Jomo Kenyatta University of Agriculture and Technology (JKUAT)

Department of Computing

ICS2406: Computer Systems Project

Implementation

REF:JKU/2/83/022

Project Title: Cheaper Exchange of Information Via Wireless Technology

Author:
Name: Kairu Joshua Wambugu Reg. No: CS281-0720/2011
Submission Date: _____ Sign: _____
Course: Bsc. Computer Science

Supervisor 1:
Name: Professor Waweru Mwangi Sign: _____ Date: _____
Supervisor 2:
Name: Doctor Petronilla Muthoni Sign: _____ Date: _____
Supervisor 3:
Name: _____ Sign: _____ Date: _____

Period: June 2015

Contents

1 Background/Introduction

This project, started around September 2015, had the aim of creating a situation where two smartphones would communicate with each other via wireless without using a third party such as a carrier network. At this time, about 7 months later, we present the implementation of the project.

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.

In the following pages and paragraphs, we will consider how the **implementation** of this project went along. We will see some of the **limitations** faced. We will reach a **conclusion** based on the data gotten from implementation. And finally we will make some **recommendations**.

2 Implementation

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

a) Device connection;

b) Varying of distance with time held constant; and

c) Varying of time with distance held constant.

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

2.1 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.

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.

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

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 1 and 2.

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 3 and 4.

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 5 and 6.

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 7 and 8.

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 9 and 10 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 11 and 12 show screenshots of this particular stage of navigation.

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 13 and 14 display what the users see at this point.

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 15 and 16.

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.

2.2 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

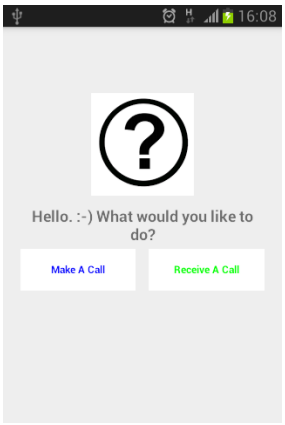

Figure 1: Server Side - HomeActivity

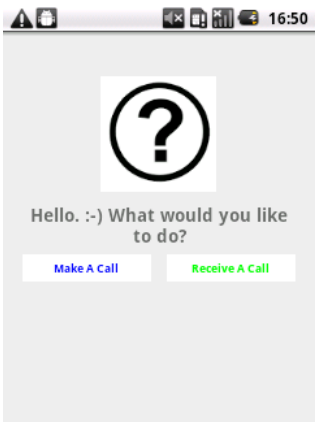

Figure 2: Client Side - HomeActivity

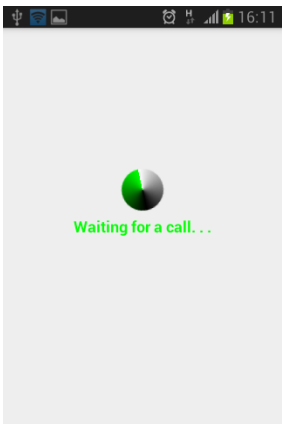

Figure 3: Server Side - ReceiveCallActivity - Waiting for a Call

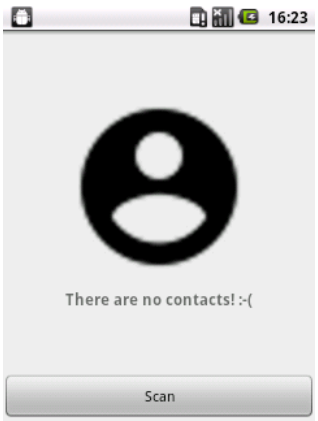

Figure 4: Client Side - MakeCallActivity - No Contacts

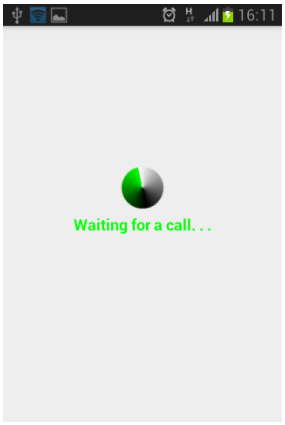

Figure 5: Server Side - ReceiveCallActivity - Waiting for a Call

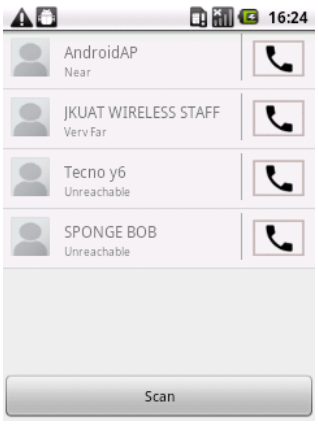

Figure 6: Client Side - MakeCallActivity - Displaying Available Contacts

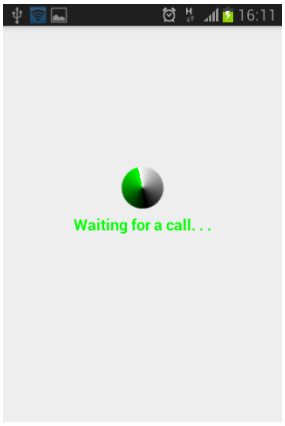

Figure 7: Server Side - ReceiveCallActivity - Waiting for a Call

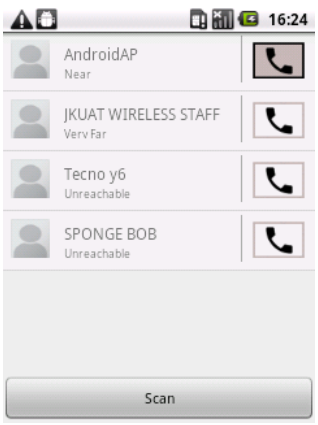

Figure 8: Client Side - MakeCallActivity - AndroidAP Selected

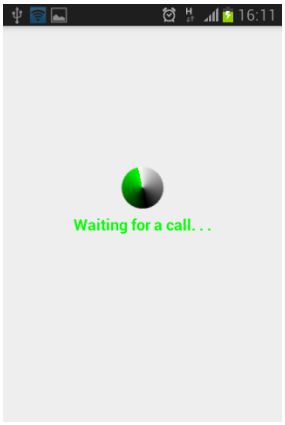

Figure 9: Server Side - ReceiveCallActivity - Waiting for a Call

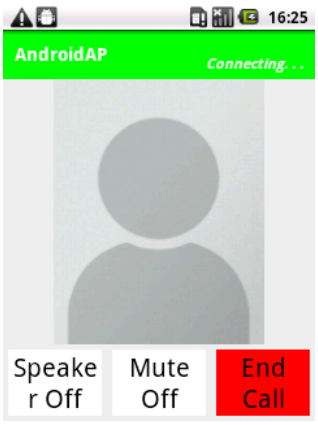

Figure 10: Client Side - MakeCallActivity - Connecting to AndroidAP

Figure 11: Server Side - IncomingCallActivity Being Called by Source of Net

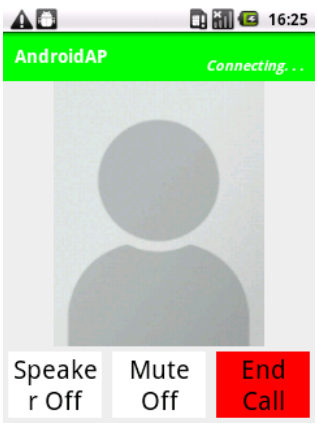

Figure 12: Client Side - MakeCallActivity - Connecting to AndroidAP

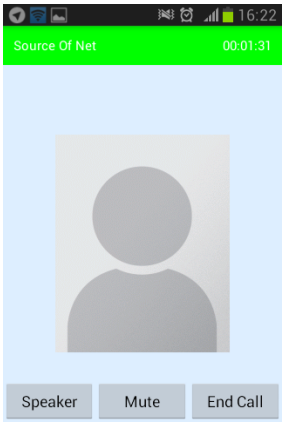

Figure 13: Server Side - CallInSessionActivity Call Ongoing

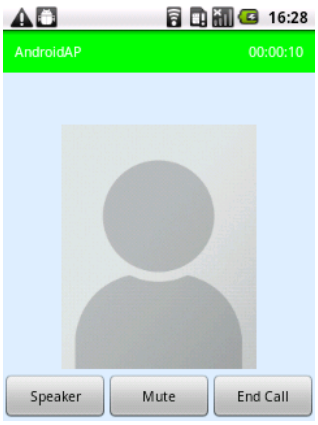

Figure 14: Client Side - CallInSessionActivity Call Ongoing

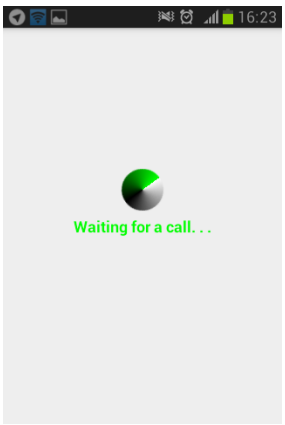

Figure 15: Server Side - ReceiveCallActivity After the Call is Over

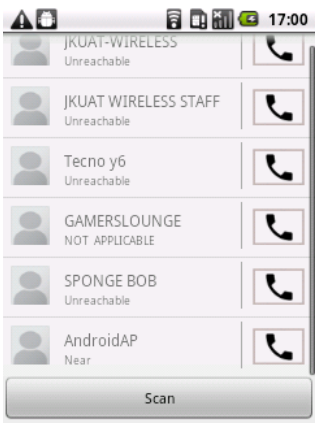

Figure 16: Client Side - MakeCallActivity After the Call is Over

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.

Results

The data gotten from this experiment was stored in a table. This table is displayed in Figure 16. 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 16 is shown in Figure 17.

2.3 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.

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 17: Table of Results of Varying Distance While Holding Time Constant

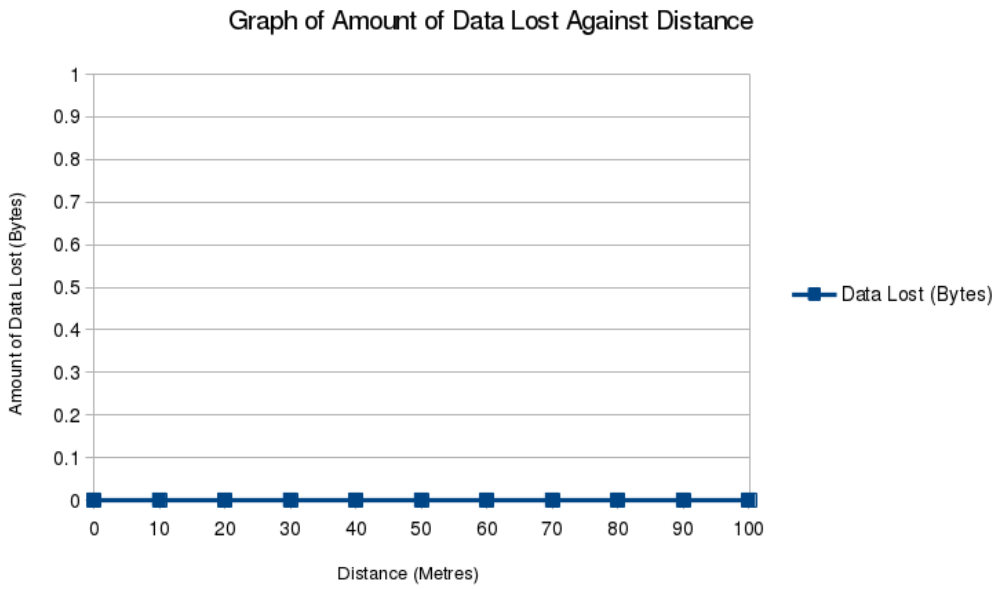

Figure 18: Graph of Results of Varying Distance While Holding Time Constant

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 19: Table of Results of Varying Time While Holding Distance Constant

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.

Results

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 18.

As can be seen from the table in Figure xxx, 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 xxx. 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.

3 Limitations

Some of the limitations that the implemented system faces are:

Figure 20: Graph of Results of the Varying Distance While Holding Time Constant

a) **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.

b) **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.

c) **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.

d) **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.

e) **Audio control.** Our team did not get the chance to adequately implement the speakerphone and mute button logic to our satisfaction.

4 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.

5 Recommendations

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

a) **Audio Streaming.** This would increase the functionality of the application immensely. It would make the app viable to the market.

b) **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.

c) **Audio control.** As mentioned in the Limitations section, we were not able to control audio volume sufficiently. This could be improved.

d) **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.