi-Fi dominates in throughput, Wi-Fi completes tasks much faster and therefore uses much less power. This supports the claim that Wi-Fi is a suitable replacement for wireless data transfer due to its efficiencies in power consumption and its data transfer speed. Lastly, it was also shown that Wi-Fi was robust and provided reliable and stable connections.

Wi-Fi is dependent on access points which allow smart phones to have access to the internet. This limits the availability of using Wi-Fi consistently as a replacement to using GSM. Consequently research has been done to allow mobile ad hoc networks to be formed in order to access certain services and files that would otherwise require the use of the internet to obtain. Specifically different systems and protocols have been created to facilitate the construction of ad hoc networks through the use of mobile devices [5], [13], [33]. Recent evaluations have shown that ad hoc networks are proven to be both flexible and robust, and provide good performance in terms of throughput and latency [33].

2.2.2 GSM vs. Bluetooth

Apart from using Wi-Fi technology to facilitate wireless data transfer, almost all smart phones are also equipped with Bluetooth technology. Bluetooth provides secure data transfer over short distances up to 10 meters and focuses on low-power consumption [32]. Its low energy consumption is due to its limited range and simpler radio architecture. In a paper presented by R. Balani [34], the energy consumption of Bluetooth, Wi-Fi and GSM radio was compared. It was shown that after optimizing Bluetooth on a hardware level, Bluetooth consumes power up to 30 times lower compared to GSM while receiving data at a rate of 1200 bytes per second. This is due to GSM having high energy per bit transmission costs and low bandwidth [34]. Therefore, Bluetooth is a solid candidate to facilitate wireless data transfer.

On top of Bluetooth's lower power usage, Bluetooth is also capable of establishing multiple connections between multiple devices [19]. Specifically, in its implementation in Android, different RFCOMM (Radio Frequency Communication) channels are created by assigning different UUIDs to each channel. All communications between two connected devices are enabled through the RFCOMM channel. According to several sources [24], [25], Bluetooth can simultaneously handle up to 7 connections.

2.2.3 Bluetooth + Wi-Fi vs. GSM

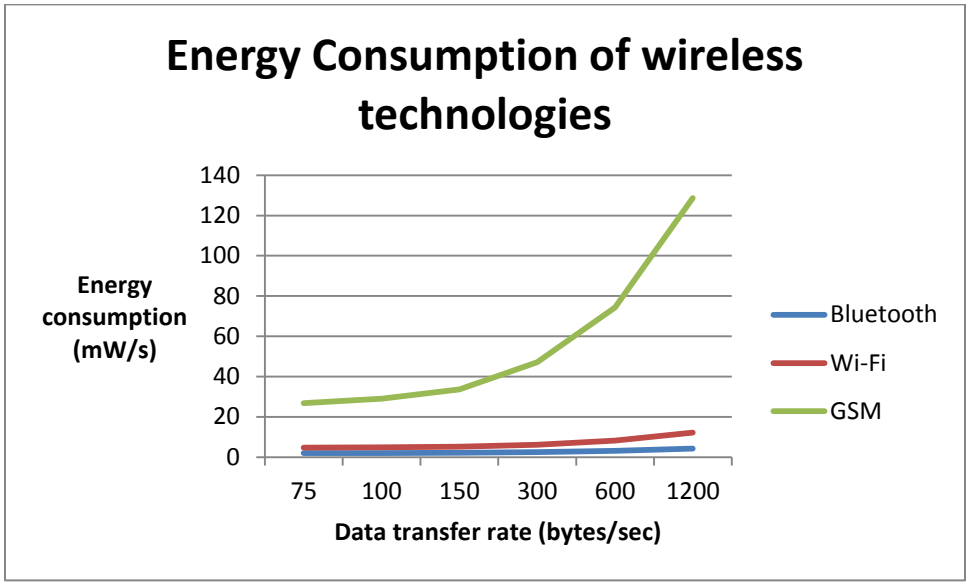
Previous works by Balani [34] aimed to analyze the average energy consumption of Bluetooth, Wi-Fi and cellular networks (GSM) for transmitting data produced at a given rate. Due to the several power saving states of Bluetooth, it was evident that Bluetooth consumed the least amount of power amongst the three candidates. This is due to Bluetooth's "sniff" mode [32, [34], whereby the Bluetooth radio sleeps for a pre-defined amount of time and is automatically activated thereafter when there is any wireless activity. Using this mode, Bluetooth can drop its power consumption to very low levels with an idle connection which still being able to switch to a fully active communication mode [32]. Different sniff intervals were tested and its corresponding power consumption analyzed by R. Balani. Bluetooth at best could transmit or receive data at 1200 bytes per second using only 4.25mW per second.

Wi-Fi has a high wakeup and connection maintenance energy compared to Bluetooth and GSM. However, Wi-Fi has low energy per bit transmission cost and high bandwidth [34]. In particular, it was shown that transmitting data at 1200 bytes per second for less than 15 seconds, Wi-Fi used up to 324.71mW per second. However, at best Wi-Fi transmits data at 12.2mW per second over 1200 seconds. This shows that transfers over time, Wi-Fi still has a much higher power consumption compared to Bluetooth. Furthermore, starting up Wi-Fi uses a significantly large amount of energy especially when compared to transferring data over long periods of time.

Cellular radios have low connection maintenance energy, but high energy per bit transmission cost and low bandwidth [34]. Experiments were conducted to verify these assumptions where data was transmitted using GSM and its power consumption was analyzed. The power consumption increased as the data production rate increased. Transmitting data at 1200 bytes per second, GSM used an average of 128.6mW per second. This is comparatively much higher than both Bluetooth and Wi-Fi.

2.2.4 Which wireless technology is best?

The choice and implementation of an energy efficient wireless technology greatly impacts the usefulness of smart phones due to their limited battery capacity. Sections 2.2.1 – Section 2.2.3 described previous research that analyzed the power consumptions and bandwidths of Bluetooth, Wi-Fi and GSM. These are the two main factors that determine the usability of a particular wireless candidate. In general, Bluetooth uses much less power compared to both Wi-Fi and GSM while transmitting data at 1200 bytes per second and is also more energy efficient in maintaining idle connections. Wi-Fi on the other hand, has high connection maintenance energy but provides smart phones with superior data transfer rate capabilities. The power consumption was compared between the three wireless technologies is graphically represented below:



Graph 2.1: Graph showing the energy consumption of Bluetooth Wi-Fi and GSM obtained by R. Balani. Note: The Wi-Fi energy consumption in the above graph is while transmitting data over 1200 seconds.

As mentioned in Section 2.2.1, GSM is not constrained by the availability of access points or other smart phones in the area to transfer data. However, due to its higher energy consumption and lower bandwidth, much research has been done towards allowing Bluetooth and Wi-Fi facilitate the creation of mobile ad hoc networks using smart phones.

Although Balani [34] claims that Bluetooth is more energy efficient even while transmitting large amounts of data, research done by Pering suggests that Wi-Fi is more energy efficient from a “pure energy/bit standpoint” [32]. A possible cause for the irregularities in research is the difference in

transfer rates used to test Wi-Fi. Pering used a significantly higher data transfer rate to test Wi-Fi which allowed Wi-Fi to use its high bandwidth capabilities to transfer data. On the other hand, Balani used data transfer rates of only up to 1200 bytes which limited the bandwidth usage of Wi-Fi. In short, higher transfer rates for Wi-Fi allows it to compensate for the higher energy consumption rate by using less time to complete the same amount of transfer.

This paper will attempt to address the short comings of the experiments done by Balani by exploring both Bluetooth and Wi-Fi data transfer for both small and large amounts of data. It is undisputed that Bluetooth and Wi-Fi are viable technologies to facilitate data transfer between smart phones. This naturally leads to the creation of mobile ad hoc networks and has been a popular branch of research in Computer Science.

2.3 Mobile ad hoc networks

Recent developments in mobile technologies have opened up many opportunities through increasingly more powerful mobile phones known as smart phones. These phones have become more capable in establishing reliable and coherent wireless connections as well as being able to handle more computational complex tasks. A mobile ad hoc network (MANET) is created using the capabilities of smart phones to facilitate file sharing and to relay information in the network. It is a group of mobile, wireless hosts which cooperatively form a network without relying on an existing infrastructure [35]. In order for a MANET to be functional, nodes are assumed to follow a protocol which allows the relaying of information between two distinct nodes [33].

2.3.1 ORION – Evaluating data transfer

Several algorithms have been implemented to aid the construction of MANETs. In Klemm's paper [37], an Optimized Routing Independent Overlay Network (ORION) protocol was implemented for peer-to-peer file sharing for mobile devices. The aim of the protocol involved providing efficient file transfer speeds in terms of transfer time and overhead, as well as reliable data transfer in terms of the probability of successful file transfers. This is because an important feature for achieving user satisfaction is to have a protocol that reliably and consistently transfer files successfully. In this paper, different file sizes were transferred using the ORION protocol and their probability of a successful transfer was evaluated. The number of devices connected to the MANET was increased and its effects on data transfer speeds were also evaluated. These are the main factors that influence the robustness of a MANET. Evaluation data transfer protocols by varying the data volume and number of devices that are connected provides insight into the feasibility of the transfer protocol. These factors will be tested in Chapter 5.

2.3.2 Minimum power networks

Rodoplu and Meng [18] described a distributed network protocol which was optimized for achieving the minimum energy for randomly deployed ad hoc networks. The protocol uses a position-based algorithm to set up and maintain a minimum energy ad hoc network. This protocol accounts for users that are randomly deployed and are moving around with arbitrary velocities. Furthermore, the protocol reconfigures the links between each mobile device dynamically as they move around while maintaining significant reductions in energy consumption. This promotes the importance of an energy efficient and

flexible suitable wireless technology to be integrated with the system. Although the protocol was not implemented, its need for a wireless technology that can sustain mobile connections is needed. Consequently, Wi-Fi may be the better option compared to Bluetooth as a candidate for the system since it can maintain connections up to 10 times further than Bluetooth [29]. The drawback of using Wi-Fi as mentioned in Section 2.2.4, is its higher energy consumption compared to Bluetooth. These wireless technologies need to be optimized and explored to facilitate the creation of mobile ad hoc networks.

2.3.3 Security

In Zhong's paper [33], the security of mobile ad hoc networks were discussed. The protocol implemented allowed the established ad hoc network to have formal proofs of security. Extensive evaluations and simulations of the system was performed and it was found that the overhead of the system is small compared to the overall power consumption of mobile networking. This shows that the security of a mobile ad hoc network can be implemented without much concern regarding energy efficiencies. The paper also managed to maintain the total throughput of the ad hoc network at an acceptable level. Optimizing data transfer speeds is essential for the success of a system that contains overheads for security.

2.4 Background summary

In this chapter, the success of ad hoc networks was discussed and its usefulness in society was evident [8]. Using the new capabilities of smart phone devices, in particular its increased computational power and wireless technologies, the creation of mobile ad hoc networks is possible. Several research papers were discussed regarding the energy consumption of the different wireless technologies that smart phones are equipped with [30], [31], [32]. In general, Bluetooth is the most efficient in terms of power consumption while Wi-Fi provides much higher bandwidth and further connection maintenance in terms of distance between smart phones. These are viable replacements for GSM radios since they both provide lower power consumption and higher bandwidth. Due to the restrictions in availability of Wi-Fi access points and applicability of using Bluetooth, there is a need for MANETs to facilitate the transfer of information. Several protocols have been implemented in the past [33], [35], [36], [37]. These protocols are designed with the following criteria in mind: Lowering power consumption, increasing throughput, designing for flexibility and robustness. It was evident that lowering the power consumption and increasing throughput whilst maintaining robust connections were the main factors that determined the success of a mobile ad hoc network. This motivates the need for a wireless technology that can facilitate these criteria.

Chapter 3

3. Design

3.1 Design aims

As discussed in Chapter 1, the aim of this project is to refine a wireless data transfer protocol that is optimized for connections between multiple smart phones. Optimizing the protocol includes an energy-aware design that minimizes energy consumption whilst maintaining a high data transfer rate. Because mobile devices are dependent on battery power, most research in energy conservation has targeted wireless networks that are structured around base station and centralized servers [1]. These implementations do not have the limitations associated with smart phone devices. Evaluations on both Bluetooth and Wi-Fi technologies will be explored in order to improve current wireless data transfer protocols. Over and above facilitating data exchange amongst a pair of smart phones, the protocol also needs to account for connections between multiple devices. Hosting multiple connections between smart phones puts limitations on: Bandwidth, computational power and scalability issues. Designing to facilitate multiple connections between multiple smart phones will also be the aim of this project.

3.2 Design constraints

Although the project was originally proposed to take advantage of the Windows Phone's unique tiling interface, its lack of libraries for wireless technologies moved the project development to using the Android development framework. In contrast, Android's rich wireless libraries create opportunities to refine the data transfer protocol which accounts of the limitations of smart phones. Specifically, the designs will target Android version 2.2 (Samsung Galaxy Tab), version 2.3.6 (Samsung Galaxy Ace) and version 4.0.3 (Samsung Galaxy S2). Due to only one of the handsets supporting Wi-Fi Direct, Wi-Fi connections will be tested using access points.

3.3 Design process

3.3.1 System architecture

The implementation of the first prototype for data transfer required thorough knowledge and experience in Android programming. Fortunately, Android applications are written in the Java language (although they do not run on the Java ME virtual machine) [2]. Research was made towards how the Bluetooth library for Android worked and how each of the classes and sub-classes interacted with each other. This was necessary to establish a connection between two smart phones. As shown in Figure 3.1 below, the Bluetooth library consists of the following main classes:

- *BluetoothAdapter:* This represents the local Bluetooth device. Using the getRemoteDevice(String MAC) function, we can obtain the BluetoothDevice we wish to connect to by parsing the MAC address of that device. It is assumed that the MAC addresses will be provided by the Device Discovery component of the project (Figure 1.1).
- *BluetoothDevice:* This represents the remote Bluetooth device the user wishes to connect to and is returned by the function getRemoteDevice().

- *BluetoothSocket:* This represents the socket that is used to send and receive data and is obtained by using the BluetoothDevice class. It is returned by the function *BluetoothDevice*.createRfcommSocketToServiceRecord(UUID uuid). In order to establish a valid connection between two devices, both devices need to create the socket using the same UUID.
- *BluetoothServerSocket:* This represents the server socket that listens for incoming requests from other devices. Once a connection request has been accepted, the Server Socket returns a BluetoothSocket.
- *InputStream/OutputStream:* This represents the streams where the devices can read and write data using the InputStream and OutputStream respectively.

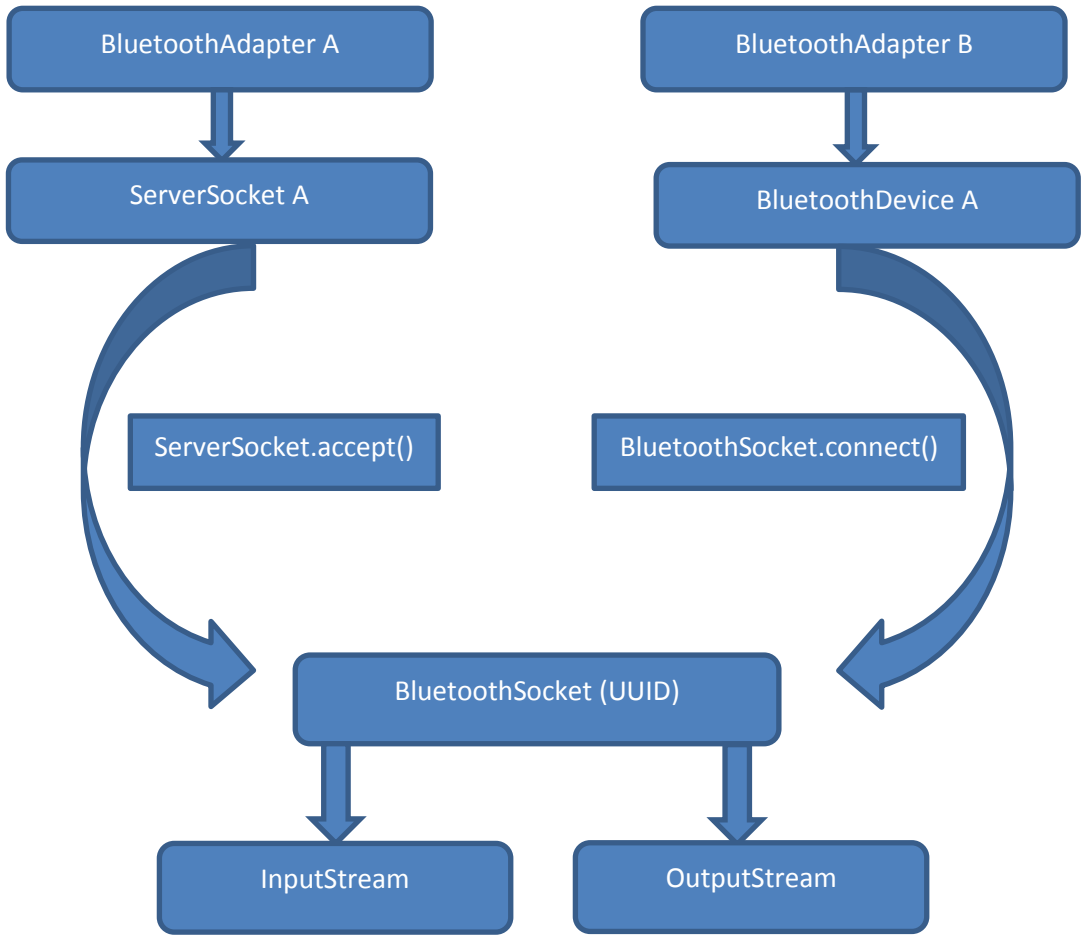


Figure 3.1: Figure showing the class structure of Bluetooth.

3.3.2 Usage of threads

Establishing multiple connections between multiple smart phones require the use of threads to handle multitasking. For testing purposes, each device may start the *AcceptThread* (by clicking on the server button, figure 3.3), where a ServerSocket will be instantiated to listen for incoming connection requests. Here, the AcceptThread acts as a server listening for a client connection request. The ServerSocket is given a unique UUID (Universally Unique Identifier) and when the client attempts to connect to the ServerSocket, it will carry a UUID that needs to match in order for the connection to be accepted [3]. In order to initiate a connection as a client, the device will start the *ConnectThread* (by clicking on the client button, figure 3.3). Once the *AcceptThread* accepts the connection, both devices will start the *ConnectedThread* whereby data exchange using the Input and Output stream is done.

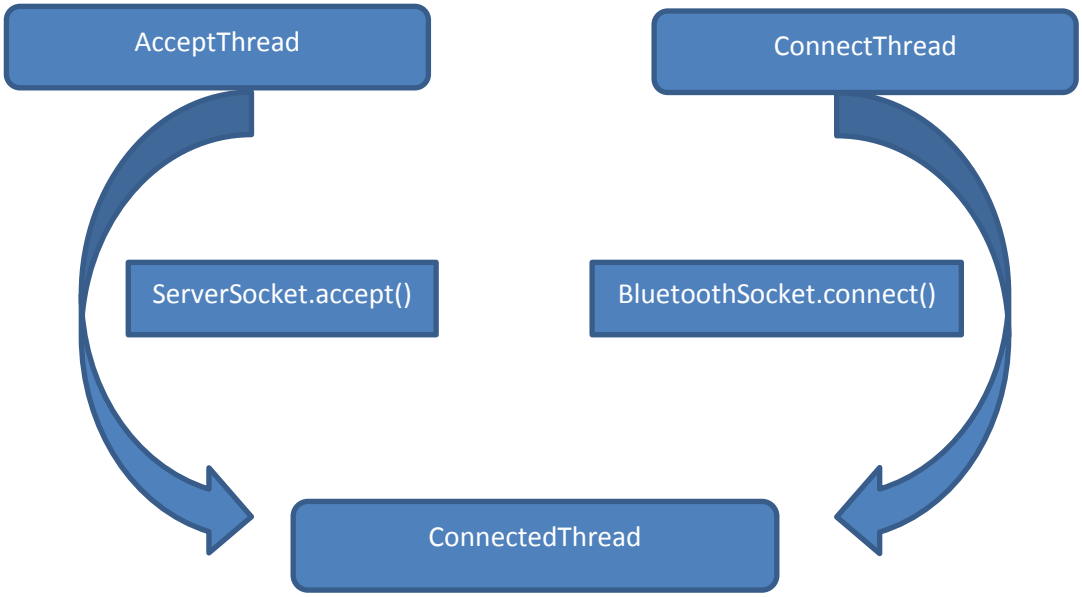


Figure 3.2: Figure showing the usage of threads

3.3.3 System interface

Although developing a system interface was out of the scope of this project, a simple UI design was created to make calls to different methods. Figure 3.3 below is a screenshot of the interface that was used to create both the client and server threads. In particular, the Server button called the AcceptThread which manages the ServerSocket. This thread listens for incoming connection requests from the ConnectThread which is called when the Client button is pressed. Thereafter a connection will be established between the server and the client and the server can then continue to listen for more connection requests. Each device may act as the server or the client. The text field was used for debugging purposes and the Bluetooth and Wireless button called variations of coding that did not implement threads.

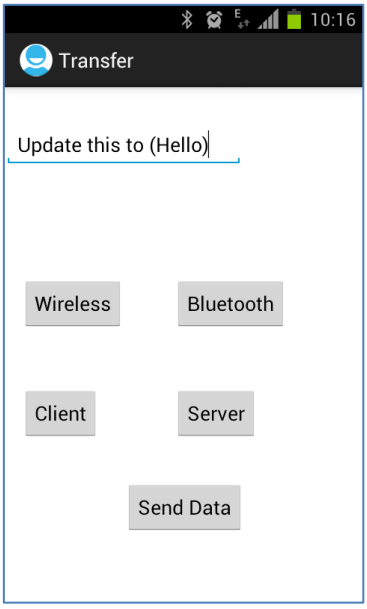


Figure 3.3: Figure showing the user interface

3.3.4 Hardware and Software

Three different Android devices were used to test wireless connections. Their hardware and software details are given by the table below:

Device name	Bluetooth version	Wi-Fi	Wi-Fi Direct	Android version
Samsung Galaxy Tab	V3.0 with A2DP	Wi-Fi 802.11	Not supported	2.2
Samsung Galaxy Ace	V2.1	Wi-Fi 802.11	Not supported	2.3.6
Samsung Galaxy S2	V3.0 + HS	Wi-Fi 802.11	Supported	4.0.3

Table 3.1: Table showing hardware and software details of the Android handsets

3.3.5 Permissions

In order for an Android application to access certain hardware and software components, the permissions for each of them are required to be stated in the Manifest file. Bluetooth, Wi-Fi and several other networking permissions were needed to allow connections to be established via Bluetooth. A list of permissions that was enabled for the application is listed below:

1. <uses-permission android:name="*android.permission.ACCESS_NETWORK_STATE*"/>
2. <uses-permission android:name="*android.permission.ACCESS_WIFI_STATE*" />
3. <uses-permission android:name="*android.permission.CHANGE_WIFI_STATE*" />
4. <uses-permission android:name="*android.permission.BLUETOOTH*" />
5. <uses-permission android:name="*android.permission.BLUETOOTH_ADMIN*" />
6. <uses-permission android:name="*android.permission.READ_PHONE_STATE*" />
7. <uses-permission android:name="*android.permission.WRITE_SYNC_SETTINGS*" />
8. <uses-permission android:name="*android.permission.WRITE_EXTERNAL_STORAGE*" />

Permissions 1-5 are needed to access the Bluetooth and Wi-Fi hardware and software components. Permissions 6-8 are needed to access external storage and to obtain write permissions to external storage. This is required for the application to store experimental data.

3.4 Data transfer

Once a connection has been established between two smart phones, to transfer data, the sender will write to the BluetoothSocket via the OutputStream and the receiver will read from the BluetoothSocket via the InputStream (see Figure 3.1). Conventionally, files will be broken up into byte blocks of a specified size given by the programmer. These bytes blocks are then sent across one at a time and the receiver will concatenate the blocks back together to form the original file. The sender also needs to specify when the last block has been sent so the receiver knows when to stop reading more blocks of data. This will be discussed in more detail in Chapter 4. For testing purposes, only blocks of data will be sent across instead of actual files. This process is shown below in Figure 3.4.

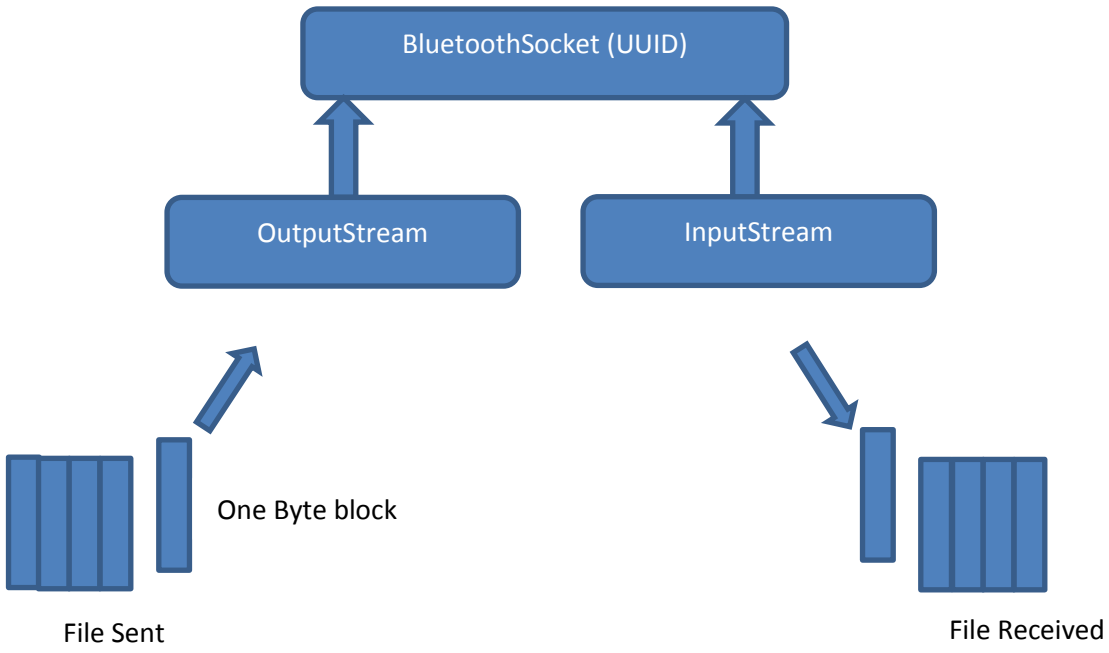


Figure 3.4: Figure showing the sending and receiving of files.

3.5 Bluetooth UUID

A UUID is an immutable representation of a 128-bit universally unique identifier [20]. They are used to uniquely define certain hardware, windows registry and are used in databases [21]. Each connection requires a UUID input from both the devices for the connection to be valid. The UUIDs were surprisingly difficult to generate. The following UUIDs were used to test out program. (Special acknowledgements to Pierre Benz – Master's student for providing me with valid UUIDs)

UUID1	**a60f35f0-b93a-11de-8a39-08002009c666**
UUID2	00001101-0000-1000-8000-00805F9B34FB

Table 3.2: Table showing the different UUIDs used.

3.6 Wi-Fi and Wi-Fi Direct

Wi-Fi technology was the second candidate that was considered for wireless data transfer. However, the software limitation of Android for versions prior to version 4.0 did not fully support data transfer between devices via Wi-Fi. The only Android device that supported Wi-Fi Direct was Samsung Galaxy S2 (from table 3.1). Consequently, all data transfers between Android devices using Wi-Fi will require setting up an access point. Thereafter, data will be transferred across using the access point and its efficiencies analyzed. There will be no refinement steps for the Wi-Fi data transfer.

3.7 Design Summary

This chapter described the design process of the software to be implemented in Android. A simple UI was developed to aid the execution of the application. The usage of threads was essential in our design to support multiple connections between multiple devices as each device needed to listen for incoming connection requests. Certain permissions were required for the application to access the hardware and software components of the Android devices. Lastly data was broken up into a specified size of byte blocks and sent across using the OutputStream. The receiver can then read the data from the InputStream and piece the blocks together to obtain the original file. Although the data transfer efficiencies of Wi-Fi will be evaluated, due to software and hardware constraints this will be done by setting up an access point for the Android devices to transfer data to.

Chapter 4

4. Implementation

The proposed design presented in chapter 3 was implemented for Bluetooth as an Android application. To develop using the Android SDK, the Android SDK together with the Eclipse IDE was downloaded and installed. Thereafter the ADT (Android Development Tools) custom plugin was downloaded and integrated the SDK and IDE together. Eclipse created an integrated environment to program in and the ADT functionalities facilitated the deployment of the software onto the Android devices. This chapter will explain the class structure of the program, different algorithms used for data transfer via Bluetooth and methods of collection for the experimental data.

4.1 Class structure

The Bluetooth program has four main classes and each of the classes is shown in Figure 4.1 below:

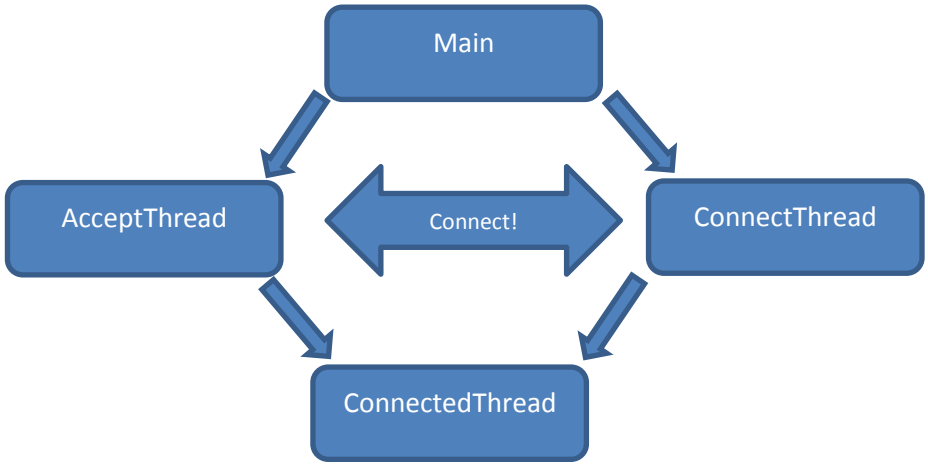


Figure 4.1: Figure showing the different classes of the Bluetooth program

The Main class is the starting point of the program and is responsible for the user interface and execution of the different threads shown in Figure 4.1. The server button starts the AcceptThread and the client buttons starts the ConnectThread. Once the AcceptThread accepts a connection from a ConnectThread, a connection is established and they each start a ConnectedThread. Once the ConnectedThread has been initialized, they can thereafter use the BluetoothSocket (attribute of the ConnectedThread) to transfer data across. Although the above diagram only illustrates a connection between two devices, each device may start multiple AcceptThreads to listen for multiple incoming connection requests. Similarly, once a ConnectedThread has been initialized, the same device may also start multiple ConnectThreads to simultaneously connect to other devices. Each thread is handled appropriately to exit once a connection is no longer needed. A more detailed description of each of the classes is given below:

4.1.1 Main.java:

This class contains all the methods that direct the program to the threads that will be executed. Firstly, the program will call the onCreate() function where the user interface is loaded. Secondly, the makeDiscover() method will use the BluetoothAdapter (Figure 3.1) to enable discoverability of the Android device as well as turn on the Bluetooth functionality if it is currently off. This allows other devices to be able to pick up the device whilst they searching for Bluetooth devices in the area. In order to access this functionality, special permissions need to be enabled (Section 3.3.5 Permissions 4 and 5). Thereafter, each of the functionalities of the buttons are loaded onto the interface and thus linking each button to a specific thread when a button is clicked. Each button is listed below (refer to Figure 3.3 for the names of the button):

Test1/Test2: These two buttons are used for testing purposes. They contain variations of code that is used for experimental results and also for refining the final protocol.

Server: This starts the AcceptThread. A BluetoothServerSocket is initialized and the device starts to listen for incoming connection requests. Once this thread has accepted a valid connection, it will proceed to create a ConnectedThread.

Client: This starts the ConnectThread. A BluetoothSocket is initialized and the device attempts to establish a connection with the server. For testing purposes, it is assumed that the ConnectThread has both the MAC address of the corresponding BluetoothServerSocket as well as the unique UUID that is required to validate the connection. Once the AcceptThread has accepted the connection request, it will proceed to create a ConnectedThread.

4.1.2 AcceptThread.java

This class acts as the server between the devices. It initializes a BluetoothServerSocket by calling the following function:

$$BluetoothServerSocket\ ServerSocket = adapter.listenUsingRfcommWithServiceRecord(\text{Test}, uui);$$

The ServerSocket then makes a blocking call [3] in an attempt to accept a connection request. The devices requesting the connection is required to have the same UUID as the one used to initialize the ServerSocket above. Once the ServerSocket has validated the UUID, it returns a BluetoothSocket where data can be read from or written to. Conventionally, the ServerSocket is then closed. For testing purposes, another AcceptThread can be started with a different UUID in order to handle multiple connections. Finally, the thread parses the BluetoothSocket to the ConnectedThread.

4.1.3 ConnectThread.java

This class acts as the client between the devices. It initializes a BluetoothSocket by firstly creating a BluetoothDevice by using the MAC address of server:

$$BluetoothDevice\ device = adapter.getRemoteDevice(\text{"78:47:1}D\text{:}A\text{5:}CD\text{:89"});$$

Thereafter, a BluetoothSocket is initialized by calling the following function,

$$BluetoothSocket\ socket = device.createRfcommSocketToServiceRecord(uuid);$$

Where the UUID needs to match the ServerSocket's mentioned in Section 4.1.2. Once the server has accepted the connection request, the BluetoothSocket is parsed to the ConnectedThread. In addition to the BluetoothSocket being parsed to the ConnectedThread, a Boolean parameter was also given to the ConnectedThread. This defined (for testing purposes) which device was sending data and which device was receiving.

4.1.4 ConnectedThread

This class handles both the sending and the receiving of data. Using the BluetoothSocket parsed by the preceding threads, both the InputStream and OutputStream can be instantiated (see figure 3.1):

$$InputStream\ instream = socket.getInputStream();$$
$$OutputStream\ outstream = socket.getOutputStream();$$

Once the ConnectedThread has obtained the InputStream and OutputStream, data can be exchanged between the two devices by writing to the OutputStream and reading the data from the InputStream. A separate thread was created within ConnectedThread in order to read from the InputStream. The reasoning behind creating a new thread is because the read function is a blocking call. If this was not executed in a thread, the program would have to wait until data was read in order to execute other parts of the class.

4.2 Evaluating execution time

To evaluate the data transfer algorithms in Section 4.3, an accurate timing method needed to be employed to access their time efficiencies. Two possible candidates were considered where one was an external program and the other a built in Java function. The latter candidate was chosen since external programs might produce overhead and unwanted time uncertainties. Although the built in Java function allowed timing to be done in many different scales, i.e. Milliseconds, nanoseconds et cetera, the nanosecond timing tool was used. Timing of specific parts of the program is shown below:

$$long\ startTime = System.nanoTime();$$

This provides the system's time accurate to the nanosecond. Thereafter, the piece of code that needed to be timed was appended onto the above code. In order to evaluate how the piece of code took to execute, we calculate the difference between the system's time after the piece of code has been executed and the startTime defined above. Thus we append this line of code:

$$long\ endTime = System.nanoTime();$$

$$long\ duration = endTime - startTime;$$

Although there may be certain overhead with calling the system function, it is shown by the table below that this overhead (in nanoseconds) is negligible for evaluating data transfer. The overhead was

calculated by calculating the duration between startTime and endTime with no code in between. Theoretically, this value will vary between different operating systems [7].

Trial number	Overhead
Trial 1	428ns
Trial 2	427ns
Trial 3	427ns
Trial 4	428ns

Table 4.1: Table showing the overhead of evaluating time execution.

4.3 Data transfer algorithms

In this section, different data transfer and data validation algorithms will be discussed and their efficiency analyzed.

In order to write data to the OutputStream, all data is required to be converted into bytes. As mentioned in Section 3.4, each file will be split up into many byte blocks (depending on the size of the file and the block size) and each block will be sent sequentially. For test purposes, no actual files will be sent. Instead, a specified number of bytes will be used to simulate a file sent across to evaluate the efficiency of the data transfer algorithm. This is done to reduce the overhead that may be caused by the complexity of the files which may affect the total execution time. Furthermore, it is much easier and faster to adjust the number of byte blocks to match the specified "file size" instead of creating files with the specified file sizes. The code for creating one byte block is given below:

$$byte[]\ byteString\ =\ block.getBytes();$$

Here, block is an N bit String. The function getBytes() maps the N bit String into a byte array of size N. For example: the String "HelloWorld" would be mapped into a byte array of size 10 where byteString[0] would be the byte representation of "H".

4.3.1 Varying block size

It was interesting to evaluate the total execution time of writing a fixed number of total bytes to the OutputStream whilst varying the number of bytes (per block) that was sequentially written. This is a tradeoff between the time required to send a lower number of larger blocks and the time required to send a higher number of smaller blocks. Block sizes of powers of two will be used to test due to the binary nature of computers. This process is shown in the figure below:

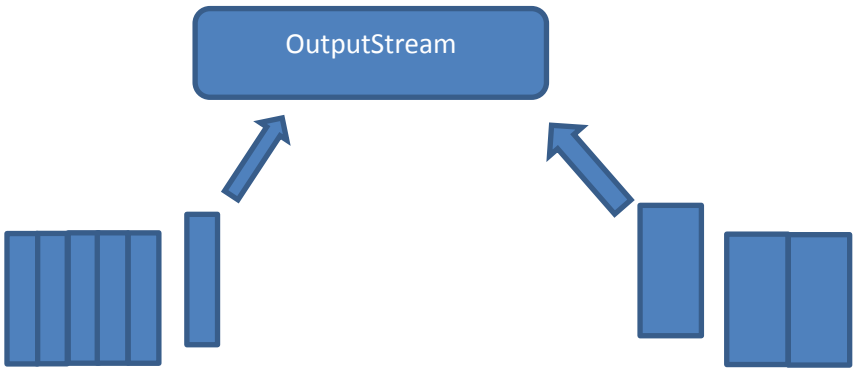


Figure 4.2: Figure showing different block sizes sent to OutputStream

4.3.2 Varying read buffer size

Once the data has been written to the OutputStream, the receiver can then retrieve it from the InputStream. Since the InputStream depends on the data that is written to the corresponding OutputStream from the sender, varying the amount of data that is read at a time from the InputStream may affect the time efficiency. In order to read from the InputStream, an initialization of a byte array which shall be referred to as the "buffer" is required. Varying the size of the buffer allows the receiver to retrieve data of different size blocks. The initialization of the buffer is given below:

$$byte[]\ buffer = new\ byte[bufferSize];$$

Different values of buffer sizes will be tested to analyze their time efficiencies. Once again, buffer sizes of powers of 2 will be used. This process is shown by the figure below:

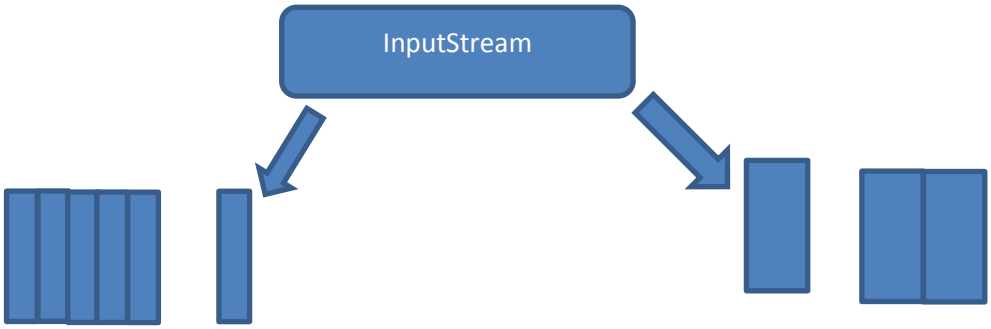


Figure 4.3: Figure showing the different buffer sizes used to read from the InputStream

4.3.3 End of file checking

Since the read function for the InputStream is a blocking call, it is important for the file receiver to be notified when the end of the file has been reached or else the receiver will continue to attempt to read from an empty InputStream. This is a potentially create a bottleneck for data exchange if an appropriate algorithm is not employed.

In this sub-section, three separate data transfer algorithms were presented and theoretically evaluated. These algorithms were named the "terminating block technique", "last byte check" and "file size block" respectively. Each algorithm proved to have their advantages and disadvantages in terms of

computational power, incomplete downloads, corrupt files and total execution time. The algorithms were also evaluated from the two different perspectives of the data sender and the data receiver. They will be evaluated in detail in Chapter 6.

4.3.3.1 Terminating block technique

Using the algorithm given by Meler, R. (2010) [2], the first proposed algorithm that was implemented uses a "terminating" block technique. The sender will append a terminating block of data that signals the end of the file. This block is predefined and is checked by the receiver and reading from the InputStream is halted. The terminating block technique is given by the figure below:

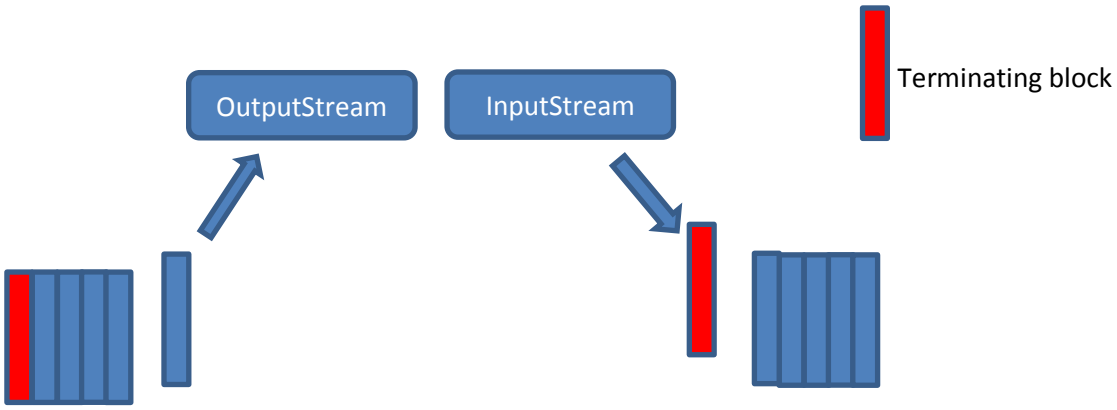


Figure 4.4: Figure showing the terminating block technique

The terminating block technique may not look particularly promising at first since if the sender is sending a large file and one of the blocks correspond to the terminating block, the receiver may stop reading any further data half-way through the file. This can cause corruption of files or incomplete files being sent. However, if the block size for the terminating block is large enough, the probability of a block matching the terminating block is extremely small. In particular, if the block size was 1024, there would be 2^{1024} different possible blocks of data.

Although this technique is simple and easy to implement, it is not particularly efficient. The receiver needs to compare every block of data that is read from the InputStream to the terminating block in order to determine whether another block needs to be read. This will not only increase the total execution time of the data transfer but also use more computational power. This is due to the N comparisons that are required to match two blocks of size N.

4.3.3.2 Last byte checking

Due to the simplicity of the terminating block technique, refinements efforts were made towards improving it's time efficiency and lowering the amount of computational power needed. This produced the last bit checking technique where the sender would instead only assign data the first N-1 bytes in the byte block of size N. I.e. Data would only fill the first 1023 bytes of a 1024 byte block. The last byte would be used to "verify" whether the current data block is a terminating block. For testing purposes,

the last byte would be assigned 1 if the current block of data was also a terminating block. The last byte would be assigned 0 otherwise.

This method is more efficient and uses less computational power for the receiver. The reasoning behind this assumption is that the receiver now only needs to check one byte of the byte block to verify whether the block is a terminating block. It also eliminates the possibility that another block could be a terminating block and thus corrupting the file or cause an incomplete transfer.

On the other hand, this algorithm places extra overhead on the sender since the sender is required to manually append an extra byte onto each data block. This may cause a bottleneck on the preparation of data before it has even been written to the OutputStream. Furthermore this also introduces a complexity into the concatenation of files on the receiver end. The receiver is required to remove the last byte of the data block before the concatenations of each data block can take place.

4.3.3.3 File size block

After analyzing the above algorithms, it can be seen that a lot of computation power is wasted in iterations of read and write. This is due to the usage of a terminating block where the sender needs to check whether a given data block is the last block to read. However, an alternative to having a terminating block at the end of the input is to have a block of data that specifies how much data needs to be read at the beginning of the input. The sender appends a file size block of data to the beginning of the file. The receiver initially only reads one block of data and can thereafter determine how much data needs to be read from the InputStream until the end of file has been reached. This prevents both the issue of incomplete data transfer as well as extra overhead that may lower the efficiency of the data transfer algorithm. This process is shown in the figure below:

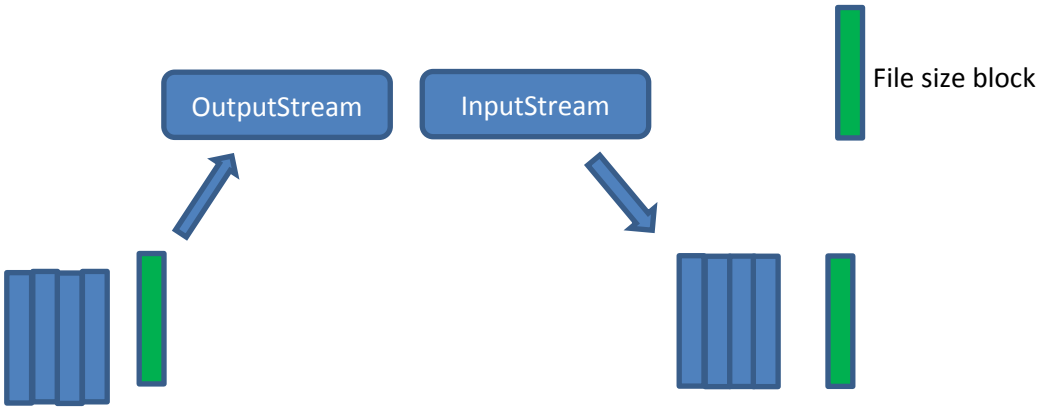


Figure 4.5: Figure showing the data transfer by appending the file size block.

Although this method appears to be the best candidate for data transfer, it is also limited to the sender knowing the size of the file that is being sent. Furthermore, if even one byte of the original file was lost due to interference, the entire file may have to be sent again.

4.4 Recording experimental data

Since the Android applications are run on the actual handsets, any experimental data that is logged needs to be written to external storage on the actual device. This requires us to enable permissions for the application to write to external storage as shown in Section 3.3.5. Experimental data is written to text files on the SD card of the Android devices and thereafter transferred to a Desktop computer for it to be graphically represented. The choice of text files was used since it would be too slow to record all the experimental data by hand.

4.4.1 Writing to external storage

Due to permission issues and security reasons, the application may only create and modify files that belong to its application folder. The following code retrieves the application directory:

$$File\ file = Environment.getExternalStorageDirectory();$$

Once the directory for the application is obtained, reading from and writing to text files onto the SD card is done through the InputStream and OutputStream.

4.4.2 Power consumption

Due to the limitations of Android's battery manager, the power consumption of the data transfer can only be approximated. The following method retrieves the level of the battery which is thereafter converted to a percentage of remaining battery life:

$$\text{level} = \text{intent.getIntExtra}(\text{BatteryManager.}EXTRA\_LEVEL, -1);$$

This is the most accurate reading of the battery that can be obtained using the battery manager. Firstly, a reading is taken before the execution of the program. Thereafter, a second reading is taken after the execution of the program and the total power consumption is calculated. This reading provides an integer percentage of the remaining battery life. Although this does not give a very accurate reading of the power consumption, it gives an idea of how much power the data transfer used.

4.5 Software issues

For each deployment of the application onto an Android device, the application was tested by starting the server and then having a client connect to the server. If the connection was not established correctly (due to an incorrect UUID or an incorrect MAC address) the BluetoothServerSocket would not be closed properly. This disallows any further connections to be created using the same UUID unless the Android device was restarted.

4.6 Implementation overview

The designs that were mentioned in Chapter 3 were implemented and each of the classes is described. The usage of threads played an important role in creating multiple connections between multiple devices and thus all classes except the main class extended the Thread class. The timing tool used to measure the total execution time of certain parts of the program proved to be accurate for data transfer testing purposes. Different data transfer algorithms were discussed. In particular, the "terminating block technique, "last byte check" and "file size block" algorithms were discussed and evaluated theoretically.

Each of these algorithms will be tested and experimental results for each of them will be obtained in Chapter 5. The method used to extract the experimental results when the application was run on Android devices was discussed. Lastly, some main hardware issues with Bluetooth were discussed.

Chapter 5

5. Experiment Design and Testing

The purpose of conducting these experiments is to evaluate wireless data transfer by using the factors in table 1.2 for smart phone users to exchange files. A comparison will be made between the efficiencies of wireless data transfer protocols and using mobile internet. Several of the main factors are discussed in the sections below. Thereafter, experimental results are used to find bottlenecks that could be possibly removed or improved upon. By using data gathered from the experimental results, a better knowledge towards the energy consumption and the data transfer speed can be achieved. This allows iterative improvements to be made towards the final data transfer protocol. Furthermore, experimental data also helps isolating issues by testing specific components before testing the whole integrated software.

5.1 Acquiring consistent results

5.1.1 Technical issues

To get consistent and accurate results from the experiments, many technical factors that could influence the results of the experiment were handled accordingly. Firstly, a constant distance between two devices that exchange data needs to be maintained as changing the distance between then devices will affect the data transfer speed [18]. This is done by measuring and marking off two positions where the first and second device will be put before any data transfer begins. Secondly, all background processes that may be running needs to be closed. This is done by clearing RAM memory in the task manager on the Android devices. Thirdly, all devices were restarted prior to the beginning of all experiments. Lastly, the battery life of all the devices is maintained at a high level to ensure the results were not affected by lower battery levels.

5.1.2 Outliers

Once the technical factors have been eliminated, several methods were employed to ensure the accuracy of the data obtained from the experiments. These methods included repeating the same experiment three times and finding the average value of the three experiments rather than only taking the first value. This aids in eliminating outliers that may have occurred in the first test. If any of the three tests provided results that was unexpected, the test would be repeated a fourth time. For example: If the Bluetooth transfer protocol transferred at 120,123,122 kilobytes per second respectively per experiment, the average of the three results was calculated and record. If the data acquired was 120, 157, 121 kilobytes per second, the experiment would be repeated a fourth time since 157 appears to be an outlier for the experiment. Any experimental value obtained that deviated more than 20% from the mean will be considered an outlier [22].

For each experiment that was conducted, it is possible that there were software implementation errors and consequently incorrect results will be obtained. In order to determine any implementation errors, a hypothesis was made before the start of every experiment. If results did not correspond to the hypothesis, the implementation details were analyzed to determine if any errors occurred. For example in Section 5.3.1, an experiment was conducted to test the different data transfer speeds for Bluetooth

V2.1 and Bluetooth V3.0. Since we expect the newer technology of Bluetooth V3.0 to yield better results, the implementation details for the data transfer would be analyzed if Bluetooth V2.1 gave higher data transfer speeds during the experiment. If no implementation errors were found, the experimental results would be analyzed to determine the cause of the unexpected results.

5.1.3 Power consumption

Although all background processes have been closed as mentioned in Section 5.1.1, there are many system processes that cannot be closed. These processes may affect the total power consumption during the experiment. The power consumption is evaluated from the start of the data transfer to the end of the data transfer. Since the total power consumption recorded includes the power used from system processes, these can be removed by subtracting the power usage when no data transfer is taking place from the results.

5.1.4 Time evaluation

The time evaluation techniques used for the experiments were discussed in Section 4.2. These values are correct to 7 decimal places since the values are given in nanoseconds. The experimental results recorded were rounded off to 3 decimal places to maintain accurate results without having to store unnecessary information.

5.1.5 Devices used

The devices used for the experiments are given in Table 3.1 Section 3.3.4. In particular, Bluetooth V3.0 testing was done by transferring data across from the Samsung Galaxy Tab to the Samsung Galaxy S3. Bluetooth V2.1 testing was done by transferring data across from the Samsung Galaxy Ace to the Samsung Galaxy Tab. Although the Galaxy Tab does not have Bluetooth V2.1, the data transfer will be capped at the lower speeds of V2.1.

5.2 Main experimental data

5.2.1 Data transfer speed

The success of a data transfer protocol is largely dependent on the rate at which data is transferred from one device to another. This is measured by finding the ratio of the total time taken to transfer the data, to the total amount of data transferred. This factor may be greatly influenced by the implementation of the data transfer algorithms and different read and write buffer sizes. The total time taken is measured employing methods that were described in Section 4.2.

5.2.2 Power consumption

Due to the limited battery life of a smart phone, the data transfer protocol can only be successful if it does not use too much power. The power consumption is measured using the technique described in Section 4.4.2. Although this is not a very accurate reading since this technique gives an integer amount of battery life that is left and it will give an idea of how much power the data transfer uses.

5.2.3 Latency

Latency is the total amount of time taken for the server and client to establish a connection once the client requests a connection until the server accepts the connection. Latency is mostly affected by the

hardware implementation of Bluetooth and not the software implementation. Consequently, latency does not largely affect the efficiency of the final data transfer protocol and no efforts were made to improve on the results obtained from the experiments. It was however interesting to evaluate the latency once multiple devices started establishing multiple connections.

5.2.4 Robustness

Bluetooth technology is known for its ability to establish stable and reliable connections [24]. For each experiment conducted, the total number of bytes sent and received was recorded to determine if any data was lost during the data transfer. The system were furthermore tested using exceptional inputs such as large data blocks to determine how the system behaves under such stressful environmental conditions [27].

5.2.5 Graphical representation

Most data acquired in the experiments were represented graphically in the form of line graphs. The goal is to make data presentation interesting and bring forward any interesting relationships between data that might've been overlooked in table form. Data visualization also provides an easier way for the reader to be informed of the results that were acquired from the data. Furthermore, good visualization techniques will not only help the reader, but also help the producer of the visualization to discover meaningful insights [26]. Most graphs had a logarithmic scale on the horizontal axis due to the choice of independent variables being powers of 2.

5.2.6 Independent and dependent variables

In order to investigate the feasibility of Bluetooth and Wi-Fi, different independent and dependent variables were used. In most cases, the independent variable that was used was the file size sent across. Different implementations of data transfer were tested for different file sizes. The dependent variable is the amount of time taken to complete the data transfer. Consequently this also makes the data transfer speed a dependent variable since it depends on the amount of time taken for the transfer. For evaluation of power consumptions, the file size was once again the independent variable with the power consumption percentage being the dependent variable.

5.3 Bluetooth testing

Several Bluetooth experiments took place to refine a final Bluetooth data transfer protocol. The first test involves testing the data transfer speeds for both Bluetooth version 2.1 and Bluetooth version 3.0. The experiment will attempt to confirm the assumption that Bluetooth 3.0 is faster than version 2.1. The next two tests involve changing the buffer's read size and the buffer's write size. Changing these values will affect data transfer speed and the optimum solution will be taken forward to be used. Using the optimum solution thus far, the three data transfer algorithms discussed in Section 4.3.3 will be experimentally evaluated. The three algorithms will be evaluated according to data transfer speeds as well as power consumption. Thereafter the best algorithm will be used to test connections between multiple devices. The latency of the final data transfer protocol will also be evaluated.

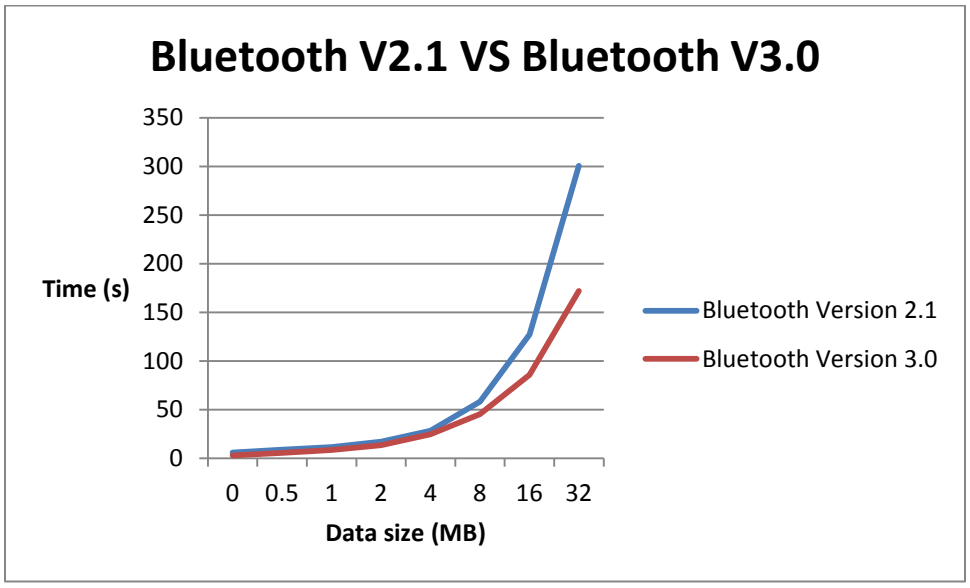
5.3.1 Bluetooth V2.1 VS Bluetooth V3.0

It is hypothesized that Bluetooth V3.0 would have a considerably higher data transfer rate compared to Bluetooth V2.1. Bluetooth V2.1 has a theoretical maximum throughput (data transfer speed) of 2.1Mbps and Bluetooth V3.0+HS has a maximum of 24Mbps [24]. Although one device (Samsung Galaxy S2) has Bluetooth V3.0+HS, the transfer speed will be capped at the lower speeds of Bluetooth V3.0 with A2DP (Samsung Galaxy Tab). In this experiment, different blocks of data were sent across using the two Bluetooth technologies. Firstly, experiments were done to evaluate how long to it takes to transfer 0MB of information across. This gives us an indication of the amount of overhead that takes place in order to set up the transfer. Thereafter, data blocks starting at 0.5MB were sent across and the data block size was doubled in iteration. The maximum block size chosen for this experiment was 32MB as it became very time consuming to test data transfers with higher data block sizes. Furthermore, Bluetooth data transfers often involve data sizes lower than 32MB [25]. The amount of time it took to send the block sizes were recorded in the table below:

Data size (MB)	Bluetooth Version 2.1	Bluetooth Version 3.0
0.0	5.607s	2.705s
0.5	8.617s	5.286s
1.0	11.318s	8.276s
2.0	17.093s	13.397s
4.0	28.299s	24.562s
8.0	58.015s	45.387s
16.0	127.140s	85.764s
32.0	300.547s	171.797s

Table 5.1: Table showing the experimental data gathered for different Bluetooth versions.

As shown below in Graph 5.1, Bluetooth V2.1 takes substantially longer to transfer data across as data block sizes get large.



Graph 5.1: Graph showing the time taken to transfer data using Bluetooth V2.1 and Bluetooth V3.0.

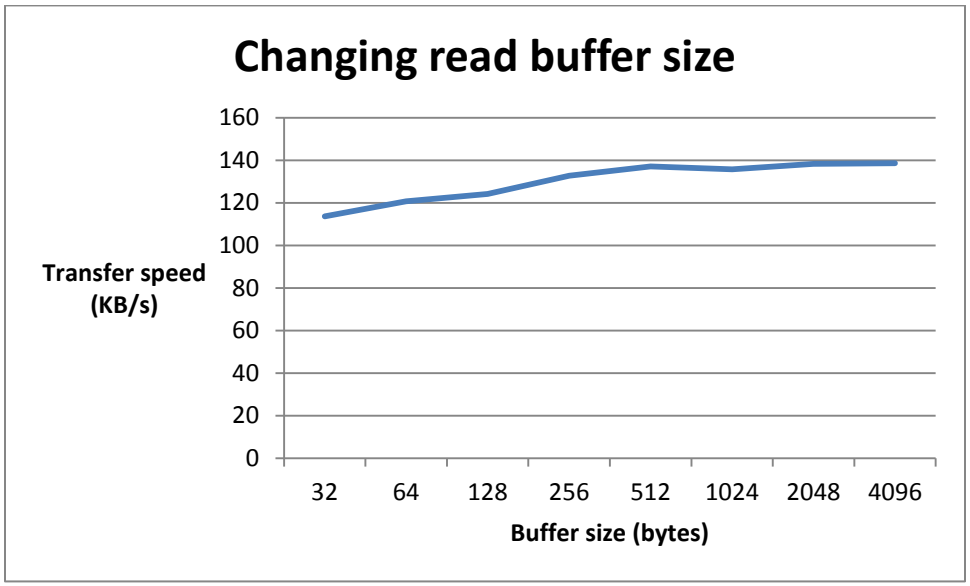
5.3.2 Changing buffer read size

For this experiment, Bluetooth version 3.0 was used and 4 megabytes of data was transferred across. The buffer write size was a constant 1024 bytes. Different read buffer sizes were tested and their transfer speeds recorded. The buffer size was increased by doubling the previous buffer size until no noticeable changes to the transfer speed were recorded. The chosen data block size was 4 megabytes since it is not affected drastically by the initial overhead and is completed in a reasonable amount of time.

Read Buffer size (bytes)	Time (seconds)	Transfer speed (KB/s)
32	36.921	113.601
64	34.744	120.719
128	33.781	124.162
256	31.601	132.726
512	30.580	137.156
1024	30.917	135.659
2048	30.325	138.309
4096	30.278	138.524

Table 5.2: Table showing the transfer speeds after changing the buffer read size.

Changing the read buffer size does not significantly affect the transfer speed. As seen in the graph below, increasing the read buffer after 512 bytes did not yield a substantial increase in data transfer speeds:



Graph 5.2: Graph showing the transfer speeds with varying read buffer sizes.

5.3.3 Changing buffer write size

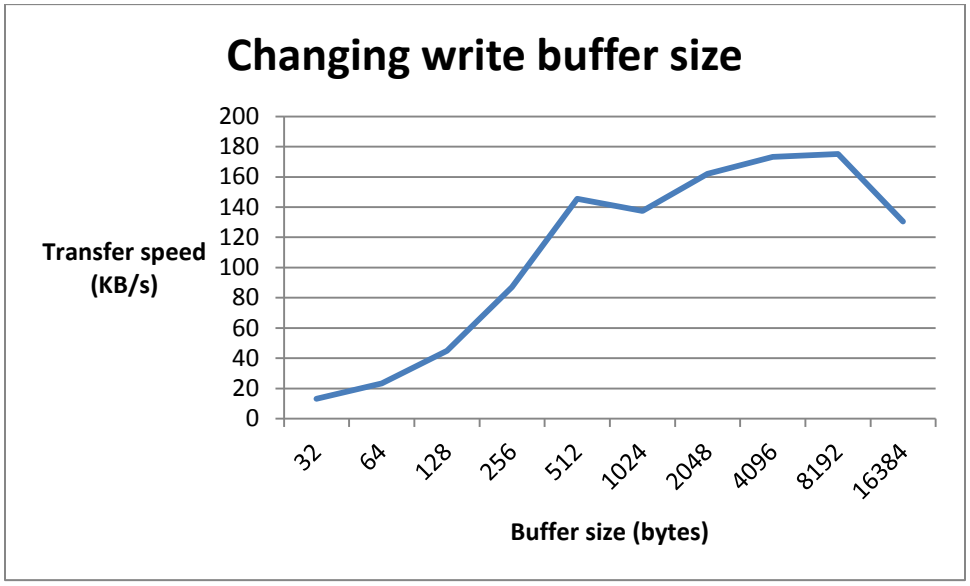
For this experiment, once again Bluetooth version 3.0 was used and 4 megabytes of data was transferred across. The buffer read size was a constant 1024 bytes since it was the optimum value according to the previous readings. Any increase in the read size of 1024 bytes did not yield a higher data transfer rate. Different write buffer sizes were tested and their transfer speeds recorded. The aim

of this experiment was to find out how much the write buffer size affected the transfer speed as well as find an optimum write buffer size for the final data transfer protocol. The transfer speeds for each of the buffer sizes are given below:

Write Buffer size (bytes)	Time (seconds)	Transfer speed (KB/s)
32	322.049	13.023
64	179.427	23.376
128	93.616	44.803
256	48.211	86.998
512	28.816	145.554
1024	30.534	137.365
2048	25.895	161.969
4096	24.199	173.324
8192	23.937	175.219
16384	32.157	130.430

Table 5.3: Table showing the transfer speeds after changing the buffer write size.

The buffer sizes that were used for the experiment started at 32 bytes. On iteration, the buffer size was doubled until no further increase in transfer speeds was recorded. This process continued until a write buffer size of 16384 bytes. The table above is graphically represented below:



Graph 5.3: Graph showing the transfer speed with varying write buffer sizes.

5.3.4 Finding the optimum buffer sizes

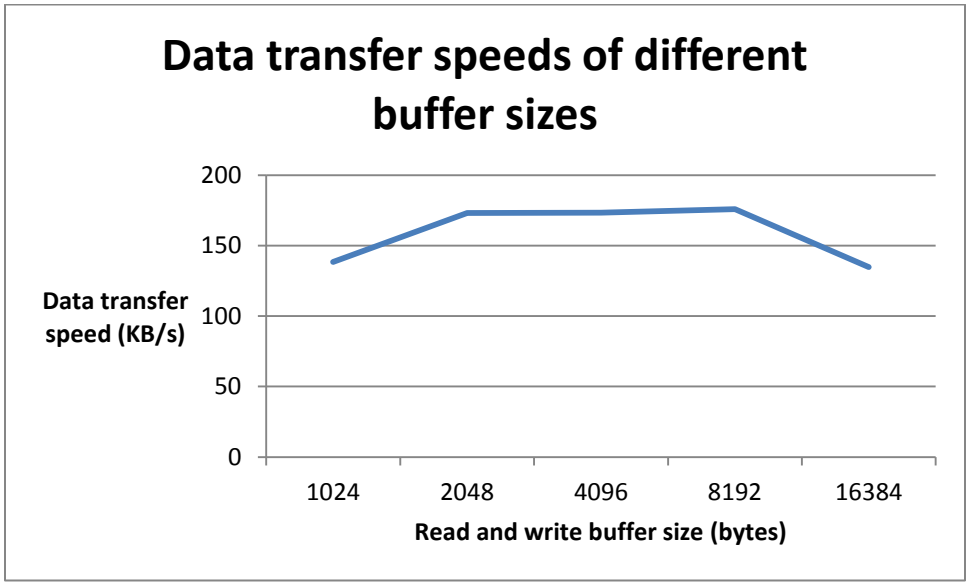
For this experiment, different combinations of values for read and write buffers were tested using Bluetooth V3.0. Once again 4 megabytes of data will be transferred. According to the data acquired in Section 5.3.3 and Section 5.3.2, lower read and write buffer sizes correspond to lower transfer speeds. Higher buffer sizes eventually reach a threshold whereby any further increases in buffer sizes did not affect the data transfer speed. Therefore, buffer sizes between 1024 bytes and 16384 bytes are

combined in order to find the optimum combination that will maximize data transfer speeds. The transfer speeds were recorded and shown in the table below:

Write Buffer size (bytes)	Read Buffer size (bytes)	Time (seconds)	Transfer speed (KB/s)
1024	**1024**	30.281	138.507
2048	**2048**	24.225	173.141
2048	**1024**	25.913	161.860
4096	**4096**	24.187	173.420
4096	**2048**	24.110	173.961
4096	**1024**	23.862	175.772
8192	**8192**	23.945	175.163
16384	**16384**	31.107	134.834

Table 5.4: Table showing the transfer speeds after changing the both the buffer write size and the buffer read size.

Since the data transfer speed is affected more substantially by the write buffer size (as seen in Section 5.3.3 and Section 5.3.2), the write buffer size was increased. From the information above, all the experimented conducted below will use read and write buffer sizes of 8192 bytes. The above table with the same read and write buffer sizes is graphically represented below:



Graph 5.4: Graph showing the different transfer speeds after changing read and write buffer sizes.

5.3.5 Terminating block technique

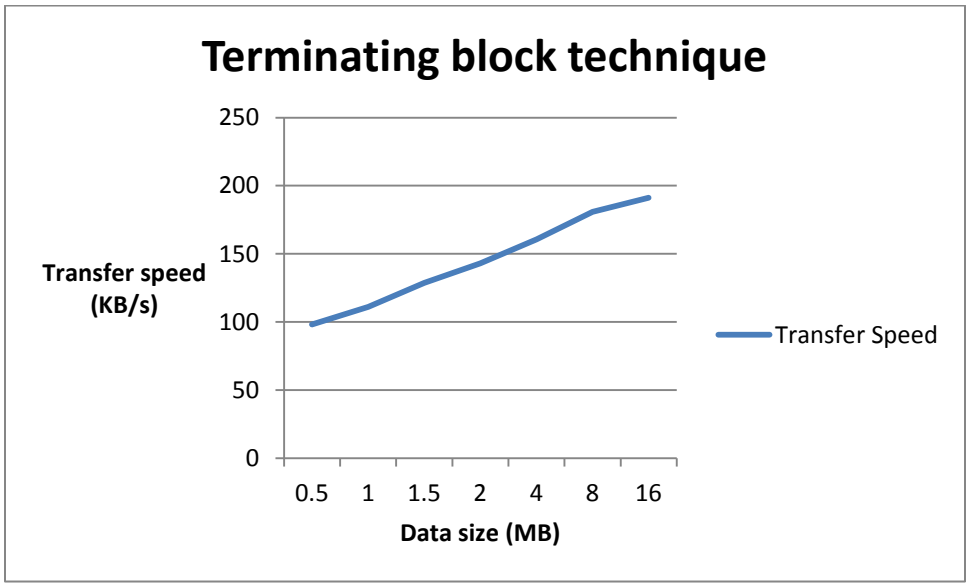
The first data transfer algorithm that was tested was the terminating block technique. A terminating block of data is appended to the end of the data block that is being transferred to signal the end of the file. The implementation details of the algorithm were discussed in Section 4.3.3.1 and this algorithm is expected to be the least efficient of the three data transfer algorithms to be tested. The algorithm will be evaluated by testing different data sizes and observing its corresponding data transfer speed. The

read and write buffer sizes will be fixed at 8192 bytes and Bluetooth V3.0 will be used. The data sizes used for this experiment started at 0.5MB and was thereafter doubled until 16MB. This gives a realistic view on the data transfer speeds of different file sizes and also gives an indication of how the algorithm scales as data sizes get larger. The transfer speeds were recorded and is given by the table below:

Data size (MB)	Time (seconds)	Transfer speed (KB/s)
0.5	5.347	98.050
1	9.443	111.040
1.5	12.228	128.627
2	14.659	143.058
4	26.129	160.524
8	46.399	180.792
16	87.753	191.186

Table 5.5: Table showing the data transfer speeds of the terminating block technique.

The above information is graphically represented below:



Graph 5.5: Graph showing the transfer speed of the terminating block technique.

5.3.6 Last byte check

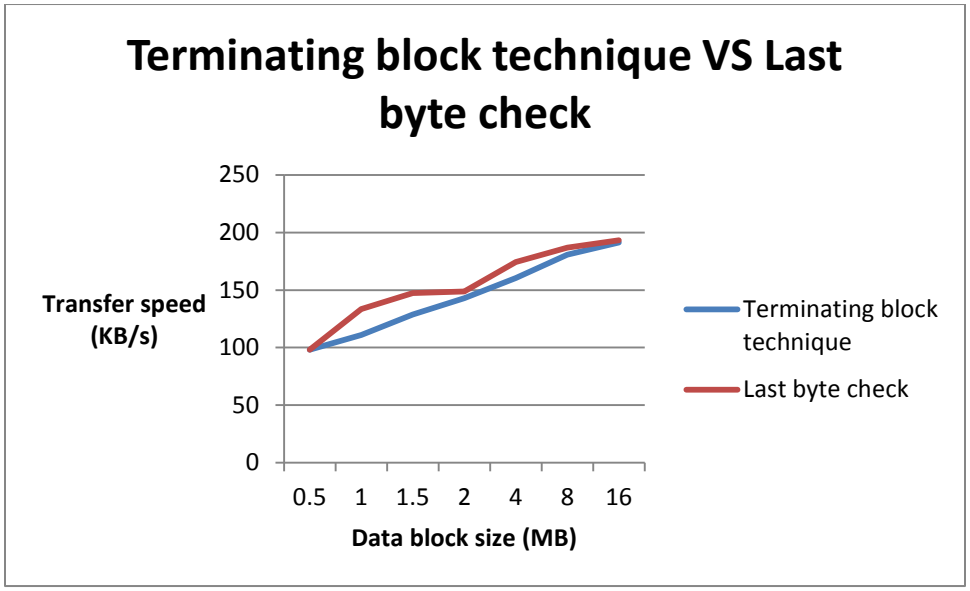
The second data transfer algorithm that was tested was the last byte check algorithm discussed in Section 4.3.3.2. This algorithm assigns the last byte of each data block that is being sent across to 0 if it is not the last block and assigns it to 1 if it is the last block of data to be received. This algorithm is expected to perform better than the terminating block technique as fewer computations are needed to determine whether the receiver has reached the end of the file. The same data block sizes were used to test the data transfer algorithm and their experimental data is recorded below:

Data size (MB)	Time (seconds)	Transfer speed (KB/s)
0.5	5.347	98.050

1	7.854	133.495
1.5	10.671	147.396
2	14.109	148.644
4	24.047	174.419
8	44.870	186.952
16	86.831	193.216

Table 5.6: Table showing the data transfer speeds of the last byte check algorithm.

As seen in the table above, this algorithm performs better than the terminating block technique. It performs consistently better in terms of data transfer speeds and this is shown in the graph below:



Graph 5.6: Graph showing the transfer speed of the terminating block technique VS the last byte check.

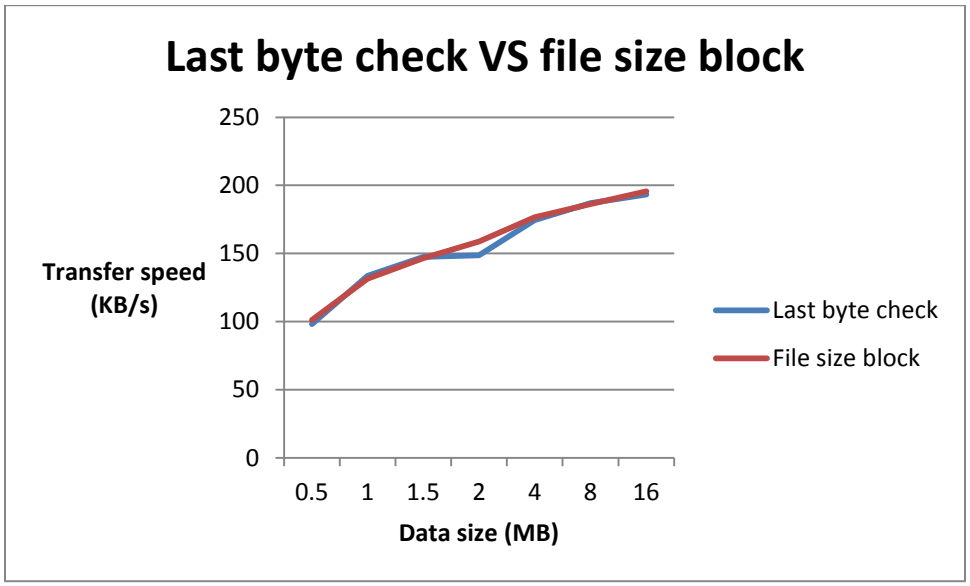
5.3.7 File size block

The last data transfer algorithm that was implemented was the file size block algorithm as discussed in Section 4.3.3.3. The first block of data sent across carried the total number of bytes that needs to be received. This allows the receiver to determine when the end of the file has been reached. It is expected that this algorithm will perform better when compared to both the data transfer algorithms above. The experiment for this algorithm used the same data block sizes to ensure consistency. The transfer speed was recorded and given in the table below:

Data size (MB)	Time (seconds)	Transfer speed (KB/s)
0.5	5.185	101.123
1	7.967	131.598
1.5	10.735	146.507
2	13.216	158.681
4	23.738	176.695
8	45.023	186.316
16	85.764	195.620

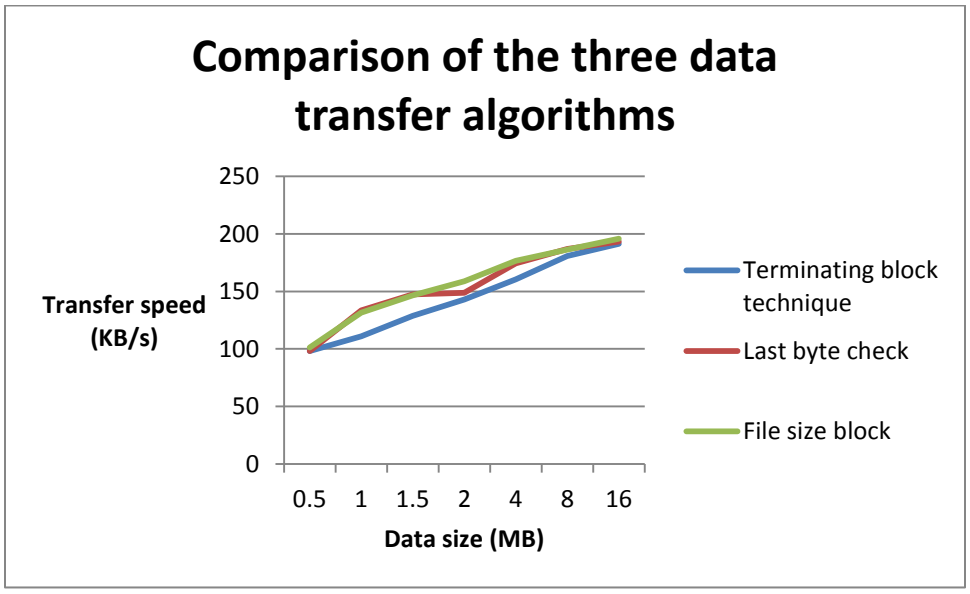
Table 5.7: Table showing the data transfer speeds of the file size block algorithm.

Whilst this algorithm does perform better when compared to the last byte check algorithm, there is only a small increase in data transfer speed. Their comparative transfer speeds are graphically represented below:



Graph 5.7: Graph showing the transfer speed of the last byte check VS file size block.

It is evident that the last byte check and the file size block algorithm perform better than the terminating block technique especially for smaller inputs. The differences in data transfer speeds is not substantial due to the read and write functions taking more time than the end of file checking algorithms. This is shown in the graph below:



Graph 5.8: Graph showing the transfer speed of the three data transfer algorithms.

5.3.8 Latency

Since the different data transfer algorithms only affect the transfer speeds and data exchange once a connection has been established, the latency is not affected by the different data transfer algorithms. Therefore latencies for the different algorithms were not measured. However, different latencies for Bluetooth V2.1 and Bluetooth V3.0 were recorded and is shown in the table below:

Bluetooth Version 2.1	Bluetooth Version 3.0
3.07secs	1.38sec
3.26secs	1.42sec
2.98secs	1.36sec
2.78secs	1.54sec
Average: 3.023secs	**Average: 1.425secs**

Table 5.7: Table showing the data transfer speeds of the file size block algorithm.

5.3.9 Multiple devices

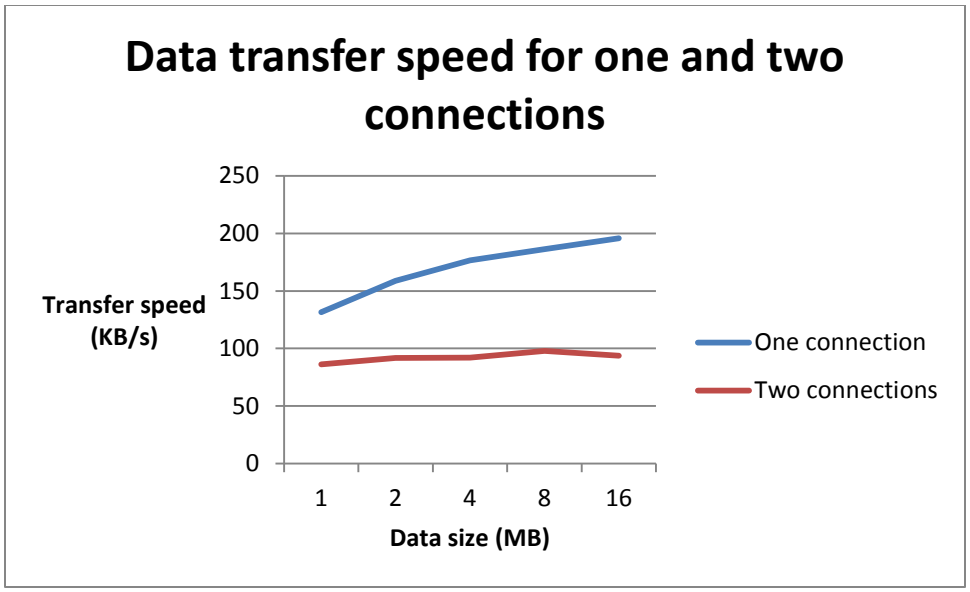
Once the data transfer algorithm was refined, experiments were extended for the program to handle connections between multiple devices. The server can only accept one connection per channel due to the limitations of Bluetooth. This implies that multiple connections need to be created on different channels for data transfer between multiple devices to occur. This is done by creating many channels that the server listens to by assigning a unique UUID to each of the channels. Thereafter devices can connect to each of these channels using different UUIDs.

Due to number of devices available, devices were tested handling up to two different connections. Different data size blocks were once again sent across. These data blocks were simultaneously sent across two devices using Bluetooth V3.0 and their data transfer efficiencies were evaluated. The first data block used was 1MB, and was doubled on iteration until 32MB. These values were then compared to the previous values obtained using only one connection. The data transfer protocol used for this test is the file size block algorithm due to its superior efficiencies compared to last byte check and the terminating block technique. Both the read and write buffer sizes were a constant 8192 bytes. The data transfer speeds for a device simultaneously transferring data between two devices were recorded below:

Data size (MB)	Time (seconds)	Transfer speed (KB/s)
1	11.595	86.244
2	21.775	91.849
4	43.503	91.947
8	81.596	98.044
16	170.737	93.711

Table 5.8: Table showing the data transfer speeds of a device handling two connections.

As shown above, there is a significant decrease in data transfer speed as expected when handling two connections. The data transfer speed while handling one connection and handling two connections are graphically represented below:



Graph 5.8: Graph showing the transfer speed of handling one connection and handling two connections.

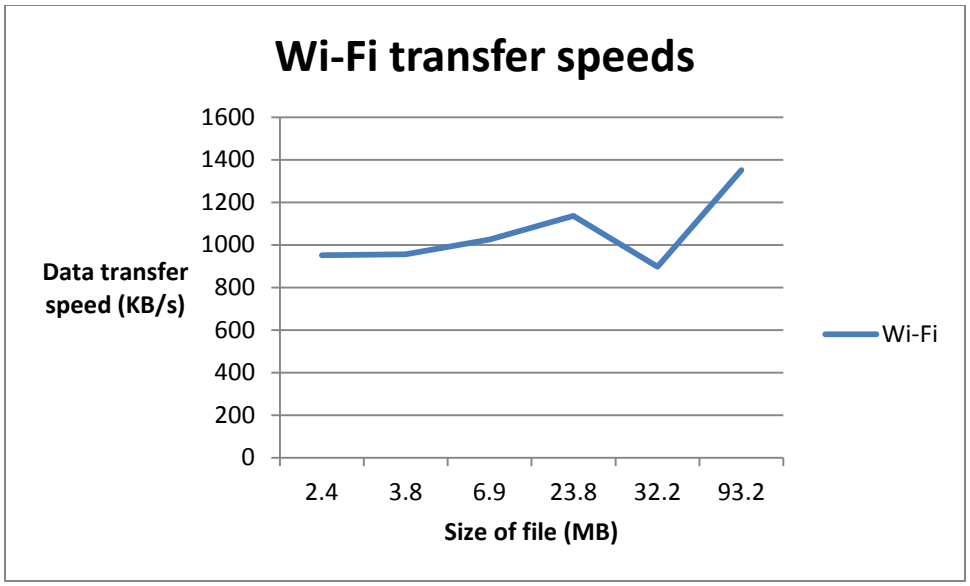
5.4 Wi-Fi testing

As mentioned in Section 3.6, only Android 4.0 supports peer-to-peer file sharing using Wi-Fi direct. Consequently, Wi-Fi testing was limited to the transfer of files via the usage of access points which was supported by the use of Samsung Kies. Iterative improvements did not take place since the data transfer protocol was not coded. Nevertheless, the capabilities of Wi-Fi was evaluated and recorded. The power consumption of using Wi-Fi was also evaluated due to its superior data transfer speeds. Different file sizes were used instead of fixed block sizes and their data transfer speeds are given in the table below:

Data size (MB)	Time (seconds)	Transfer speed (KB/s)
2.4	2.522	951.626
3.8	3.977	955.494
6.9	6.733	1024.803
23.8	20.951	1135.983
32.2	35.887	897.608
93.2	68.953	1351.645

Table 5.9: Table showing the data transfer speeds using Wi-Fi.

Although Wi-Fi has a much higher data transfer speed compared to Bluetooth, it is shown in the graph below that the transfer speeds are inconsistent:



Graph 5.10: Graph showing the Wi-Fi transfer speeds for varies file sizes

5.5 Power consumption

The battery information retrieved from the battery manager is only a percentage of the remaining battery. Although this does give a precise indication of how much battery is used during the transfer of data, it gives an indication of whether the amount of battery used is feasible. If the battery usage did not decrease by at least 1%, the experiment was repeated until there was a drop in battery life of at least 1%. Thereafter the total battery usage of an iteration was determined by dividing 1% by the number of iterations.

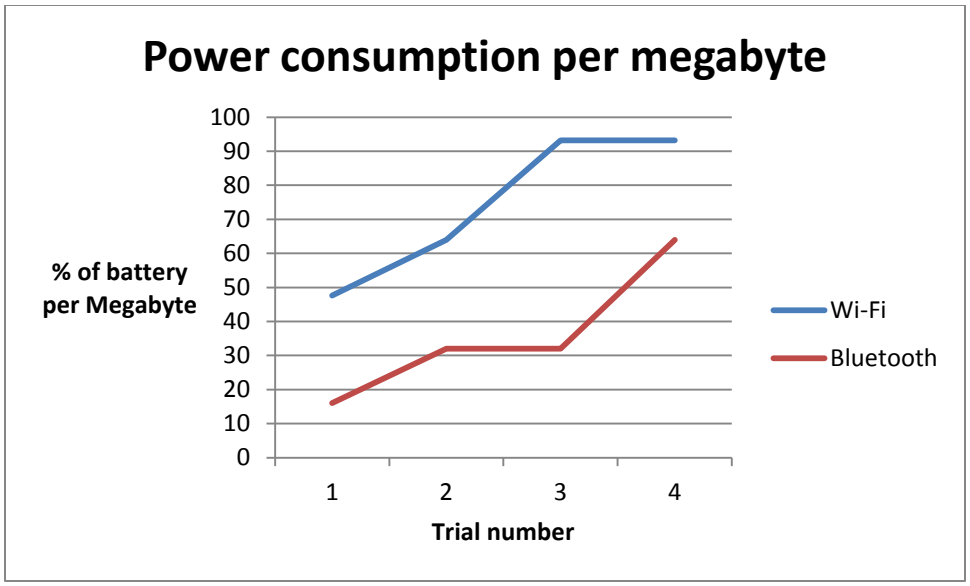
5.5.1 Bluetooth vs. Wi-Fi

Different size data were sent across using either Bluetooth and Wi-Fi and their battery usages are given by the table below. Although the data sizes vary between Bluetooth and Wi-Fi, the time taken to complete the transfer vary due to Wi-Fi having a superior transfer speed compared to Bluetooth.

Data size (MB)	Bluetooth/Wi-Fi	Time taken	Battery usage
8	Bluetooth	23.738	<0.5%
16	Bluetooth	45.023	<0.5%
32	Bluetooth	85.764	<1%
64	Bluetooth	171.797s	~1%
23.8	Wi-Fi	19.950	<0.5%
32	Wi-Fi	35.88	<0.5%
93.2	Wi-Fi	68.953	<1.0%
186.4	Wi-Fi	137.906	<2.0%

Table 5.10: Table showing the battery usage of Bluetooth and Wi-Fi.

The amount of power consumption used per megabyte is shown below:



Graph 5.11: Graph showing power consumption of Bluetooth and Wi-Fi per megabyte.

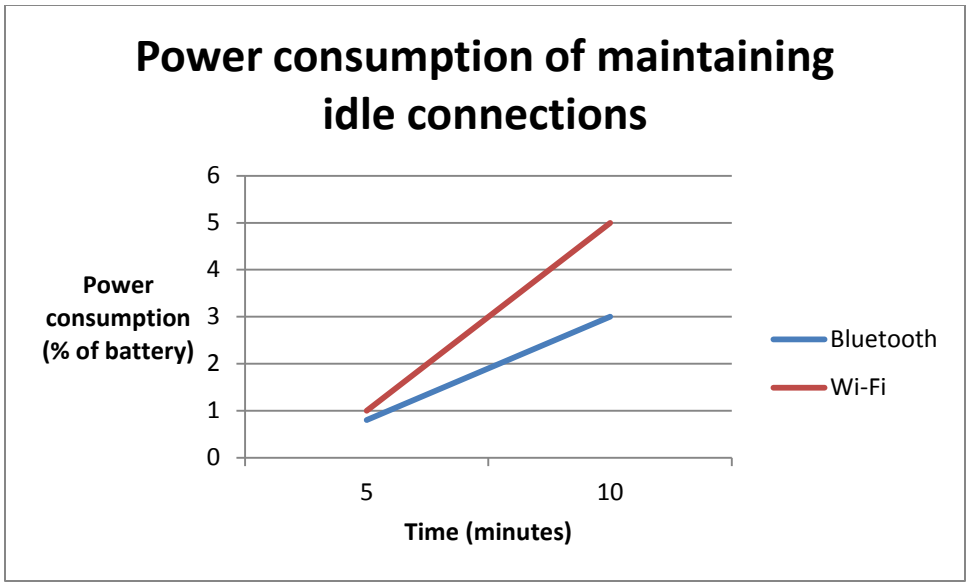
5.5.2 Maintaining idle connection

On top of evaluating the power consumption of Bluetooth and Wi-Fi while data is being transferred, maintaining an idle connection between two devices will also reduce battery life. An experiment was conducted whereby a connection was established using Bluetooth and Wi-Fi and no data transfer took place. This allowed an evaluation to be made on the battery usage when a connection is idle. Since it is not expected that connections are remained idle for long periods of time, only two readings (5 minutes and 10minutes) were taken. Their results were given by the table below:

Wireless technology	Time Idle	Battery Usage
Wi-Fi	10minutes	<5%
Bluetooth	10minutes	~1%
Wi-Fi	5minutes	<3%
Bluetooth	5minutes	<1%

Table 5.11: Table showing the power consumption of maintaining idle connections.

The above table is graphically represented below:



Graph 5.12: Graph showing power consumption of Bluetooth and Wi-Fi maintaining idle connections.

5.5.3 Managing multiple connections

Bluetooth was also tested for scalability issues whereby the power consumption by a device was evaluated while it was handling multiple connections. Two simultaneous connections were simulated on the Samsung Galaxy S2 and its power consumption is given below:

Number of connections	Data size (MB)	Battery Usage
1	16	<1%
2	16	<1%
1	32	<1%
2	32	<1.5%

Table 5.12: Table showing the power consumption of multiple connections.

5.6 Experiment summary

Many experiments were conducted to evaluate the data transfer speeds and power consumption of Wi-Fi and the different implementations of Bluetooth. Experiments were conducted in a way such that most external factors that could influence the results were removed. These factors were discussed in Section 5.1 and include background processes, distance between devices and the remaining battery life of devices. In particular, the experiments focused mainly on the data transfer speeds produced by varying several implementation details. These were tested in Section 5.3 where different read and write buffer sizes were used to obtain the optimum solution for the data transfer. On top of data transfer speeds, the latency of connecting two devices together was also evaluated for Bluetooth V3.0 and Bluetooth V2.1. This gave an indication of the initial overhead of connecting the devices. Lastly, multiple connections (up to 2 due to hardware limitations) were handled by a device how multiple connections influenced data transfer speeds were analyzed.

Section 5.4 and Section 5.5 focused on data transfer using Wi-Fi and also evaluating its power consumption. The power consumption used by Bluetooth were compared to the power consumption

used by Wi-Fi to transfer the same amount of data. Although Wi-Fi comparatively transfers more data using the same amount of battery, Bluetooth uses less battery over the same amount of time. This is further proved in Section 5.5.2 where the power consumption of maintaining idle connections were evaluated. Bluetooth uses much less power to maintain idle connections compared to Wi-Fi.

Chapter 6

6. Discussions and recommendations

Chapter 5 discussed the design of experiments and how accurate data would be obtained for the experiments. Thereafter, experiments were conducted and their results were displayed in tables or were graphically represented. These results reflect the feasibility of using Bluetooth and Wi-Fi for data transfer between devices. In particular, the power consumption and data transfer speeds were the focus of the experiments and are the main factors that determine the feasibility of using wireless technologies for data transfer. This Chapter serves to discuss interesting results and provide insight into the data that was obtained in Chapter 5. Furthermore this Chapter provides recommendations on how data transfers should be handled.

6.1 Changing read and write buffer sizes

Data is transferred by firstly establishing a connection between two devices using Bluetooth. One device acts as the server waiting for a connection request and the other device acts as the client initiating a connection with the server. These devices can thereafter transfer data by writing to the corresponding OutputStream that is given by the BluetoothSocket. Data is written or read at fixed data buffer sizes from the InputStream or OutputStream. The choice of buffer sizes influence the data transfer speed obtained between the two devices. Particularly, a read buffer size of 8192 bytes and a write buffer size of 8192 bytes were used. When a file is sent across, the file is broken up in 8192 byte blocks and written to the OutputStream. Thereafter, the data is read using an 8192 byte buffer from the InputStream.

The reason behind the data transfer speeds being influenced by the change in buffer sizes is the time tradeoff between preparing large chunks of data to calling the read and write function of the InputStream and OutputStream. More time is required to write larger data blocks at a time but more calls to the write function is needed to write the same amount of data. Similarly this applies to reading the data from the InputStream. From the data acquired in Section 5.3.4, an increase in buffer size initially increased the data transfer speeds. Once data blocks were increased over 8192, there was a decrease in data transfer speeds. A combination of 8192 bytes for the write buffer and a corresponding 8192 bytes for the read buffer was determined as the optimum solution. The data transfer speed for this combination when transferring a 4 megabyte file using Bluetooth V3.0 was 175.163 kilobytes per second.

6.2 Data transfer algorithms

In the implementation of Bluetooth data transfer, three different algorithms were implemented. These algorithms provided methods for the receiver to determine where the end of the file is in order to halt any further readings from the InputStream. Each of these algorithms provided consistent ways to check for end of file. However, the terminating block technique, although easy to implement, performed the worst of the three algorithms. This is due to the many comparisons necessary to compare whole byte blocks. Specifically, using a read buffer of 8192 bytes, the terminating block technique would require 8192 comparisons to determine whether the current block read from the InputStream was the

terminating block. Furthermore, this technique uses extra memory as an extra block is needed to be appended at the end of the file.

The last byte check technique showed better performance according to Graph 5.5. This is considered a speed up of the terminating block technique as on iteration, the last byte of each byte block was checked to determine whether the current block was the terminating block. Since the last byte check technique only checks one byte, this is much more efficient than the terminating block technique, which requires up to 8192 byte checks.

Lastly the file size block algorithm was tested and also showed slightly better performance according to Graph 5.6. This was done by appending the file size to the beginning of the file. Although this algorithm had slightly better performance, it is constrained to the sender of the file knowing the file size. This is not always the case. For example: If the sender wishes to transmit data for a certain fixed amount of time, it is uncertain how much data would be transferred across.

In conclusion, the last byte check technique is considered the best technique due to its improved efficiencies compared to the terminating block technique as well as its flexibility in not having to know the size of the file prior to sending it. Although the file size block algorithm provided better efficiencies, these were only slight improvements. The last byte check technique provided data transfer speeds of up to 195KB/s when transferring files of 16 megabytes using Bluetooth V3.0.

6.3 Handling multiple devices

Connections between devices via Bluetooth are established via a communication channel that is defined by the UUID of the connection. Each channel can only facilitate one connection at a time and if another devices wishes to connect under the same UUID, the first connection is required to be closed. To manage multiple connections between devices, different UUIDs were used between multiple devices. Their data transfer efficiencies were shown in Graph 5.8. Although only two simultaneous connections were tested, the drop in data transfer speeds were evident according to graph 5.8. The decrease in data transfer speeds scaled to around half of the normal speed. For a 16 megabyte data block, the transfer rate using Bluetooth V3.0 was 195KB/s. Simultaneously handling two connection under two separate channels lowered the transfer rate to 98KB/s, which is about half of the normal transfer speed. This is expected since the effective transfer speeds of the two connections added together give the result of the data transfer speed of one connection.

6.4 Power consumption

The power consumption of the data transfers were obtained using the battery manager as mentioned in Section 4.4.2. Evaluating the Bluetooth data transfers showed that Bluetooth uses a small amount of battery power for the data transfers. All the data transfers up to 64 megabytes were completed using less than 1% of the total battery life. Furthermore maintaining the connection over long periods of time did not use much battery power either. This deemed Bluetooth a feasible candidate for wireless data transfer.

Experiments were thereafter extended to obtain results for the amount of power consumption while handling multiple connections. Using Bluetooth again, data was transferred over two connections whereby a total of 64 megabytes of data was transferred across to two separate devices. This used less than 1.5% of the total battery life. This shows that Bluetooth does not use much power even when connected to multiple devices.

6.5 Bluetooth vs. Wi-Fi

Both Bluetooth and Wi-Fi provided excellent results in terms of data transfer speeds, power consumption and robustness. Each of these factors determine whether or not wireless data transfer is feasible.

6.5.1 Data transfer speed

Whilst many efforts were made to improve the data transfer speeds of Bluetooth by changing many implementation details, Wi-Fi has a superior data transfer speed. Wi-Fi provided data transfer speeds of approximately 1 megabyte per second while Bluetooth provided approximately 195KB/s (Using Bluetooth V3.0) after many stages of refinements.

6.5.2 Power consumption

Although Wi-Fi has a superior data transfer speed compared to Bluetooth, it consequently also uses a lot more battery power. In comparison although Wi-Fi uses slightly more power to transfer a same sized file, Bluetooth takes much longer to transfer the same file. Therefore whilst Wi-Fi completes the task using slightly more power than Bluetooth, Bluetooth uses much less power per second compared to Wi-Fi. This is particularly evident in Table 5.11 where Wi-Fi uses at least 5 times the amount of power to sustain an idle connection compared to Bluetooth. This is because wireless signals are stronger and travel a lot further than Bluetooth signals [29]. Bluetooth signals travel up to 30 feet whereas wireless signal can travel up to 300feet, about 10 times as far [29].

6.5.3 Robustness

While each of the experiments were conducted in Section 5.3, the total data sent and the total data received were compared. If the number of bytes received matched the number of bytes sent, the data transfer is deemed reliable as no data was lost. This was true for both Bluetooth and Wi-Fi.

6.6 Recommendations

Since Wi-Fi uses considerably more power compared to Bluetooth, it is recommended that all connections are established using Bluetooth. Thereafter, depending on the file size, a choice should be made between Bluetooth and Wi-Fi to facilitate the data transfer. In general, if the file size is below 16 megabytes, it is suitable to use Bluetooth to transfer the data. Although Wi-Fi has a considerably higher data transfer speed, transferring smaller files using Bluetooth reduces energy consumption. Files of size greater than 16 megabytes should be transferred using Wi-Fi. This is because Wi-Fi will only use slightly more energy to transfer the same amount of data whilst reducing the amount of time taken to complete the data transfer. Lastly, once a connection has been established Bluetooth should be used to handle idle connections as Wi-Fi uses at least 5 times more power to handle idle connections.

Chapter 7

7. Conclusion

The purpose of this research project was to evaluate the feasibility of using Wi-Fi and Bluetooth for wireless data transfer between smart phones as well as to refine a data transfer protocol for Bluetooth. Due to the popularity of wireless technologies and the increased capabilities of smart phones, much research has been done in the field of mobile ad hoc networks. Protocols have been developed [33], [35], [36], and a feasible wireless technology needs to be integrated with the protocol to create mobile ad hoc networks. As a result, this project designed a system in Android that refined data transfer via Bluetooth and furthermore used access points to provide insight into Wi-Fi technology. Refinements were aimed at increasing the data transfer rate of Bluetooth and lowering the energy consumption while providing reliable and robust connections.

In order to refine data transfer between smart phones, the Bluetooth library for Android was used. Specifically, data transfer is facilitated by the use of a BluetoothSocket class, whereby the InputStream and OutputStream could be obtained. The sender of data would write to the OutputStream and the corresponding receiver would read data from the InputStream. In order to obtain the BluetoothSocket mentioned above, one device acts as the server waiting for a connection request, while the other acts as a client requesting a connection. This process is represented in Figure 3.1.

Refinements towards the final data transfer protocol firstly involved the changing of read and write buffer sizes. This provided insight into the time tradeoff between reading and writing many small data blocks to the BluetoothSocket and reading and writing fewer larger data blocks. Experimental results show that the optimum read and write buffer sizes is 8192 bytes. This produced an average data transfer speed of up to 195Kb/s while transferring a 4 megabyte file. Experimental data showed that increasing the buffer sizes produced higher data transfer rates while any buffer sizes larger than 8192 bytes reduced the data transfer speed.

Thereafter, three algorithms were implemented to notify the receiver when the end of the file had been reached. This halts the process of the receiver attempting to read from the InputStream which may cause unnecessary overhead. The first algorithm implemented was the terminating block technique and was followed by the last byte check technique. Whilst the terminating block technique is easy to implement and understand, it performed the worst of the three algorithms. The last byte check algorithm provided significant speedups since it made fewer comparisons to determine when the end of the file had been reached. Lastly the file size block algorithm was implemented and showed some speedups compared to the last byte check algorithm. Due to the constraints of the sender needing to know the file size prior to sending the file, the file size block algorithm may not always be applicable to data transfer. Consequently, the last byte check algorithm was deemed the best algorithm to facilitate file transfer. Despite performing slightly worse compared to the file size block, lack of constraints provided the flexibility necessary to facilitate reliable data transfer.

Since mobile ad hoc networks require multiple connections between multiple devices, tests were conducted to evaluate the transfer speeds and power consumption of a device managing two simultaneous connections. Experimental data showed that data transfer speed is approximately halved while two connections are transmitting data. This is expected as the effective transfer speed of the two connections added together gives the transfer speed of one connection. Although the analysis of power consumption was not very accurate, experimental data implied that power consumption increased about 50% while handling two connections. This shows that Bluetooth does not use much power even when connected to multiple devices.

Wi-Fi connections were tested using access points. Experimental results showed that Wi-Fi has a much higher data transfer rate compared to Bluetooth. Consequently, Wi-Fi also uses more power over the same amount of time but since it supports higher bandwidth, Wi-Fi uses comparably similar amounts of power to Bluetooth for the same volume of data. This supports Pering's claim [32] that Wi-Fi is slightly more energy efficient from a "pure energy/bit standpoint". However, experimental results also show that Wi-Fi has uses a lot of power to maintain an idle connection compared to Bluetooth. This supports Balani's [34] claim that Bluetooth is much more energy efficient in terms of connection maintenance.

7.1 Research questions and recommendations

The two main research questions that this research project aimed to answer were:

1. How do Bluetooth and Wi-Fi compare in using network data in terms of data transfer speeds and power consumption?
2. Are connections between multiple devices possible? If so, how does this affect their data transfer efficiencies as the number of devices connected increases?

The research in this component of GROUT confirms the feasibility of Wi-Fi and Bluetooth as viable wireless technologies for establishing a mobile ad hoc network. Each candidate provided higher data transfer speeds while maintaining a lower power consumption rate compared to using network data. While Bluetooth provided a more energy efficient method of transferring data, Wi-Fi provided higher bandwidths which compensates for the amount of time that Wi-Fi requires transferring the same data volume. Connections between multiple devices are available through different RFCOMM channels. Bluetooth utilizes these channels by assigning a unique UUID to separate simultaneous connections while being able to listen for more incoming connection requests using different UUIDs. Experimental results show that Bluetooth data transfer speeds are halved if two simultaneous connections are being handled. In terms of power consumption, Bluetooth does not use significantly more power while handling two connections. Lastly it was evident multiple connections did not affect the robustness of Bluetooth since all data transfers were successful during experiments. In general, both Bluetooth and Wi-Fi had a 100% success rate while transferring data between smart phones.

Due to Wi-Fi's superior data transfer speed capabilities and energy efficiency for larger files, it is recommended that files greater than 16 megabytes are transferred using Wi-Fi. Smaller files should be transferred using Bluetooth despite lower data transfer speeds due to Wi-Fi's significant initial overhead. Since Wi-Fi uses considerably more power compared to Bluetooth, it is recommended that all

connections are established using Bluetooth where possible. Thereafter a choice should be made between Bluetooth and Wi-Fi to facilitate the data transfer. Idle connections in particular should always be handled using Bluetooth due to its significantly lower power usage compared to Wi-Fi.

7.1 Future work

In this research project, it is evident that constraints have been placed on Bluetooth and Wi-Fi. Specifically, Bluetooth is constrained to connections over a shorter distance compared to Wi-Fi and has much lower Bandwidth compared to Wi-Fi. With new technological advances which includes Bluetooth v4.0, the range of Bluetooth has been enhanced while still providing ultra-low power consumptions [38]. This provides opportunities for Bluetooth to be integrated with new devices that are required to operate with low energy consumption. As a result, this promotes networking opportunities in areas such as health care, sports and fitness, security and home entertainment [39]. The different refinements in data transfer protocols can not only be extended into Bluetooth v4.0 but also to other mobile platforms. In general, mobile ad hoc networks are more useful if a cross platform application is developed.

The main factor that constrains Wi-Fi from being the only wireless technology used to facilitate data transfers is its high energy consumption. Due to the time constraints and hardware availability placed on GROUT, Wi-Fi Direct had not been explored in detail. Future works include experimenting with Wi-Fi Direct and optimizing a protocol that lowers power consumption. Wi-Fi's superior connection range and bandwidth are both useful properties needed to create a mobile ad hoc network. Lowering the power consumption of Wi-Fi is essential to the applicability of Wi-Fi to MANETs. The approach of using different algorithms to check for the end of file can apply to Wi-Fi to optimize data transfer speeds.

Integration of the optimized data transfer protocol discussed in this project with the several MANET protocols provide a foundation to establish reliable and energy efficient MANETs. These MANETs need not be restricted to file transfer but can be extended to smart phones offering services. The data transfer protocol provided in this project facilitates flexible data transfer which enhances service offering opportunities via smart phones.

8. References

[1] Feeney, L. M., & Nilsson, M. (n.d.). Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, *3*, 1548-1557. Ieee. doi:10.1109/INFCOM.2001.916651

[2] Meler, R. (2010). Android 2 Application Development. ISBN: 978-0-470-56552-0, Chapter 1 p1-16, Chapter 13 p425-455

[3] Android development (n.d.). Connectivity-Bluetooth.[online] Available at: <http://developer.android.com/guide/topics/connectivity/bluetooth.html> [Accessed 10 September 2012]

[4] Miller, B.A., Bisdikian, C. (2001). Bluetooth revealed, 2nd edition. ISBN: 0130672378

[5] Zbruba, G. V., Basagni, S., & Chlamtac, I. (n.d.). Bluetrees-Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks, 273-277.

[6] Haartsen, J. (1998). BLUETOOTH — The universal radio interface for ad hoc , wireless connectivity, (3), 110-117.

[7] Varoquaux, G. (2012). Timing problems with a computer. [online] Available at: <http://gael-varoquaux.info/computers/real-time/index.html> [Accessed 14 October 2012].

[8] Mishra, A. (2008). Introduction to Ad hoc Networks, 1-53.

[9] Yu, Z., Zhou, X., Zhang, D., Chin, C.-yau, Wang, X., & Men, J. (2006). for Smart Phones.

[10] Marsh, A. J. (2005). IEEE Pervasive Computing 4 (2) 20-27., *4*, 20-27.

[11] Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., & Estrin, D. (2010). Diversity in smartphone usage. *Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10*, 179. New York, New York, USA: ACM Press. doi:10.1145/1814433.1814453

[12] Adoption of mobile Internet services: An exploratory study of mobile commerce early adopters Per E. Pedersen Agder University College. (n.d.).

[13] Pucha, H., Das, S. M., Hu, Y. C., Lafayette, W., Hash, D., & Dhts, T. (2004). How to Implement DHTs in Mobile Ad Hoc Networks ?, (1), 2-3.

[14] News, L., News, L., Tech, B., Tech, G., & Apple, L. (2011). Microsoft collects locations of Windows phone users, 1-5.

[15] Carroll, A. (n.d.). An Analysis of Power Consumption in a Smartphone.

[16] Zhang, L., Dick, R. P., Mao, Z. M., Wang, Z., & Arbor, A. (n.d.). Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones.

[17] Wooley, T. (2010). A Comparative Study of the Android and iPhone Operating Systems.

[18] Rodoplu, V., & Meng, T. H. (1999). Minimum Energy Mobile Wireless Networks, *17*(8), 1333-1344.

[19] In, O. (2002). Bluetooth in Wireless Communication, (June), 90-96.

[20] Android development, (n.d.). UUID [online] available at <http://developer.android.com/reference/java/util/UUID.html> [Accessed 5 October 2012]

[21] Anonymous, (n.d.). What is an UUID? [online] available at http://www.famkruithof.net/guid-uuid-info.html [accessed 5 October 2012].

[22] Finn, K, (2012). How to calculate outliers. [online] available at http://www.ehow.com/how_5201412_calculate-outliers.html [accessed 17th October]

[23] Jones, M., Marsden, G., Jones, M., & Marsden, G. (2005). *Mobile Interaction Desig. Chapter 8.6.4 p238-p242*

[24] Anonymous, (n.d.). Bluetooth [online] available at http://en.wikipedia.org/wiki/Bluetooth#Bluetooth_v2.1_.2B_EDR [accessed 17 October 2012]

[25] Lee, U., Jung, S., Cho, D.-ki, Chang, A., Choi, J., & Gerla, M. (n.d.). Bluetooth-based P2P Content Distribution to Mobile Users.

[26] Visualization, D., & Data, T. M. (n.d.). Representing Data Graphically, 1-34.

[27] Micskei, Z,. (2012). Robustness testing, [online] available at http://mit.bme.hu/~micskeiz/pages/robustness_testing.html [accessed 19 October 2012]

[28] Lu, X. (2010). Robustness Checks and Robustness Tests in Applied Economics, 1-39.

[29] Volgar, E. (n.d.). Bluetooth vs. Wi-Fi Power Consumption. [online] avilable at http://techtips.salon.com/bluetooth-vs-wifi-power-consumption-20975.html [accessed 20 October 2012]

[30] Carrol, A., Heiser G., (2012). An analysis of power consumption in a smart phone.

[31] Agarwal, Y., Schurgers C., and Gupta R., "Dynamic Power Management using On Demand Paging for Networked Embedded Systems", In Proc. of Asia-South acific Design Automation Conference (ASPDAC). 2005.

[32] Pering, T. (2006). CoolSpots : Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces.

[33] Zhong, S., Chen, J., Yang, Y., R., (2003). Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks.

[34] Balani, R., (n.d.), Energy Consumption Analysis for Bluetooth, WiFi and Cellular Networks. University of California at Los Angeles

[35] Feeney, L. M., & Nilsson, M. (n.d.). Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, *3*, 1548-1557. Ieee. doi:10.1109/INFCOM.2001.916651

[36] Pucha, H., Das, S. M., Hu, Y. C., Lafayette, W., Hash, D., & Dhts, T. (2004). How to Implement DHTs in Mobile Ad Hoc Networks ?, (1), 2-3.

[37] Klemm, A., Lindemann, C., Waldhorst, O., P., (n.d.). A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks

[38] Phan, R. C., & Mingard, P. (2010). This item was submitted to Loughborough ' s Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions . For the full text of this licence , please go to : Analyzing the Secure Simple Pairing in Bluetooth v4 . 0.

[39] Bluetooth official website, (n.d.). [online] available at http://www.bluetooth.com/Pages/low-energy.aspx, [accessed 27 October 2012].