CSE 486/586 Distributed Systems

Programming Assignment 1

Simple Messenger on Android

Introduction

In this assignment, you will write a simple messenger app on Android. The goal of this app is simple: enabling two Android devices to send messages to each other. The purpose of this assignment is to help you see if you have the right background for this course. If you can finish this all by yourself without getting any help from others, then it is probably the case that you have the right background. Please see for yourself!

There are four high-level challenges in this assignment:

- There will be a lot of reading involved in this assignment. This is because you need to read many tutorials/articles online.
- There will be a lot of infrastructure setup. This involves installing various software components and configuring them.
- There will be many trials and errors to make Android do what you want to do. This is true not just for Android but also for any platform/framework. Luckily, Android provides excellent official documentation. You will need to get used to reading the official developer website.
- There are networking restrictions that the Android emulator environment imposes. You need to get used to it for the assignment.

So here we go!

Step 1: Getting Started

- Unless you are already familiar with Android programming, the first thing you want to do is to follow the tutorials.
    - It's good to follow all the tutorials, but it's not mandatory for this assignment. However, please do follow the ones mentioned below.
- Please follow the first tutorial "Building Your First App" which will guide you through the process of installing necessary software and creating a simple app.
    - Important Note 1: Please use Android Studio. Eclipse is not supported any more.
    - Important Note 2: We use API ~~21~~ 19 for this semester.
    - Important Note 3: There is one part of this tutorial where it instructs you to create an Android emulator instance (or AVD, Android Virtual Device). Please first read both this and this to create an x86-based AVD, not an ARM-based AVD. Reading both articles is important to understand what needs to be done for your setting.

x86-based AVDs run a lot faster with less resources, so it's going to be easier to run on your laptop. For this to work, you will need to have a machine that supports VT-x or equivalent.
    - If you're using Ubuntu 14.xx 64-bit, you might have to install some additional packages. Please refer to this.
- Please follow the fourth tutorial "Managing the Activity Lifecycle" which will give you some basic concepts to understand the code given (as described in Step 2 below).
- Please also follow the tutorial "Saving Data" which you will need from PA2. But this is not necessary for this PA.
- For more information on Android programming, please refer to the following pages.
    - Application Fundamentals
    - Activities
    - Processes and Threads
- For the success of all the programming assignments, it is critical that you know how to use the Android emulator and debug your app because you will spend lots of time using the emulator and debugging. The following pages will give you the information on debugging.
    - Using the Android Emulator
    - Debugging, especially with Android Studio using the Log class.

Step 2: Setting up a Testing Environment

You will need to run two AVDs in order to test your app. Unfortunately, Android does not provide a flexible networking environment for AVDs, so there are some hurdles to jump over in order to set up the right environment. The following are the instructions.

- You need to have the Android SDK and Python 2.x (not 3.x; Python 3.x versions are not compatible with the scripts provided.) installed on your machine. If you have not installed these, please do it first and proceed to the next step.
- Add <your Android SDK directory>/tools to your $PATH so you can run Android's development tools anywhere.
    - A good reference on how to change $PATH is here.
- Add <your Android SDK directory>/platform-tools to your $PATH so you can run Android's platform tools anywhere.
- Download and save create_avd.py, run_avd.py, and set_redir.py.
- To create AVDs, enter: `python create_avd.py 5 <your Android SDK directory>`
    - The above command should be executed only once because you do not need to create AVDs multiple times.
    - In the middle of the script, it will ask "Do you wish to create a custom hardware profile [no]" multiple times. The script handles it automatically, so please do not enter anything to answer that question.
    - 5 AVDs should have been created by the above command. The names are avd0, avd1, avd2, avd3, & avd4. You can manipulate them in Android Studio, but please do not edit or delete them because they are necessary for our scripts to work.

    - If you can't create x86-based AVDs, please enter: `python create_avd.py -a arm 5 <your Android SDK directory>` instead; this will create ARM-based AVDs. However, we strongly encourage you to not do this, but to create x86 AVDs.
- For all the programming assignments except this first one, you will need to run 5 AVDs at the same time. This means that you need to have access to a machine that can handle 5 AVDs running simultaneously.
- In order to test the above, please enter: `python run_avd.py 5`
    - The above command will launch 5 AVDs.
    - Please play around with all 5 AVDs and make sure that your machine can handle them comfortably. Most of the recent laptops will run 5 AVDs without much difficulty.
    - After you are done checking, close all AVDs.
- For this assignment you need two AVDs, so enter: `python run_avd.py 2`
    - The above command will start two AVDs, avd0 & avd1.
    - In general, to run the AVDs created by create_avd.py, enter: `python run_avd.py <number of AVDs>`
- After all AVDs finish launching, create a network that connects the AVDs by entering: `python set_redir.py 10000`
    - The above command will set up port redirections, but there are some restrictions in terms of socket programming. We will detail the restrictions in Step 3 below.

Step 3: Writing the Messenger App

The actual implementation is writing the messenger app. For this, we have a project template you can import to Android Studio.

1. Download the project template zip file to a temporary directory.
2. Extract the template zip file and copy it to your Android Studio projects directory.
    a. Make sure that you copy the correct directory. After unzipping, the directory name should be "SimpleMessenger", and right underneath, it should contain a number of directories and files such as "app", "build", "gradle", "build.gradle", "gradlew", etc.
3. After copying, delete the downloaded template zip file and unzipped directories and files. This is to make sure that you do not submit the template you just downloaded. (There were many people who did this before.)
4. Open the project template you just copied in Android Studio.
5. Add your code to the project. If you look at the template code, there are a few "TODO" sections. Those are the places where you need to add your code.
6. Before implementing anything, please understand the template code first.
    a. The main Activity is in SimpleMessengerActivity.java.
    b. Please read the code and the comments carefully.

The project requirements are below. You must follow everything below exactly. Otherwise, you will get no point on this assignment. And when we say no point, we actually mean it :-)

1. There should be only one app that you develop and need to install for grading. If you just

use the project template and add your code there, you will be able to satisfy this requirement.

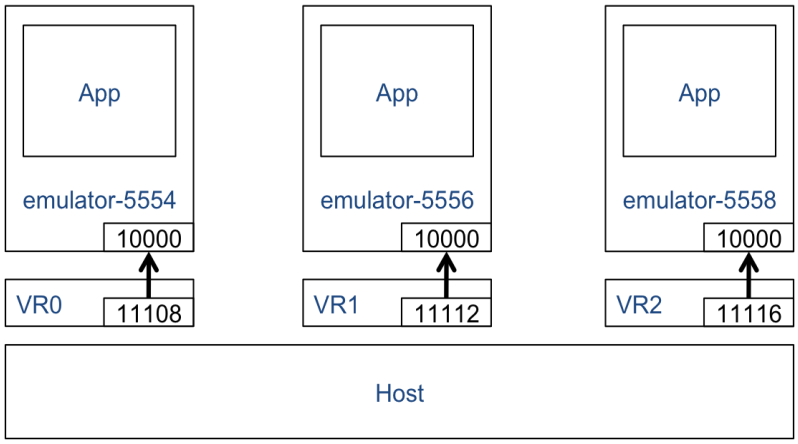
2. When creating your new Android application project in Android Studio, use the following:
    - Application Name: SimpleMessenger
    - Project Name: SimpleMessenger
    - Package Name: edu.buffalo.cse.cse486586.simplemessenger
    - API level ~~21~~ 19 as the minimum & target SDK.
    - If you just use the project template, you will be able to satisfy this requirement.
3. There should be only one text box on screen where a user of the device can write a text message to the other device. If you just use the project template, you will satisfy this requirement.
4. The other device should be able to display on screen what was received and vice versa. The project template contains basic code for displaying messages on screen.
5. You need to use the Java Socket API.
6. All communication should be over TCP.
7. You can assume that the size of a message will never exceed 128 bytes (characters).

As mentioned above, the Android emulator environment is not flexible for networking among multiple AVDs. Although set_redir.py enables networking among multiple AVDs, it is very different from a typical networking setup. When you write your socket code, you have the following restrictions:

- In your app, you can open only one server socket that listens on port 10000 regardless of which AVD your app runs on.
- The app on avd0 can connect to the listening server socket of the app on avd1 by connecting to <ip>:<port> == 10.0.2.2:11112.
- The app on avd1 can connect to the listening server socket of the app on avd0 by connecting to <ip>:<port> == 10.0.2.2:11108.
- Your app knows which AVD it is running on via the following code snippet. If portStr is “5554”, then it is avd0. If portStr is “5556”, then it is avd1:

```
TelephonyManager tel =
        (TelephonyManager) this.getSystemService(Context.TELEPHONY_SERVICE);
String portStr = tel.getLine1Number().substring(tel.getLine1Number().length() - 4);
```

- The project template already implements the above, but you are expected to understand the code as this is critical for the rest of the programming assignments.
- This document explains the Android emulator networking environment in more detail.
- In general, set_redir.py creates an emulated, port-redirected network like this (VR stands for Virtual Router):



Testing

We have testing programs to help you see how your code does with our grading criteria. If you find any rough edge with the testing programs, please report it on Piazza so the teaching staff can fix it. The instructions are the following:

1. Download a testing program for your platform. If your platform does not run it, please report it on Piazza.
    a. Windows: We've tested it on 64-bit Windows 8.
    b. Linux: We've tested it on 64-bit Ubuntu 12.04.
    c. OS X: We've tested it on 64-bit OS X 10.9 Mavericks.
2. Before you run the program, please make sure that you are running two AVDs (avd0 & avd1). `python run_avd.py 2` will do it.
3. Please also make sure that you have installed your SimpleMessenger on those two AVDs.
4. Run the testing program from the command line.
5. At the end of the run, it will give you one of the three outputs.
    a. No communication verified: if SimpleMessenger instances cannot communicate with each other. This is 0 point.
    b. One-way communication verified: if SimpleMessenger on avd0 can send a message to SimpleMessenger on avd1. This is 2 points.
    c. Two-way communication verified: if both AVDs can communicate with each other. This is additional 3 points.

Submission

We use the CSE submit script. You need to use either "submit_cse486" or "submit_cse586", depending on your registration status. If you haven't used it, the instructions on how to use it is here: https://wiki.cse.buffalo.edu/services/content/submit-script

You need to submit one file described below. Once again, you must follow everything below exactly. Otherwise, you will get no point on this assignment.

- Your entire Android Studio project source code tree zipped up in .zip: The name should be SimpleMessenger.zip.
    a. Never create your zip file from inside “SimpleMessenger” directory.
    b. Instead, make sure to zip “SimpleMessenger” directory itself. This means that you need to go to the directory that contains “SimpleMessenger” directory and zip it from there.
    c. Please do not use any other compression tool other than zip, i.e., no 7-Zip, no RAR, etc.

Deadline: ~~2/1/2016 (Monday) 11:59:59am~~ 2/5/2016 (Friday) 11:59:59am

This is one hour before class. The deadline is firm; if your timestamp is 12pm, it is a late submission.

Grading

This assignment is 5% of your final grade. The breakdown for this assignment is:

- 2%: if your messenger app can send one-way messages from avd0 to avd1.
- (additional) 3%: if your messenger app can send two-way messages between avd0 and avd1.

Notes

- There is a cap on the number of AsyncTasks that can run at the same time, even when you use THREAD_POOL_EXECUTOR. The limit is "roughly" 5. Thus, if you need to create more than 5 AsyncTasks (roughly, once again), then you will have to use something else like Thread. However, I really do not think that it is necessary to create that many AsyncTasks for the PAs in this course. Thus, if your code doesn't work because you hit the AsyncTask limit, then please think hard why you're creating that many threads in the first place.

  This document gives you more details on the limit and you might (or might not, depending on your background) understand why I say it's "roughly" 5.

  http://developer.android.com/reference/java/util/concurrent/ThreadPoolExecutor.html
  (Read "Core and maximum pool sizes.")
- For Windows users: In the past, it was discovered that sometimes you cannot run a grader and Android Studio at the same time. As far as I know, this happens rarely, but there is no guarantee that you will not encounter this issue. Thus, if you think that a grader is not running properly and you don't know why, first try closing Android Studio and run the grader.