Android Programming in Bluetooth Cochlea Group

Zijian Zhao

Abstract: My project is mainly android programming work in the Bluetooth Cochlea Group. In this report I will first introduce the background of the bluetooth cochlea. Then I will introduce some related android knowledge. Next I will show the results of my project. Finally I will finish the report with a conclusion.

1 Introduction

There are many people who cannot hear voice all over the world. And every year many babies are born with deafness. So it is always a important task to help these people to hear voice and feel the beautiful world. The artificial cochlea is a surgically implanted electronic device that provides a sense of sound to a person who is profoundly deaf or severely hard of hearing in both ears. And the task of our group is to develop an android application to control the cochlea hardware.

Figure 1: Artificial Cochlea

2 Android Programming

2.1 Android

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. It has been the best-selling OS on tablets and on smartphones since 2013, and has the largest installed base.

Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. The operating system's current design language is Google's Material Design. Android's primary app store is Google Play, with over one million Android applications published and 50 billion downloads as of July 2013. In addition to touchscreen devices, Google has further developed Android for television, cars, and wristwatches,

each with a specialized yet similar interface. Variants and forked versions of Android are also used on notebooks, game consoles, digital cameras, and other electronics.

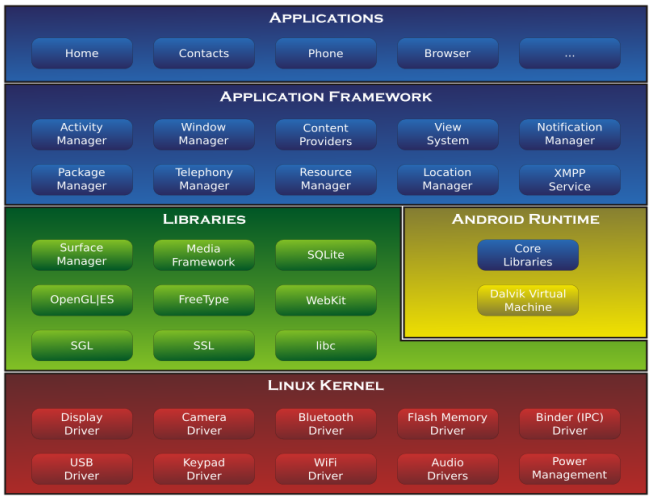


Figure 2: Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.

1. **Linux Kernel**
   At the bottom of the layers is Linux - Linux 2.6 with approximately 115 patches. This provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

2. **Libraries**
   On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

3. **Android Runtime**
   This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android.
   The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.
   The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

4. **Application Framework**
   The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

5. **Applications**
   You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.

2.2 Android Programming Basic

2.2.1 Android Programming Environment

1. Android SDK
   - Provides the Java framework classes
   - Compiles to java bytecode
   - Class framework is updated with every OS release
2. Android NDK
   - C/C++ toolchain for compiling to machine code
3. Android platform tools
   - adb (android debug bridge) : runs and debugs apps from your dev machine
4. Android developer tools
   - Eclipse plug-in for Android
   - Android studio (not yet fully support all NDK features)

2.2.2 Android Components

Components	Description
Activities	They they dictate the UI and handle the user interaction to the smartphone screen
Services	They handle background processing associated with an application.
Broadcast Receivers	They handle communication between Android OS and applications.
Content Providers	They handle data and database management issues.

Application components are the essential building blocks of an Android application. There are following four main components that can be used within an Android application:

1. **Activities**
   An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.
2. **Services**
   A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.
3. **Broadcast Receivers**
   Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

4. **Content Providers**
   A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.

2.3 Android Volley Library

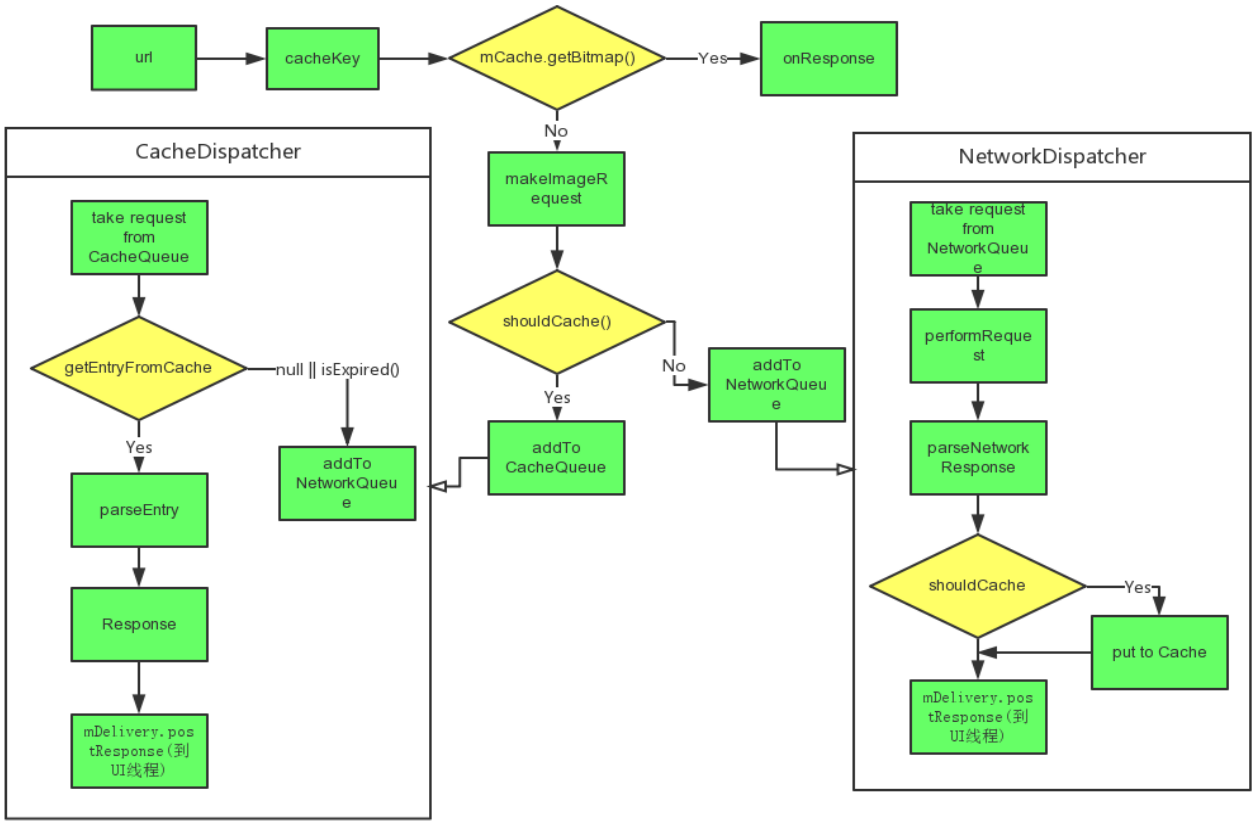


Figure 3: Volley Workflow

Most apps need network access to send and receive data, which is a complex task. Until HoneyComb, network calls ran from the main thread and in subsequent Android versions, network requests executed asynchronously from the main thread. To make a network call, a developer needs to implement an *Asynctask* in a different thread from the main application thread or a *NetworkOnMainThreadException* will be thrown.

Volley will help avoid all this frustrating work. Android volley is a networking library was introduced to make networking calls much easier, faster without writing tons of code. By default all the volley network calls works asynchronously, so we don't have to worry about using *Asynctask* anymore. It bundles the most important features you'll need such as accessing JSON APIs, loading images and String requests in an easier-to-use package.

Designed for RPC (Remote Procedure Call) network operations, Volley is perfect for populating UI elements. It's not for streaming operations like downloading a video or a MP3.

In your application code, you create a Volley Request, queue it on the main thread and receive responses. The work happens on the main thread and this aspect of Volley is perfect for populating the UI of any app because you can't touch the UI from a background thread.

3 Project Work

3.1 Interface of Scene Selection

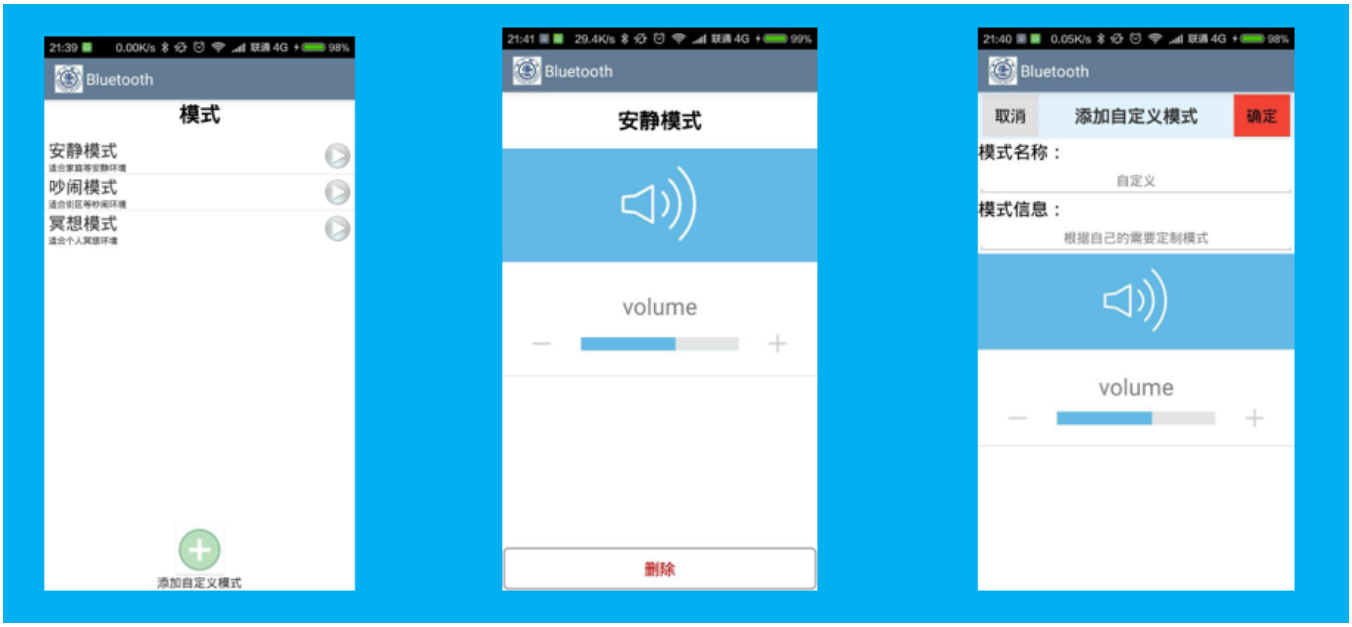


Figure 4: Interface of Scene Selection

This scene selection interface is provided for users to change different sound settings according to the place where they are. For example, if the user is in quiet places such as home, the sound setting should be changed to make the sound smaller; but if the user is in noisy places such as street, the sound setting should be changed to make the sound larger. So we have prepared three scenes, that is quiet scene, noisy scene and meditation scene. The users can edit the detailed settings according to their preference. And the users can also add the own customized scenes.

The work is implemented using the Android ListView. Android **ListView** is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an **Adapter** that pulls content from a source such as an array or database.

An adapter actually bridges between UI components and the data source that fill data into UI Component. Adapter holds the data and send the data to adapter view, the view can takes the data from adapter view and shows the data on different views like as spinner, list view, grid view etc.

The ListView and GridView are subclasses of AdapterView and they can be populated by binding them to an Adapter, which retrieves data from an external source and creates a View that represents each data entry.

Android provides several subclasses of Adapter that are useful for retrieving different kinds of data and building views for an AdapterView (i.e. ListView or GridView). The common adapters are ArrayAdapter, BaseAdapter, CursorAdapter, SimpleCursorAdapter, SpinnerAdapter and WrapperListAdapter. We will see separate examples for both the adapters.

We use the ArrayAdapter as our data source is an array. By default, ArrayAdapter creates a view for each array item by calling toString() on each item and placing the contents in a TextView. Once we have array adapter created, then simply call setAdapter() on our ListView object. On the other hand, we define our list view under layout directory in an XML file. And we implement the click response functions and persistent data manipulations. After all this work, we can get the effect as we see here.

3.2 Login and Register

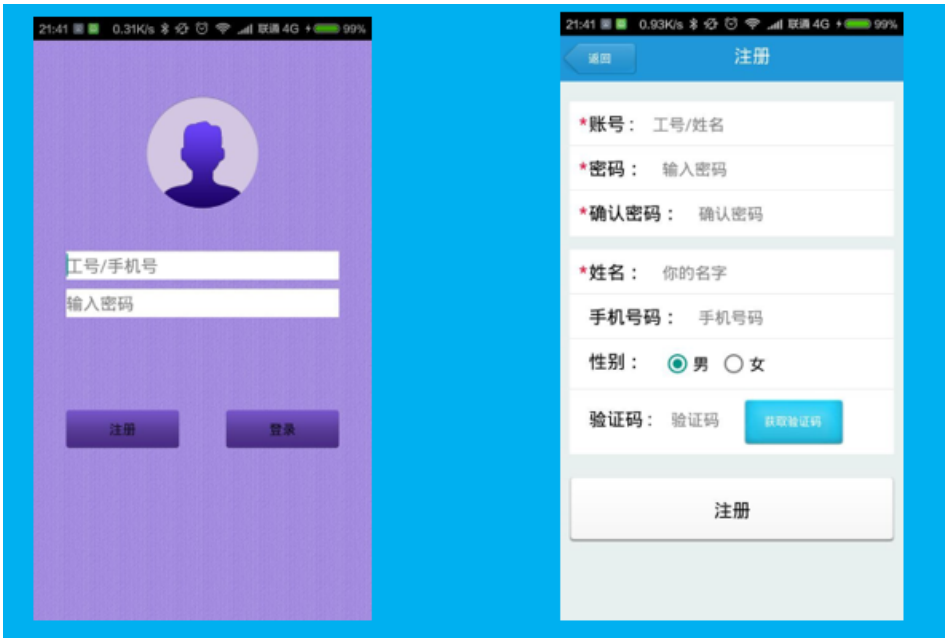
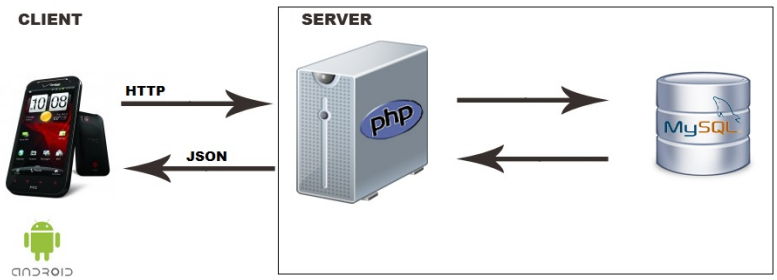


Figure 5: Login and Register

This is the login and register function of our application. There many ways to implement this function but most of them is difficult and inconvenient. However, google has released volley framework to help us solve this problem. Volley is very useful when we want to process data of small size in the network. And since the login and register function only need to transmit very little message, the volley library is a good choice.
The workflow of this function is as follows:

1. The client first send the user account and password information to the server;
2. Then the server query the database to see whether the user is in it or whether the password is matched to the account if the the user is in it;
3. The server then send back the authentication state information to the client;
4. The client deals with the returned information and decide whether it is a successful operation.

We create a *StringRequest* message contains data and operations we need. Then the *RequestManager* add the request to the queue. The at some latter time the request will be picked out for process. The Network will process the request and get *NetworkResponse*. Then *onErrorResponse* and *onResponse* will process the response information to decide what to do next. If we get no errors in this procedure, we will successfully login.



4 Conclusion

In this semester, my project in the Wireless Communications and Mobile Internet course is mainly android programming work in the Bluetooth Cochlea Group. In this project, I learn a lot more about android programming and get some fresh knowledge of the bluetooth cochlea. And we also have group meetings each week. From the meetings I know the interesting things others are doing and learn something from others, such as wireless charging. I believe all these experience are useful whether in study or in research. In summary, I think it is a meaningful project and feel happy to do some work in the project.