Mobile Systems Task 5 Report
Communicating with Android Smartphones

Wenchang Liu
ID: 10406141
2019/04/30

Content

1. Overview

This lab is to develop further on the Android app that we made in task 4, allowing it to send text/audio using bit strings via different channel. I also learned simple encryption and decryption in this lab.

2. Implementation

2.1 Introduction

The development has 4 stages, including extending task 4 app to send text via different channels, decryption experiment, showing images and transmit audio.
Development environment:
android studio 3.4.0.18 + ubuntu 18.04 + API 23(emulator) / API 26(real phone)
(You will have to change SERVERIP to local ip of the server if you want to use a real phone)

2.2 Development and Code Documents

2.2.1 Send text messages via channels

This stage I just extend the task 4 with:

1. A text to bit string method:
    - Code Document for this see Answer Question Part 5.1
    - Convert the user message into binary string that can be send to BarryBot via different channels.
2. A bit string to text method:
    - Code Document for this see Answer Question Part 5.1
    - Convert the response(bit string) sent by BarryBot via channels back to text string.
3. Add a drop down list for user to select a specific channel(or none as task 4)
4. In NetworkConnectionAndReceiver class, add if statement to parse responses from BarryBot via channel:

```
} else if(message2.indexOf("BarryBot") == 0) {
    // barry bot reply via channel
    botChannelReply = message2.split( regex: " ");
    style = new SpannableStringBuilder( text: "\n" + "Reply via channel: " + toTextString(botChannelReply[3]));
    style.setSpan(new ForegroundColorSpan(Color.WHITE), start: 0, end: toTextString(botChannelReply[3]).length()+1+19, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
} else if(message2.indexOf("0") == 0 || message2.indexOf("1") == 0) {
    // other user reply via channel
    style = new SpannableStringBuilder( text: "\n" + "Reply via channel: " + toTextString(message2));
    style.setSpan(new ForegroundColorSpan(Color.WHITE), start: 0, end: toTextString(message2).length()+1+19, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
```

Figure 2-1 Code for parsing responses from BarryBot via channel

2.2.2 Decryption

I do the decryption task using telnet and python:

1. First of all, we need to guess the signature, it is easy to guess the “XXX” in the lab manual is “The University of Manchester” and can be verified in the BarryBot terminal logs.
2. Then we can make the python program which takes two arguments, 2 encrypted bit strings s1 and s2, then we can convert the text format signature into bit string version and mark it as a1, since the encryption is to use the unencrypted xor the key, so if we use s1 xor s2, it will have the same value as the unencrypted xor string a1 xor a2, in that way, we can get a2 by using s1 xor s2 xor a1, then we convert a2 back to text to see the decrypted messages.

```
def text_to_bin(string):
    result = ""
    for character in string:
        result += bin(ord(character))[2:].zfill(8)
    return result

# s1, s2
sig_encrpyt = sys.argv[1]
text_encrpyt = sys.argv[2]
# a1 xor a2 = s1 xor s2
xor_string = int(sig_encrpyt, 2) ^ int(text_encrpyt, 2)
s1_xor_s2 = bin(xor_string)[2:].zfill(len(sig_encrpyt))
# signature: Sent by BarryBot, School of Computer Science, The University of Manchester
signature = text_to_bin("Sent by BarryBot, School of Computer Science, The University of Manchester")
# random text = s1 xor s2 xor signature
# print(signature)
text_bin = int(s1_xor_s2, 2) ^ int(signature, 2)
text = bin(text_bin)[2:].zfill(len(signature))
for i in range(0, len(text)-7, 8):
    character = text[i: i+8]
    n = chr(int(character, 2))
    print(n, end="")
```

Figure 2-2 Code for decrypting messages

3. The final step is to observe which bit string is the encrypted string s1 of the signature, to do this, we can constantly use the bit string sent by BarryBot as input s1 and s2 of the python program and see if we can get the signature text, if we do, then we can locate the encrypted version of the signature and then we can decrypt the rest messages.

2.2.3 Images

In this part, since the transmission of pictures have already been done, so we just need to show the images in the imageViews:

1. I create a new activity to show the five images, I just use the provided function msImage to read the raw images into java array and then show them in imageViews.
2. Use intent to switch between activities:

```
// Image button to skip to image activity
Button imgButton = findViewById(R.id.btnImage);
imgButton.setOnClickListener((v) → {
        Intent intent = new Intent( packageContext: MainActivity.this,ImageActivity.class);
        startActivity(intent);
});
```

Figure 2-3 Code for switch activity

We need to Create a new layout file for the new activity, create a new java class for the new activity, edit the manifest file, and finally set java intent to switch to the new activity.

2.2.4 Transmit Audio

In this part, we have to play a wav file, play a audio via java array, smooth damaged speech, do recording and its play back, send the recording to BarryBot and get it back via channels and play 12kHz sample rate version:

1. Media Player: I use a drop down list that allow user to select audio to play, then create two methods for play button and stop button:
   (1) Play: create the selected audio resource if mySound is null, if there are no current playing audio(avoid overlapped audio), then play the new audio resource.

```
Button playButton = findViewById(R.id.btnPlay);
playButton.setOnClickListener((v) → {
            // OnClick actions here
            if(mySound == null) {
                if(audioFile.getSelectedItem().toString().equals("HQ music")) {
                    audioToPlay = R.raw.hqmusic;
                } else if(audioFile.getSelectedItem().toString().equals("damaged speech")) {
                    audioToPlay = R.raw.damagedspeech;
                }
                mySound = MediaPlayer.create(getApplicationContext(), audioToPlay);
            }
            if(mySound.isPlaying() == false) {
                mySound.start();
            }
});
```

Figure 2-4 Code for playing MediaPlayer

   (2) Stop: if mySound is not null, then stop playing the audio, release the resource, and then set msSound back to null so that it can be created again when play button is clicked.

```
// stop playing
public void stopSound(View v) {
    if(mySound != null) {
        mySound.stop();
        mySound.release();
        mySound = null;
    }
}
```

Figure 2-5 Code for stopping MediaPlayer

2. Play array track and smoothing: I just use the provided msSound to read the wav file into a java array instead of using the random samples in the example, and then use the provided arrayplay method to play the java array version audio.

After this, we want to resolve bit errors and lose packets problems, I just called a smoothAudio method before arrayplay is called, in this method, we first fill in the lost packets(200 consecutive zeros), then we smooth the bit errors by checking if a sample is too much different than its neighbors(if data[i] - average of data[i-1] and data[i+1] is larger than the threshold), we will change it to the average value of its neighbors if it is too different(more likely to be a bit error):

```
// Method for dealing with "spikes"
public void smoothAudio() {
    int threshold = 12000;
    int hasError = 1;
    for(i = 200; i < length; i+=200) {
        for(int j = 0; j < 200; j++) {
            if(data[i+j] != 0) {
                hasError = 0;
                break;
            }
        }
        if(hasError==1) {
            for(int j = 0; j < 200; j++) {
                data[i+j] = data[i+j-200];
            }
        }
        hasError = 1;
    }
    for(i = 1; i < length-1; i++) {
        short temp = (short) ((data[i-1] + data[i]) / 2);
        if(Math.abs(data[i] - temp) > threshold) {
            data[i] = temp;
        }
    }
}
```

Figure 2-6 Code for smoothing damaged speech

3. Recording and Play back: I just use the provided code to do this, the short array version of the audio is stored in the soundSample variable.
4. Send the audio to BarryBot via channel:
   First I use a spinner to select the channel like part 5.1, then we will have to divide the soundSample into several packets because the server and BarryBot cannot process a string that is too long.
   We will need 2 method that is very similar to those in part 5.1 that convert short array to bit string and vice versa:

```
// Method for transform short array into 01string
public static String shortToString(short[] data) {
    String str;
    String result = "";
    Log.i( tag: "short to str", Integer.toString(data.length));
    for(int i = 0; i < data.length; i++) {
        str = Integer.toBinaryString( i: 0xFFFF & data[i]);
        while(str.length() < 16) {
            str = "0" + str;
        }
        Log.i( tag: "short to string", Integer.toString(i));
        result = result + str.substring(str.length()-16);
    }
    Log.i( tag: "short to string",  msg: "convert to 01 string completed");
    return result;
}
```

Figure 2-7 Code for converting short array to string

Here we just use Integer.toBinaryString method to convert each short variable in the array to binary string, then we add zeros until its length is 16, then we can concatenate it to the result string that we need to return in the end.

```
// Method for transform bitstring into short array
public static void stringToshort(String data, short[] result) {

    Log.i( tag: "str to short", Integer.toString(data.length()));
    //If the message does not have a length that is a multiple of 8, we can't decrypt it
    if(data.length() % 8 != 0) {
        Log.e( tag: "str to short", msg: "msg's length is not a multiple of 8");
        return;
    }

    //Splits the string into 16 bit segments with spaces in between
    String characters = "";
    for(int i = 0; i < data.length() - 15; i += 16) {
        characters += data.substring(i, i + 16) + " ";
    }

    //Creates a string array with bytes that represent each of the characters in the message
    String[] bytes = characters.split( regex: " ");
    int j = 0;
    for(int i = 0; i < bytes.length; i++) {
        result[j] = (short) Integer.parseInt(bytes[i], radix: 2);
          Log.i("test transfer", Short.toString(result[j]));
        j++;
    }
    Log.i( tag: "str to short", msg: "convert to short array completed");
}
```

Figure 2-8 Code for converting bit string to short array

Here we split the string into segments whose length is 16, the we use parse and cast it to short type, and put it into the result short array that we need to return in the end.

With the help of those methods, what I do is to select a packet of 40 short samples, and then convert it to bit string, and use the transmitter class to send it to the server, then I get the response from the server, convert bit string back to short array and put the short array segments together, and finally we can play the short array we received from BarryBot.

2.3 Testing

1. Send “hello barry bot” to BarryBot via channel 1 2 3 three times:

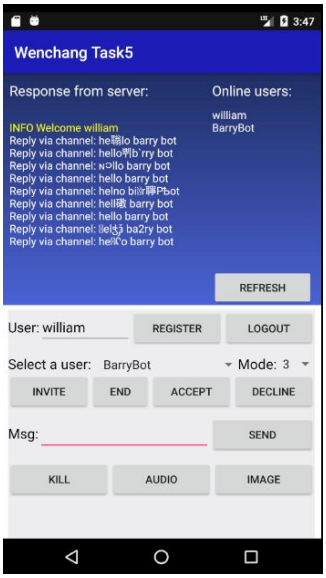

Figure 2-9 Part 5.1 test

We can observe that using channel 1, since the bit error rate is very low, so it will seldom get bit errors; using channel 2, since the bit error rate is high, so every time we will get the message so wrong; using channel 3, the bit error is “burst”, so we will always get part of the message very wrong while the other parts are correct.

2. Decryption using telnet and python: see Answer Question Part 5.2
3. See the images using mobile phone:

Figure 2-10 showing different images

We can see that the original image of jpg and bmp version is similar, however, when we introduce some bit errors, the jpg version will be affected much more than bmp version: every color seems severely wrong in damaged jpg image, but only some particles showed in the damaged bmp image.

4. Play the audio and do the recording: I am afraid that I cannot show the audio playing status in the report. But everything works well.

2.4 Evaluation

All the requirements are implemented and functional, it is just a pity that I do not have more time to adjust my layouts and UI design. The audio transmission part is interesting but also took me much time due to some problems with the server(like mangled messages, buffer sizes, etc.). To be honest, this lab demands a lot of time of effort, and it is so rewarding that I learned how to convert to bit strings, divided into packets and then transmit, etc.

3. Answer Questions

Part 5.1

1. Explain and demonstrate the code for the conversion and re-conversion.
   (1) Convert text to bit string

```java
// method that convert a text string into bit string
public static String toBitString(String inString) {
    //Creates array of all the characters in the message we want to convert to binary
    char[] characters = inString.toCharArray();
    String result = "";
    String preProcessed = "";

    for(int i = 0; i < characters.length; i++) {
        //Converts the character to a binary string
        preProcessed = Integer.toBinaryString((int)characters[i]);
        //Adds enough zeros to the front of the string to make it a byte(length 16 bits)
        String zerosToAdd = "";
        if(preProcessed.length() < 16) {
            for(int j = 0; j < (16 - preProcessed.length()); j++) {
                zerosToAdd += "0";
            }
        }
        result += zerosToAdd + preProcessed;
    }

    //Returns a string with a length that is a multiple of 16
    Log.i(LOGTAG, msg: "convert result: " + result);
    return result;
}
```

Figure 3-1 Code for converting text to bit string

The first thing we do is to convert the String to char array using the “toCharArray” method, then we just convert each “char” in the char array to binary bit string using Integer.toBinaryString method and also adding zeros if its length less than 16, then we can concatenate the bit string fragment into the result string that we need to return in the end. When we finished dealing with the char array, we can return the String result.

(2) Convert bit string to text

```
// method that convert a bit string back to text
public String toTextString(String message) {
    //Check to make sure that the message is all 1s and 0s.
    for(int i = 0; i < message.length(); i++) {
        if(message.charAt(i) != '1' && message.charAt(i) != '0') {
            Log.e(LOGTAG, msg: "msg is all 0 or all 1");
            return null;
        }
    }

    //If the message does not have a length that is a multiple of 8, we can't decrypt it
    if(message.length() % 16 != 0) {
        Log.e(LOGTAG, msg: "msg's length is not a multiple of 16");
        return null;
    }

    //Splits the string into 16 bit segments with spaces in between
    String result = "";
    String characters = "";
    for(int i = 0; i < message.length() - 15; i += 16) {
        characters += message.substring(i, i + 16) + " ";
    }

    //Creates a string array with bytes that represent each of the characters in the message
    String[] bytes = characters.split( regex: " ");
    for(int i = 0; i < bytes.length; i++) {
    // Decrypts each character and adds it to the string to get the original message
        result += (char)Integer.parseInt(bytes[i], radix: 2);
    }
    Log.i(LOGTAG, msg: "convert result is:  " + result);
    return result;
}
```

Figure 3-2 Code for converting bit string to text

First we just need to see if the bit string is normal, say if it is a multiple of 16 or not, then we can cut it into 16 digits segments, and for each segment, we will have to using (char)Integer,parseInt(str, 2) to convert the bit string back to text and add it to the result string. When we finished processing the segments, we can return the text string result.

2. How did you enter the message and did you get back the same message exactly?
    - If using telnet, the message sent through channels shall be bit strings.
    - If sending messages within the app, the message can be text messages, and the app will do the conversion and re-conversion.
    - I can get back the same message exactly if there were no bit errors, it depends on the channel we select.
3. Although the simulated channel should not deliberately introduce any bit-errors in mode 0, what would happen if a real bit error occurred due, perhaps, to a malfunction of the mobile phone? How would your program deal with this occurrence?
    - I think the probability of real bit errors occur is not high, and since we do not have any checking method (e.g. crc) so we cannot deal with this occurrence in this app.
4. How does this form of transmission increase the required bit-rate over the simulated channel?
    - Because we use a 16 digits binary string to represent a “char” while using the simulated channel, and since each digit of the binary string is also a “char”, so we actually increase 15 times more than the original bit-rate.
5. Why is this form of transmission useful for experimental purposes?

- This form of transmission can be used on different types of messages, such as audio, images, and text messages.
- Also it is easy to introduce different kinds of bit-errors or packets losts and easy to track the string transmitted and received via logcat.

6. What are your brief observations about the effect of bit-errors?
    - The low error rate will result in low number of errors, and burst errors will only affect part of the message.
7. How realistic is it to assume that bit-errors will normally be evenly spread out in time?
    - In this server, the bit-errors will be evenly spread in this app, however, it it not likely to be evenly spread in reality, because in reality, the bit-errors will be burst bit-errors(e.g. suddenly a car pass by).

Part 5.2

1. What were the messages?
    - Standard 64-bit WEP uses a 40 bit key (also known as WEP-40), which is concatenated with a 24-bit initialization vector (IV) to form the RC4 key. At the time that the original WEP standard was drafted, the U.S. Government's export restrictions on cryptographic technology limited the key size. Once the restrictions were lifted, manufacturers of access points implemented an extended 128-bit WEP protocol using a 104-bit key size (WEP-104)
    - The signature is: Sent by BarryBot, School of Computer Science, The University of Manchester
2. Assuming that this experiment demonstrates that using the same key more than once is dangerous, how could you avoid this danger while still allowing the receiver to de-encrypt your messages?
    - In this experiment, we can decrypt the message if we know how the transmitter do the encryption, to avoid this danger, we can use asymmetric encryption: the receiver will generate 2 keys, the public key for the transmitter to do encryption and the private key for the receiver to decrypt the message. In this case, even we know how the transmitter encrypt the message(public key), we still cannot know the content if we do not hold the private key.
3. Briefly summarize your de-encryption method and indicate whether it really worked.
    - The explanation of how to develop the python program is in 2.2.2, the method I use is that if messages a1 and a2 are encrypted by the same key k to obtain s1 = a1 xor k and s2 = a2 xor k, then: s1 xor s2 = (a1 xor K) xor (a2 xor k) = (a1 xor a2) xor (k xor k) = a1 xor a2. And in this case we can guess that a1 is the signature(“Sent by Barry Bot, ...”), so I can get any a2 as (a1 xor a2) xor a1. (s1 is one of the bit string we received from BarryBot and we can try it out)
    - First register with telnet and send twice “0MSG BarryBot encrypt” to BarryBot

Figure 3-3 use telnet to send encrypt to BarryBot

- Use python program and the received bit string as input, if we can get a readable text string, then we will know that the encrypted signature is one of the 2 input bit string then we can simply try one of them with another encrypted message to see if it can successfully decrypt the message, then we will know which one is the encrypted signature(s1); and if we do not get a readable string, we will need to send “0MSG BarryBot encrypt” again and again until we can receive the encrypted signature.

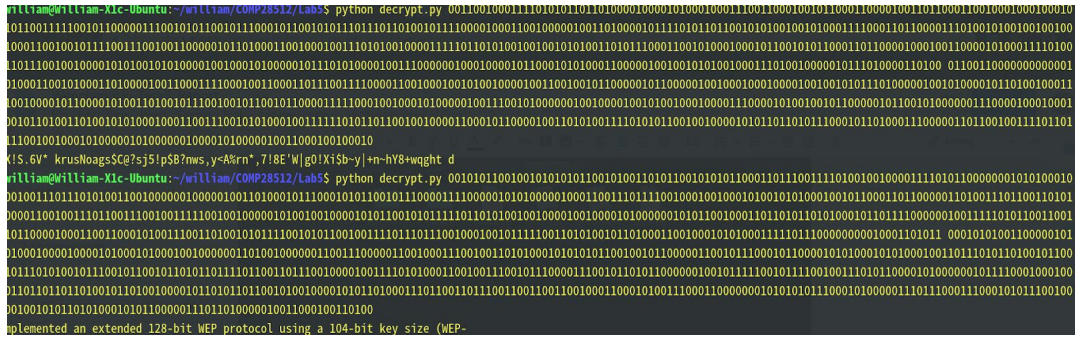

Figure 3-4 Failed to get a readable message

Figure 3-5 Get a readable message and locate s1

- With the encrypted signature bit string, we can use the python program to decrypt any other encrypted text by passing s1 with the bit string you want to decrypt.

4. If you could not predict that an email signature will be sent regularly, what other weaknesses could make the encryption vulnerable to attack.

- It will be difficult to attack but xor is the most vulnerable encryption and there might be some way such as we calculate the frequency of each part of the text, or using other properties(e.g. Locating space if char a1 xor char a2 is still a char(A-Z)).

Part 5.3

1. What happens to your sound player if you keep pressing the ‘start’ button?
    - Since I want the app can still play the audio after pressing “stop”, so I “new” a audio source within the “start” onClick method, so if we keep pressing the “start” button, there will be many playing audio overlapped each other. So, I managed to fix this by adding a “if” statement to see that if there is a audio playing right now, it will only play a new one if the current one is not playing.
2. Why does a ‘wav’ file have a 44 byte header, and how are the sound samples themselves stored? Are the samples stored in text form?
    - The “wav” file needs its 44 byte header to save some data about the audio including information about sample rate, channels, bytes per second, and so on. The header format is called “RIFF” (Resource Interchange File Format), which has RIFF WAVE Chunk, Format Chunk, Fact Chunk(optional), Data Chunk.
    - The sound samples are stored in bytes, they can be regarded as text but not readable.
3. How did you display the original undamaged images?
    - I used the provided msImage method which read the image file into array and then show the image in a imageView.
4. Did you detect any difference between the bit-map image and the jpeg version?
    - No.
5. What was the effect of the bit-errors and lost packets on the bit-map image?
    - There will be particles on the damaged bit-map image.
6. How could you conceal the distortion caused by these transmission-errors (if you had time)?
    - Bit-map can remove the “particles” by comparing the damaged pixel to its surrounding pixels, if it is so different with other pixels, we can replace it with the average of its surrounding pixels.
    - However, for jpg, the colors are so wrong that it is not likely to be concealed.
7. What was the effect of the transmission errors on the JPEG image?
    - The color of the image is completely in chaos.
8. Why are ‘jpg’ images much more sensitive to bit-errors and lost packets than ‘bmp’ images?
    - Because the “jpg” images are compressed image format, however, the “bmp” images are uncompressed image format, so that if an error occurs, it will have more impact on compressed version.

Part 5.4

1. How does the receiver know when a packet has been lost in these experiments, and how does the mechanism provided differ from that used by the real time transport protocol RTP.
    - If there are 200 consequent zeros, it means that this 200 samples packet has lost.
    - RTP mechanism is to let the receiver send ACK message back to the transmitter to ensure that the receiver has received every packet.
2. How successful is your simple method of dealing with lost packets in speech transmissions?
    - If we do not deal with the bit errors, replace the lost packet with the previous packet can even make the audio worse.
3. How successful is your simple method of dealing with bit-errors in speech transmissions?
    - The effects are not that good, it indeed can reduce some damage, especially in the beginning, however, it cannot remove them completely, and sometimes it can even make the damage of the audio worse.

Part 5.5

1. What was, or could be, the effect of the design flaw.
    - The conversion from short array to bit string and vice versa is really slow, so sending a voice message will take a long time due to the server only receive bit string.
    - The buffer of the server and barrybot is still not large enough, so we cannot transmit a voice message just once.
    - The playing java array audio cannot be stop because there is a while loop.
2. How could you correct the design flaw?
    - I think we could directly transmit bits through server not using string.
    - Improve on the server side.
3. Demonstrate the effect of the bit-errors and packet-loss in modes 0 to 5.
    - Mode 0: not bit errors, sound all the same.
    - Mode 1: a few bit errors, contents are clear and noises are less than Mode 2, 3.
    - Mode 2: many bit errors, can hear the contents with loud noises.
    - Mode 3: burst bit errors, contents are clear but noises are loud.
    - Mode 4: lose some packets, not many noises or “spikes”.
    - Mode 5: lose more packets, like the voice is “shaking”, different with bit errors.
4. What was the effect of increasing the sampling rate to 12 kHz on play-back?
    - The voice sounds very differently, and speed of the audio is faster.

5. Why is it useful to be able to develop your own application for speech recording or processing, rather than relying on downloaded apps. Suggest a useful application that you now know how to develop.
    - We can protect our privacy, because we can trust our own server but if the server we use to transmit is owned by others, every message we sent can be seen by the owner of server.
    - We can freely implement functions we like, e.g. send longer voice message a time, famous downloaded apps like wechat can only allow us to send 60s voice message a time.