An Android-based Instant Message Application

Qi Lai, Mao Zheng and Tom Gendreau
Department of Computer Science
University of Wisconsin - La Crosse
La Crosse, WI 54601
mzheng@uwlax.edu

Abstract

One of the learning outcomes listed by ACM mobile computing education is to "Implement a simple application that relies on mobile and wireless data communications". This Android-based Instant Message Application is aimed to meet this outcome.

Since the Android mobile platform was first open sourced by Google in November 2007, it has attracted more than 180,000 developers and the deployment of 50,000 mobile applications in the Android Market. Today more than 60 smart phones from major manufactures run the Android platform. All these numbers show that the Android project has gained momentum and has moved forward. In addition to its openness, all the tools in the Android development are free and no special hardware is required. These factors motivated us to practice this application on the Android platform.

This paper presents an application that is developed based on the Android operating system, using Eclipse with a plugin. The programming language is Java. The project is tested on an Android Emulator which is a tool that allows developers to easily test an application without having to install the application on a real device.

The Android-based instant message application uses the client/server architecture. The client can add a registered user to be his/her friend and send or receive a text message while the friend is on-line. Currently, the communication in this application is using TCP. A MySQL database is used as a backbone to store the user information.

1 Introduction

One of the learning outcomes listed by ACM mobile computing education [1] is to "Implement a simple application that relies on mobile and wireless data communications".

Smartphone devices such as iPhone, Blackberry, and those that support the Android operating system are ubiquitous. In addition to serving as a phone device, smartphones are also capable of video/picture/text exchanges, accessing the Internet and executing sophisticated embedded software applications. A large percentage of these users are young adults that include college students. Hence, the interest on engaging in the development of the next generation of software applications for embedded and mobiles devices is arising among students.

Since the Android mobile platform was first open sourced by Google in November 2007, it has attracted more than 180,000 developers and the deployment of 50,000 mobile applications in the Android Market. Today more than 60 smart phones from major manufactures run the Android platform. All these numbers show the Android project has gained momentum and has moved forward.

We believe Android provides a rich platform with a variety of concepts, techniques, and resources which can be combined to produce useful and marketable applications. In addition to its openness, all the tools in the Android development are free and no special hardware is required. These factors motivated us to practice an instant message application on the Android platform to explore Android's main components and various building blocks, and to acquire a working knowledge of its developing environment.

2 Android Operating System

The Android operating system is developed by the Open Handset Alliance led by Google. It includes a large set of features for supporting mobile applications. Android consists of a kernel based on the Linux kernel, with middleware, libraries and APIs written in C and application software running on an *Application Framework* which includes Java-compatible libraries based on Apache Harmony [2]. Android uses the Dalvik virtual machine with just-in-time compilation to run compiled Java code. The Android development environment includes a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE. The programming language is Java.

The emulator available in the Android SDK is a tool that allows developers to easily test applications without having to install it to a real device. With the proper configuration for an emulator, it is also possible to test situations which are hard to reproduce on a physical device.

2.1 Android's Main Components

Application components are the essential building blocks of an Android application. There are four different types of application components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed [2].

2.1.1 Activities

An *activity* represents a single screen with a user interface. A multi-screen application will consist of a number of activities that work together to form a cohesive user experience. However, each activity is independent of others. An application can start any one of these activities.

2.1.2 Services

A *service* is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. Example: a service might play music in the background while the user is in a different application. An activity can start the service and let it run or bind to it in order to interact with it.

2.1.3 Content Providers

A *content provider* manages a shared set of application data. Through the content provider, other applications can query or even modify the data (if the content provider allows it).

2.1.4 Broadcast Receivers

A *broadcast receiver* is a component that responds to system-wide broadcast announcements.

Three of the four component types—activities, services, and broadcast receivers—are activated by an asynchronous message called an *intent*. Intents bind individual components to each other at runtime.

2.2 Android Emulator

The Android SDK includes a virtual mobile device emulator that runs on the computer. The emulator lets a developer prototype, develop, and test Android applications without using a physical device.

The Android emulator mimics all of the hardware and software features of a typical mobile device, except that it cannot place actual phone calls. It provides a variety of navigation and control keys, which the user can "press" using the mouse or keyboard to

generate events for an application. It also provides a screen in which an application is displayed, together with any other Android applications running. The emulator utilizes Android Virtual Device (AVD) configurations. AVDs let the developer define certain hardware aspects of the emulated phone and allow the developer to create many configurations to test many Android platforms and hardware permutations.

3 Instant Message Application

The instant message application allows a user to register to the system in order to use the application. After a user has logged into the system, he/she can add another registered user to be his/her friend. The user can then send or receive a text message while the friend is on-line. A MySQL database is used as a backbone to store the user information.

3.1 Architecture

The proposed Instant Message Application uses a Client/Server architecture. The database and web server are on the same machine in this project, but it can also be hosted on different machines. The client can run the application in any other computer, communicate with the server via the network.

3.2 Avatar Display

During the registration, the user can choose a picture from the Android gallery to be his/her avatar. The user can also chop or resize the picture.

The following code shows how to get the photo from Android gallery.

```
protected void doPickPhotoFromGallery() {
        try {
                // Launch picker to choose photo for selected contact
                final Intent intent = getPhotoPickIntent();
                startActivityForResult(intent, PHOTO_PICKED_WITH_DATA);
        } catch (ActivityNotFoundException e) {
                Toast.makeText(this, R.string.photoPickerNotFoundText1,
                                Toast.LENGTH_LONG).show();
        }
}
```

Here, the object ***intent*** is used to pass data from one activity to the other. The method getPhotoPickIntent() is used to get the photo from the gallery.

The photo is stored as String type in the database. In order to retrieve the photo, it needs to convert the photo from String to an array of Byte, then display as Bitmap. The related program segment is shown as below:

String friendphoto = jsonObject.getString("friendpicture"); //get back from database

byte[] friendphotobyte = Base64.decode(friendphoto);

Bitmap photo = getBitmapFromByte(friendphotobyte);

Figure 1 shown below is the screen shot for choosing a picture from Android galley while registering a new user.

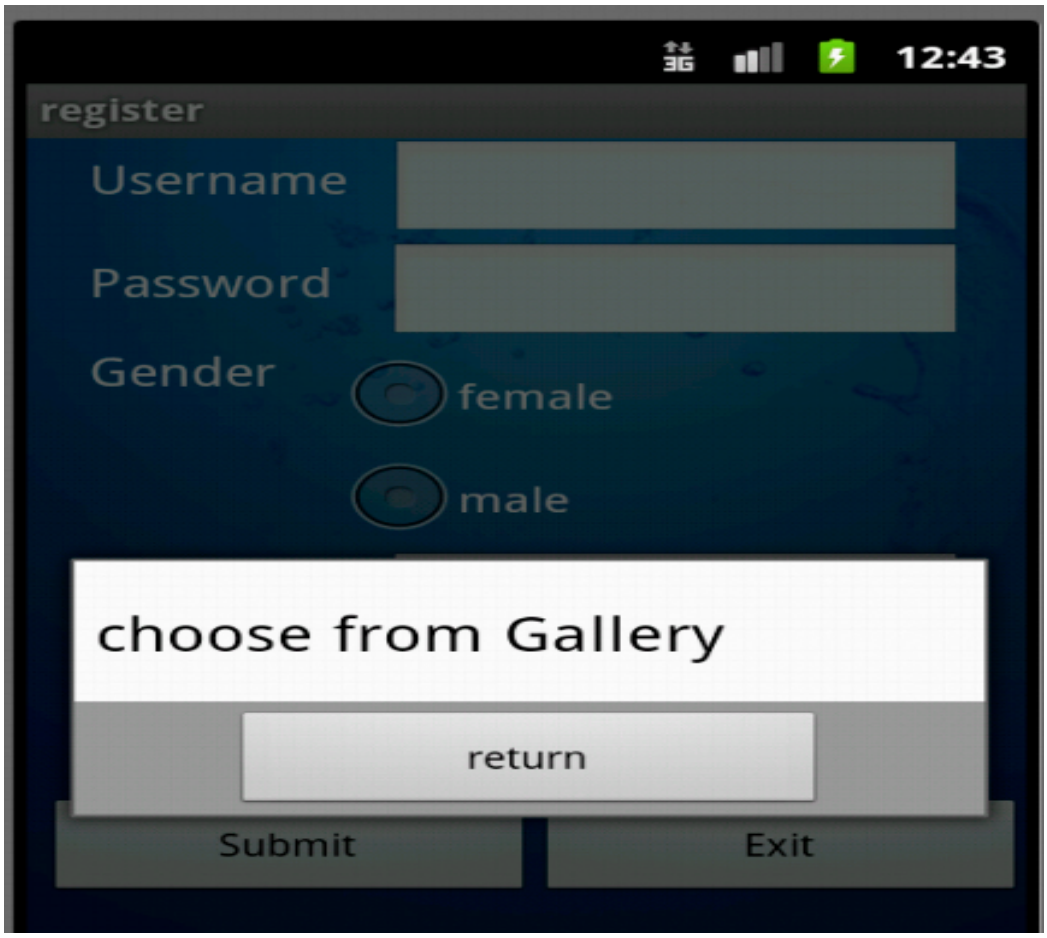

Figure 1: Choose a picture from the Android Galley

3.3 GPS Location

When a registered user login the system, the system will retrieve the user's current location. This is done by using the Dalvik Debug Monitor Server (DDMS) to send the AVD a mock location. DDMS is a debugged tool shipped within Android. Figure 2 below shows how to set the location by manually specifying decimal or sexagesimal longitude and latitude values in the DDMS.

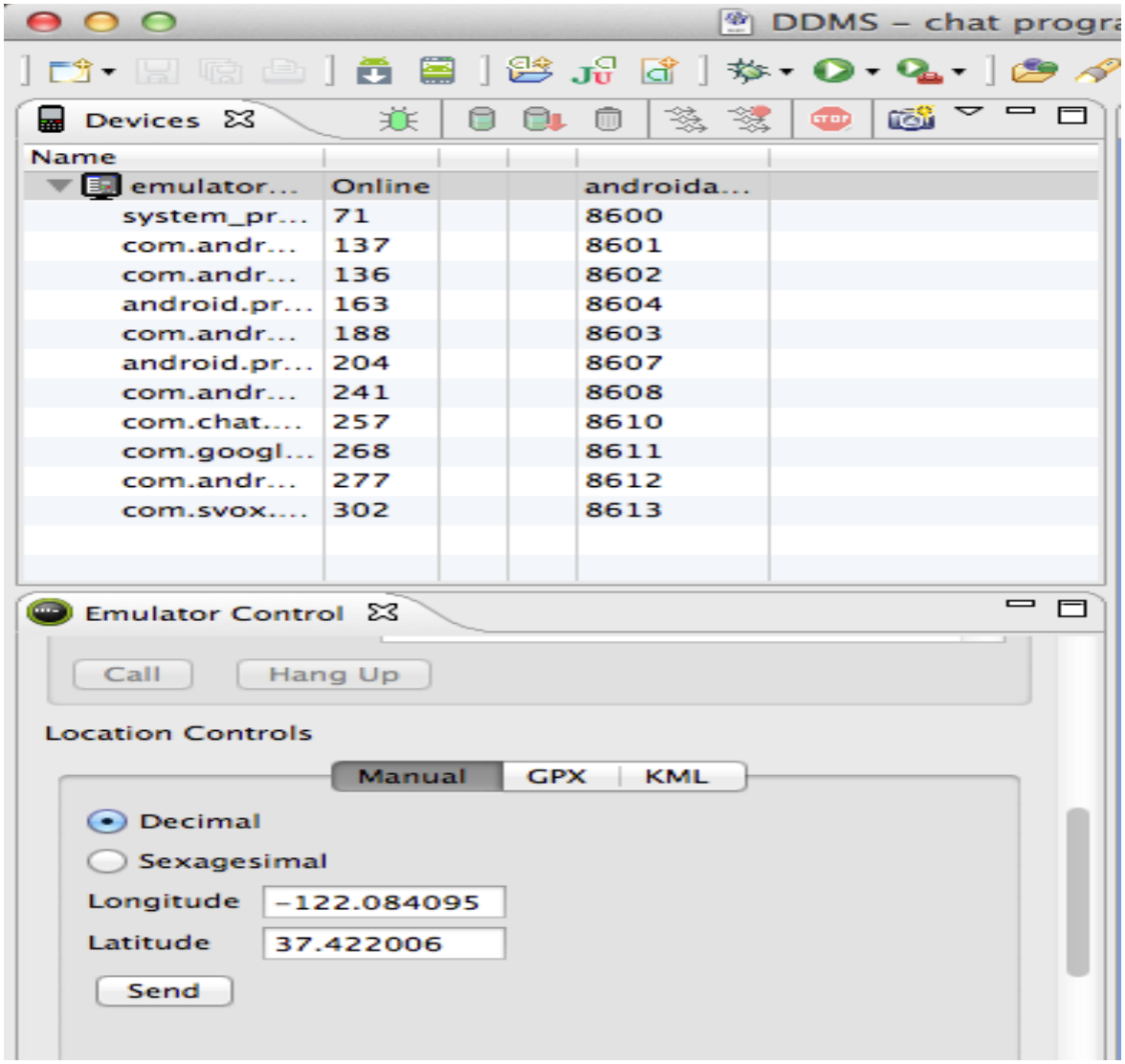

Figure 2: Use DDMS to send a mock location to AVD

Then we use LocationManager to read the user's GPS location. The code segment is shown below:

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
    setContentView(R.layout.gpslocation);
    retrieveLocationButton = (Button) findViewById(R.id.retrieveplaceBtn);
    addplaceBtn = (Button) findViewById(R.id.addplaceBtn);
    addplaceBtn.setOnClickListener(addplaceBtnClick);

   locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
   locationManager.requestLocationUpdates(
            LocationManager.GPS_PROVIDER,
            MINIMUM_TIME_BETWEEN_UPDATES,
            MINIMUM_DISTANCE_CHANGE_FOR_UPDATES,
            new MyLocationListener()
    );
retrieveLocationButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            showCurrentLocation();
        }
});
}
```

We are also able to calculate the distance between a user and his friend based on their GPS locations.

3.4 Communication based on TCP

Currently the communication of the instant message application is using TCP. The client will send the message to the server along with the friend's ID. The server then sends the message to the intended user. A HashMap is used on the server side to store the source and destination clients information. It is shown in the Figure 3 below:

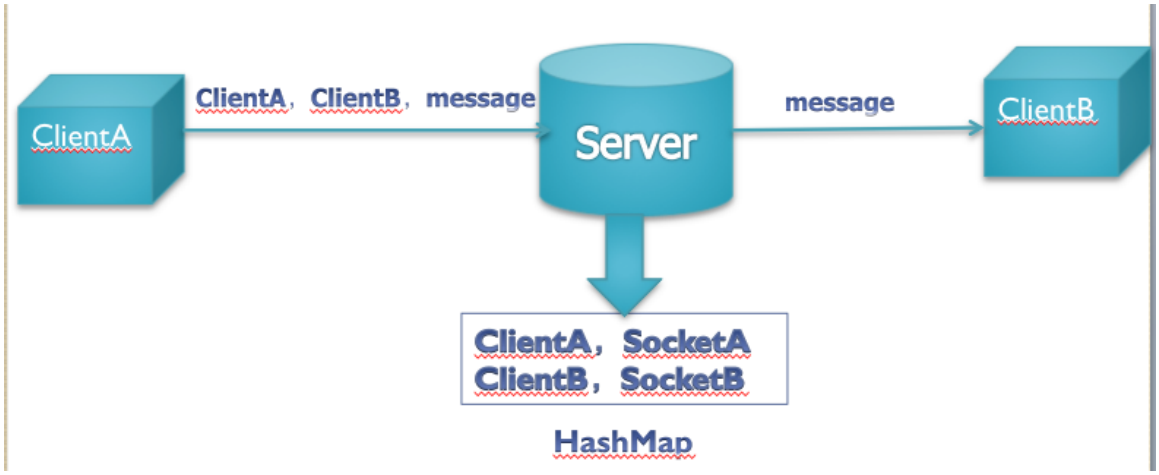

Figure 3: Communication between Two Clients

The associated code segment is shown below.

```
public ClientHandler(Socket socket, Map<String, ClientHandler> clients) throws IOException {

    this.socket = socket;

    this.clients = clients;

    this.os = socket.getOutputStream();

    this.is = new BufferedReader(new InputStreamReader(socket

        .getInputStream()));

  }

public static void main(String[] args) throws IOException {

  final Map<String, ClientHandler> clients = new ConcurrentHashMap<String,
ClientHandler>();

  ServerSocket ss = new ServerSocket(port);

  for (Socket socket = ss.accept(); socket != null; socket = ss.accept()) {

        Runnable handler = new ClientHandler(socket, clients);

         new Thread(handler).start();

  }

}
```

3.5 Communication among Android's Emulators

Android OS assigns each emulator the same private IP address: 10.2.2.15. With this setting, even if two emulators are on the same machine, they can not communicate with each other through a Socket. In order to make two emulators in the same machine communicate, we need to redirect the emulator's port number.

3.5.1 Redirect Emulator's Port Number

Assume the emulator-5554 is the server, and the emulator-5556 is the client. The following steps are needed in the console:

1. telnet localhost 5554
2. redir add tcp:5000:6000

This way, the server's port number 5000 is redirected to be 6000.

Below is the source code on the Client side:

```
public class client extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);


      try {
         Socket socket = new Socket("10.0.2.2", 5000);

          outputStream = socket.getOutputStream();
       } catch (UnknownHostException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      } catch (IOException e) {
         // TODO Auto-generated catch block
         e.printStackTrace();
      }
   }
}
```

Please note: the client side is using the port number 5000. We need to add a line below to the AndroidManifest.xml file.

<uses-permission android:name="android.permission.INTERNET"></uses-permission>

The source code on the Server side is shown below (Remember that we have already redirected the server port 5000 to be 6000:

```
public class server extends Activity {
   /** Called when the activity is first created. */

   public static final int SERVERPORT = 6000;

@Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      try {
      ServerSocket serverSocket = new ServerSocket(SERVERPORT);
      while (true) {
         Socket client = serverSocket.accept();
```

```
    }
}catch(Exception e) {
 System.out.println("S: Error");
   e.printStackTrace();
}
```

4 Conclusion

The current communication in the application is using TCP. However it is not desirable to have all the communication through the server. The next version of this application is to use UDP instead of TCP: once the communication between two clients is established, the messages are directly exchanging between two clients. We also hope to explore audio and video exchange in the next version.

Through the development of this Android-based Instant Message Application, we have gained a working knowledge on Android's API and the developing environment.

References

[1] Computer Science 2008, An Interim Revision of CS 2001(http://www.acm.org/education/curricula/ComputerScience2008.pdf)

[2] Android Developer's Guide. http://developer.android.com/guide/index.html

[3] Abelson, W.F., Collins, C., Sen, R. *Unlocking Android – A Developer's Guide*. Manning Pub. April 2009.

[4] Victor Matos, Rebecca Grasser, Building Applications for the Android OS Mobile Platform: A Primer and Course Materials, *Journal of Computing Sciences in Colleges*, Volume 26 Issue 1, pp: 23-29, October 2010