REF:JKU/2/83/022

Jomo Kenyatta University of Agriculture and Technology (JKUAT)

School of Computing and Information Technology (SCIT)

Department of Computing

Computer Science Final Year Project Report

Student Name: Kairu Joshua Wambugu	Reg. No: CS281-0720/2011
Course: BSc. Computer Science	
Title/Research Topic: Cheaper Exchange of Information Via Wireless Technology	
Supervisor 1: Professor Waweru Mwangi	Sign:_____
Supervisor 2: Doctor Petronilla Muthoni	Sign:_____
Date Submitted:_____	

Abstract

This project is an effort to reduce cost. One of the ways we incur expenses in our daily lives is through communication. Specifically, communicating through a third party increases costs since it calls for compensation of that third entity. Removing the third party during communication would reduce communication costs substantially. The most common communication devices among humanity at the moment are mobile phones. And smart phones will be just as common very soon. For individuals to contact each other via smartphones, a third party – such as a carrier – is usually involved. While there are many ways to remove this third party, the project to follow will focus on establishing an audio communication between two smartphones over a wireless connection without a third party. It is hoped that the idea behind this project will, in the (probably very) long run reduce communication costs between owners of smart devices.

Acknowledgements

I would wish to take the next few lines of text to acknowledge the individuals who have made it possible for me to finish this project.

- Jehovah God, the true God, the Creator and the Maintainer of the Universe, for giving me the wisdom, the courage, the supportive environment and the opportunity to finish this project.
- My parents, Hosea Kairu Wambugu and Joyce Njoki Kairu, for sharing their experiences, giving words of wisdom, and always pushing me when I became slightly lazy.
- The project coordinator, Harriet Ratemo, setting the infrastructure needed for my project mates and I to go through the project process without worrying about technical details outside the project's scope.
- My supervisors Professor Waweru Mwangi and Doctor Petronilla Muriithi for providing invaluable advice on, among other things, how to present my project in an acceptable manner.
- Julian Keya for assisting me do the experiments whose results are presented in this project.

Contents

Lists of Figures and Tables

List of Figures

List of Tables

Definition of Abbreviations

- 3G – 3rd Generation.
- 3GPP – 3rd Generation Partnership Project.
- AAC – Advanced Audio Coding.
- ACELP – Algebraic Code Excited Linear Prediction.
- AES-CCMP – Advanced Encryption Standard Counter Block Chaining Message Authentication Protocol.
- AMR – Adaptive Multi-Rate.
- AMR-NB – Adaptive Multi-Rate Narrowband.
- AMR-WB – Adaptive Multi-Rate Wideband.
- AP – Access Point.
- API – Application Program Interface.
- AUC – Authentication Center.
- BSC – Base Station Controller.
- BSS – Base Station System.
- BTS – Base Transceiver Station.
- CDMA – Code Division Multiple Access.
- CELP – Code Excited Linear Prediction.
- CNG – Comfort Noise Generation.
- DHCP – Dynamic Host Configuration Protocol.
- DTX – Discontinuous Transmission.
- ECC – Electronic Communications Committee.
- EDGE – Enhanced GSM Data Environment.
- EIR – Equipment Identity Register.
- ETSI – European Telecommunications Standards Institute.
- GSM – Global System for Mobile communications.
- HDTV – High Definition Television.

- HLR – Home Location Register.
- IDE – Integrated Development Environment.
- IEC – International Engineering Consortium.
- IMDCT – Inverse Modified Discrete Cosine Transform.
- IP – Internet Protocol.
- Kbps – Kilobits per second.
- MDCT – Modified Discrete Cosine Transform.
- ME – Mobile Equipment.
- MP3 – Motion Picture Experts Group-1/2 Layer-3.
- MPEG – Motion Picture Experts Group.
- MSC – Mobile services Switching Center.
- NFC – Near Field Communication.
- OoBTC – Out-of-Band Transcoder Control.
- OSS – Operation and Support System.
- P2P – Peer-To-Peer.
- P2P GO – P2P Group Owner.
- PCM – Pulse Code Modulation.
- PCMCIA – Personal Computer Memory Card International Association.
- PIN – Personal Identification Number
- PSK – Pre-Shared Key.
- QoS – Quality of Service.
- SIM – Subscriber Identity Module.
- SS – Switching System.
- TCP – Transmission Control Protocol.
- TCP/IP – Transmission Control Protocol/Internet Protocol.
- TDMA – Time Division Multiple Access.
- TFO – Tandem-Free Operation.

- TNS – Temporal Noise Shaping.
- TrFO – Transcoder-Free Operation.
- UI - User Interface.
- USB – Universal Serial Bus.
- UTRAN – Universal Terrestrial Radio Access Network.
- VAD – Voice Activity Detector.
- VLR – Visitor Location Register.
- WCDMA – Wide Code Division Multiple Access.
- Wi-Fi – Wireless Fidelity.
- WLAN – Wireless Local Area Network.
- WPA2 – Wi-Fi Protected Access II.
- WPS – Wi-Fi Protected Security.

1 Introduction

The 21st Century has been characterised by very fast developments in technology. Services that we thought were impossible 100 or even 50 years ago are now commonplace, resulting in a lot of expediency. Among the devices that have brought this convenience – or inconvenience depending on one's viewpoint – is the mobile phone.

The mobile phone technology allows people to use handheld telephones to communicate with others despite the physical distance between them. This communication is done either via voice by one user calling another; or via reading by one user sending another a message using the Short Message Service (SMS). Both of these forms of communication usually involve the user having to pay the network service provider some money in exchange for the use of the provider's infrastructure to call or SMS. For the most part, the payment to the network provider outweighs the cost and inconvenience of a cell phone user having to set up their own equipment so as to achieve the same communication. But what if the two users were just a few meters away, maybe just separated by a wall? Would it make sense to have to pay the network service provider to communicate with someone so near? For quite some time, it seemed that mobile phone users had no choice.

Within the past half-decade or so, cellular phones have grown from just being devices to send and receive voice and SMS data to being devices that can do much more. Smartphones – loosely defined as cell phones with computer capabilities – have given people the capability to not only call and text but also check their email, surf the Internet, listen to music, keep up to date with the latest news, just to mention a few things. Smartphones have revolutionised the mobile phone revolution. With this mind, consider: It is currently possible for two networked computers to send and receive not only audio but also video data between themselves via a process known as streaming. The network between the two computers could either be wired or wireless. Could the same streaming be done between smartphones? A "Yes" answer with the corresponding implementation could conveniently put the network service provider out of the picture. And that question dovetails nicely into this project's problem statement.

1.1 Problem Statement

The research question that will guide this project is:

How can smartphones communicate with each other over short distances without incurring network service provider costs?

By the end of this project, the following objectives should have been met:

- Two smartphones should be able to connect with each other via wireless without the aid of an infrastructure device such as a wireless router or a wireless hotspot.
- The abovementioned smartphones should then be able to send and receive data – initially audio data – between themselves.

1.2 Justification

Generally, when a person wants to get information from another and both of them are in within each other's social space – between 2.2 and 3.7 meters (7 and 12 feet) according to Beck and Grajeda (2008, p. 43) – face to face communication is used. This form of communication is free and quite convenient. What about when the two people are within the same distance but separated by at most one concrete wall? What if the two people are not within each other's social space but are within 50 meters (about 164 feet) of each other and their smartphones have a clear line of sight to each other? In such circumstances people still go for calling each other. That definitely is inconvenient and costs money. But people will choose to call since there is no other alternative. Is it possible for us to reduce the sting of such an unpleasant circumstance by at least removing the cost part?

It is possible to come up with basic software that allows two desktop computers to communicate with each other over wireless without using a wireless router. Such a network is called an ad hoc wireless network. Such a network would be free to its users since the only infrastructure in place is that of the two computers participating in the communication. Since computers can do this, theoretically smartphones should be able to accomplish the same. The purpose of this project is to see whether this theory can work in the real world.

Should the aforementioned theory be proved then this project will go on to try send and receive audio data between the connected smartphones, thus achieving the basic requirements of a phone call.

2 Literature Review

This literature review will focus on the following items.

1. Three wireless technologies — NFC, Bluetooth, and WiFi;

2. Two peer-to-peer technologies — SMPP and WiFi P2P;
3. Two audio encoding techniques — AMR-NB and AMR-WB; and
4. Two audio file formats — MP3 and AAC.

2.1 Wireless Technologies

A number of wireless technologies were mentioned in the project proposal. These were:

1. Near Field Communication(NFC) technology;
2. Wireless Fidelity (Wi-Fi) technology; and
3. Bluetooth technology.

These technologies will be considered to some detail below.

1. **Near Field Communication(NFC) technology**

   According to a programmer's guide to Android (Deitel et al. 2012, p. 11), Near Field Communication, or NFC, is a short-range radio frequency (RF) wireless connectivity standard that enables communication between two devices. It can also be used between a device and a tag — which stores data that can be read by NFC-enabled devices. NFC operates within a range of a few centimetres. NFC-enabled gadgets can operate in three modes:

   (a) **Reader/writer** — such as when a device reads data from a tag (Deitel et al. 2012);

   (b) **Peer to peer** — where devices exchange information without involving a third party server (Deitel et al. 2012); and

   (c) **Card emulation** — where devices act like smart cards, accomplishing various smart card operations (Deitel et al. 2012).

   Currently, Android devices support reader/writer and peer-to-peer NFC modes.

   According to the ISO NFC standard governing NFC protocols (ISO/IEC-18092 2013), NFC devices can have one of the following roles in an NFC network:

   (a) **Initiator** — that is, the generator of the RF field in which NFC signals will be passed between the communicating devices. The Initiator is also the starter of NFC communication.

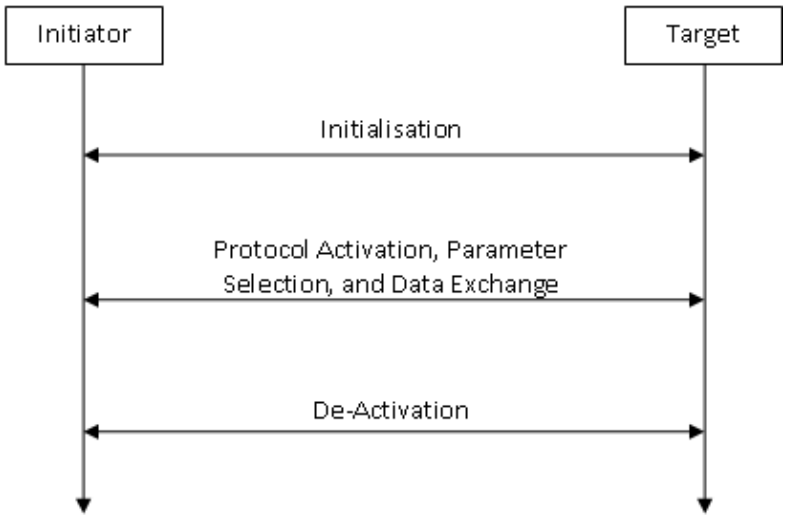

Figure 1: NFC Communication Procedures

(b) **Target** — which responds to Initiator commands either using RF generated by the Initiator through a method is called the load modulation scheme; or using modulation of an RF field generated by the Target itself.

The same ISO standard (ISO/IEC-18092 2013) defines two modes the Initiator and Target devices can communicate with:

(a) **Active communication mode** — in which both the Initiator and the Target devices use their own RF field to enable communication; and

(b) **Passive communication mode** —- where the Initiator generates the RF field and the Target responds to an Initiator command in a load modulation scheme.

NFC Targets and Initiators usually have an implementation of both the Active and Passive communication modes.

According to the aforementioned standard (ISO/IEC-18092 2013),transactions between NFC devices start with device initialization. Initiators select one of three bit rates ($\frac{frequencyofRFfield}{128}$, $\frac{frequencyofRFfield}{64}$, or $\frac{frequencyofRFfield}{32}$ bits) to start the transaction. Initiators may change the bit rate in the middle of the transaction using certain commands. However, communication modes — Active or Passive — cannot be changed during one transaction.

Figure 1 gives a simple illustration how NFC communication usually happens. The figure is a simple one and leaves out some details of what is

involved during each step of NFC communication since a lot of smaller processes come into play during NFC setup and teardown. The following list touches just a few of these processes:

- **RF collision avoidance** – which checks to ensure an Initiator communicates with only one Target at a time.
- **The Single Device Detection (SDD) algorithm** – which is used by the Initiator to detect one out of several Targets in the Initiator's RF field.
- **Active or Passive mode activation** – which is done using the Attribute Request and Attribute Response messages. (Both messages are called ATRs) Activation also involves the NFCID3 – a random ID used to finish the transport protocol activation process.
- **Parameter selection** – which is done using the Parameter Selection Request (PSL_REQ) and Parameter Selection Response (PSL_RES) commands.
- **Data Exchange Protocol selection** – which is done using the Data Exchange Protocol Request (DEP_REQ) and the Data Exchange Protocol Response (DEP_RES) commands.
- **Deselection and release** – which are done during device deactivation.

According to an article in the April 2012 issue of the International Journal of Advanced Research in Computer Science and Software Engineering (Preethi, Sinha, & Varma 2012), NFC has a range of up to 10 centimetres. (This translates to roughly 4 inches)

The NFC Forum, which champions NFC technology, was founded in 2004 but had to wait until 2006 before NFC tags came on the scene.

NFC-enabled devices include, but are not limited to, credit cards, smart posters, smart phones, and even on some computers.

The following are two advantages of using NFC;

- NFC provides security since its range is quite small. Piggybacking – which, according to a Computer Science journal article (Arul Oli, 2013), is the situation where unauthorized devices can access a wireless network by virtue of being within the operating range of that network – is almost impossible with NFC. This is because NFC operates within a very small range. Intruders would have to be very close to the victim devices to access them via NFC.

- NFC helps make device use intuitive. In English, "communicate" can mean "get in touch." NFC helps two devices communicate by getting in touch. The concept is thus instinctive and therefore easy to adapt to daily life.

The following are two disadvantages of NFC;

- The NFC technology is relatively new. It is not common. Anecdotally, relatively few people have NFC enabled smart phones in Kenya.
- NFC can only transfer small quantities of data. It does not work well with transfer of data in the millions of bytes. This is because NFC has a relatively small maximum transfer rate of 424 kilobits per second. (Preethi, Sinha, & Varma, 2012)

2. **Bluetooth technology**

A study of Bluetooth (Singh, Sharma, & Agrawal, 2011) defined Bluetooth as a wireless communication protocol aimed at low-powered, short range applications. It was initially developed by Ericsson but is now governed by the Bluetooth Special Interest Group (SIG). Initially, it was proposed as a technology to replace cables among computer components — think of a computer's monitor, motherboard, mouse, and keyboard working seamlessly without having to be connected via physical cabling. Bluetooth has grown past that goal — in part due to its low power consumption and potential low cost.

The Bluetooth we know today started in Scandinavia around 1996 when a certain Jim Kardach developed a system to allow mobile phones to communicate with computers. The name Bluetooth is based on the tenth-century Scandinavian king known in English as Harald Bluetooth. He united the whole of Denmark, achieving with the Danes what Kardach and his colleagues intended to with computers and cell phones.

According to a survey on Bluetooth security, (Ibn Minar & Tarique, 2012), Bluetooth was officially approved in the summer of 1999. Since then, the Bluetooth SIG has grown to have over 14,00 members, including some leading companies in telecommunications, computing, automotive, music, industrial automation, and network industries.

The same survey noted that Bluetooth is a combination of both hardware and software. On the one hand, the hardware is placed on a radio chip. On the other hand, the main control and security protocols have been implemented as software.

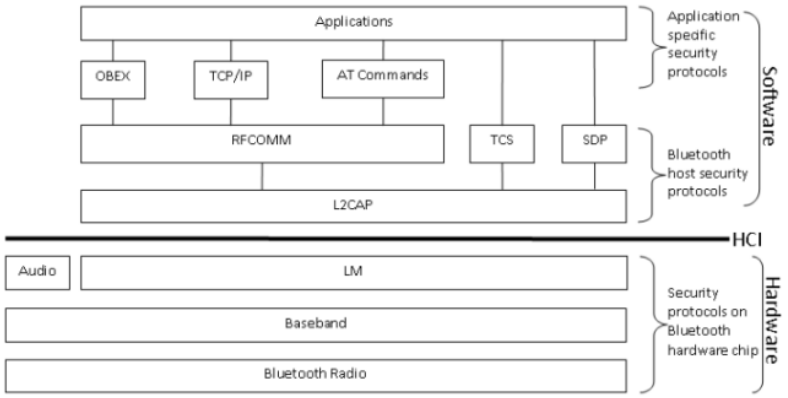

Figure 2: Bluetooth Protocol Stack

As per the previously mentioned Bluetooth security article, Bluetooth support involves both hardware and software. The hardware rides on a radio chip while software is used to implement control and security protocols. Using both hardware and software makes Bluetooth quite flexible. Over the next few paragraphs we will consider the hardware and software parts of Bluetooth.

Figure 2 – adapted from Ibn Minar and Tarique (2012) – shows the Bluetooth protocol stack. According to a study by those two (Ibn Minar & Tarique 2012), a protocol stack is a combination of software and hardware implementations of the actual protocols defined in a standard as well as a definition of how devices using a certain standard should communicate with each other based on the said standard. Figure 2 has some protocols above and below a comparatively thicker line labelled HCI. HCI stands for the Host Controller Interface. All protocols above the HCI line are included in the host's device software package. All protocols below the HCI are built into the Bluetooth microchip. The next few bullets discuss the Bluetooth protocol stack from the bottom up. They also define the abbreviations used in the figure.

- **Bluetooth radio.** It transmits data in the form of bits by using a RF. This functionality is defined in the radio layer. Bluetooth radio systems generally use the Gaussian Frequency Shift Keying (GFSK) technique to transmit and receive RFs.
- **Baseband.** This layer does frequency hopping for interference mitigation, medium access control, and data packet formation. In addition, the Baseband layer also control link, channel, and error correction and flow control.

- **Link Manager (LM).** This layer acts as a go-between for the application and the link controller in the local Bluetooth device. Remember, the Baseband layer does link control.
- **Audio.** The audio layer is almost on the same level with the LM layer. However, Audio is separated from LM so as to avoid the overhead of upper layer protocols. This is important since the Audio layer hosts protocols used to provide real time two way voice communication. The separation of Audio from LM ensures voice communication does to experience lag due to LM protocols.
- **Logical Link Control and Adaptation Protocol (L2CAP).** This protocol is on a layer of its own. It normally resides on the host. L2CAP acts as a conduit for data on the connection link between the Baseband and host applications. L2CAP is used to ensure both connection-oriented and connection-less services. This protocol also initiates security services for any Bluetooth communication session.
- **Radio Frequency COMMunication (RFCOMM).** This is a transport protocol that is used to emulate RS-232 serial ports. It enables Bluetooth devices to connect with external gadgets such as printers and scanners.
- **Telephony Control Specification (TCS).** This protocol defines the call control signalling needed for the establishment and/or release of speech and data calls between Bluetooth devices. It also proved functionality for exchanging signalling information that is not related to ongoing calls.
- **Service Discovery Protocol (SDP).** This protocol is essential since it discovers the Bluetooth services available within the RF proximity and determines the characteristics of the available services. SDP is what allows Bluetooth devices to form ad-hoc, or peer-to-peer, networks.
- **Object EXchange Protocol (OBEX).** OBEX is used to exchange objects between Bluetooth devices. These objects include calendar notes, business cards, and data files. The exchange is done based on a client-server model.
- **Transport Control Protocol/Internet Protocol (TCP/IP).** This well-known protocol provides a reliable stream of data to Bluetooth applications from the RFCOMM layer.
- **ATtention (AT) commands.** These are not protocols as such but

are a set of commands used in general telecommunications to produce commands for management of communication sessions.

- **Applications.** These are the Bluetooth applications used by end users.

Data from a sending Bluetooth device traverses the protocol stack from top to bottom – changing from intelligible information to bits. At the receiver's end, the data traverses the protocol stack from bottom to top – changing from ones and zeros to data the end user can understand.

Bluetooth operates within the Industrial, Scientific, and Medical (ISM) RF band. This section of the electromagnetic spectrum ranges from 2,400 to 2,483.5 MHz and is divided into 79 channels, each with a bandwidth of 1 MHz. Since the ISM band is also home to other technologies such as microwaves and Wi-Fi, it is possible that Bluetooth communication may get some interference. To avoid this, Bluetooth interfaces employ frequency hopping every few seconds. This ensures that if one channel among the 79 has interference, data can be re-sent via another channel that will likely not have interference. Bluetooth uses a hopping rate of 1600 hops per second. Its developers decided to use the Frequency Hopping Spread Spectrum (FHSS) channel management technique where sender and receiver are synchronized to know which channels they will be hopping to at any given moment during their communication. FHSS leads to efficient channel use and is not affected by the distance between sender and receiver. This is unlike the other common channel management method: the Direct Sequence Spread Spectrum (DSSS) technique.

Bluetooth usually operates on a 'master-slave' concept. The master device works as the moderator during communication between itself and the slave as well as among slaves themselves. For devices to connect to each other, Bluetooth demands that the two share secret codes referred to as PINs. Successful PIN exchange leads to two devices being connected over Bluetooth — a process referred to as 'pairing.'

Bluetooth has a range of up to around 30 metres. (Preethi, Sinha, & Varma, 2012)

The previously mentioned Bluetooth study and Bluetooth security survey — (Ibn Minar & Tarique 2012) and (Singh, Sharma, & Agrawal 2011) respectively — mention the following as some of the gadgets in which Bluetooth technology has been implemented: mobile phones, game controllers, Personal Digital Assistants(PDAs), personal computers, laptops, keyboards, mice, printers, scanners, notebooks, palmtops, cameras, and DVD players.

Here are two advantages of using Bluetooth:

- Bluetooth is quite flexible since it operates on both the hardware and the software level. As mentioned earlier, Bluetooth rides on a radio chip and has its control and security implemented in code.
- Bluetooth is quite common in smart phones within Kenya.

Bluetooth has some disadvantages. These include, but are not restricted to:

- The technology having a rather small range of operation. While common Bluetooth covers a larger distance than, say, NFC, it is not an ideal solution for wireless communication over 30 metres.
- Security issues since it is vulnerable to sniffing and information leaks.

3. **Wireless Fidelity (Wi-Fi) technology.**

Wireless Fidelity technology, commonly known as WiFi, is a communication technology known to many. A study on Wi-Fi (Song & Isaac, 2014) defines it as the IEEE 802.11x standard and a short-range wireless transmission technology. The study further tells that Wi-Fi is a brand held by the WiFi Alliance, whose purpose is to improve interoperability between wireless network products based on the IEEE 802.11 standard.

The concept of wireless access networks emerged in the late 1980s as a by-product of cellular wireless technology. As the demand for cellular service grew exponentially, the cost of wireless network components decreased, while the cost of setting up and maintaining conventional copper-based subscriber networks increased. (Skariah & Suriyakala, 2013) It was time for a wireless technology to enter the communications foray. The initial Wi-Fi standard, 802.11, was released in 1997. It was improved to 802.11a in 1999. (Song & Isaac, 2014) From then on various enhancements have been introduced in the form of new standards such as IEEE 802.11b, IEEE 802.11g, and IEEE 802.11n. The .11n standard is the most common nowadays. It operates within both the 2.4 GHz and 5 GHz frequency range with speeds of 400 to 600 Mbps. (Song & Isaac, 2014)

Wi-Fi operates within the unlicensed radio band between 2.4 and 5 GHz. All Wi-Fi networks use contention-based Half Duplex Time Division Duplex (TDD) techniques. TDD involves vying for shared media. All devices in a Wi-Fi network attempt to use shared media (the air) at specific time intervals. Because of this operation, Wi-Fi network devices can only send or receive data at one moment. Thus they are half duplex. The contention based nature of WiFi can cause subscriber devices far from an AP to be repeatedly

interrupted by closer devices. This makes services such as Voice over Internet Protocol (VoIP) or Internet Protocol Television (IPTV) – which depend on an essential constant Quality of Service (QoS) – difficult to maintain for more than a few devices. (Skariah & Suriyakala, 2013)

To reduce the limitations imposed by half duplex communication, the 802.11n WiFi standard – a common Wi-Fi standard – optimizes technology found in the physical and MAC layers of the Open Systems Interconnection (OSI) model. It does this by implementing features such as Multiple Input Multiple Output (MIMO) and MIMO-Orthogonal Frequency Division Multiplexing (MIMO-OFDM), 40 MHz channels, and short guard intervals. These combined optimizations result in enhanced throughput of up to 600 MHz for Wi-Fi-based wireless local area networks (WLANs). (Song & Isaac, 2014)

Wi-Fi uses either Direct Sequence Spread Spectrum (DSSS) or Orthogonal Frequency Division Multiplexing (OFDM) to manage the channels allocated to it in the radio band it uses. (Skariah & Suriyakala, 2013) OFDM is the more favoured of the channel management technologies. This is because it offers high-speed transmission rates. OFDM takes a given frequency domain and divides it into orthogonal sub-channels. Each sub-channel uses a sub-carrier to modulate signals, and each sub-carrier performs transmission parallel to other sub-channels. (Song & Isaac, 2014)

Speaking of channels, Wi-Fi standards define a fixed channel bandwidth of 25 MHz for 802.11b and 20 MHz for either 802.11a or g.

Skariah and Suriyakala, (Skariah & Suriyakala, 2013), say that modulation of bit streams across WiFi networks is done using some of the following methods:

- **Quadrature Phase Shift Keying. (QPSK)** This is used mostly in the 802.11b standard.
- **Binary Phase Shift Keying (BPSK)**, used by 802.11a and g.
- **Quadrature Amplitude Modulation (QAM)**, which comes in two flavours — 16-QAM and 64-QAM — and is used by 802.11a and g.

Wi-Fi operates by having an access point (AP, also known as a hotspot) which emits Wi-Fi signals. Devices desiring to connect to a Wi-Fi network send their requests to that network's AP. A series of handshakes takes place which mostly involve authentication. Finally, the connecting device is issued with data that will enable it to connect to the said AP.

Figure 3 shows a simple example of how Wi-Fi networks can be deployed.

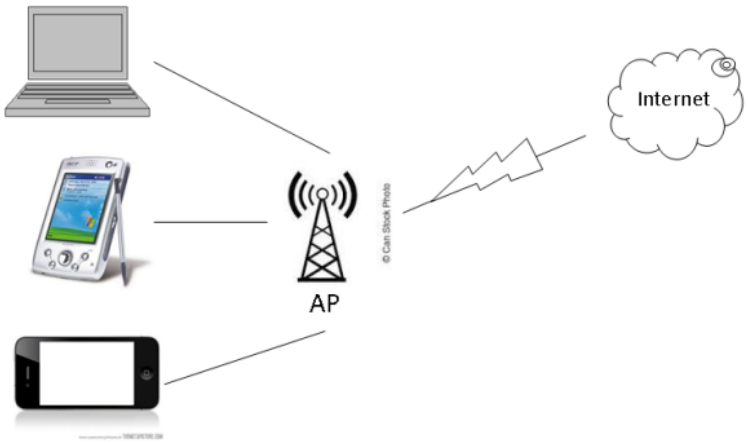

Figure 3: A Simple WiFi Deployment

Table 1 shows how the five Wi-Fi standards mentioned so far — 802.11, 802.11a, 802.11b, 802.11g, and 802.11n — compare.

Various studies ((Song & Isaac, 2014), (Skariah & Suriyakala, 2013), and (Banerji & Chowdhury, 2013)), have found that Wi-Fi can is present in devices such as personal computers, video game consoles, smart phones, tablets, printers, PDAs, and routers.

Some advantages of Wi-Fi are:

- It has the longest range of the four common wireless networks referred to here.
- Wi-Fi is a feature in almost all smart phones.

Some disadvantages of Wi-Fi are:

- Wi-Fi is inherently insecure. Because of its huge operation range, piggybacking is very possible — and piggybacking could lead to data sniffing. This could result in a breach in security.
- Wi-Fi has the highest power draw of the mentioned technologies. Some estimate Wi-Fi to use as much as 40 times more power than Bluetooth.

2.2 Peer-To-Peer Technologies

Two peer-to-peer (P2P) technologies will be considered:

1. Short Message Peer to Peer (SMPP) protocol; and

Table 1: Comparing 802.11 standards

IEEE Standard	**Maximum Speed (Megabytes per second)**	**Frequency (GigaHertz)**	**Backward Compatible with**
802.11	2	2.4	-
802.11a	54	5	-
802.11b	11	2.4	-
802.11g	54	2.4	802.11b
802.11n	600	2.4 and 5	802.11a/b/g

2. Wi-Fi Direct.

Their consideration is here below.

1. **Short Message Peer to Peer (SMPP) protocol**

   SMPP is a Short Message Service (SMS) protocol that is used to send messages over a TCP/IP connection. It is an open, industry standard protocol designed to offer a flexible data communication interface for the transfer of SMS data between a Message Centre (which acts as a store for SMSes) and a SMS application system (such as a system that sends bulk SMSes to subscribers). Examples of SMS application systems include External Short Message Entities (ESMEs), Routing Entities (REs), and Message Centres (MCs). As mentioned earlier, SMPP transmits messages TCP/IP. The IP link used for this can either be a leased line or the Internet. SMPP has no security measures specified, and this allows fast delivery of bulk SMSes. (Samanta, Mohandas, & Pais, 2012) However, this is one of its major draw-

backs – and will be discussed a bit later.

Note: an ESME in this context of this letter refers to external sources and sinks of short messages. Such sources and sinks include Voice Processing Systems, Wireless Application Protocol (WAP) Proxy Servers, or Message Handling computers. In this document, ESME excludes Short Message Entities (SMEs) – which are located within the mobile network. An example of an SME is a Mobile Station (MS), commonly known as a mobile phone. SMS first appeared in Europe in 1992. It was included in Global System for Mobile (GSM) communications right from GSM's beginning. SMS was later ported to wireless technologies such as Code Division Multiple Access (CDMA) and Time Division Multiple Access (TDMA). A standard SMS message should be a maximum 160 characters long if each character has 7 bits (which is suitable for encoding Latin characters such as English alphabets); or a maximum 70 characters long if each character has 16 bits(suitable for encoding Unicode Universal Character Set (UCS) 2 characters such as Chinese characters). (Samanta, Mohandas, & Pais, 2012)

SMS based services have proliferated in the past few years. These services include mobile banking, delivery services, airtime status checks, and mobile ticketing. Let us take an example of a SMS sent by a user enquiring after the status of their airtime balance. The user sends a message to 144.

```
User SMS: "Balance"
```

The SMS leaves from the MS via the GSM network to the SMS Center (SMSC). The SMSC serves as the point at which all SMSes sent in a mobile network arrive for processing. The SMSC sends SMSes using the "store and forward" mechanism which involves receiving a message, storing it for some time while determining its intended recipient, then forwarding the message to the identified recipient. The SMS Centre forwards the SMS to the ESME with the destination unique number "144". At the ESME that has number "144", the message is parsed and checked for a matching query and a response is found. This response is forwarded to the user's MS using the path ESME to SMSC to MS. The user now knows their airtime balance. (Samanta, Mohandas, & Pais, 2012) But where is SMPP involved in this?

As mentioned in the outset, SMPP is what is used when an ESME wants to interact with the SMSC. To make such communication happen, an ESME first establishes a session then communication between ESME and SMSC is done over this session. This communication is performed usually over a TCP/IP or an X.25 connection. For TCP/IP, application port 2775 is the default port assigned for SMPP. (Samanta, Mohandas, & Pais, 2012)

Operations over SMPP can be categorized into four groups:

- **Session Management:** These operations assist in the setting up of an SMPP session between an ESME and the SMSC. Operations here also provide error handling functionalities.
- **Message Submission:** This set of operations allows an ESME to submit messages to the SMSC.
- **Message Delivery:** This set of operations allows the SMSC to submit messages to an ESME. They do the inverse of the Message Submission operations.
- **Ancillary Operations:** The operations provide a set of additional features such as cancellation queries or message replacements. (Samanta, Mohandas, & Pais, 2012)

ESMEs and SMSCs interact with each other by exchanging commands. Some of the important commands the two entities exchange with each other are:

- *bind*. The purpose of this operation is to register an instance of an ESME with the SMSC system and request an SMPP session with the SMSC over a specified network connection. (Samanta, Mohandas, & Pais, 2012)
- *submit_sm*. This operation is only used by an ESME to submit a short message to the SMSC for onward transmission to a specified SME. (SMSForum, 2002)
- *deliver_sm*. This operation is used when an ESME wants to send message data to the SMSC. (Samanta, Mohandas, & Pais, 2012)
- *data_sm*. This operation is used to transfer data between a SMSC and an ESME. The *data_sm* operation is an alternative to the *submit_sm* and *deliver_sm* operations. (SMSForum, 2002)

Figure 4, gotten from study of SMPP security flaws (Samanta, Mohandas, & Pais, 2012) illustrates how SMPP messages flow.

As noted earlier, SMPP is used among Mobile Stations, Routing Entities, SMS Centres, External Short Message Entities, Voice Processing Systems, Wireless Application Protocol (WAP) Proxy Servers, and Message Handling computers. All these are found within GSM networks.

Some of the advantages of SMPP are:

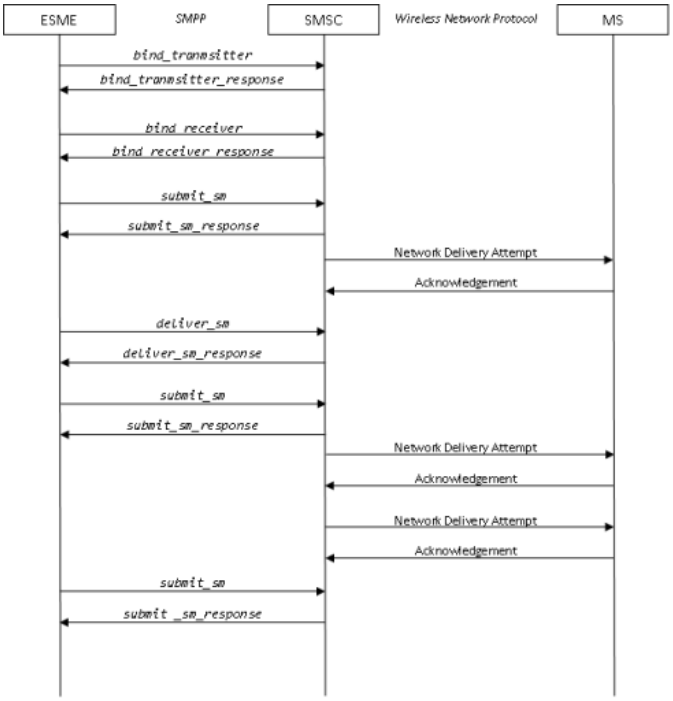

Figure 4: SMPP Message Flow

- It is an open, accessible standard. It is also being actively supported by its originators. (SMSForum, 2002)
- It is flexible and thus can take care of various consumer SMS services. (SMSForum, 2002)
- It works over the TCP/IP suite therefore it is not limited to GSM networks. (Samanta, Mohandas, & Pais, 2012)
- SMPP allows the sending of messages in bulk. (Samanta, Mohandas, & Pais, 2012)

Some of the disadvantages of SMPP are:

- It uses traditional SMS technology which does not provide security. Traditional SMS sends messages as plain text. (Samanta, Mohandas, & Pais, 2012)
- SMPP is usually implemented in the Application layer of the TCP/IP suite therefore it assumes that reliability will be maintained in lower layers of the suite. Assumptions are risky. (Samanta, Mohandas, & Pais, 2012)

- SMPP is vulnerable to a Man-In-The-Middle attack because of its lack of end to end authentication. (Samanta, Mohandas, & Pais, 2012)
- Messages sent via SMPP run the risk of being tampered with because of their being plaintext. (Samanta, Mohandas, & Pais, 2012)

2. **Wi-Fi Direct**

   According to its white paper (Wi-Fi Alliance, 2010) Wi-Fi Direct is a game changing new technology that enables Wi-Fi devices to connect directly, making it simple and convenient to do things like print, share, sync, and display. Products that have the Wi-Fi Direct functionality can be identified by looking for the Wi-Fi CERTIFIED Wi-Fi Direct designation. Such devices can connect to each other without having to join a traditional home, office, or hotspot network connection. And overview of this technology in an IEEE journal (Camps-Mur, Garcia-Saavedra, & Serrano, 2013) states that WiFi-Direct involved enabling device to device connectivity without requiring the presence of an AP. Device to device connectivity has been possible within the 802.11 standard mentioned in the Wi-Fi section of this literature review. Such connectivity would be done by means of the ad-hoc operation mode. (Camps-Mur, Garcia-Saavedra, & Serrano, 2013) However, ad-hoc has some challenges, such as complicated setup processes, inefficient power use, lack of extended QoS services. Wi-Fi Direct addresses these challenges and also provides a seamless way for devices with older technology to connect with Wi-Fi Direct. (Camps-Mur, Garcia-Saavedra, & Serrano, 2013)

   The Wi-Fi Direct white paper (Wi-Fi Alliance, 2010) – which has already been quoted in this section and will continue to be referenced below – was published in October 2010. It is therefore assumed that the Wi-Fi Direct technology was released around October 2010. In a traditional Wi-Fi network, clients look for and connect to WLANs, which are created and announced by APs. This creates a clear distinction between WLAN AP and WLAN client. Each of those WLAN components has distinctly different roles and functionalities. Within Wi-Fi Direct, however, things are different. The AP and client roles are specified dynamically since Wi-Fi Direct works as a peer to peer connection between two devices over a shared Wi-Fi connection. As a result a Wi-Fi Direct device has to implement both the AP and the client role. These roles cease from being physical ones to being logical ones. A device can even run both roles at the same time, such as a moment when a device acts as the originator of a Wi-Fi Direct connection while still accessing a different Wi-Fi Direct connection. The simultaneous execution of roles can be implemented by using different frequencies (if the device has

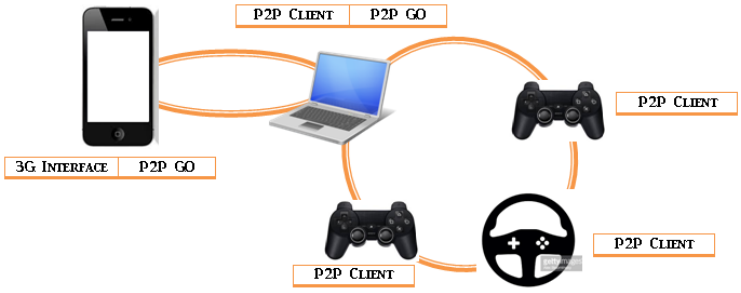

Figure 5: A P2P Deployment

a number of physical radios) or time-sharing a single radio channel. (Camps-Mur, Garcia-Saavedra, & Serrano, 2013) But how do Wi-Fi Direct gadgets decide which gadget takes on which role?

Wi-Fi Direct devices, or, more formally, **P2P devices**, come to a consensus on which roles to take. In order to understand how this consensus comes about, we need to first understand the structure of a Wi-Fi Direct network. This will be explained in the following paragraphs: P2P devices communicate by establishing **P2P Groups**, which perform the same work as traditional Wi-Fi infrastructure networks – that is, P2P Groups have a device implementing something resembling a WLAN AP as well as devices that act in a manner resembling WLAN clients. The device implementing the AP is called the **P2P Group Owner (P2P GO)** while the clients are called **P2P Clients**. As mentioned in the last few paragraphs, these roles are dynamic, therefore when 2 P2P devices discover each other they **negotiate** their roles – P2P GO or P2P Client – before the P2P Group is set up. Once the Group is established, new P2P Clients can join the group just the same way devices can join a WLAN. Figure 5 illustrates how P2P devices can be deployed. In Figure 5, all the devices have Wi-Fi Direct. We assume that the users of these devices would wish to play a game on the Internet with each other. The smartphone has a 3rd Generation (3G) interface with which it connects to the Internet. The smartphone negotiates with the laptop and they both decide the former will be the P2P GO and the latter will be a P2P Client. The users connect their gaming devices to the laptop – during which process the laptop becomes the P2P GO and the gaming gadgets the P2P Clients. This figure clearly shows that it is possible for a device to act both as a P2P GO and a P2P Client.

Legacy devices – Wi-Fi enabled devices that do not have the Wi-Fi Direct technology – do not formally belong to the P2P Group but can communicate with the P2P GO as long as they do not exclusively implement the 802.11b

standard and support the security measures specified by the P2P GO. Such legacy devices see the P2P GO as a standard issue WLAN AP. (Camps-Mur, Garcia-Saavedra, & Serrano, 2013)

Just as traditional APs use the Dynamic Host Configuration Protocol (DHCP) to give clients IP addresses, the P2P GO assigns P2P Clients IP addresses via DHCP.

There are at least three ways in which two devices can form P2P Groups (Camps-Mur, Garcia-Saavedra, & Serrano, 2013):

- **Standard**. This is the case where P2P devices first have to find each other, then negotiate over which of them will be P2P GO.
- **Autonomous**. This is the situation where a P2P Device creates a P2P Group on its own and immediately becomes its P2P GO. The device starts transmitting signals indicating its group's availability. Other gadgets can discover and connect to a group set up in this manner using the traditional Wi-Fi scanning, authentication, and address configuration method.
- **Persistent**. Wi-Fi Direct devices can set up a P2P Group and store network credentials and assigned roles such that the next time they set up the group, devices will already know which of them is GO and which are Clients. Such a P2P Group is called persistent.

For security, Wi-Fi Direct devices are needed to implement Wi-Fi Protected Setup (WPS). This setup method assures users of a safe connection with little intervention on their part. WPS establishes a secure connection using methods such as the introduction of a PIN in the P2P Client side, or a button push between two connecting P2P devices. WPS is based on the Wi-Fi Protected Access II (WPA2) which employs Advanced Encryption Standard Counter Block Chaining Message Authentication Protocol (AES-CCMP) for encryption and randomly generated Pre-Shared Keys (PSKs) for mutual authentication. (Camps-Mur, Garcia-Saavedra, & Serrano, 2013)

Devices which implement Wi-Fi Direct include smartphones, printers, monitors, cameras, gaming devices, digital photo frames, desktop computers, notebooks, and netbooks. (Camps-Mur, Garcia-Saavedra, & Serrano, 2013) (Wi-Fi Alliance, 2010) There also are open source implementations of the standard which can used on WLAN hardware such as Personal Computer Memory Card International Association (PCMCIA) cards. (Camps-Mur, Garcia-Saavedra, & Serrano, 2013)

Some of Wi-Fi Direct's advantages are:

- Wi-Fi Direct gives its users mobility and portability. (Jichkar, 2014)
- Wi-Fi Direct helps people use devices immediately, thus saving time. (Jichkar, 2014)
- Wi-Fi Direct is easy to use. Setting it up is quite hassle free. (Jichkar, 2014)
- Wi-Fi Direct provides simple, secure connections. (Jichkar, 2014) Wi-Fi Direct's disadvantages include:

Wi-Fi Direct's disadvantages include:

- As with all network connections, allowing Wi-Fi Direct links with untrusted devices can result in breaches of security. (Wi-Fi Alliance, 2010)
- Implementing both client- and server-side code makes Wi-Fi Direct a little heavier than traditional WLAN technology. (Camps-Mur, Garcia-Saavedra, & Serrano, 2013)
- Setting up P2P Groups calls for substantial communication between devices. This takes time.

2.3 Audio Encoding Techniques

Speech encoding is the process of compacting a speech signal so that it can be transmitted with a substantially smaller memory. (Choudhary & Kumar, 2014) Encoding is needed because space is one of the things we have only a finite amount of. Of course, speech that is encoded at its source will need to be decoded at its destination. Because of this, speech encoding tends to refer to the process of encoding and decoding speech. The word "codec" is used to denote an encoder and a decoder. Figure 6, based on an article about AMR coding (Choudhary & Kumar, 2014), is an abstraction of the speech encoding and decoding system.

Speech coding is a lossy kind of coding. This means that the output signal sounds close to the input but is not a one to one mapping of the input.

The most widely used type of speech coding is the Adaptive Multi-Rate (AMR) coding. (Choudhary & Kumar, 2014) It was adopted by the 3rd Generation Partnership Project (3GPP) in 1988. AMR uses the Algebraic Code Excited Linear Prediction (ACELP) algorithm for voice coding. It is an improvement of the Code Excited Linear Prediction (CELP) algorithm. CELP uses analysis by synthesis to encode voice by perceptually optimising the decoded signal is a closed loop system.

By doing this, CELP-based coders produce good quality output even with low bit rates. Its drawback is a signal delay of 50 milliseconds. This delay was addressed by ACELP, which uses specific algebraic structures in its codebooks to process input signals. The result is CELP-quality output with a signal delay of about 2 milliseconds. (Choudhary & Kumar, 2014)

Audio in AMR is further processed using:

- A **Voice Activity Detector (VAD)** which differentiates speech from silence;
- **Comfort Noise Generation (CNG)** which generates some background static on purpose to counter the effects of suddenly swinging from silence to speech; and
- **Discontinuous Transmission (DTX)** which controls the transmitter so that it does not use the battery during times of silence.

AMR technologies are used in GSM networks (Choudhary & Kumar, 2014), during conference calls, in Universal Terrestrial Radio Access Network (UTRAN) devices, and in Enhanced GSM Data Environment (EDGE) devices. (Birkehammar, Bruhn, Eneroth, Hellwig, & Johansson, 2006)

AMR comes in two forms, the latter being an improvement of the former. These forms are:

1. Adaptive Multi-Rate Narrowband (AMR-NB) encoding; and
2. Adaptive Multi-Rate Wideband (AMR-WB) encoding.

1. **Adaptive Multi-Rate Narrowband (AMR-NB) encoding**

   This coding technology was the first AMR one. It operates within a bandwidth of 200 – 3400 Hz and has a total of eight rates. (Choudhary & Kumar, 2014) It works in two rates:

   - **Full-rate** – with a bit rate of 22.8 Kilobits per second (Kbps) and eight of the eight rates available; or
   - **Half-rate** – with a bit rate of 11.4 Kbps and six of the eight rates available

   Some of the advantages of AMR-NB are:

   - It saves space and memory for long distance communications. (Choudhary & Kumar, 2014)

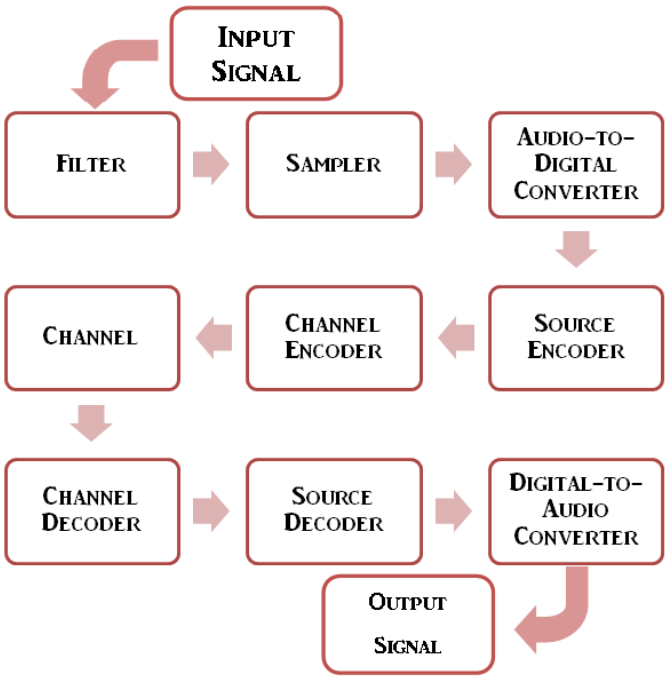

Figure 6: The Encoding Decoding Process

- It is supported by active standards bodies such as the European Telecommunications Standards Institute (ETSI) and the 3GPP. (ETSI, 2002) It has functionality to save phone battery life – what with VAD and DTX. (ETSI, 2002)

Some of AMR-NB's drawbacks are:

- It is vulnerable to frame stealing. (ETSI, 2002)
- Technologies such as VAD, DTX, and CNG demand space and processing power. (ETSI, 2002)
- Transmission errors can result in the loss of frames. (ETSI, 2002)

2. **Adaptive Multi-Rate Wideband (AMR-WB) encoding**

Wideband AMR was specified as an improvement to AMR-NB, AMR-WB extends mobile phone bandwidth from 200 – 3400 Hz to 50 – 7000 Hz. This results in an audio output of higher quality, intelligibility, and naturalness than that of the earlier technology. AMR-WB has been standardized by 3GPP for GSM and Wide Code Division Multiple Access (WCDMA) 3G systems. (Byun, Eo, Bum, & Minsoo, 2005)

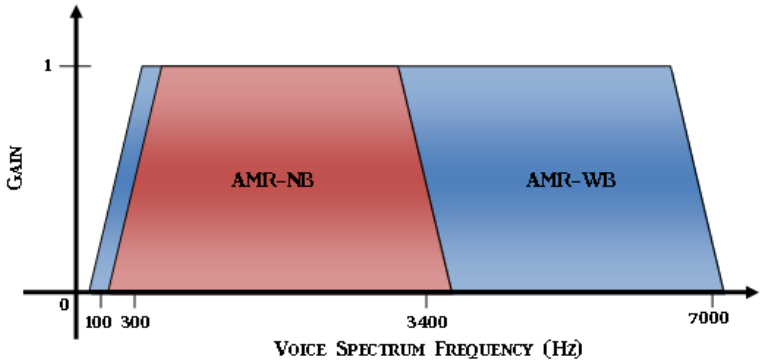

Figure 7: A Sketch Graph of the Pass band Characteristics of AMR Technologies

A lab test done by Ericsson showed that AMR-WB outperformed every AMR-NB mode up to 12.2 Kbps. Even in error-prone GSM channels the former technology was better than the latter. Ericsson thus believes the adoption of AMR-WB will result in positive changes in calling patterns. (Birkehammar, Bruhn, Eneroth, Hellwig, & Johansson, 2006)

AMR-WB operates similar to AMR-NB but has a greater bandwidth. It also offers more signal processing functionalities in the form of in-band signalling, fixed rate speech, link adaptation and fixed channel codec modes. It splits the available bandwidth into smaller sections thus reducing coding intricacies and increasing optimality. (Choudhary & Kumar, 2014)

AMR-WB within a GSM network operates based on one of two 3GPP standards (Birkehammar, Bruhn, Eneroth, Hellwig, & Johansson, 2006):

(a) **Tandem-Free Operation (TFO)**, an in band signalling protocol that allows voice codec parameters to pass unmodified through Pulse Code Modulation (PCM) links in traditional voice networks. TFO preserves quality but does not reduce transport bit rate within the network.

(b) **Transcoder-Free Operation (TrFO)**, which is a combination of out-of-band transcoder control (OoBTC) signalling and enhanced transport technology. This combination ensures that the voice signal encoding in the transmitting MS is transported without modification to the receiving MS. TrFO is initially more expensive to lay out but the OoBTC feature guarantees a higher success rate for AMR-WB calls.

Figure 7 shows a sketch graph of how AMR-NB and AMR-WB compare in terms of vocal spectrum frequencies. The figure clearly illustrates that Wideband coding capture more voice frequency than AMR-NB.

Apart from the advantages that come with being an AMR coding technology, AMR-WB's advantages include:

- A higher voice quality.
- It allows end users of mobile phones to have privacy, discretion, and comfort when making phone calls. (Birkehammar, Bruhn, Eneroth, Hellwig, & Johansson, 2006)
- It codes a higher voice range as shown in Figure 7.

Some of AMR-WB's pitfalls are:

- It is still relatively new in the phone communication industry. This is why TFO and TrFO are needed to incorporate AMR-WB into GSM networks. (Birkehammar, Bruhn, Eneroth, Hellwig, & Johansson, 2006)
- The improvements AMR-WB has over AMR-NB need more processing. (Choudhary & Kumar, 2014)

2.4 Audio File Formats

After audio is decoded, it needs to be played. Operating systems will need to know which files are audio files so as to play them. Audio material coded by Motion Picture Experts Group (MPEG) – that is, audio material having an .mp3 or an .aac extension – has spread widely in the Internet since 1995. Almost everyone on the planet has heard or is owning some audio file with the abovementioned extension. The initial MPEG audio file format was defined in 1992. Since then, research on audio file formats has increased tremendously, giving rise to new and better ways of saving high quality sound files in minimal space. (Brandenburg, 1999)

MPEG audio technologies are based on the principle of perceptual coding. This idea is based on psychoacoustics, which is the study of how the human mind responds to sound. It turns out that some inaudible edits of a piece of music do not affect how we hear it. Perceptual coding capitalizes on this phenomenon to produce music files whose file size has been compressed without compromising the sound of the music they store. It has to be noted that perceptual coding does not perform lossless compression – the result of perceptual encoding is a file that is smaller than the original. The catch is that the human ear will not get the difference between the original Song & its perceptually encoded form.

A perceptual audio codec is made up of:

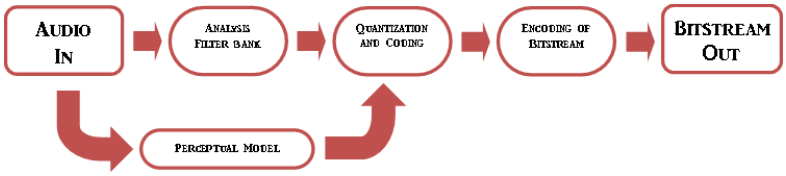

Figure 8: A Simple Perceptual Encoding System

Figure 9: A Simple Perceptual Decoding System

- A **filter bank** which analyses and decomposes the input signal into sub-sampled spectral components during encoding; and synthesises the spectral components into an audio signal in the decoder. (Brandenburg, 1999)
- A **perceptual model** which uses either the time domain input signal and/or the analysis filter bank to come up with an estimate of a masking threshold from rules defined by psychoacoustics. (Brandenburg, 1999) The masking threshold is the threshold above which changes in the audio file will be noticeable by human ears. The perceptual model is only found in the encoding section since it is only the input audio that needs to be encoded perceptually.
- **Quantization and coding** mechanisms. Quantization is the process of converting the sampled values of an audio signal into bit representations. Quantization introduces some noise in the spectral components. Coding involves assigning each of the bit representations gotten from quantization a binary code. Coding is usually done to keep quantization noise below the masking threshold. In the encoder the spectral components are quantized and then coded. In the decoder, the inverse is done. The input bitstream will be decoded (the inverse of the coding just defined) and then it will undergo inverse quantization to produce the corresponding spectral components.
- **Encoding and decoding** mechanisms. In the encoder, a bitstream formatter is used to arrange the bits gotten from quantized and coded spectral components. The assembled bits are then sent out of the perceptual encoding system. In the decoder, a bitstream is received, decoded into its quantized form, and released for inverse quantization.

Figures 8 and 9 show how a perceptual coding system encodes and decodes audio data respectively.

MPEG audio file formats are used to process music in audio players, mobile phones

(Mehta & Kharote, 2014), High Definition Television (HDTV), videos (Brandenburg, 1999), personal computers (Youngseok & Jongweon, 2014), digital homes, streaming applications, and the Internet (Geiger, et al., 2007)

We will consider two common MPEG audio file formats in this letter:

1. Motion Picture Experts Group-1/2 Layer-3 (MP3); and
2. Advanced Audio Coding. (AAC)

1. **Motion Picture Experts Group-1/2 Layer-3 (MP3)**

   This standard defined a data representation having a number of options (Brandenburg, 1999) such as:

   - **Operation mode**. This enables MP3 audio to work in both mono and stereo mode.
   - **Sampling rates**. This helps MP3 audio to work on a couple of sampling frequencies such as 44.1 KHz for MPEG-1 and 22.05 KHz for MPEG-2.
   - **Bit rate**. This option allows the implementers of MP3 standards to decide the bit rate of the audio compressed by their implementation of the standard.

   MP3 encoding and decoding follows the perceptual model. Below are some details about how MP3 encodes audio (Brandenburg, 1999).

   (a) The **filter bank** is built by cascading two kinds of filter banks – the **polyphase filter bank** and then a **Modified Discrete Cosine Transform (MDCT)**. The polyphase filter bank is what is used in MPEG-1/2 Layer-1 and MPEG-1/2 Layer-2. It is used in Layer-3 to make it a little similar to the previous two layers. The MDCT ensures the audio data is stored in an overlapping manner so as to save space.

   (b) The **perceptual model** is what determines the quality of the encoder implementation by setting the **level of noise allowed** for each partition of encoded audio.

   (c) Quantization is done using a **power-law quantizer** and coding is done using **Huffman coding**. Before quantization is done for a given audio data block from the perceptual model, the optimum gain and noise control for that data block are determined using two loops, one inside the other.

- The inner loop determines the quantization rate by adjusting the number of bits resulting from a coding operation until the resulting bits are equal to the number of bits allowable for each coded audio data block. This loop is called the **rate loop**.
- The outer loop shapes the quantization noise according to the masking threshold set by the perceptual model. Each data block has a scale factor attached to it. Each scale factor starts off with its value being 1.0. If the quantization noise for a particular data block is found to be higher than the threshold then that block's scale factor is adjusted to reduce the quantization noise. Since getting a smaller quantization noise needs a larger number of quantization steps and thus a higher bit-rate, the rate loop has to be repeated every time scale factors change. The outer loop manages the quantization noise and is thus called the **noise control loop**.

(d) At this point the **bitstream** is funnelled **out** of the encoder. Possible destinations include a remote audio output device or a local storage device.

When an MP3 bitstream gets to a decoder, it contains a sequence of data frames put one after another. Each frame corresponds to two granules, or long blocks, of audio. Each granule has exactly 576 consecutive audio samples. A granule may be further divided into three short blocks which have exactly 192 samples each. The following few steps explain how MP3 decoding works. (Mehta & Kharote, 2014)

(a) **Synchronization** of the decoder **with the start** of the MP3 **frame**. This is done to decode MP3 header information.

(b) **Decoding** of MP3 **side information**. This side information is found on the side of MP3 audio data and may contain scale factor selection information and block splitting information.

(c) **Decoding** of the main data for each granule. Main data includes **Huffman bits** and **scale factors**.

(d) **Inverse quantization of transform coefficients**. In the case of short blocks, transform coefficients may be re-ordered and divided into three sets of coefficients, one set for each block. An alias reduction is done for long blocks.

(e) The **Inverse Modified Discrete Cosine Transform (IMDCT)**, the inverse of the MDCT done during encoding, is applied for the transform

coefficients acquired in step 4 of decoding.

(f) The inverse poly-phase filter bank, the inverse of the encoding poly-phase filter bank, is applied to IMDCT's output to produce a full-bandwidth signal.

MP3 has some advantages for those who use it. These include:

- It is a very common audio file format. (Youngseok & Jongweon, 2014)
- It produces audio files that are small in size and good in quality. (Mehta & Kharote, 2014)
- It is an open standard and can therefore be implemented by anyone with the right skills.

MP3's shortfalls consist of:

- Since MP3 uses perceptual coding system, it leaves out some data from audio files coded by it. (Mehta & Kharote, 2014) This may lead to an overall loss of quality with repeated MP3 encoding.
- The small size of MP3 files has contributed to a lot of illegal possession of MP3-file music. (Brandenburg, 1999)
- The rate and noise control loops need tuning so as to avoid indefinite looping. (Brandenburg, 1999)

2. **Advanced Audio Coding (AAC)**

AAC emerged as the spiritual successor of the very successful MP3 file format. It has been called the new all-round coder to take the mantle from MP3. (Herre & Dietz, 2008) AAC has been developed to support functionalities such as scalability, low-delay operation, and lossless signal representation. (Herre & Dietz, 2008) The original version of AAC was published in 1997 and finalized in 1999. (Herre & Dietz, 2008)

To encode audio, AAC does the following:

(a) It gets a PCM digital audio signal.

(b) It uses a **MDCT filter bank** to transform the PCM signal into a spectral representation. This representation will be used to apply psychoacoustic principles and redundancy reduction algorithms. AAC uses a 1,024 spectral line MDCT filter bank to create spectra corresponding to 1,024 PCM input signals. (Geiger, et al., 2007)

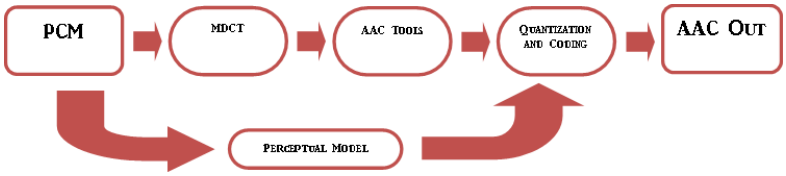

Figure 10: A Simplified Structure of an AAC Encoder

Figure 11: A Simplified Structure of an AAC Decoder

(c) Quantization and coding processes in AAC are similar to those found in MP3.

(d) Before the audio signal is fully converted into bits, it will have passed through some new tools unique to AAC. These tools include:

- **Temporal Noise Shaping (TNS)**. This tool allows AAC to shape quantization noise by doing an open loop prediction along the input signal's frequency domain. This tool especially helps AAC to improve output quality at low bit rates. (Brandenburg, 1999)
- **Block switching**. Instead of using MP3's cascading filter bank, AAC uses a standard switched MDCT filter bank with an impulse response (for short blocks) of 5.3 milliseconds at a 48 KHz sampling frequency. This is better than MP3's 18.6 short block impulse response. (Brandenburg, 1999)

AAC decoding happens as follows:

(a) The bitstream is received by the codec.

(b) It goes through **inverse quantization** and then **decoding**.

(c) Any **AAC tools** applied to the bitstream are **reversed**. Error mapping is also done at this point.

(d) The inversely quantized, decoded bitstream is passed through the **IMDCT**.

(e) A full audio signal is produced.

Figures 10 and 11 show a simplified AAC encoder and decoder respectively. They attempt put into pictures what the above few lines have put in words.

In addition to the advantages brought about by being based on the MP3 standard, some advantages of using AAC codecs are:

- AAC encoding and decoding is flexible. That has helped to develop more refined forms of AAC codecs from the basic AAC model. (Geiger, et al., 2007)
- AAC has a higher coding efficiency than MP3 due to the use of prediction. (Brandenburg, 1999)
- They have near-lossless audio representation. (Geiger, et al., 2007)

Since AAC inherits the MP3 model, it also inherits MP3's disadvantages. To add to these are the following demerits unique to AAC:

- AAC has more features included in it such as prediction and TNS. These increase the processing power needed to encode and decode audio using AAC. (Brandenburg, 1999)
- The features mentioned above result in higher quality audio. However, this higher quality output needs more space on memory. (Brandenburg, 1999)

—

After considering the technologies discussed above, it was decided that the project will use the following technologies to implement the project idea:

- Wi-Fi for wireless communication because of its proliferation and our familiarity with it;
- AMR-WB for speech coding since it is the newest and most preferred encoding technique in Android; and
- AAC as the audio file format since it is also the most recent most common audio file format.

From now on, the project will focus on transferring audio information over wireless technology.

No peer to peer technology was chosen since we desire to implement peer to peer over Wi-Fi during this project. Also, as noted in its disadvantages, Wi-Fi Direct is not so common yet.

A lot of words have been written in this project concerning various technologies that assist in audio exchange, what with Bluetooth, Wi-Fi, and AMR. But one major, very widespread technology has not been touched on: GSM.

What is GSM? How does it work? Does it have any advantages? Disadvantages? And, most importantly, how does GSM compare with what this project is trying to implement? These questions will be answered over the next few pages.

2.5 GSM

In the beginning of the 1980s there were several systems for mobile communications in Europe. There was an acute need for a common communications system. (Willassen, 2003) In 1982, a group of European states came up with a new standards organization, "Groupe Speciale Mobile" (GSM). Its work was to develop a communications standard common for the member countries. (Willassen, 2003) In 1988, GSM was put in the ETSI, making GSM a standard for all telecommunications across Europe. (Willassen, 2003)

Unlike the other telecommunications systems that came up during that time, GSM was, and still is, a fully digital system allowing speech and data transfer as well as roaming across networks and countries (Willassen, 2003). Currently, GSM means "Global System for Mobile communication" and is a trademark. The ETSI group working on telecommunications standards has been renamed SMG (Special Mobile Group) so as to avoid confusion between it and GSM (Willassen, 2003).

Table 2, gotten from an International Engineering Consortium (IEC) GSM guide (IEC, 1999), shows some of GSM's milestones between the late 1980s and early 1990s.

The GSM network is divided into three major systems (IEC, 1999). These are:

1. The Switching System (SS);
2. The Base Station System (BSS); and
3. The Operation and Support System (OSS).

We will consider these systems below:

1. **The Switching System (SS)**

   It is responsible for performing call processing and subscriber-related functions (IEC, 1999). This system contains the following subsystems:

   - **Home Location Register (HLR)** – This is a database used to store and manage subscriber data. (IEC, 1999)
   - **Mobile Switching Center (MSC)** – The MSC performs the switching of phone calls. (IEC, 1999)

Table 2: GSM Milestones

Year	Milestone
1982	GSM formed
1986	Field test
1987	TDMA chosen as access method
1988	Memorandum of understanding signed
1989	Validation of GSM system
1990	Preoperation system
1991	Commercial system start-up
1992	Coverage of larger cities/airports
1993	Coverage of main roads
1995	Coverage of rural areas

- **Visitor Location Registry (VLR)** – This is a database that has the temporary information about subscribers that is needed by the MSC in order to take care of subscribers who are visiting. (IEC, 1999) Visitor subscribers are those who are not in the HLR database.
- **Authentication Center (AUC)** – This subsystem provides authentication and encryption parameters used to verify the user's identity and ensure the confidentiality of each call. (IEC, 1999)
- **Equipment Identity Register (EIR)** – This is yet another database. It contains information about the identity of mobile equipment that ensures that no calls are made from stolen, unauthorized, or defective MSs. (IEC, 1999)

2. **The Base Station System (BSS)**

This system performs all radio-related functions (IEC, 1999). It has the following subsystems (IEC, 1999):

- **Base Station Controller (BSC)** – This subsystem controls several Base Transmission Stations (BTSs). (BTSs will be discussed below.) The BSC handles procedures regarding call setup, location update and handover for each individual MS (Willassen, 2003).
- **Base Transmission Station (BTS)** – The BTS is the radio equipment needed to service each MS in the network (IEC, 1999). It contains transceivers and antennas (Willassen, 2003). In layman terms, the BTS is the booster.

3. **The Operation and Support System (OSS)**

   This is functional entity from which the network operator watches over and regulates the system. (IEC, 1999) The OSS also provides a network overview and gives support for the maintenance activities of different maintenance organizations (IEC, 1999).

Figure 12, derived from an IEC document (IEC, 1999), shows the GSM network's elements. A couple of items in the figure have not yet been mentioned. These are:

- **Message Center (MXE)** – This is a node that takes care of SMS, cell broadcast, voice mail, fax mail, e-mail, and notification. (IEC, 1999)
- **Mobile Service Node (MSN)** – This handles mobile intelligent network services. (IEC, 1999)
- **Gateway Mobile services Switching Center (GMSC)** – This node is used to provide a connection between networks. (IEC, 1999)
- **GSM InterWorking Unit (GIWU)** – This node provides an interface to various networks for data communication. It helps users to alternate between data and speech during the very same call. (IEC, 1999)
- **Public Switched Telephone Network (PSTN)** – This is the interconnection of voice-based public telephone networks in all parts of the world.
- **Public Land Mobile Networks (PLMNs)** – These are any wireless communications systems intended for use by subscribers on land. PLMNs may be stand-alone but are usually connected with systems such as the PSTN.
- **Packet Switched Public Data Network (PSPDN)** – This is a network that allows for the transfer of packet data between data networks.

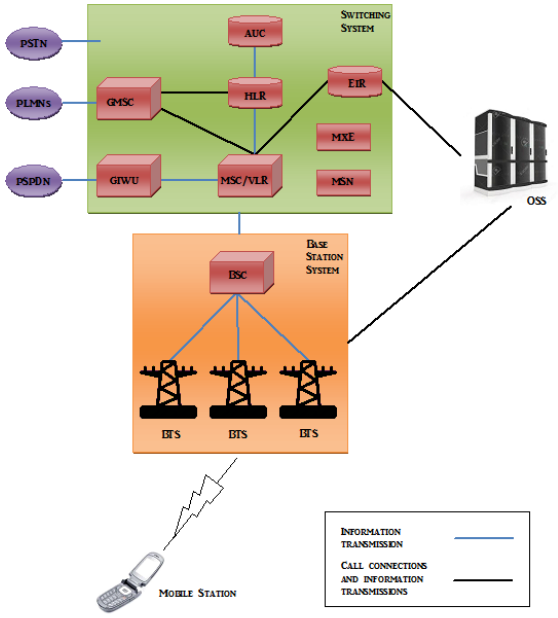

Figure 12: GSM Network Elements

- **Mobile Station** – This is the user equipment. It has two elements: the Mobile Equipment (ME) (the phone itself) and the Subscriber Identity Module (SIM) (Willassen, 2003).

When a user makes a call, their information goes through the Mobile Station to the Base Transmission Station then to the Base Station Controller. The BSC sends the information to a Mobile Switching Center which determines how to route caller information so that it can reach the individual being called (IEC, 1999). This involves checking location registers, authenticating the caller, and possibly routing the information to far away networks. On the receiving side, the receiving MSC sends the information through the BSC then to the BTS then to the called individual's phone. Communication thus takes place.

Concerning security, GSM provides authentication and encryption. These are closely related to the AUC. The user and the network have a shared secret key called the Ki (Willassen, 2003). The Ki is stored in the SIM and is not acc directly accessible to the user (Willassen, 2003).

- **Authentication**
  - Each time the MS connects to the network, the network authenticates the user by sending a random number to the MS. The SIM then uses the A3 encryption algorithm to compute an authentication token using the

random number and the Ki. The MS sends the authentication token back to the network (Willassen, 2003). The network computes an authentication token independently and then compares its own token with the one sent by the MS. If they match then the MS is authenticated.

- **Encryption**
  - Immediately after authentication, an encryption key Kc is computed (Willassen, 2003).The Kc is used to encrypt subsequent data moving from the MS to the network (Willassen, 2003).

GSM works based on cells. Each cell can be viewed as a geographical location in which a particular BTS's effects are felt. Cell terminologies include:

- **Cell radius** – The distance between a BTS and the outermost point of that BTS's coverage range;
- **Cell range** – The distance between two outermost points of a BTS's coverage range. The cell range is twice the cell radius; and
- **Inter-site distance** – The distance between two BTS's. This distance is usually three times the cell radius (ECC, 2010).

The cell radius can vary based on network demand. Some studies have put the cell radius at about 580 metres for highly populated areas such as towns and 5000 metres for areas with low population such as the countryside (ECC, 2010). The idea of cells is illustrated in Figure 13, based on a GSM comparison written by the Electronic Communications Committee (ECC) (2010). Figure 13 shows cells having a hexagonal shape. This shape is used since it is the same one used in the ECC GSM comparison guide of 2010 quoted in the previous sentence.

GSM is used in cellular phones, microcontrollers (Amin & Khan, 2014), and modems (Verma & Bhatia, 2013).

Some of the advantages of the GSM technology are:

- It is a very common technology. A source says GSM is the world's largest system for mobile communication today (Willassen, 2003).
- Unlike the analogue communications it replaced, GSM uses digital technology. This means that GSM can scale effectively while keeping signalling mechanisms, interference, and switching operations at manageable levels (IEC, 1999).
- The GSM standard is abstracted enough to allow designers as much freedom as possible while still making it possible for the GSM operators to buy

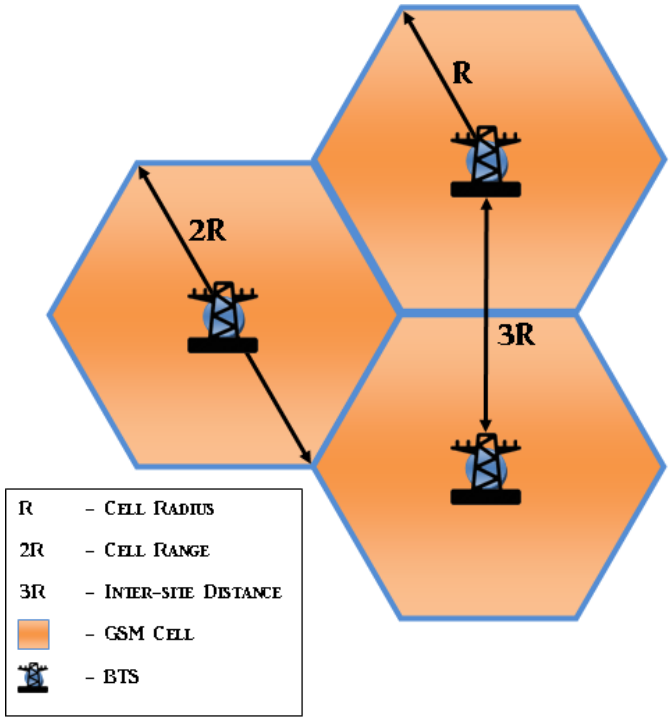

Figure 13: GSM Cells

equipment from various suppliers (IEC, 1999).

- GSM ensures security of communication by using the Ki key as was explained earlier (Willassen, 2003).

Some of GSM's shortfalls include:

- Since it is so common, GSM can be used by criminals to harass other citizens. For example, in Kenya, cases of fraudsters staging mock kidnaps over cell phone have been on the rise. A person gets a seemingly innocent message from Customer Care saying they need him/her to switch off his/her phone for some hours because of some network maintenance. As soon as the person does this, the caller calls the victim's relatives saying they have kidnapped the victim and need a certain ransom. Since the victim has switched his/her phone off, there is no way the relatives can contact him/her. The unwary relative(s) may end up sending the ransom money, only to learn that it was a scam.
- Old SIM cards have limited space to store contact and SMS information (Willassen, 2003).

- Ki encryption algorithms, such as A3, have weaknesses and can be cracked (Willassen, 2003).

Lastly, we will compare GSM with this project idea. This comparison will follow three categories:

- **Distance.**
  - On the one hand, GSM is a global network. As mentioned earlier, GSM cells have a range of up to about 5 kilometres. These cells join up to form PLMNs, covering thousands of kilometres. To add to that, the interconnection of PLMNs means that GSM users can access other GSM users in almost any part of the world.
  - On the other hand, our project uses Wi-Fi. The Wi-Fi section of this letter showed that Wi-Fi is limited to about 100 metres.

  Conclusion: The project idea's range vanishes into insignificance in the face of GSM's reach.

- **Security Issues.**
  - On the one hand, as mentioned earlier, GSM uses the Ki, authentication algorithms, and resultant keys to ensure confidentiality of the data sent through it. However, we noted in GSM's disadvantages that GSM's security algorithms can be cracked.
  - On the other hand, as was noted in the Wi-Fi section of the project, Wi-Fi is very prone to data sniffing because if its huge operation range relative to Bluetooth and NFC. However, we noted that Wi-Fi Direct uses WPS to ensure security. WPS involves using AES-CCMP for encryption and PSKs for authentication. Our project will try to implement a version of WPS to safeguard the audio data sent between devices. Figure 14 shows the security options available for Wi-Fi hotspots in Android 4.1.2.

  Conclusion: Both GSM and this project's idea have some security weaknesses.

- **Cost.**
  - On the one hand, for the implementers, GSM is quite costly. The cost incurred by the implementers can be seen in the following ways:
    - Getting the hardware in place is understandably expensive.

Figure 14: Android 4.1.2 Wi-Fi Hotspot Security Options

  * Providing the software that will perform real time switching of both voice and data costs additional time and money.

  For the users, GSM is rather cheap. All that is needed is:

  * A basic mobile phone,
  * Some electricity to charge the phone's battery, and
  * Some airtime.

- On the other hand, for the implementers, this project's idea is a bit cheaper compared to GSM.

  * First, the hardware needed is two Wi-Fi enabled (preferably Android) smartphones. This is way cheaper than the switches, routers, and registers of GSM.
  * Second, the code to implement the peer to peer communication between those two devices is miles less complicated than that involved in GSM's real time voice and data transmission.

  For the users, this project's idea might be a bit expensive.

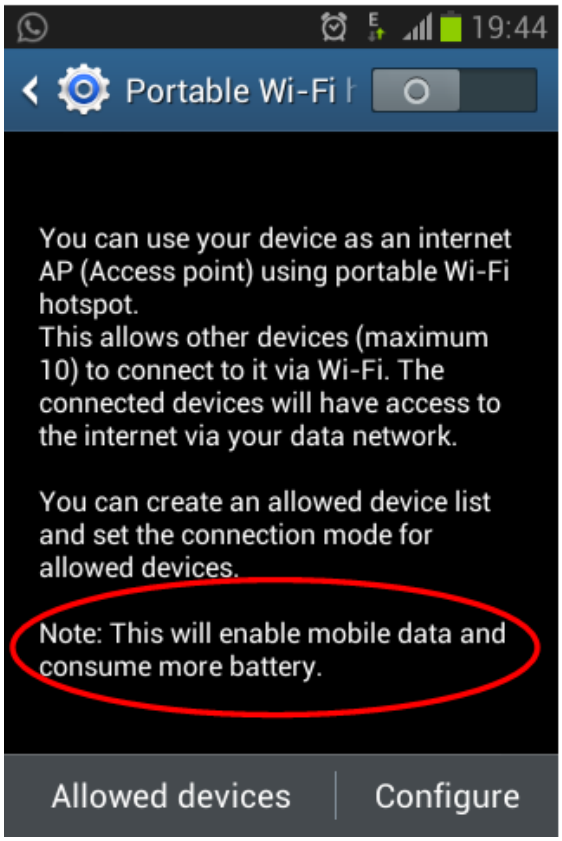

Figure 15: Android 4.1.2 Wi-Fi Power Draw Warning

* First, the idea cannot be of any use unless one has a Wi-Fi enabled (preferably Android) smartphone.
* Second, for those having such devices, this application can only work within the Wi-Fi range of maximum 100 meters. Users will have to know if other users are in that range first.
* Third, most smartphones have an application used for dialling and calling. This project will make a dialling application. The hassle of navigating to this application to make a call as opposed to navigating to the default dialling application may deter some users.
* Fourth, the project idea uses Wi-Fi, meaning that it will draw a lot of power from devices. This might frustrate the user. In fact, Android 4.1.2 warns of the power draw, as seen in Figure 15.

Conclusion: GSM and the project's idea somehow cancel each other out in

terms of costs. With the current trend towards smartphones and more long-lasting batteries, however, the project's idea might edge GSM ever so slightly in this aspect.

In summary, GSM and its derivatives will continue to be the go-to technology for mobile communication for the foreseeable future. This project does not intend to replace GSM. This project intends to provide cheaper communication over short distances among devices that are Wi-Fi enabled. This provision can be used in ways such as:

- Making **phone calls.** Two devices in the same range can send and receive audio data simultaneously, thus a phone call.
- Creating **public address systems in small rooms**. This is possible by having only one user speaking into a device and having the other users listening from a second device. It is our hope that the project will be expanded to allow for a point-to-multipoint architecture that will really implement this public address system functionality.
- **Real time recording and streaming.** One device can be put in a room and another in an adjacent room. The device in the adjacent room can play all the sounds in the first room as soon as they happen. In this way, a phone can be used as a bug, or listening device.

The above functionalities may make the project idea very useful in companies that have a small size since the project works within a 100 metre radius.

The project is not planning to limit itself to just phone calls.

3 Research Methodology

At least three methods of research will be used in this project:

1. **Experimentation**;
2. **Web Search**; and
3. **Interviews.**

These methods are discussed here underneath.

3.1 Experimentation.

This which will be used to test whether the project's implementation will work on actual devices. Experimentation will be the research method most extensively used in this project.

At least three experiments are planned. These are:

1. Device connection and communication;
2. Effect of distance on communication; and
3. Effect of amount of audio data on communication.

Here is how the three experiments are to be executed.

1. **Device connection and communication**

   Premise

   The idea of this experiment is to try to see if two Android smart phones can connect and communicate via Wi-Fi without using any third party intermediary devices.

   Figure 16 shows what we are trying to achieve.

   Inputs

   These include:

   - Relevant code.
   - Commands to connect the devices.

   Tools

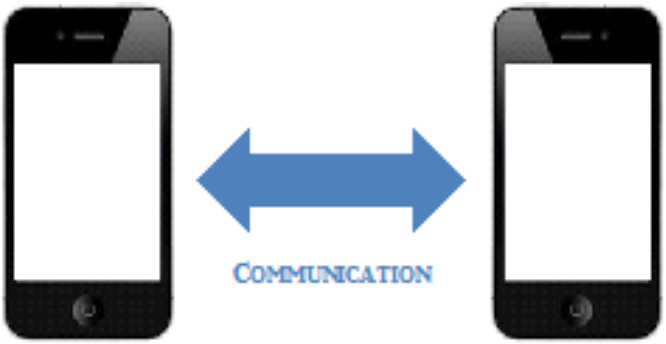

Figure 16: Establishment of Communication Between Two Android Devices

These will be two Android smart phones and an Integrated Development Environment (IDE).

Process

The process to be followed will be as follows:

(a) Execute the code for both server and client on each of the two devices.

(b) Attempt to connect one device to the other.

Outputs

The expected outputs include:

- A User Interface display informing us that the two Android devices have connected with each other.
- Sound from one device playing on the other device.

2. **Effect of distance on communication**

Premise

The idea of this experiment is to see how much distance will affect the quality of communication. Usually, the quality of wireless communication degrades when the two communicating devices increase the distance between them. We want to establish if this is so in our system.

One of the assumptions made here is that quality goes down when only parts of a file sent are received. Therefore we will check the amount of bytes in the audio file sent from the sender and compare it with the amount of bytes in the audio file received by the receiver. Since our system will be sending audio files every second, we will need to fix the amount of audio data sent

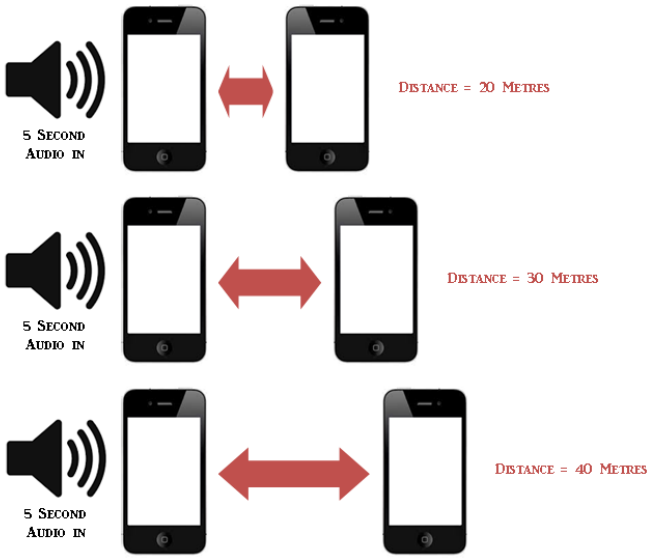

Figure 17: Varying Device-to-Device Distance while Keeping Audio Input Fixed so as to Determine Effect of Distance on Communication

by fixing the amount of time audio will be recorded at the sender side. This time will be set at a stationary five seconds. The distance between devices will be varied from zero metres to the Wi-Fi maximum range of 100 metres.

Figure 17 shows a part of the experiment we plan to carry out. Audio is input into the smart phone on the left for a fixed five seconds. The distance between devices is then varied by adding ten metres to it after each iteration. The "Process" segment of this experiment will give more details concerning the experiment.

Inputs

There will be only one input: a five second audio input at the sender's end.

Tools

The tools used will include:

- Two Android smart phones;
- An IDE;
- A timer if necessary;
- A 100 meter tape measure; and
- The method `length()` of the Java class `File`.

Process

The experiment will be carried out as follows:

(a) Write code on the sender's side that will record the amount of digital data gotten from converting the analogue five second audio input into a digital format. This code will involve use the `length()` method.

(b) Write code on the receiver's side that will record the amount of data received from the sender. This code will also use `length()`.

(c) Set the devices zero metres apart.

(d) Establish a connection between the two devices.

(e) Play the five second audio into the sender device. This step can also be executed by talking into the sender device for exactly five seconds. This alternative calls for the use of a timer to ensure exactly five seconds of audio are input.

(f) The code written in part (a) should be able to record the amount of digital data gotten at the sender.

(g) The recorded data will be sent to the receiver device thanks to the application code.

(h) The code written in part (b) should be able to record the amount of digital data received at the receiver's side.

(i) The receiver device should attempt to play the data.

(j) The recorded sent and received data amount should now be put in permanent storage for further processing.

(k) If the current distance between devices is less than 100 metres then the distance between the two devices should be increased by ten metres and the experiment should go back to step (d). Otherwise the experiment should terminate.

Outputs

The expected output will include the following:

- The amount of bytes lost at every distance.
- Hopefully a graph of the same.

3. **Effect of amount of audio data on communication**

Premise

This experiment will test how well the application handles volumes of data. To simulate various volumes of data we have decided to vary the amount of time audio data recorded. The time taken to record the audio data will be called the talk time. We will start with a talk time of ten seconds. The audio data will be recorded at the sender side.

However, we will keep the distance between devices fixed at 50 metres (half the maximum Wi-Fi range). We will then compare the mean lost bytes at various audio recording times with the mean lost bytes at 50 metres that was established in experiment 2. We will refer to the mean lost bytes at 50 metres as the base mean and the mean lost bytes at various audio recording times as the varied records means. The measure of how much the varied records means deviate from the base mean will tell us how communication quality is affected by amount of audio data. A lower variation will mean that little quality is lost with increase in audio data. This is because a variance of zero implies that all sampled data is identical, and so a variance close to zero will imply that the varied records means will be close to identical to the base mean.

Figure 18 attempts to illustrate how the experiment will work. As said in the above paragraphs, the distance between the two testing devices will be fixed at 50 metres. The figure shows that. What will be varied is the amount of time audio will be played into the sender device. Figure 18 shows the audio input at the sender side being increased by five seconds at every iteration of the experiment.

Inputs

The inputs include:

- A fixed device-to-device distance of 50 metres; and
- Audio.

Tools

The tools include:

- Two Android devices;
- An IDE;
- A timer; and
- The method `length()` of the Java class `File`.

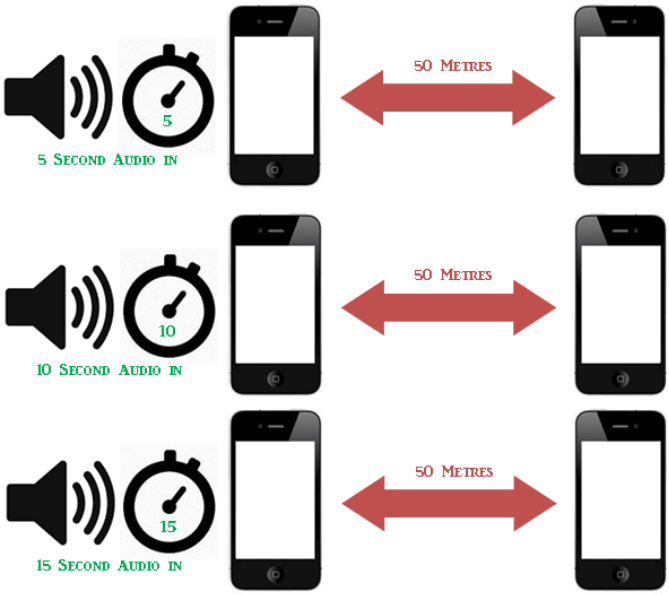

Figure 18: Varying Audio Input while Keeping Device-to-Device Distance Fixed so as to Determine Effect of Amount of Input Data on Communication

Process

The experiment should happen as follows:

(a) Write code on the sender's side that will record the amount of digital data gotten from converting analogue audio input into a digital format. This code will involve using the `length()` method.

(b) Write code on the receiver's side that will record the amount of data received from the sender. This code will also use `length()`.

(c) Set the devices such that the distance between them is equal to the fixed distance defined in the Inputs section of this experiment.

(d) Establish a connection between the two devices.

(e) Start with a talk time of ten seconds.

(f) Talk into the sender device for ten seconds.

(g) The code written in part (a) should be able to record the amount of digital data gotten at the sender.

(h) The recorded data will be sent to the receiver device.

(i) The code written in part (b) should be able to record the amount of digital data received at the receiver's side.

(j) Store the recorded data in permanent storage.

(k) If the talk time is less than 100 seconds then increase the talk time by ten seconds and go back to step (f). Otherwise terminate the experiment.

Outputs

The outputs anticipated include:

- Data on the amount of data sent and received at various recording times.
- A graph of comparing this data with the various talk times.

3.2 Web Search.

Here, various websites will be visited to get solutions to project problems as well as get inspiration to work around implementation issues.

Some of these sites include:

- http://stackoverflow.com/, a software developer community where programmers post their coding problems and get answers from the community;
- https://en.wikipedia.org, home to Wikipedia – the free encyclopedia; and
- http://developer.android.com/, the official Android development site.

We expect to use the following inputs for this method of research:

- An internet enabled device.
- Questions to search answers to.

Below are the tools we plan to use during web search:

- An internet enabled device.
- A web browser.

Here is the data we expect to get from this research method:

- Answers to the questions searched for – hopefully including snippets of code implementing those answers.Questions here might include queries such as:
  - How is timing implemented in Android?

  - What is the difference between using the Android Activity Constructor and using the Android Activity `onCreate` method?
  - How do I extract an mp3 file from a byte array?
  - How do I use Android `DialogFragments`?
  - How do I convert an `ImageIcon` to a `BitmapDrawable` in Android?
- More questions from those answers.

3.3 Interviews

These will be done in an informal setting to acquire opinions from potential end users concerning the user interface, any possible limitations, and other useful pieces of information.

The target population for my interviews will generally be university students since these know how to use smart phones the most.

I intend to carry out interviews after every major application update. That way I will be able to get user input more often.

The expected inputs to interviews are:

- Preparation of interview questions.

Interviews might need the following tools:

- A notebook and a pen to record interviewee responses.

We expect to get the below-mentioned data from conducting interviews:

- Various varying opinions on questions asked to interviewees.
- More questions to ask interviewees, such as when seeking clarification.
- Suggestions on improvements of the system

4 System Analysis

In this section, we will look at two requirements needed by any computer system:

1. Functional requirements.
2. Non-functional requirements.

These are discussed starting now.

4.1 Functional requirements

The system has two functional requirements.

First, it should allow **two smartphones** to **connect to each other over wireless**. This should be done **without** the use of **an intermediary device**. This will be achieved using relevant code on both phones, as well as a WiFi connection between the two.

Second, the system should allow **audio data** to be **transferred between the two connected phones**. This is also to be done using code.

4.2 Non-functional requirements

These are also two.

First, the system should be **secure**. This is important because audio can easily be recorded and misused by malicious individuals. Security over WiFi has always been a challenge. Since we cannot control the circumstances and motives of all who will end up using this system, we can only hope that they will set up their WiFi connections over a protected wireless link. During the implementation phase of this project, we plan to use a connection protected by WPA2 PSK.

Second, the system should provide for **audio clarity**. The sound heard by individuals on both ends of the device should be clear enough for them to recognize. We hope that the inbuilt sound processing systems of the devices we will use have this facility in place.

5 System Design

To show the design of this system, we will use the following diagrams:

1. The Network Topology Diagram;
2. Class Diagrams;
3. Activity Diagrams;
4. The Sequence Diagram; and
5. The Navigation Diagram.

5.1 The Network Topology Diagram

Since the project is centered on creating a connection between two devices, we have to outline how the network will look. This is highlighted in the network topology diagram in Figure 19.

The figure shows two phones serving as server and client. The numbers close to the points where arrows touch the phones indicate the number of servers and clients allowed to communicate with each other at a moment. The 1's close to where the arrows touch the phones show that there can only be one server communicating with one client at any particular point in time.

This understanding is important as we design the system further since it ensures that we will not focus on broadcasts, multicasts, or any other one-to-many communication form.

5.2 Class Diagrams

Class diagrams show the various classes used in a software system as well as how these classes are related.

Our system has a total of 11 classes. These classes are explained below.

1. `HomeActivity` **class.** This class is the point at which the user lands when they fire up the system.

   The class diagram for the `HomeActivity` class is seen in Figure 20

2. `ReceiveCallActivity` **class.** This class waits for a call to arrive and processes the requests of the caller.

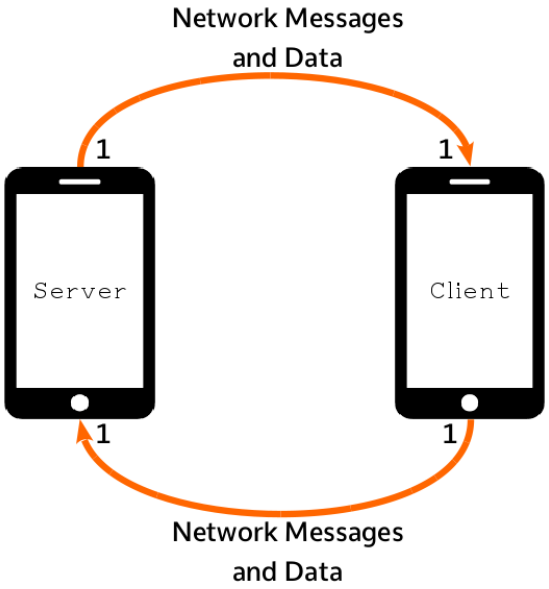

Figure 19: The Network Topology Diagram

The class diagram for the `ReceiveCallActivity` class is seen in Figure 20.

3. `IncomingCallActivity` **class.** This class alerts the user to an incoming call. It allows the user to their accept or reject the call.

   The class diagram for the `IncomingCallActivity` class is seen in Figure 20.

4. `CallInSessionActivity` **class.** This is the class where audio data is sent and received between the two connected devices.

   The class diagram for the `CallInSessionActivity` class is seen in Figure 20.

5. `MakeCallActivity` **class.** This class allows the user to select a contact to call.

   The class diagram for the `MakeCallActivity` class is seen in Figure 20.

6. `CallingActivity` **class.** In this class, the user calls the contact selected in the `MakeCallActivity` class. If the contact is not a viable receiver or is busy, this class informs the user of the same. When the called individual picks up, this class directs the user to an instance of `CallInSessionActivity`.

   The class diagram for the `CallingActivity` class is seen in Figure 21.

7. `Contact` **class**. This class represents a contact on the contact list found in `MakeCallActivity`.

Figure 20: Class Diagrams for `HomeActivity`, `ReceiveCallActivity`, `IncomingCallActivity`, `CallInSessionActivity` and `MakeCallActivity`

Figure 21: Class Diagrams for `CallingActivity`, `Contact`, `SearchForContactsThread`, `WifiStateChangeBroadcastReceiver`, `SendRecordedSoundTimerTask` and `SocketServerThread`

The class diagram for the `Contact` class is seen in Figure 21.

8. `SearchForContactsThread` **class**. This class extends the Java `Thread` class and is used to search for contacts from available hotspots. Searching for contacts is fairly expensive in terms of time. This means that doing it from the same thread as the UI thread would make the app look like it is hanging. To avoid this circumstance, in this class we search for contacts using a thread different from the UI thread.

   The class diagram for the `SearchForContactsThread` class is seen in Figure 21.

9. `WifiStateChangeBroadcastReceiver` **class**. This class extends the Android `BroadcastReceiver` class. Whenever the state of an Android device's WiFi changes, the new state is broadcast throughout the applications of the device. To receive that broadcast, an application needs to extends the `BroadcastReceiver` class. Since our system needs to be informed the moment WiFi is turned on so as to scan for contacts, we use the `WifiStateChangeBroadcastReceiver` class and configure it to receive the "WiFi on" state.

   The class diagram for the `WifiStateChangeBroadcastReceiver` class is seen in Figure 21.

10. `SendRecordedSoundTimerTask` **class**. This class extends the Android `TimerTask` class. `TimerTask` classes are used to perform actions at specific intervals of time. We use the `SendRecordedSoundTimerTask` class to record audio every second and to send the recorded audio every second.

    The class diagram for the `SendRecordedSoundTimerTask` class is seen in Figure 21.

11. `SocketServerThread` **class**. Like the `SearchForContactsThread` class above, this extends the Java `Thread` class. It is used by the `ReceiveCallActivity` class to listen for any connections coming in from the `CallingActivity` class.

    The class diagram for the `SocketServerThread` class is seen in Figure 21.

A summarized class diagram showing relationships between classes is called an elided class diagram. Our elided class diagram is seen in Figure 22. The next few paragraphs explain the elided diagram.

To begin with, we see `HomeActivity` can start one of either `ReceiveCallActivity` or `MakeCallActivity`. `HomeActivity` cannot start both these two classes at the

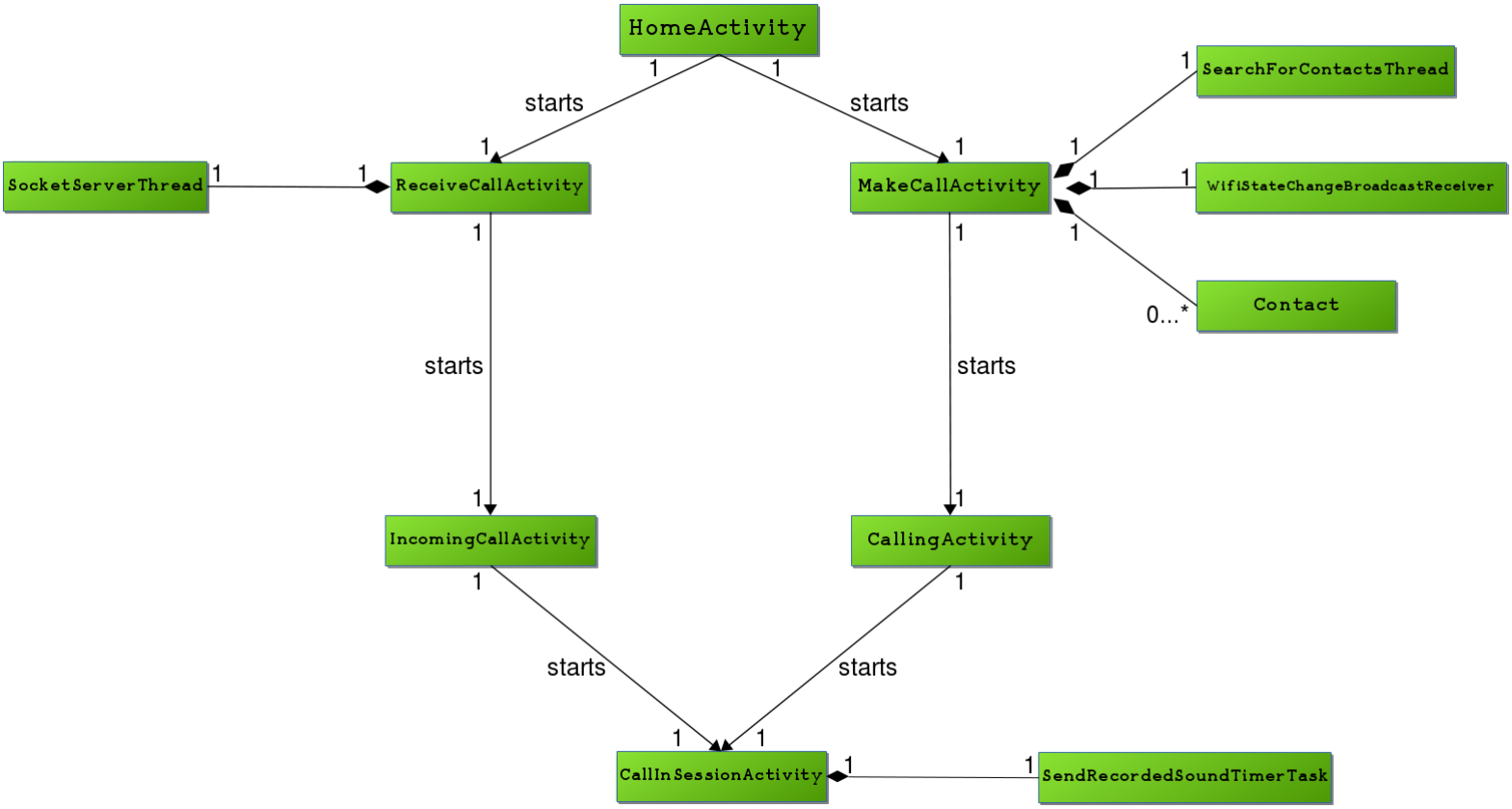

Figure 22: Elided Class Diagram

same time.

The filled diamond on the line connecting `ReceiveCallActivity` to `SocketServerThread` tells us that `ReceiveCallActivity` has a `SocketServerThread` instance and that the `SocketServerThread` instance cannot exist without the `ReceiveCallActivity` instance. This is so because the `SocketServerThread` class exists to assist the `ReceiveCallActivity` class connect to clients. The `ReceiveCallActivity` can start only one instance of the `IncomingCallActivity` class.

`IncomingCallActivity` can start only one instance of `CallInSessionActivity`. There is not much else `IncomingCallActivity` can do in terms of inter-class interactions.

`MakeCallActivity` has one object of the `SearchForContactsThread` class and one object of the `WifiStateChangeBroadcastReceiver` class. However, it can have as many `Contact` objects as possible. `MakeCallActivity` uses a `SearchForContactsThread` object to search for contacts and display them. It uses a `WifiStateChangeBroadcastReceiver` instance to be alerted whenever the WiFi is turned on. And `MakeCallActivity` can have as many `Contact` objects as possible since we cannot determine beforehand how many hotspots there will be around the place of application execution. The last thing about `MakeCallActivity` is that it can start only one instance of the `CallingActivity` class.

Just like `IncomingCallActivity`, `CallingActivity` can start only one `CallInSessionActivity` object and can do nothing much else.

From the diagram, we see that `CallInSessionActivity` instances can be instantiated by either `IncomingCallActivity` instances or `CallingActivity` instances. Each of those last two mentioned classes can start only one `CallInSessionActivity` object each. `CallInSessionActivity` has a `SendRecordedSoundTimerTask` object which it uses to send recorded sound every second and to receive recorded sound and play it every second.

5.3 Activity Diagrams

Activity diagrams highlight the order in which instructions are executed in an instance of a class.

In our case, all classes which have a set of instructions to execute have their activity diagrams drawn. Explanations are as follows.

1. `HomeActivity` **class.**

   The activity diagram for the HomeActivity class is seen in Figure 23

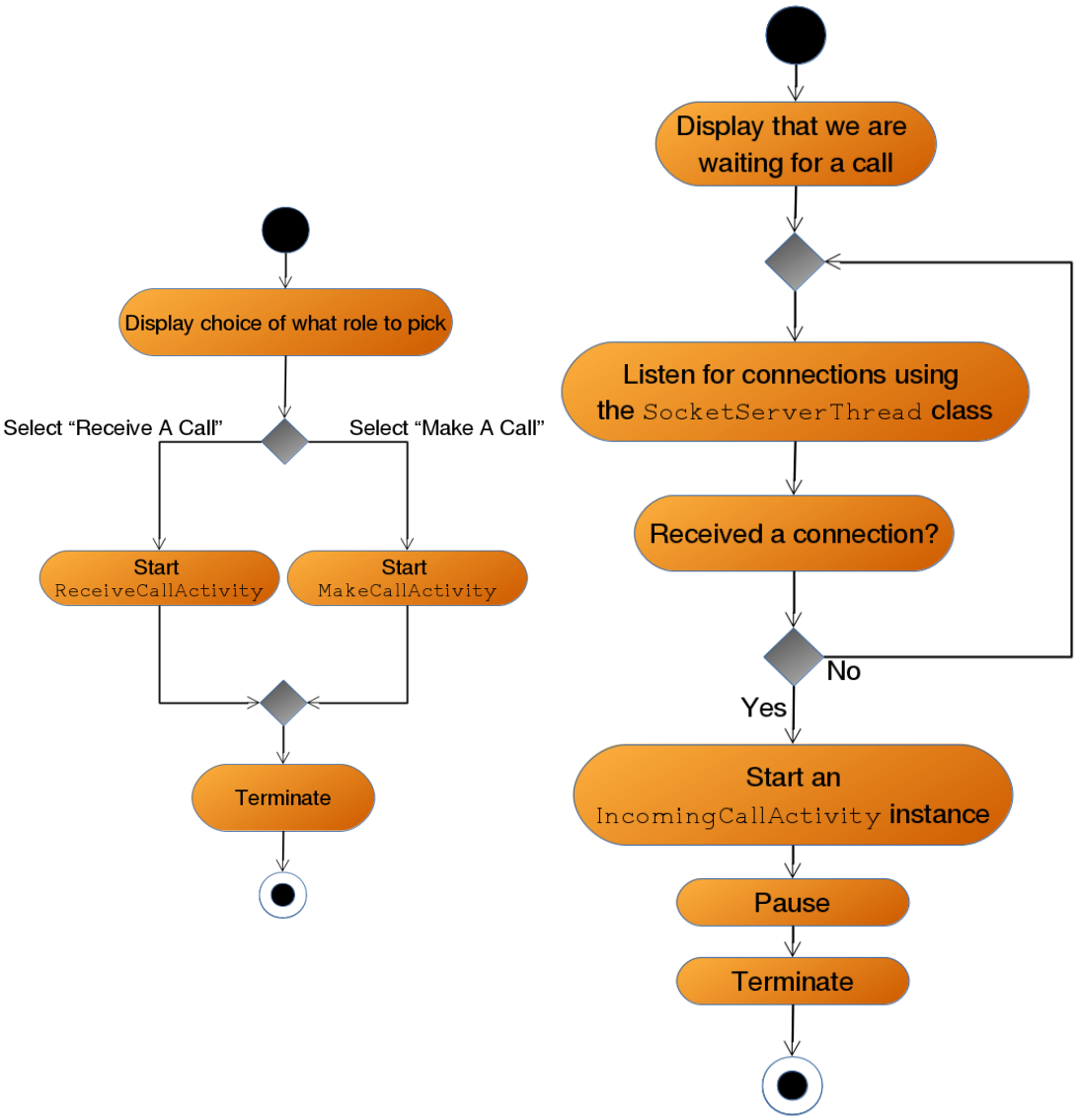

Figure 23: Activity Diagram for `HomeActivity` on the left and `ReceiveCallActivity` on the right

The class starts off displaying to the user a UI that allows him/her to choose which role they want to be: caller or receiver. If the user chooses to make a call, the class starts the `MakeCallActivity` class. If the user chooses to receive a call, the class starts the `ReceiveCallActivity`. After processing user requests, `HomeActivity` terminates.

2. `ReceiveCallActivity` **class.**

   The activity diagram for the `ReceiveCallActivity` class is seen in Figure 23.

   We begin by displaying to the user that the system is waiting for a call. We then begin listening for any incoming connections using an instance of the `SocketServerThread` class. We keep doing this listening until we receive a connection from the `CallingActivity` class. On receiving such a connection, we start an instance of the `IncomingCallActivity` class and pause the `ReceiveCallActivity` class. An option is there to terminate the `ReceiveCallActivity` if necessary.

3. `IncomingCallActivity` **class.**

   The activity diagram for the `IncomingCallActivity` class is seen in Figure 24

   In this class, we start by displaying the needed UI so that the user might know that a call is incoming. We then retrieve references to the connection established in the parent activity of this class. The parent activity is the `ReceiveCallActivity` since instances of the `IncomingCallActivity` class are started off by instances of the `ReceiveCallActivity` class. Part of the UI of the `IncomingCallActivity` class contains two buttons: one for accepting the incoming call and the other for rejecting it. If the user taps the "Accept" button, the `IncomingCallActivity` class starts a `CallInSessionActivity` activity then terminates. If the user selects "Reject" then `IncomingCallActivity` simply terminates. In the latter case, since `IncomingCallActivity` was started by `ReceiveCallActivity` and since `ReceiveCallActivity` had been paused after starting `IncomingCallActivity`, `ReceiveCallActivity` is resumed.

4. `CallInSessionActivity` **class.**

   The activity diagram for the `CallInSessionActivity` class is seen in Figure 24.

   This class begins by initializing the speaker and mute to be off. It sets up the display, initializes the media recorder used for recording sound and initializes

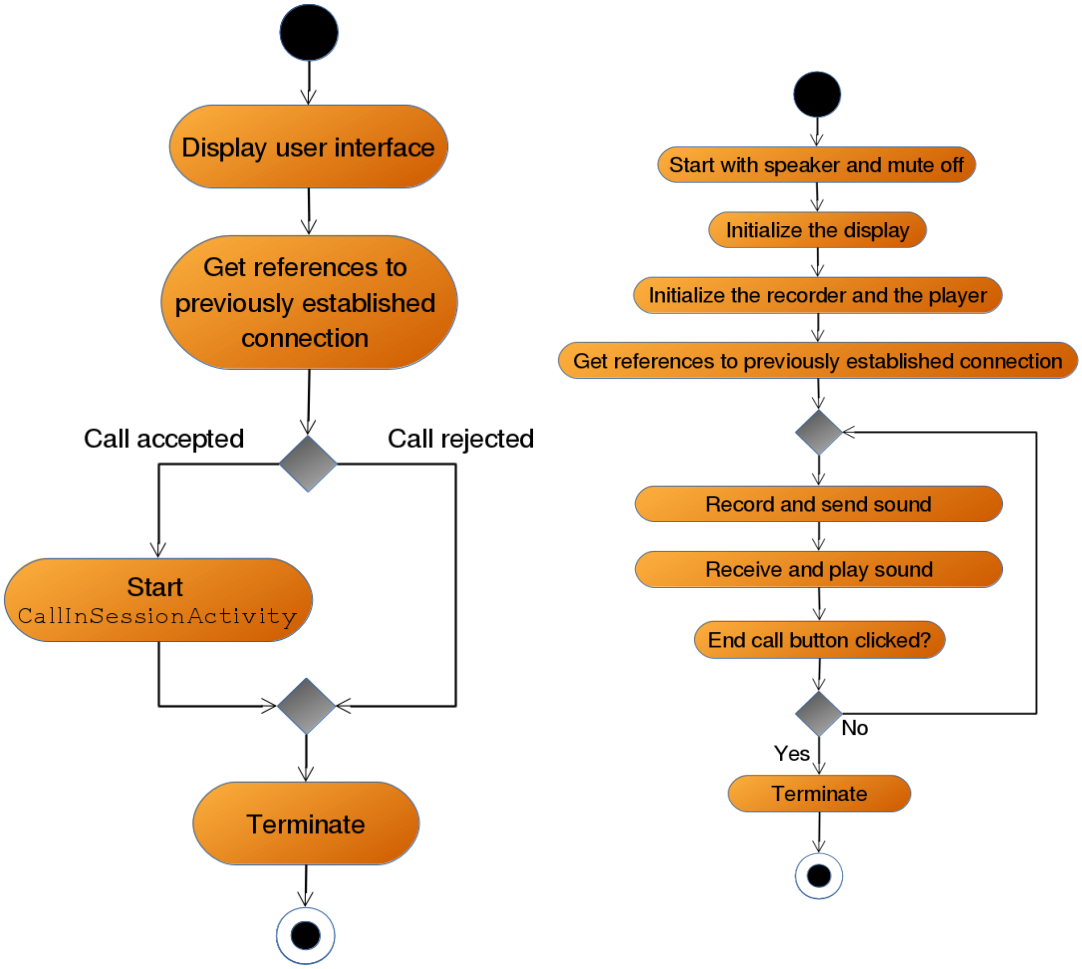

Figure 24: Activity Diagram for `IncomingCallActivity` on the left and `CallInSessionActivity` on the right

the media player used to play sound. It also gets references to the previously established connection. These references come from the class that started this class. As we saw in the class diagrams, `CallInSessionActivity` can be started by either `CallingActivity` (from the caller's side) or `IncomingCallActivity` (from the receiver's side). Either parent class has references to the established network connection and passes these to the `CallInSessionActivity` class. The UI in `CallInSessionActivity` has a button for ending the call. As long as this button is not clicked, the class records sound and sends it to the `CallInSessionActivity` instance on the other side of the network. The class also receives sent sound and plays it. As soon as the end button is clicked, the class terminates.

5. `MakeCallActivity` class. This class allows the user to select a contact to call.

   The activity diagram for the `MakeCallActivity` class is seen in Figure 25.

   We begin by displaying a user interface which will show the user the contacts available for him/her to call and allow the user to scan for contacts. The class also gets the SSID of the user's device. This will be used to identify the user when he/she calls another. After this, the class initializes a `SearchForContactsThread` object, which will be used to search for contacts when the user elects to scan for some. The final initialization step is to turn on the WiFi. If the user scans for contacts and none are found, they can keep scanning for them. Any contacts that the class finds will be displayed in a list. When the user selects a contact to call, `MakeCallActivity` establishes a network connection with the selected contact, starts `CallingActivity`, then pauses itself. The user has the option of eventually terminating this class if he/she sees fit.

6. `CallingActivity` **class.**

   The activity diagram for the `CallingActivity` class is seen in Figure 25.

   As usual for an `Activity` class, this class begins by displaying the relevant UI. It then waits for five seconds to simulate the behavior users might expect during the making of a call. Without this delay, the app will execute very quickly and that might leave users a bit confused. After the five seconds, we get references to the previously set up connection. This connection is the one set up at `MakeCallActivity`. Then we process the messages coming from the `IncomingCallActivity`, which by this time should be running on the device on the other side of the network. If the `IncomingCallActivity` sends a "PICKED_UP" message, this means that the other user has picked

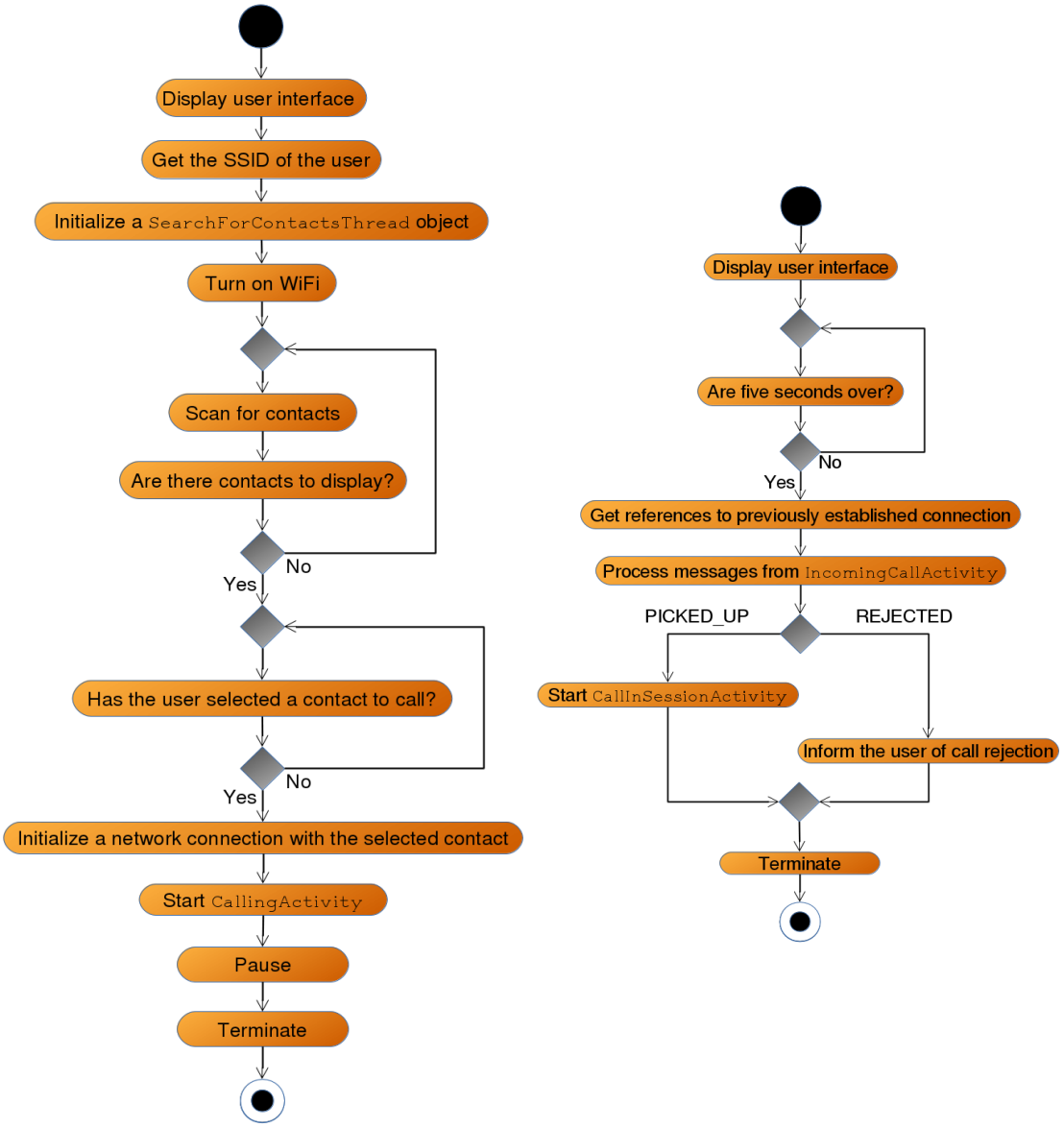

Figure 25: Activity Diagram for `MakeCallActivity` on the left and `CallingActivity` on the right

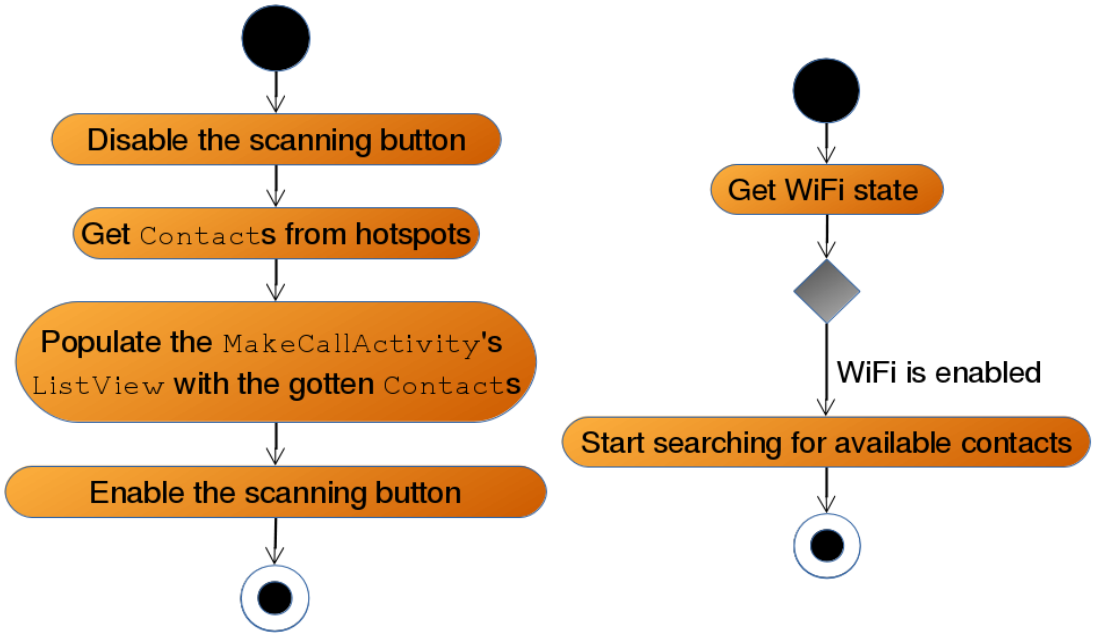

Figure 26: Activity Diagram for `SearchForContactsThread` on the left and `WifiStateChangeBroadcastReceiver` on the right

up the call and so we start `CallInSessionActivity`. If we receive a "REJECTED" instead, this means that the user does not desire to communicate now so we change the UI inform the caller of this. After processing the messages, we terminate the `CallingActivity` instance, leaving either the newly started `CallInSessionActivity` instance or the previously paused `MakeCallActivity` instance to continue interacting with the user.

7. **`SearchForContactsThread` class.** The activity diagram for the `SearchForContactsThread` class is seen in Figure 26.

   This class starts out by disabling the button used to scan for contacts. This is done to show the user that they cannot scan until the previous scan request has been fully processed. Thankfully, processing scan requests in a separate thread takes a rather short time. The class then initializes a set of `Contact` instances from the hotspots found after scanning. After this, the class populates the `ListView` of `MakeCallActivity` with this list of `Contacts`. The last thing done by this class is enabling the scanning button.

8. **`WifiStateChangeBroadcastReceiver` class.**

   The activity diagram for the `WifiStateChangeBroadcastReceiver` class is seen in Figure 26.

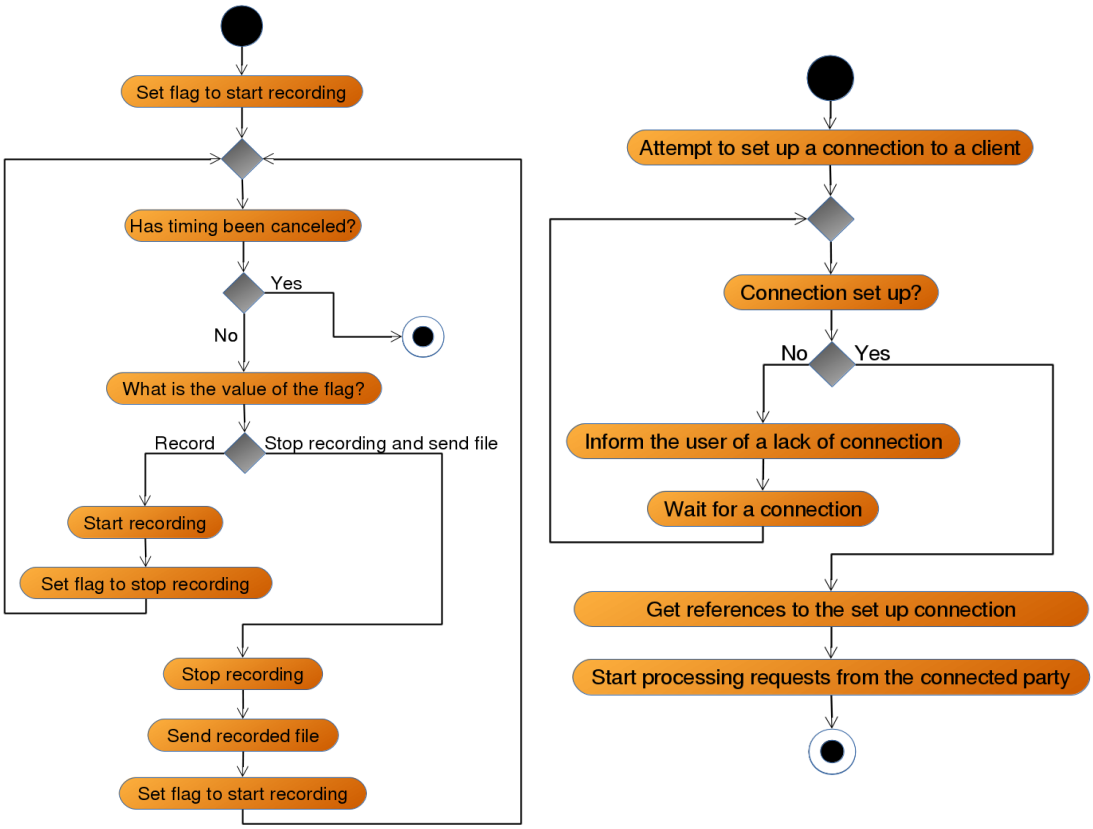

Figure 27: Activity Diagram for `SendRecordedSoundTimerTask` on the left and `SocketServerThread` on the right

What happens in this class is fairly simple. We get the wait for the WiFi state to change. It changes to the state where it is enabled, we start searching for available contacts and then terminate after doing so.

9. **`SendRecordedSoundTimerTask` class.**

   The activity diagram for the `SendRecordedSoundTimerTask` class is seen in Figure 27.

   This class is very important for the sending and receiving of audio data between `CallInSessionActivity` instances. What happens in this class is generally controlled by a flag which oscillates between the "record" and the "stop recording and send recorded file" states. Since the class does its job based on fixed intervals of time, if timing is canceled then the class terminates. The class starts off with the flag set to the "record" state. In this state, the class starts recording sound for a fixed amount of time. When

that is finished, it sets the flag to the "stop recording and send file" state. As long as the timing is not canceled, processing continues. The value of the flag is read and found to be "stop recording and send file". As long as the timing is not canceled, processing continues. With timing still not canceled, the class checks the flag, sees that it reads "record", and goes about executing the "record" state. This cycle goes on until the timing gets canceled.

10. `SocketServerThread` **class.**

    The activity diagram for the `SocketServerThread` class is seen in Figure 27.

    This class is where most of the networking is done. We first attempt to connect to a client. If the attempt fails, we inform the user of a failure and then we keep waiting for a connection. If we are able to set up the connection, we then get references to that connection. These references are important since they help us send data to and receive data from device at the other end. With the references acquired, we can now start processing requests from the connected party.

5.4 The Sequence Diagram

A sequence diagram is used to show how different objects in a system interact with each other. The sequence diagram for this project is shown in Figure 28. The table in Figure 29 explains what each of the text styles in the sequence diagram represents.

Below is a brief explanation of what happens.

- `HomeActivity`: The system starts at this activity. The user chooses if they want to make a call or receive a call. Depending on the user's choice, the system calls a `startActivity()` for an instance of either the `MakeCallActivity` class or the `ReceiveCallActivity` class. After doing this, the `HomeActivity` activity finishes and closes down.
- `MakeCallActivity`: This activity is started when the user decides to make a call. Here, the user can select which individual to call. When the individual to be called is chosen, the `MakeCallActivity` activity calls the `CallingActivity` activity which will attempt to call the chosen individual. The sequence diagram shows that two extras are passed to the `CallingActivity` activity. An extra is a piece of data passed between Android activities. These two extras are the name of the called individual as well as the name of the individual calling. After starting the `CallingActivity` activity, the `MakeCallActivity` activity suspends itself but does not close down.

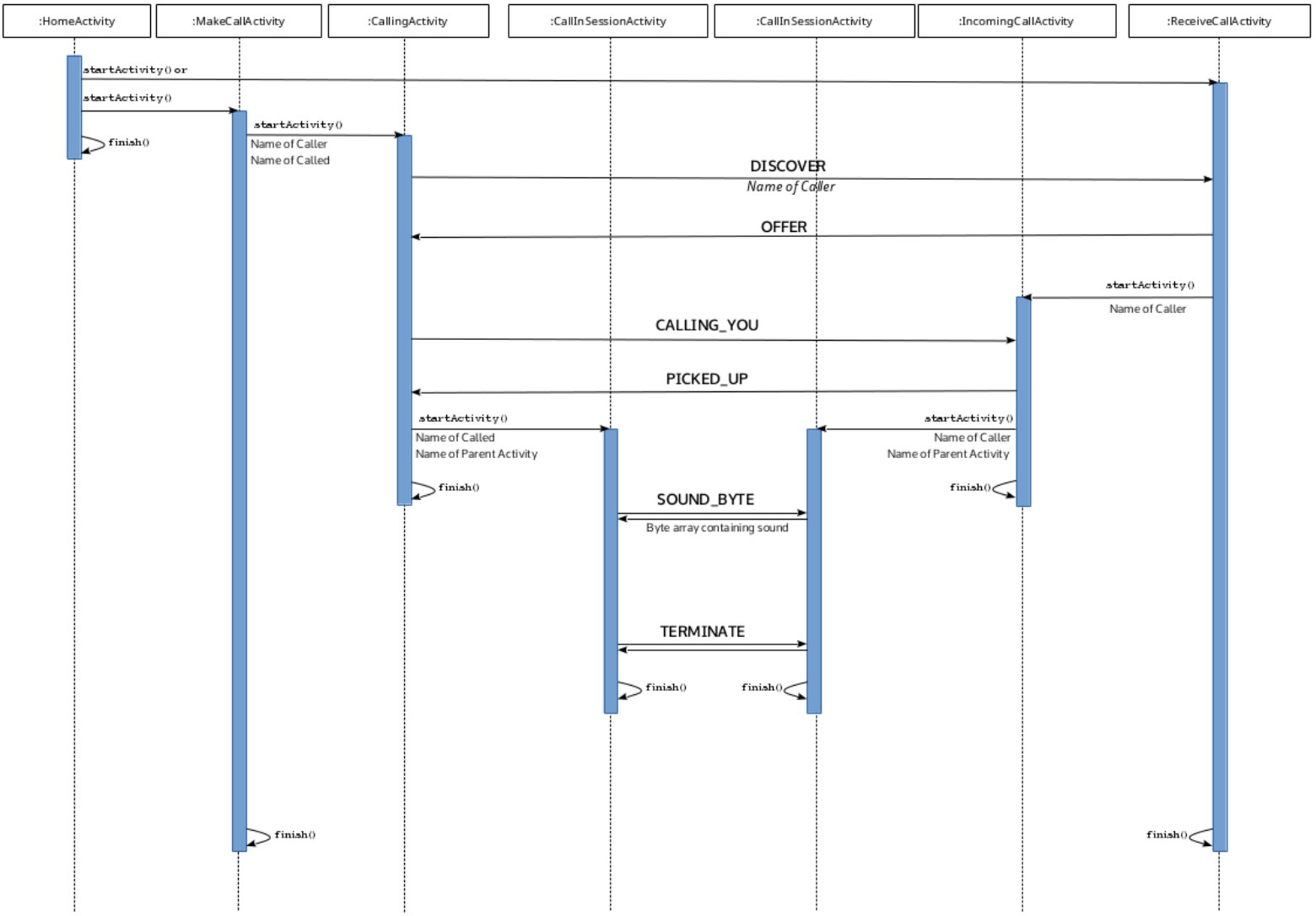

Figure 28: The Sequence Diagram

Component	*Text Style*
Activity object	:HomeActivity
Method	`startActivity()`
Extra passed to activity object	Name of Caller
Network message	**DISCOVER**
Data sent together with network message	*Name of Caller*

Figure 29: The Sequence Diagram Legend

- `CallingActivity`: This activity sends a "DISCOVER" network message to the `ReceiveCallActivity` activity of the individual being called. Along with the "DISCOVER", the `CallingActivity` activity sends the name of the caller to the `ReceiveCallActivity` activity. The `ReceiveCallActivity` activity responds with an "OFFER" which confirms that it is active and has received the name of the called. As the diagram shows, the `ReceiveCallActivity` instance starts an `IncomingCallActivity` activity as soon as it sends the "OFFER". The `CallingActivity` activity then sends a "CALLING YOU" network message, which will be received by an instance of the `IncomingCallActivity`. When the `IncomingCallActivity` activity sends the `CallingActivity` activity a "PICKED UP" in response to the "CALLING YOU", the `CallingActivity` activity starts an instance of the `CallInSessionActivity` activity. The `CallingActivity` passes extras to the `CallInSessionActivity` containing the name of the called individual as well as the name of the parent activity. Since the user got to the `CallInSessionActivity` activity via the `MakeCallActivity` activity, the parent activity in this case is the `MakeCallActivity`. After passing these extras, the `CallingActivity` activity finishes and closes down. Remember that the `MakeCallActivity` instance is still suspended in the background.

- `ReceiveCallActivity`: This activity is started when the user decides to wait to receive a call. When the `ReceiveCallActivity` activity receives a "DISCOVER" from a `CallingActivity` instance, it responds with a "OFFER" and starts an `IncomingCallActivity` instance – passing an extra containing the name of the caller. After starting the `IncomingCallActivity` activity, the `ReceiveCallActivity` activity suspends itself but does not close down.

- `IncomingCallActivity`: This activity waits for a "CALLING YOU" from the `CallingActivity` activity and responds to it with a "PICKED UP" as soon as the called individual picks up the call. After sending the "PICKED UP" the `IncomingCallActivity` instance starts the `CallInSessionActivity` activity, passing extras containing the name of the caller and the name of the parent activity. Since the user got to the `CallInSessionActivity` activity via the `ReceiveCallActivity` activity, the parent activity in this case is the `ReceiveCallActivity`. After passing these extras, the `IncomingCallActivity` activity finishes and closes down. The `ReceiveCallActivity` instance still remains suspended in the background.

- `CallInSessionActivity`: Here is where the actual voice transfer happens. `CallInSessionActivity` instances on the side of the caller and the called alternate in sending sound snippets. They do this by sending to each other a "SOUND BYTE" network message and a byte array containing a sound

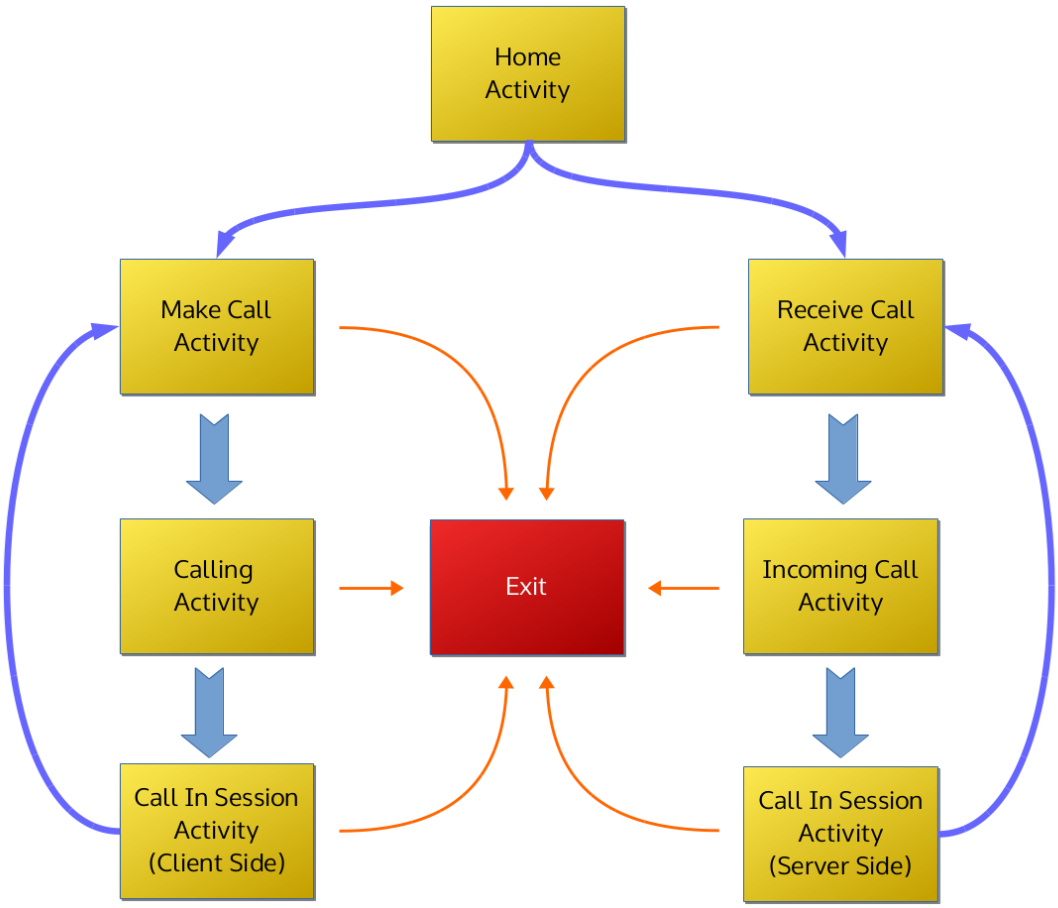

Figure 30: The Navigation Diagram

snippet immediately after that. This happens in fixed intervals so that sound is heard continuously. When the users decide to end the call, both `CallInSessionActivity` instances send each other "TERMINATE" network messages and close down by calling `finish()`. The system then goes back to the corresponding background activity, either `MakeCallActivity` for the caller or `ReceiveCallActivity` for the person being called.

5.5 The Navigation Diagram

An application's navigation diagram shows how a user can maneuver through the said application. Figure 30 shows the navigation options available to the user of the application being created in this project.

The user starts off at the `HomeActivity` activity and selects whether to make a call or to receive a call.

If, on the one hand, the user elects to make a call, they are directed to the `MakeCallActivity` activity where they choose a contact to call. Choosing a person to call leads them to the `CallingActivity` activity where the system attempts to connect to the phone being called. Successful connection leads to the user getting to the `CallInSessionActivity` activity where they can talk with the other individual. As soon as the call ends, the user is returned to the `MakeCallActivity` activity. Unsuccessful connection returns the user to the `MakeCallActivity` activity.

If, on the other hand, the user decides to receive a call, they are taken to the `ReceiveCallActivity` activity where they wait for an individual to call them. When a call comes in, the user is taken to the `IncomingCallActivity` activity where they can decide to reject or accept the call. Should the user reject the call, they are returned to the `ReceiveCallActivity` activity. Should the user accept the call, they are presented with the `CallInSessionActivity` activity where they can talk with the person on the other side of the line. At the end of the call, the user is taken back to the `ReceiveCallActivity` activity to wait for another call.

The orange arrows show that it is – at least currently – possible for the user to exit the system at any time.

6 Implementation

Implementation of this system was done using the Android programming language. This is the programming language used for creating applications for Android smartphones. The Android language has some aspects of the eXtensible Markup Language (XML) language and the Java language. The application that resulted from the coding was tested on a couple of devices running various versions of the Android Operating System. Some of these devices included:

- A Huawei Ideos U8150, running Android version 2.2;
- A Samsung GT-S6790, running Android version 4.1.2; and
- An InnJoo Fire Plus 3G, running Android 4.4.2.

As was established during Research Methodology, this project was to boil down to three experiments.

1. Device connection;
2. Varying of distance with time held constant; and
3. Varying of time with distance held constant.

We will use these experiments to guide us through this implementation section.

6.1 Device connection

Explanation

In this first experiment the idea was to try to see if two Android smart phones can connect and communicate via Wi-Fi without using any third party intermediary devices.

The expected outputs of this experiment were:

- A User Interface (UI) display informing us that the two Android devices have connected with each other.
- Sound from one device playing on the other device.

Our expectations going into this experiment were that the two devices would connect and transmit clear audio to each other.

6.2 Varying of distance with time held constant

Explanation

The idea of this experiment is to see how much distance will affect the quality of communication. We would connect two devices using the application, input a fixed amount of audio data into one of the devices, record the amount of bytes we had input, then record the amount of data that was received on the other device. We would then vary the distance between the two devices to see if physical separation would affect the amount of data transferred between the gadgets. We would limit the maximum distance between the two devices to 100 metres since this was we had established earlier as the maximum WiFi radius.

This experiment had the following expected outputs:

- The amount of bytes lost at every distance.
- Hopefully a graph of the same.

This experiment was done using an Android 2.2 Huawei Ideos U8150 and an Android 4.4.2 InnJoo Fire Plus 3G.

Our expectation for this experiment was that the devices would lose practically no data when they were in close proximity but would suffer data loss as the distance between them increased.

6.3 Varying of time with distance held constant

Explanation

This experiment was to assist us to see whether the application would handle volumes of data gracefully. We would assume that if we vary the length of time recording and sending would take, we would vary the amount of data processed by the system. This would make sense since a five second recording would generate less data than, say, a ten second recording. The plan would be to fix the distance between the two devices and vary the time taken for audio input. We fixed the distance at 50 metres and recorded audio over 100 seconds, noting any differences between the amount of data sent and received.

The outputs anticipated include:

- Data on the amount of data sent and received at various recording times.
- A graph of comparing this data with the various talk times.

Our expectation was that the system was nimble and was robust enough to handle volumes of audio data.

This experiment was also done using an Android 2.2 Huawei Ideos U8150 and an Android 4.4.2 InnJoo Fire Plus 3G.

7 Results and Analysis

The results of the above experiments are found below. Together with them are comments on the results.

7.1 Device connection

After finishing this expected, we expected the following:

- A User Interface (UI) display informing us that the two Android devices have connected with each other.
- Sound from one device playing on the other device.

Before we go far, it has to be mentioned that – as of the time of writing, Tuesday, April 5, 2016 – one has to manually set up the WiFi connection between the two devices before they can communicate. With that fact clearly understood, the following set of screenshots show how the UI looks like as a Huawei Ideos and a Samsung GT-S6790 try to connect to each other using the implemented application. The screenshots on the left are taken from the Samsung while those on the right are from the Huawei. We will use the names of the individual phone's hotspots as the unique identifiers of each of the phones. The Samsung's hotspot name is AndroidAP while the Huawei's hotspot name is Source of Net. The Samsung owner will wait for a call from the Huawei owner while the Huawei owner will try to call the Samsung owner. It should also be noted that the Samsung GT-S6790's screen size is larger than that of the Huawei Ideos U8150. We have tried to increase the size of the Huawei screenshots for better visibility. This accounts for the difference is size of the screenshots.

Results

According to the navigation diagram seen in the Analysis and Design section, at the start both users should see the HomeActivity activity. This shown in Figures 31 and 32. The code snippet below does this.

```
makeCallButton = ( Button ) findViewById( R.id.h_b_make_call );

// begin method makeCallButton.setOnClickListener
makeCallButton.setOnClickListener(

// begin View.OnClickListener
new View.OnClickListener() {

@Override
// begin method onClick
public void onClick( View v ) {

// 1. start the make call activity
// 2. stop this activity

// 1. start the make call activity
```

```
Intent i = new Intent( HomeActivity.this, MakeCallActivity.class );
startActivity( i );

// 2. stop this activity

finish();

} // end method onClick

} // end View.OnClickListener

); // end method makeCallButton.setOnClickListener

receiveCallButton = ( Button )findViewById( R.id.h_b_receive_call );

// begin method receiveCallButton.setOnClickListener
receiveCallButton.setOnClickListener(

// begin View.OnClickListener
new View.OnClickListener() {

@Override
// begin method onClick
public void onClick(View v) {

// 1. start the receive call activity
// 2. stop this activity

// 1. start the receive call activity

Intent i = new Intent(HomeActivity.this, ReceiveCallActivity.class);
startActivity(i);

// 2. stop this activity

finish();

} // end method onClick

} // end View.OnClickListener
```

```
); // end method receiveCallButton.setOnClickListener
```

The Samsung user chooses to receive a call – taking him/her to the ReceiveCallActivity activity. The Huawei user chooses to make a call – taking him/her to the MakeCallActivity activity. The MakeCallActivity activity starts off with no contacts to display. These situations are shown in Figures 33 and 34.

The Samsung continues to wait for a call. The Huawei does a scan of available networks and lists them to the user. This is illustrated in Figures 35 and 36. The code that scans for and displays available networks is shown below.

```
ArrayList< Contact > contacts = new ArrayList< Contact >();

// initialize the wifi manager

final WifiManager wifiManager = (WifiManager) getSystemService( Context.WIFI_SERVIC

...

// scan for networks

// begin try to start an active scan
try {

Method startScanActiveMethod = wifiManager.getClass().getMethod( "startScanActive"

startScanActiveMethod.invoke( wifiManager );

} // end try to start an active scan

// catch any exceptions
catch ( Exception exception ) { exception.printStackTrace();...; }

// 4. get a list of scan results

List< ScanResult > results = wifiManager.getScanResults();

// 5. use the scan results to populate the contacts

// begin if to check if the scan results are available
if ( results != null ) {
```

```
// begin for to go through the scan results
// use each scan result to get a contact
for (int i = 0; i < results.size(); i++) {

Contact newContact = new Contact();

ScanResult currentResult = results.get( i );

newContact.setName( currentResult.SSID );
newContact.setRangeStatus( WifiManager.calculateSignalLevel(currentResult.level, 5

contacts.add( newContact );

} // end for to go through the scan results

} // end if to check if the scan results are available

contacts.trimToSize();

...

// pass context and data to the custom adapter

ContactsArrayAdapter contactsArrayAdapter = new ContactsArrayAdapter( this, gotten

// set list adapter

setListAdapter( contactsArrayAdapter );
```

The Samsung user keeps patiently waiting for a call while the Huawei user selects "AndroidAP" as the name of the hotspot he/she would like to call. We can see this set of events in Figures 37 and 38.

The Samsung user continues waiting for a call to come. The Huawei user is taken to the CallingActivity activity where the app attempts to connect to the "AndroidAP" hotspot. Figures 39 and 40 show us this.

By this time the Samsung has received the "DISCOVER" network message from the Huawei and has responded with an "OFFER" network message. The Samsung switches to the IncomingCallActivity activity and displays it to the user. The Huawei user waits for the Samsung user to pick up the phone. As the screenshots

show, the call coming into the Samsung is from a device called “Source of Net” - this is the name of the Huawei’s hotspot. Figures 41 and 42 show screenshots of this particular stage of navigation. The code that implements the switch from RecieveCallActivity to IncomingCallActivity is shown below.

```
inputRequest = ( String ) localObjectInputStream.readObject();

// begin switch to determine how to respond
switch ( inputRequest ) {

// 3d. when a DISCOVER is received, get the caller’s name immediately after the DIS

// case a DISCOVER is received
case HomeActivity.DISCOVER:

String incomingCallerName = ( String ) localObjectInputStream.readObject();

...

// switch to incoming call

// send an OFFER

sendDataToClient( HomeActivity.OFFER );

// create the Incoming Call Activity

Intent i = new Intent( this, IncomingCallActivity.class );

// pass the caller’s name to the activity

i.putExtra(HomeActivity.CALLER_NAME, getCallerName());

// start the Incoming Call Activity

startActivity(i);

...

} // end switch to determine what to do
```

Since we assume that the Samsung user wants to communicate with the Huawei user, the Samsung user chooses to accept the call from "Source of Net". This leads both devices to display the CallInSessionActivity activity as seen below. The interface has buttons to end the activate the speakerphone, to mute the call, and to end the call. The user is also shown how much time the current call has taken so far. The screenshots in Figures 43 and 44 display what the users see at this point. The code showing how data is sent between CallInSessionActivity instances is here below.

```
// begin method doStopRecordingAndSendRecord
// will stop recording and send the recorded file
public void doStopRecordingAndSendRecord() {

// stop recording sound
currentActivity.stopRecordingSound();

List recordedBytesAsList  = null;

// try to convert the file to a byte array
try { recordedBytesAsList = Arrays.asList( HomeActivity.getByteArrayFromFile( myInt

// catch I/O issues
catch ( IOException e ) { e.printStackTrace(); HomeActivity.logError( SendRecordedS

// send sound to client
currentActivity.sendDataToClient( HomeActivity.SOUND_BYTE );
currentActivity.sendDataToClient( recordedBytesAsList );

// nullify the file
myInternalSendingRecordFile = null;

} // end method doStopRecordingAndSendRecord
```

The code for handling received sound bytes is in the following snippet.

```
inputRequest = ( String ) localObjectInputStream.readObject();

// begin switch to handle the input request
switch ( inputRequest ) {

// 3a. when a SOUND_BYTE is received
```

```
// case sound byte

case HomeActivity.SOUND_BYTE:

// 1. get the list
// 2. convert it to a byte array to the received file's path
// 3. play the received file
// 4. delete the received file

// 1. get the list

// maybe the assignment here puts the array from the client in the first slot of th
List< Object > byteArrayAsList = ( List< Object > ) localObjectInputStream.readObje

// 2. convert it to a byte array to the received file's path

byte[] byteArray = ( byte[] )byteArrayAsList.get( 0 );

if( myInternalReceivingRecordFile == null ) { myInternalReceivingRecordFile = new I

HomeActivity.writeByteArrayToFile(myInternalReceivingRecordFile, byteArray);

// 3. play the received file

FileInputStream fileInputStream = new FileInputStream( myInternalReceivingRecordFil

HomeActivity.logError( CallInSessionActivity.class, "myInternalReceivingRecordFile

startPlayingSound(fileInputStream.getFD());

HomeActivity.logError(CallInSessionActivity.class, "after startPlayingSound(fileInp

// delete the received file

myInternalReceivingRecordFile.delete();
myInternalReceivingRecordFile = null;

break;

...
```

```
} // switch to handle the input request
```

At the end of the call, either user can elect to tap on the "End Call" button. This will send the "TERMINATE" network message to the other user and end the call. After the call ends, the Samsung user will be returned to the ReceiveCallActivity activity to wait for another call, while the Huawei user will be returned to the MakeCallActivity activity to make another call. This fact is highlighted in Figures 45 and 46.

This experiment's results were expected and unexpected. First, they were expected since we were able to connect two devices just as we had proposed at the outset. However, sound transmission between those two devices was found to be intermittent, resulting in inconsistent sound. This was unexpected.

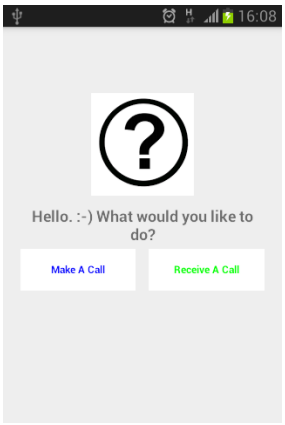

Figure 31: Server Side - `HomeActivity`

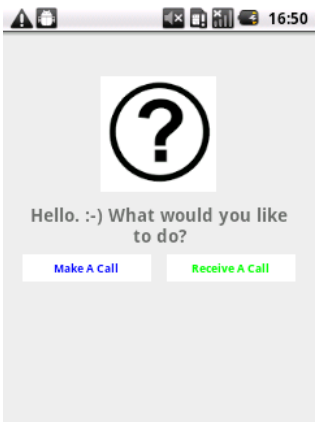

Figure 32: Client Side - `HomeActivity`

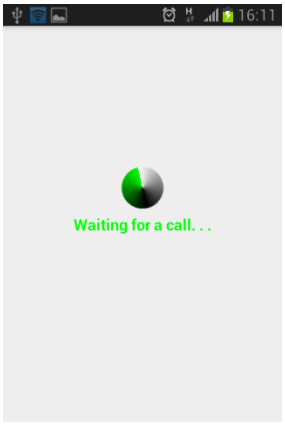

Figure 33: Server Side - `ReceiveCallActivity` - Waiting for a Call

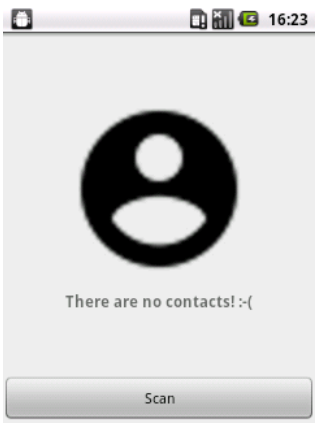

Figure 34: Client Side - `MakeCallActivity` - No Contacts

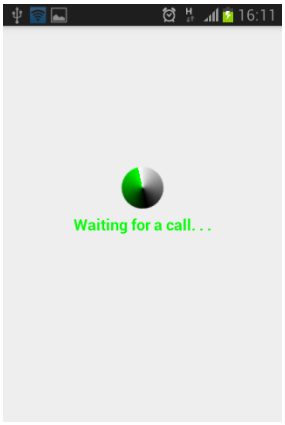

Figure 35: Server Side - `ReceiveCallActivity` - Waiting for a Call

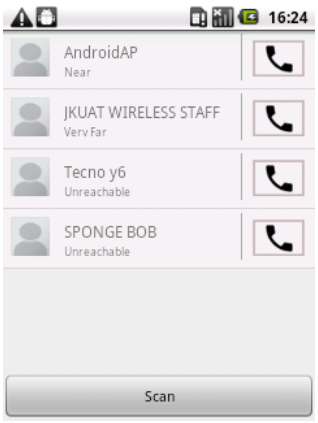

Figure 36: Client Side - `MakeCallActivity` - Displaying Available Contacts

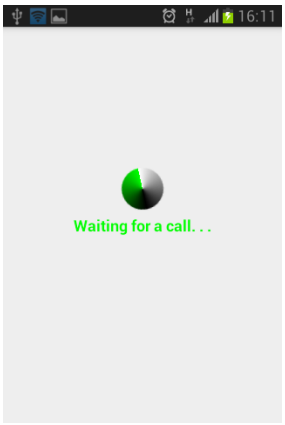

Figure 37: Server Side - `ReceiveCallActivity` - Waiting for a Call

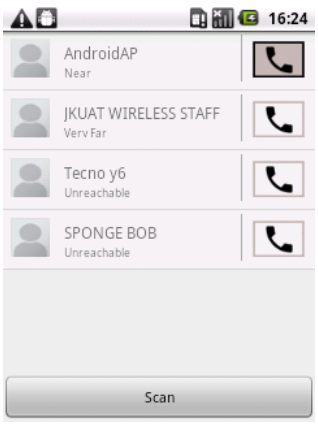

Figure 38: Client Side - `MakeCallActivity` - AndroidAP Selected

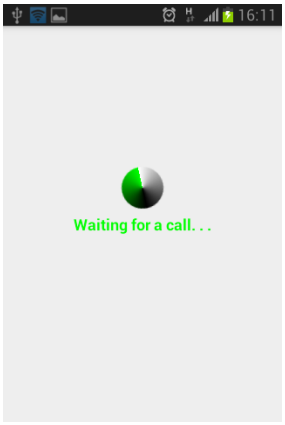

Figure 39: Server Side - `ReceiveCallActivity` - Waiting for a Call

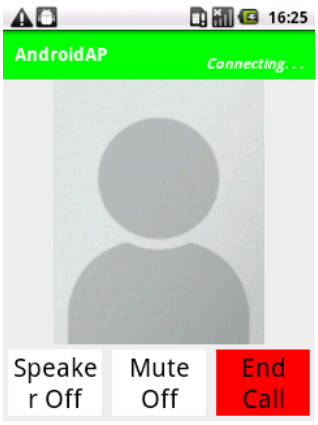

Figure 40: Client Side - `MakeCallActivity` - Connecting to AndroidAP

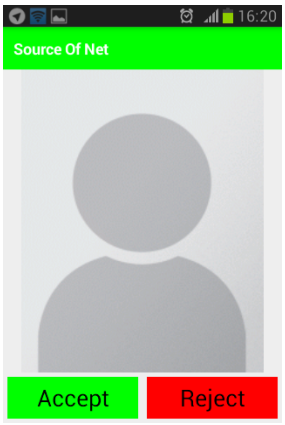

Figure 41: Server Side - `IncomingCallActivity` Being Called by Source of Net

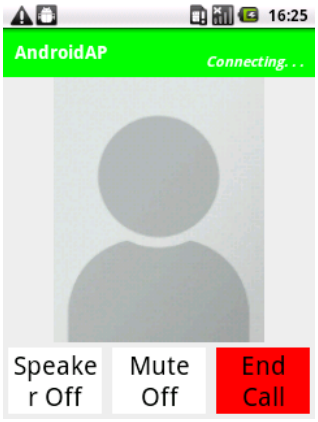

Figure 42: Client Side - `MakeCallActivity` - Connecting to AndroidAP

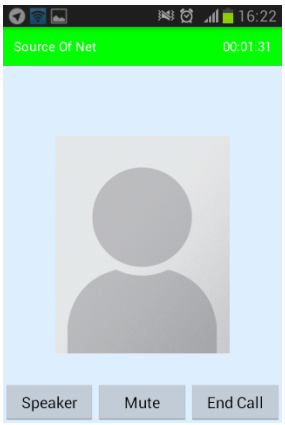

Figure 43: Server Side - `CallInSessionActivity` Call Ongoing

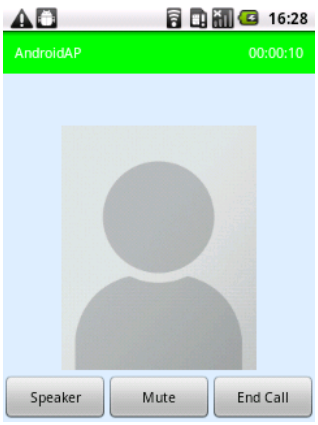

Figure 44: Client Side - `CallInSessionActivity` Call Ongoing

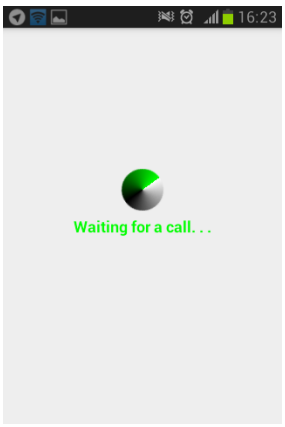

Figure 45: Server Side - `ReceiveCallActivity` After the Call is Over

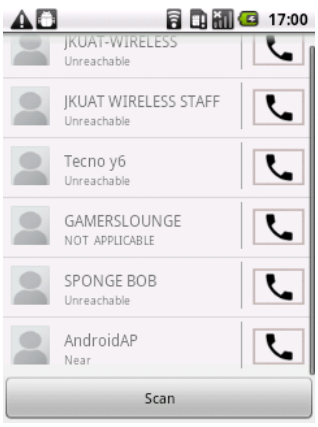

Figure 46: Client Side - `MakeCallActivity` After the Call is Over

Distance (Metres)	Data lost (Bytes)
0	0
10	0
20	0
30	0
40	0
50	0
60	0
70	0
80	0
90	0
100	0

Figure 47: Table of Results of Varying Distance While Holding Time Constant

7.2 Varying of distance with time held constant

This experiment had the following outputs as expected ones:

- The amount of bytes lost at every distance.
- Hopefully a graph of the same.

Results

The data gotten from this experiment was stored in a table. This table is displayed in Figure 47. Frankly the results were a bit surprising. It turned out that no data loss was experienced despite the increase in distance. This might be due to the fact that we used Java Socket objects to create connections between the two devices. According to the Java Application Program Interfaces (APIs), Java Socket classes implement sockets – endpoints for communication between two devices. Java Sockets generally use TCP. This protocol ensures reliability, reliability meaning that all data sent from a sender is received by the receiver no matter the communication challenges such transmissions may face. The use of TCP in Java Sockets may be why no data was lost.

A graphical form of the table in Figure 47 is shown in Figure 48.

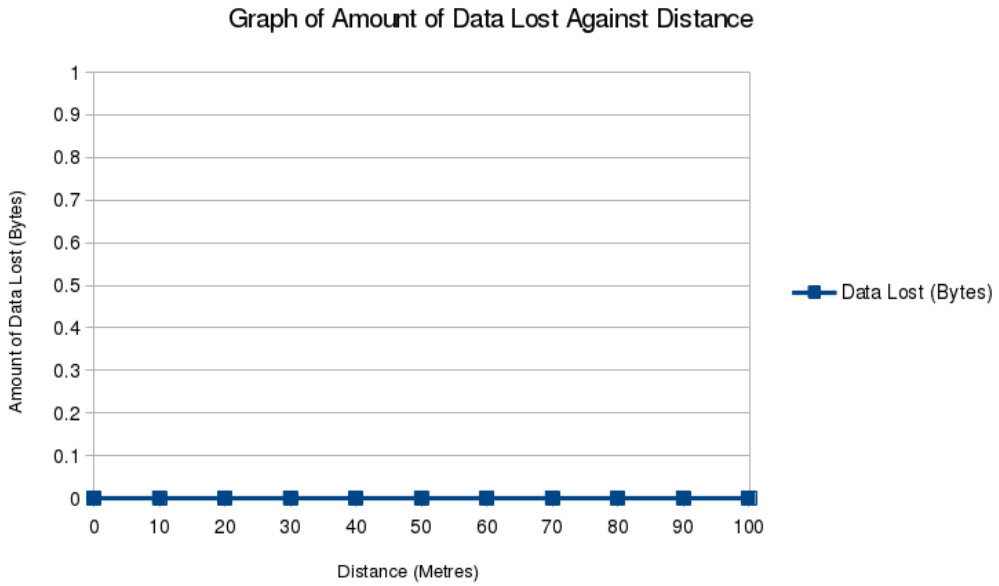

Figure 48: Graph of Results of Varying Distance While Holding Time Constant

7.3 Varying of time with distance held constant

The outputs anticipated are:

- Data on the amount of data sent and received at various recording times.
- A graph of comparing this data with the various talk times.

The result gotten was that over 100 seconds not a byte of data was lost when the device-to-device distance was fixed at 50 metres. This result was as shown in the table in Figure 48.

As can be seen from the table in Figure 49, the amount of data sent from the sender is exactly the same amount received by the receiver. This is what we expected. We again attribute this to TCP.

The graph of this table is seen in Figure 50. In the graph, notice that the lines showing data sent sand data received follow the same path. This is because they have similar values. The two lines have been given different thicknesses and colors to differentiate them.

Time (Seconds)	Data Sent From Sender (Bytes)	Data Received By Receiver (Bytes)
0	2609	2609
10	26090	26090
20	46962	46962
30	70443	70443
40	83488	83488
50	130450	130450
60	140886	140886
70	182630	182630
80	187848	187848
90	211329	211329
100	208720	208720

Figure 49: Table of Results of Varying Time While Holding Distance Constant

Figure 50: Graph of Results of the Varying Distance While Holding Time Constant

7.4 Limitations

Some of the limitations that the implemented system faces are:

1. **Technological limitations.** Each of the devices used had a different Android OS version. This meant that each device's limits were different from the others. What this meant was that we could only implement the features found in the oldest version of the OSs we had – Android 2.2. We therefore missed out on the niftier features of the newer Android releases. This facts fits nicely with our second limitation.
2. **Choppy, intermittent sound.** Due to the use of old Android technology, advancements in Android audio streaming were unreachable for our system. This forced us to play sound only in discrete pieces that were not always audible.
3. **Manual setup.** As mentioned some paragraphs earlier, the application currently needs one to set up the wireless connection manually first before firing up the app. This limitation should be very solvable given enough time.
4. **Restriction to either calling or receiving.** At the moment, users of the app will be restricted to either making a call or receiving a call. They cannot do both. This is due to at least two reasons:
    - The Android 2.2 device used during testing had slow processing power so it could not switch between the WiFi station mode and the WiFi hotspot mode quickly enough.
    - The basic WiFi technology found in most Android devices (including the ones used for this project) is not designed for switching between the aforementioned WiFi modes in real time. So even with high speed gadgets, such a process would be inefficient.
5. **Audio control.** Our team did not get the chance to adequately implement the speakerphone and mute button logic to our satisfaction.

8 Conclusion and Recommendations/Future Work

8.1 Conclusion

Around September 2015 we came up with the idea that became this project. What were the objectives of the project? The objectives were as follows:

- Two smartphones should be able to connect with each other via wireless without the aid of an infrastructure device such as a wireless router or a wireless hotspot.
- The aforementioned smartphones should then be able to send and receive data – initially audio data – between themselves.

The experiments above have established that:

- Two smartphones can connect – or, more accurately, have connected – with each other via wireless without the aid of an infrastructure device such as a wireless router or a wireless hotspot; and
- The aforementioned smartphones have sent and received data – audio data – between themselves.

The project objectives have been met.

8.2 Recommendations

We recommend the following actions for any future interest in this project:

1. **Audio Streaming.** This would increase the functionality of the application immensely. It would make the app viable to the market.
2. **Video Streaming.** This would be another great feature to add to the application. With this in place, people would be able to video chat over short distances.
3. **Audio control.** As mentioned in the Limitations section, we were not able to control audio volume sufficiently. This could be improved.
4. **WiFi Direct.** As mentioned in the Literature Review, this technology has established itself as the next important WiFi technology. We believe that implementing WiFi Direct might solve the choppy audio problem we faced. However, WiFi Direct will mean that we do away with older Android OS versions. This might be a concern Android code in based on Android 2.2 covers almost 100% of all Android devices.

9 Bibliography

References

[1] Amin, A & Khan, M N 2014, 'A Survey of GSM Technology to Control Remote Devices', *International Journal of u- and e- Service, Science and Technology*, vol. 7, no. 5, pp. 153-162, viewed 7 December 2015, http://www.sersc.org/journals/IJUNESST/vol7_no6/14.pdf

[2] Arul Oli, V C K P 2013, 'Wireless Fidelity Real Time Security System', *International Journal of Computer Science Trends and Technology*, vol. 1, no. 1, pp. 43-50, viewed 24 October 2015, http://arxiv.org/ftp/arxiv/papers/1405/1405.1019.pdf

[3] Banerji, S & Chowdhury, R S 2013, 'Wi-Fi and WiMAX: A Comparative Study', *Indian Journal of Engineering*, vol. 2, no. 5, viewed 25 October 2015, http://arxiv.org/ftp/arxiv/papers/1302/1302.2247.pdf

[4] Beck, J & Grajeda, T 2008, *Lowering the Boom: Critical Studies in Film Sound*, p. 43, University of Illinois Press, Champaign, IL, USA.

[5] Birkehammar, C, Bruhn, S, Eneroth, P, Hellwig, K, & Johansson, S 2006, 'New high-quality voice service for mobile networks', *Ericsson Review*, vol. 3, pp. 97-100, viewed 30 October 2015, http://www.ericsson.com/ericsson/corpinfo/publications/review/2006_03/files/2_amrwb.pdf

[6] Brandenburg, K. 1999, 'MP3 and AAC Explained', *Audio Engineering Society Journal*, pp. 1-12, viewed 30 October 2015, https://graphics.ethz.ch/teaching/mmcom12/slides/mp3_and_aac_brandenburg.pdf

[7] Byun, K J, Eo, I S, Bum, J H, & Minsoo, H 2005, 'An Embedded ACELP Speech Coding Based on the AMR-WB Codec', *Electronics and Telecommunications Research Institute (ETRI) Journal*, vol. 27, no. 6, pp. 231-234, viewed 30 October 2015, http://koasas.kaist.ac.kr/bitstream/10203/18247/1/An%20Embedded%20ACELP%20Speech
WB%20Codec.pdf

[8] Camps-Mur, D, Garcia-Saavedra, A, & Serrano, P 2013, 'Device to device communications with WiFi Direct: overview and experimentation', *Wireless Communications Magazine, IEEE*, vol. 20, no. 3, pp. 96-104, viewed 9 November 2015, http://www.it.uc3m.es/pablo/papers/pdf/2012_camps_commag_wifidirect.pdf

[9] Choudhary, D & Kumar, A 2014, 'Study and Performance of AMR Codecs for GSM' *International Journal of Advanced Research in Computer and Communication Engineering*, pp. 8105-8110, viewed 30 October 2015, http://www.ijarcce.com/upload/2014/october/IJARCCE1F%20s%20-divu-%20abhinav%20-%20STUDY%20AND%20PERFORMANCE%20OF%20AMR%20COD

[10] Deitel, P, Deitel, H, Deitel, A & Morgano, M 2012, *Android for Programmers An App-Driven Approach*, Pearsion Education, Inc., Crawfordsville, IN, USA.

[11] ECC 2010, *Compatibility Study for UMTS Operating Within the GSM 900 and GSM 1800 Frequency Bands*, Roskilde: Electronic Communications Committee (ECC), viewed 8 December 2015, http://www.erodocdb.dk/docs/doc98/official/pdf/ECCRep082.pdf

[12] ETSI, 2002, Universal Mobile Telecommunications System (UMTS); AMR speech Codec; General description (3GPP TS 26.071 version 5.0.0 Release 5), Sophia Antipolis Cedex, France, viewed 8 November 2015, http://www.etsi.org/deliver/etsi_ts/126000_126099/126071/05.00.00_60/ts_126071v050000p.p

[13] Geiger, R, Rongshan, Y, Herre, J, Rahardja, S, Sang-Wook, K, Xiao, L, et al 2007, 'ISO/IEC MPEG-4 High-Definition Scalable Advanced Audio Coding', *Audio Engineering Society Journal*, vol. 55, pp. 27-43, viewed 30 October 2015, http://www.ece.rochester.edu/courses/ECE472/Site/Assignments/Entries/2009/1/15_Week_

[14] Herre, J, & Dietz, M 2008, 'MPEG-4 High-Efficiency AAC Coding', *IEEE SIGNAL PROCESSING MAGAZINE*, pp. 137-142, viewed 30 October 2015, http://www.img.lx.it.pt/ fp/cav/ano2008_2009/Trabalhos_MEEC_2009/Artigo_MEEC_11/pa

[15] Ibn Minar, N B & Tarique, M 2012, 'Bluetooth Security Threats ans Solutions: A Survey', *International Journal of Distributed and Parallel Systems (IJDPS)* , vol. 3, no. 1, pp. 127-148, viewed 24 October 2015, http://www.airccse.org/journal/ijdps/papers/0112ijdps10.pdf

[16] ISO/IEC-18092 2013, 'Information technology — Telecommunications and information exchange between systems — Near Field Communication — Interface and Protocol (NFCIP-1)', Geneva, Switzerland, viewed 24 October 2015, http://standards.iso.org/ittf/PubliclyAvailableStandards/c056692_ISO_IEC_18092_2013.zip

[17] Jichkar, B R 2014, 'Paper on Proposed System for Placing Free Call over Wi-Fi Network Using Voip and SIP', *International Journal of Engineering Re-*

search and Applications, vol. 4, no. 1, pp. 132-135, viewed 8 November 2015, http://www.ijera.com/papers/Vol4_issue1/Version%203/V4103132135.pdf

[18] Mehta, A V & Kharote, P R 2014, 'ARM 7 Based MP3 Player', *International Journal of Engineering Research and Applications*, vol. 4, no. 2, pp. 01-05, viewed 13 November 2015, http://www.ijera.com/papers/Vol4_issue2/Version%206/A42060105.pdf

[19] Preethi, K, Sinha, A, & Varma, N 2012, 'Contactless Communication through Near Field Communication', *International Journal of Advanced Research in Computer Science and Software Engineering*, pp. 158-163, viewed 24 October 2015, http://www.ijarcsse.com/docs/papers/April2012/Volume_2_issue_4/V2I40047.pdf

[20] Samanta, S, Mohandas, R, & Pais, A R 2012, 'Secure Short Message Peer-To-Peer Protocol', *International Journal of Electronic Commerce Studies*, vol. 3, no. 1, pp. 45-60, viewed 28 October 2015, http://www.academicjournals.org/ojs2/index.php/ijecs/article/viewFile/1013/101

[21] Singh, P, Sharma, D, & Agrawal, S 2011, 'A Modern Study of Bluetooth Wireless Technology', *International Journal of Computer Science, Engineering and Information Technology*, vol. 1, no. 3, pp. 55-63, viewed 24 October 2015, http://airccse.org/journal/ijcseit/papers/0811ijcseit06.pdf

[22] Skariah, M & Suriyakala, C D 2013, 'An Exploration on Wi-Fi/802.11b and WiMAX/802.16 Networks with Performance Enhancements', *International Journal of Engineering Sciences & Research Technology*, vol. 2, no. 12, pp. 3658-3664, viewed 25 October 2015, http://www.ijesrt.com/issues%20pdf%20file/Archives%202013/dec-2013/71.pdf

[23] SMSForum 2002, 'SMPP v3.4 Protocol Implementation guide for GSM/UMTS.', viewed 28 October 2015, http://opensmpp.org/specs/smppv34_gsmumts_ig_v10.pdf

[24] Song, S & Isaac, B 2014, 'Analysis of Wi-Fi and WIMAX and Wireless Network Coexistence', *International Journal of Computer Networks & Communications (IJCNC)*, vol. 6, no. 6, pp. 63-78, viewed 25 October 2015, http://www.ijesrt.com/issues%20pdf%20file/Archives%202013/dec-2013/71.pdf

[25] Verma, P & Bhatia, J S 2013, 'Design and Development of GPS-GSM Based Tracking System with Google Map', *International Journal of Computer Science, Engineering and Applications*

(IJCSEA), vol. 3, no. 3, pp. 33-40, viewed 7 December 2015, http://airccse.org/journal/ijcsea/papers/3313ijcsea04.pdf

[26] Wi-Fi Alliance 2010, 'Wi-Fi CERTIFIED Wi-Fi Direct™', viewed 9 November 2015, http://www.broadcom.com/blog/wp-content/uploads/2013/10/Wi-Fi-Direct-White-Paper.pdf

[27] Willassen, S Y 2003, Forensics and the GSM mobile telephone system, *International Journal of Digital Evidence*, vol. 2, no. 1, pp. 1-17, viewed 7 December 2015, https://www.utica.edu/academic/institutes/ecii/publications/articles/A0658858-BFF6-C537-7CF86A78D6DE746D.pdf

[28] Youngseok, L & Jongweon, K 2014, 'MP3 File Identification Based on Concurrence Order of Metadata', *International Journal of Multimedia and Ubiquitous Engineering*, vol. 9, no. 9, pp. 41-50, viewed 25 October 2015, http://www.sersc.org/journals/IJSH/vol7_no3_2013/5.pdf