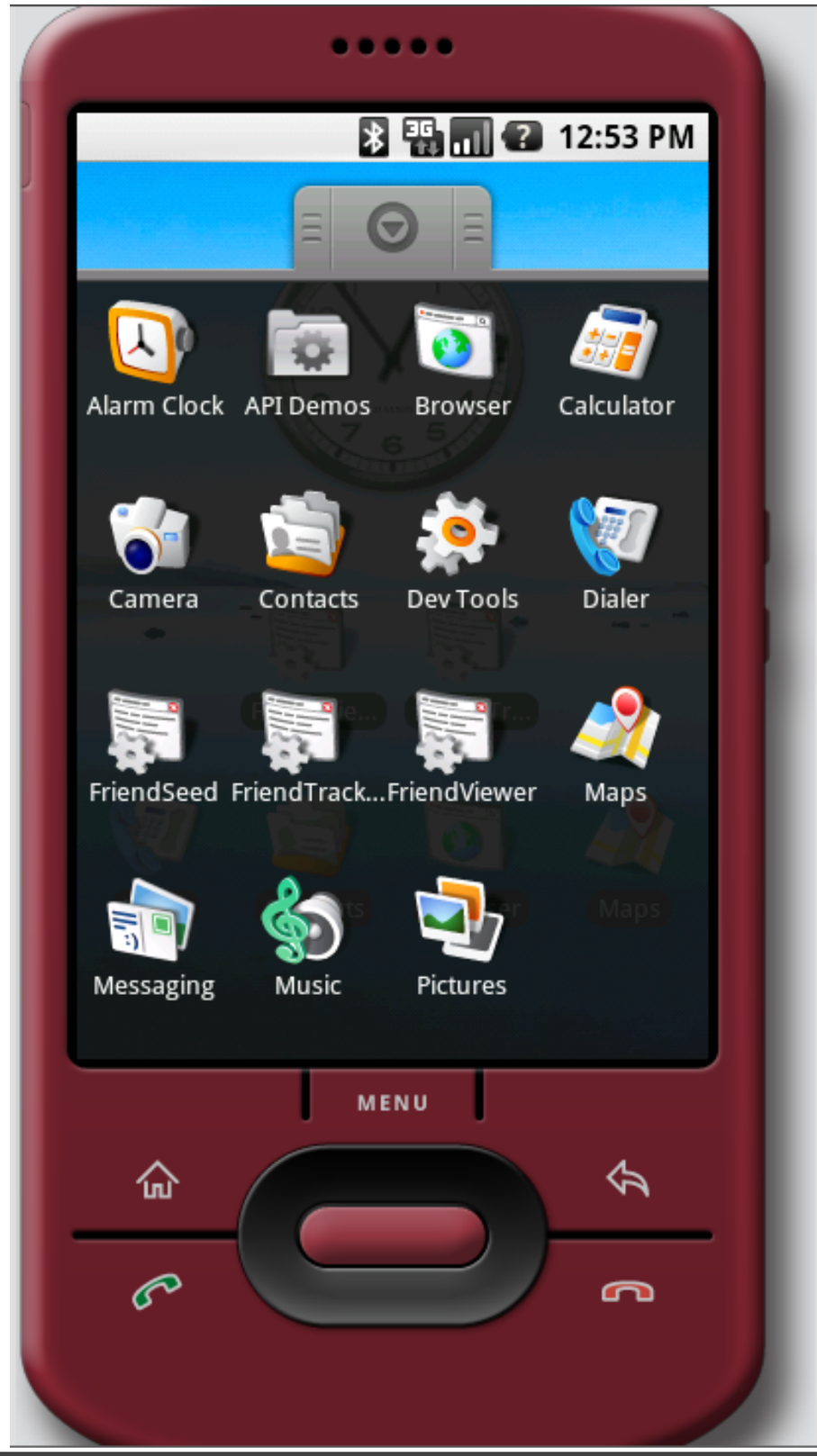
Intro to Developing Android Applications

William Enck

October 2014

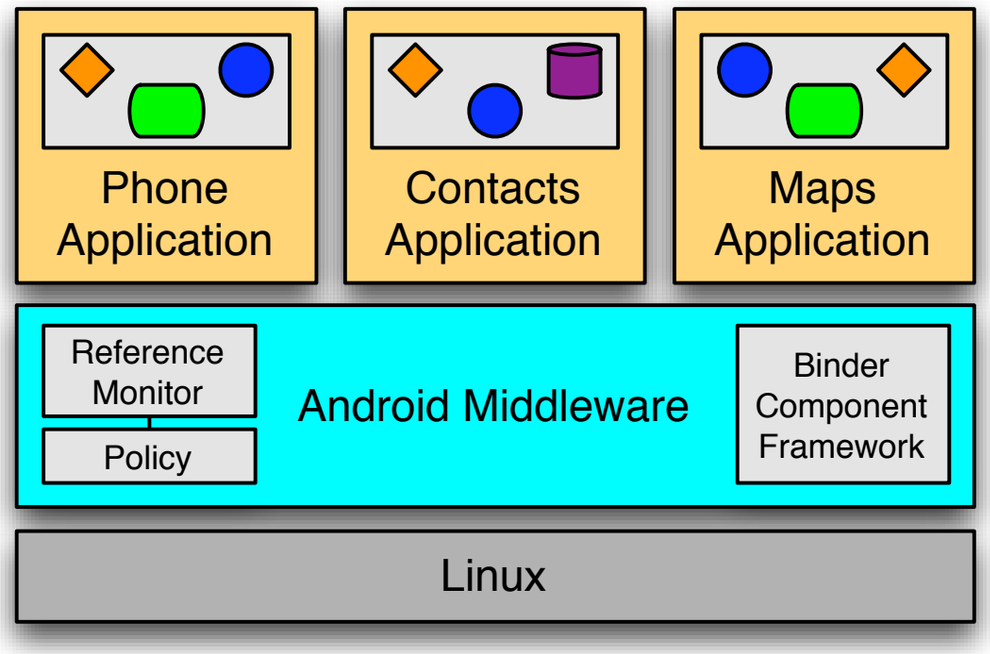
Android Phones

- An Android contains a number of "*applications*"
  - Android comes installed with a number of basic systems tools, e.g., dialer, address book, etc.
  - Developers use the Android API to construct applications.
    - All apps are written in *Java* and executed within a custom Java virtual machine.
  - Each application *package* is contained in a jar file (.apk)
- Applications are *installed* by the user
  - No "app store" required, just build and go.
  - Open access to data and voice services



Architecture

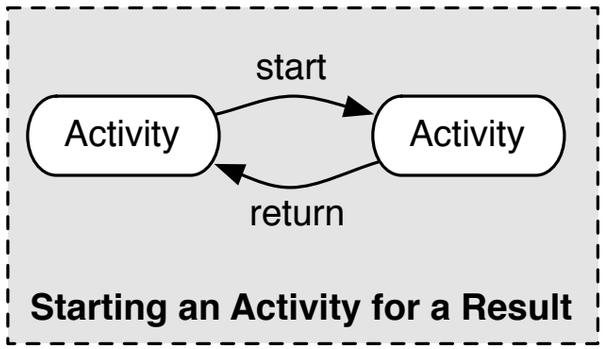
- The Android smartphone operating system is built upon Linux and includes many libraries and a core set of applications.
- The middleware makes it interesting
  - Not focused on UNIX processes
  - Uses the Binder component framework
    - Originally part of BeOS, then enhanced by Palm, now used in Android
  - Applications consist of many *components* of different types
  - Applications interact via components

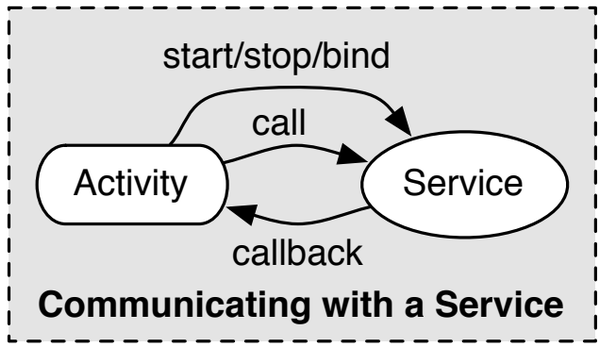


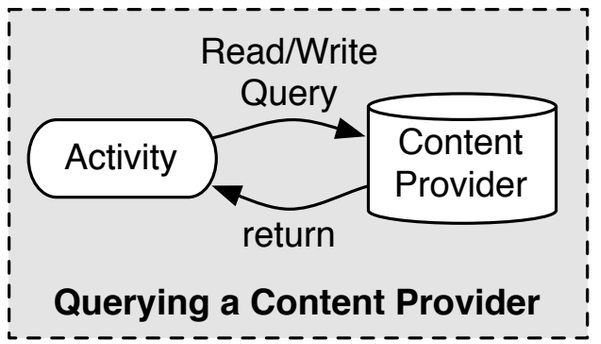
- We focus on security with respect to the component API

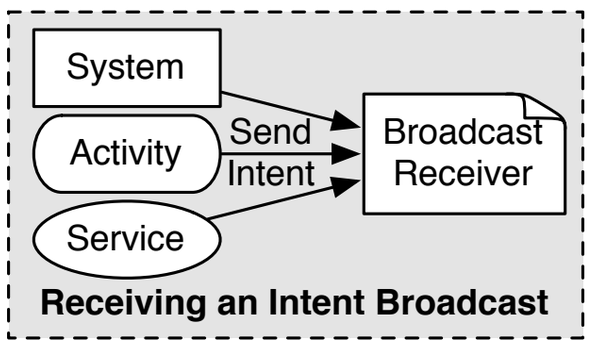
Component Model

- While each application runs as its own UNIX uid, sharing can occur through *application-level* interactions
  - Interactions based on components
  - Different component types
    - Activity
    - Service
    - Content Provider
    - Broadcast Receiver
  - Target component in the same or different application
  - *but first …*









Intents

- Intents are objects used as inter-component signaling
  - Starting the user interface for an application
  - Sending a message between components
  - Starting a background service

- Two types
  - Explicit: names the target component class
  - Implicit: specifies an "action string" (e.g., ACTION_VIEW)

Activity Component

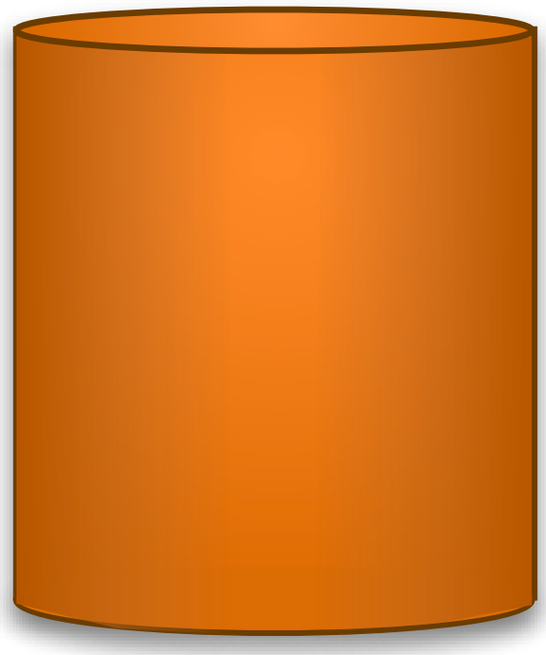
- The user interface consists of a series of *Activity* components.
- Each Activity is a "screen".
- User actions tell an Activity to *start* another Activity, possibly with the expectation of a *result*.
- The target Activity is not necessarily in the same application.
- Directly or via Intent "action strings".
- Processing stops when another Activity is "on top".

Service Component

- Background processing occurs in *Service* components.
  - Downloading a file, playing music, tracking location, polling, etc.
  - Local vs. Remote Services (process-level distinction)
- Also provides a "service" interface between applications
  - Arbitrary interfaces for data transfer
    - Android Interface Definition Language (AIDL)
  - Register callback methods
  - Core functionality often implemented as Service components
    - e.g., Location API, Alarm service
- Multiple interfaces
  - Control: start, stop
  - Method invocation: bind

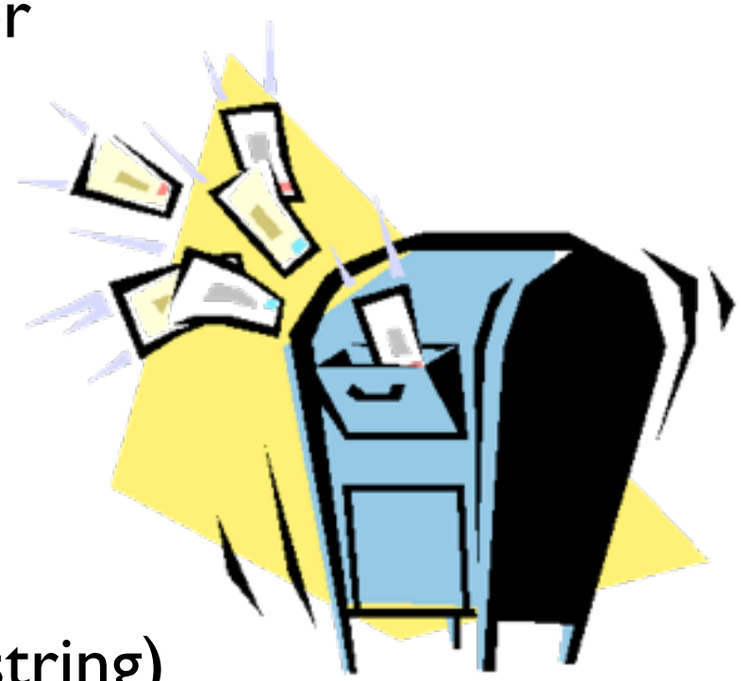
Content Provider Component

- *Content Provider* components provide a standardized interface for sharing data, i.e., content (between applications).
- Models content in a relational DB
  - Users of Content Providers can perform queries equivalent to SELECT, UPDATE, INSERT, DELETE
  - Works well when content is tabular
  - Also works as means of addressing "files"
- URI addressing scheme
  - `content://<authority>/<table>/[<id>]`
  - `content://contacts/people/10`



Broadcast Receiver Component

- *Broadcast Receiver* components act as specialized event *Intent* handlers (also think of as a message mailbox).
- Broadcast Receiver components "*subscribe*" to specific action strings (possibly multiple)
  - action strings are defined by the system or developer
  - component is automatically called by the system
- Recall that Android provides automatic Activity resolution using "action strings".
  - The action string was assigned to an *Intent* object
  - Sender can specify component recipient (no action string)



The Android Manifest

- Manifest files are the technique for describing the contents of an application *package* (i.e., resource file)
- Each Android application has a special `AndroidManifest.xml` file (included in the .apk package)
  - describes the contained components
    - components cannot execute unless they are listed
  - specifies rules for "auto-resolution"
  - specifies access rules
  - describes runtime dependencies
  - optional runtime libraries
  - required system permissions

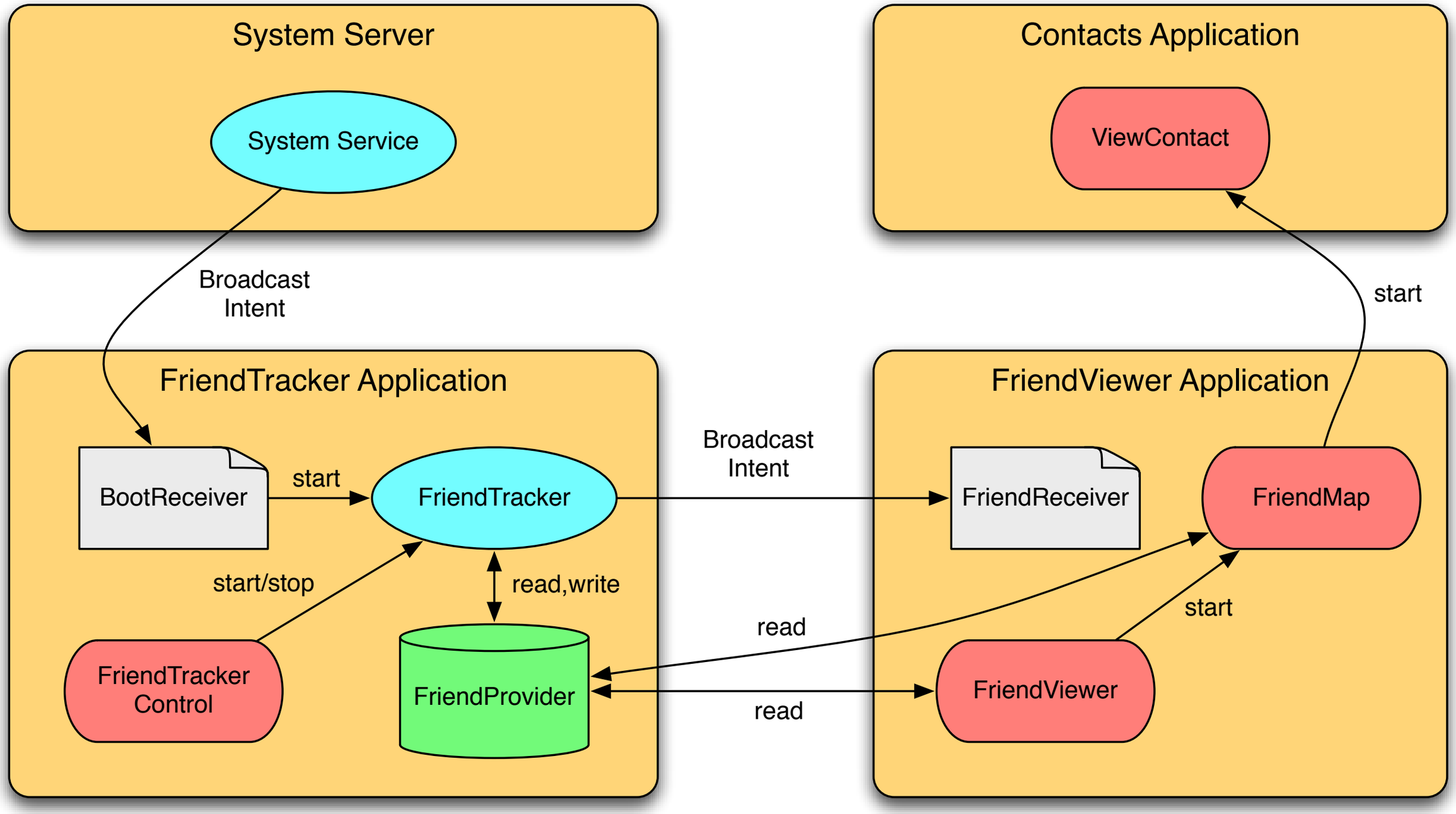
Manifest Specification

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="org.siislab.tutorial.friendtracker"
      android:versionCode="1"
      android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".FriendTrackerControl"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <provider android:authorities="friends"
              android:name="FriendProvider"
              android:writePermission="org.siislab.tutorial.permission.WRITE_FRIENDS"
              android:readPermission="org.siislab.tutorial.permission.READ_FRIENDS">
        </provider>
        <service android:name="FriendTracker" android:process=":remote"
              android:permission="org.siislab.tutorial.permission.FRIEND_SERVICE">
        </service>
        <receiver android:name="BootReceiver">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"></action>
            </intent-filter>
        </receiver>
    </application>

    <!--  Define Permissions -->
    <permission android:name="org.siislab.tutorial.permission.READ_FRIENDS"></permission>
    <permission android:name="org.siislab.tutorial.permission.WRITE_FRIENDS"></permission>
    <permission android:name="org.siislab.tutorial.permission.FRIEND_SERVICE"></permission>

    <!--  Uses Permissions -->
    <uses-permission android:name="org.siislab.tutorial.permission.READ_FRIENDS"></uses-permission>
    <uses-permission android:name="org.siislab.tutorial.permission.WRITE_FRIENDS"></uses-permission>
    <uses-permission android:name="org.siislab.tutorial.permission.FRIEND_SERVICE"></uses-permission>

    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"></uses-permission>
    <uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
</manifest>
```

Example Applications

- FriendTracker Application
  - *FriendTracker* Service to poll for friend locations
    - Broadcasts an Intent when near a friend
  - *FriendProvider* Content Provider to store location of friends
    - Cross references friends with system Contacts Provider
  - *FriendTrackerControl* Activity to start and stop the Service
  - *BootReceiver* Broadcast Receiver to start the service on boot
- FriendViewer Application
  - *FriendViewer* Activity to display list of friend locations
  - *FriendMap* Activity to show friends on a map (on right)
  - *FriendReceiver* Broadcast Receiver to display when near
- Available from http://siis.cse.psu.edu/android_sec_tutorial.html

Component Interaction



Hands-on Exercise

http://developer.android.com/training

Additional Recommended Modules

- Getting Started
  - Saving Data
  - Interacting with Other Apps
- Building Apps with Content Sharing
  - Sharing Simple Data
  - Sharing Files
- Building Apps with Connectivity & the Cloud
  - Performing Network Operations

Also Recommended

http://developer.android.com/samples

Thank You!

William Enck
Assistant Professor
Department of Computer Science
North Carolina State University
http://www.enck.org
enck@cs.ncsu.edu