Joon Ki Hong
CS 4803 Project 1 Design Doc
Professor Kishore

Project 1: Networked Android App

Purpose:

The purpose of this Android application is to create a basic TCP client that communicates with a provided server (IP: 143.215.129.129:8080) by sending a message and displaying the returning result from the server. This is to demonstrate a basic understanding of the higher-level framework of an Android app as well as basic networking concepts.

APIs or packages used:

- Critical packages
  - Java.net
    - Java.net provided the Socket class, which was essential to establishing a connection with the server. This was the first step to creating the TCP connection.
  - Android.os.AsyncTask
    - As of Android 3.0 (Honeycomb), android requires asynchronous tasks to be abstracted into an AsyncTask object. If not the app will crash. The Async Task decouples the task, in our case the message transmission and reception, from the UI thread. It then uses inherited methods to update the UI post-task.
  - Java.io
    - Java.io allowed the app to use the input and output streams provided by the socket.
- Non-critical packages:
  - Android.Log
    - Logs allow any errors or warnings to be shown by Logcat in Eclipse. This was very useful for debugging.
  - Android.view
    - View allowed the app to determine which visible element was being selected (Button listener).
  - Android.widget
    - Widget allowed the app to use buttons. In this app, the single button initiates the socket and sends the message.
- Default Android packages
  - Android.os.Bundle
  - Android.app.Activity

Application Flow:

The application is made up of two text fields and one button. The top text field represents the message being sent whereas the bottom text field, which by default shows 'result…', is replaced by the messaged received from the server. When the button is clicked, the app initializes an AsyncTask object and sends/retrieves the message in the background, separate from the UI thread so that the app doesn't lock up. This is done in the doInBackground method in the AsyncSend class (extension of AsyncTask). The doInBackground method closes the socket then returns the retrieved message to the onPostExecute method to update the bottom text field with the retrieved message (done in the UI thread once asynchronous execution is complete).

Difficulties/Solutions:

The initial hurdle to overcome was getting situated with learning the high level constructs of a simple Android app. Things like the various states (onCreate, onStop, etc.). The next hurdle was getting familiar with the Java Socket API and learning how to use the socket to communicate to the server. The largest hurdle however was dealing with the AsyncTask. Initially I wrote the entire codebase within the onCreate method thinking the amount of time the message transmission would stall the app would be insignificant. However, this had several design issues. The first issue would be that the transmission would only work once, specifically when the app is first started up. The second issue was that the message transmission was being done in the UI thread. This would cause the app to lock up depending on how long the transmission takes. As of Android 3.0 however, this is not allowed and an AsyncTask is required to decouple the transmission from the UI thread properly.