Georgia Tech College of Computing

CS 3251: Computer Networking I

Android Overview

Brad Reaves, PhD Student

11/5/13

(slides from Prof. Patrick Traynor)

Non-sequitur... (?)

- Today's lecture is going to feel like a systems or software lecture -- because it is.
- Our goal is to get you acquainted with Android and how applications work on that platform.

- Aside: Why do we do object-oriented programming?

What is Android?

- Android is a Google-led operating system for mobile devices
  - Complete software stack
  - Open source (Apache v2 license) ... mostly

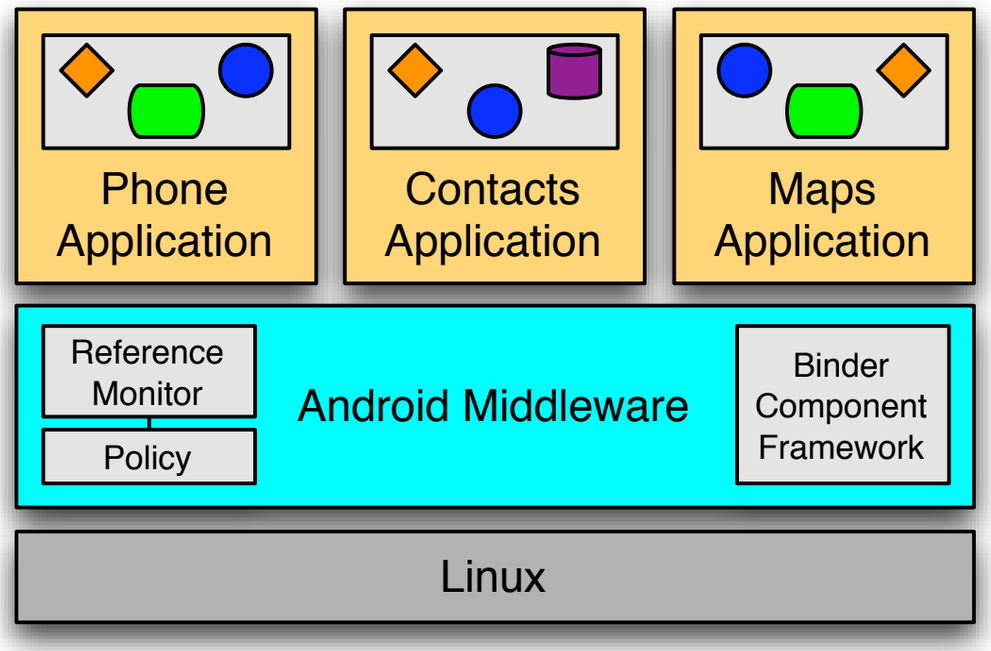
- Open Handset Alliance
  ... 30+ industrial partners
  - Google, T-Mobile, Sprint, HTC, LG, Motorola, Samsung, Broadcom, Intent, NVIDIA, Qualcomm, and many more.

Why Android?

- There are (arguably) three main platforms for mobile development: iPhone (iOS), Android, and Windows Phone.
- Android *is* the most widely deployed mobile OS
  - Estimates by Strategy Analytics give Android 81.3% of market share
  - iOS takes 13.4% and Windows Phone takes 4.1%
- You can mess with the internals!
  - That makes it great as a teaching and research platform!

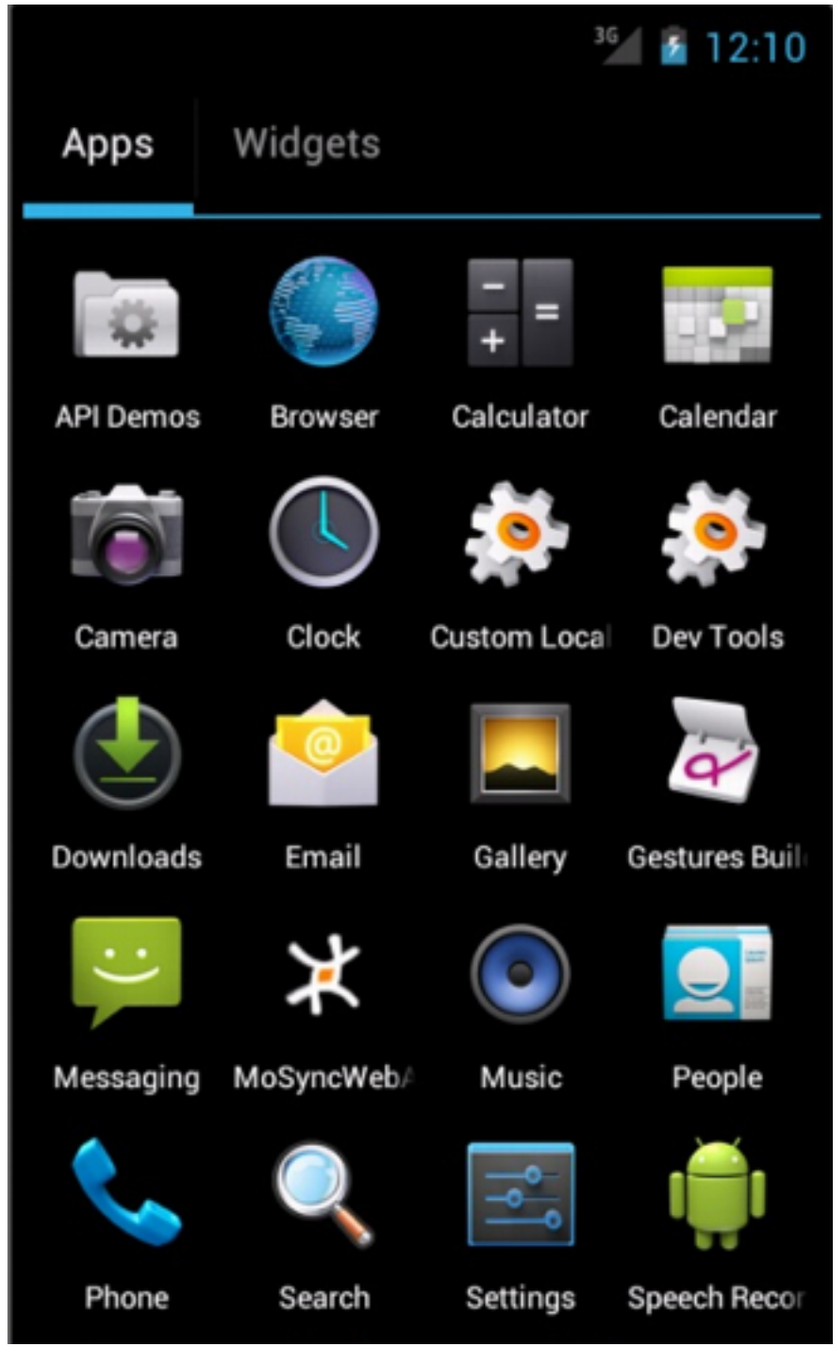
Architecture

- 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



Applications

- An Android device contains a number of "*applications*"
  - Android comes installed with a number of basic systems tools
    - Dialer, Address book, etc.
  - Developers use the Android API to construct applications.
  - 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

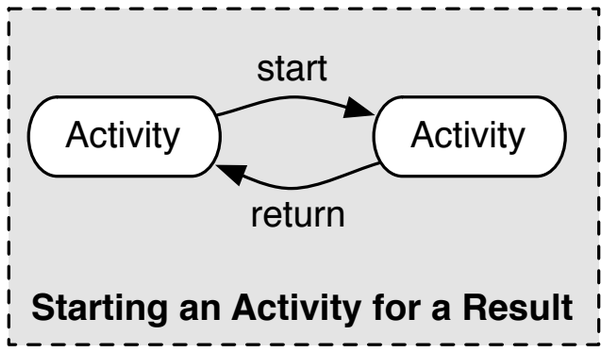


Application Environment

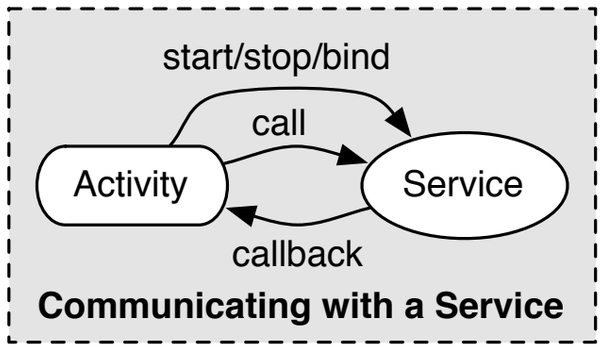
- Android Apps are written in Java
  - Executed in an optimized VM called "Dalvik"
  - Native code is permitted (C/C++ compiled to ARM/x86)
- Apps are "sandboxed" by the OS
- Each application is run as a unique Unix user and group
  - File system protections are used to ensure apps can't write to other apps' data
- Applications are strictly isolated from each other
  - There are well-defined ways for apps to communicate

Component Model

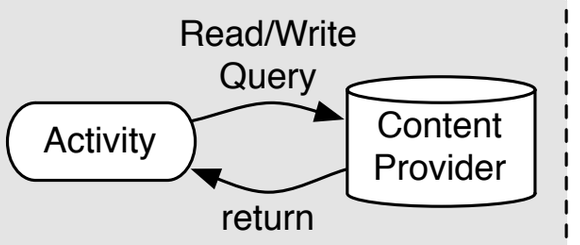
- While each application runs as its own UNIX uid, sharing can occur through *application-level* interactions
  - Interactions based on components
  - Different component types
    - ***Activity***
    - ***Service***
    - ***Content Provider***
    - ***Broadcast Receiver***
  - Target component in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



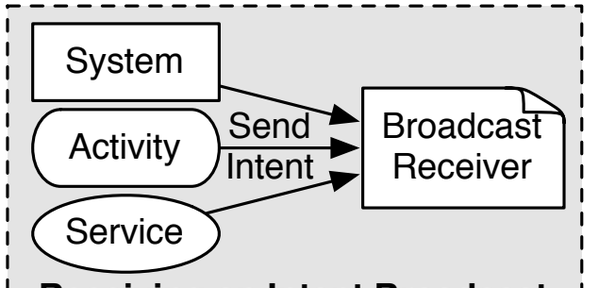
Starting an Activity for a Result



Communicating with a Service



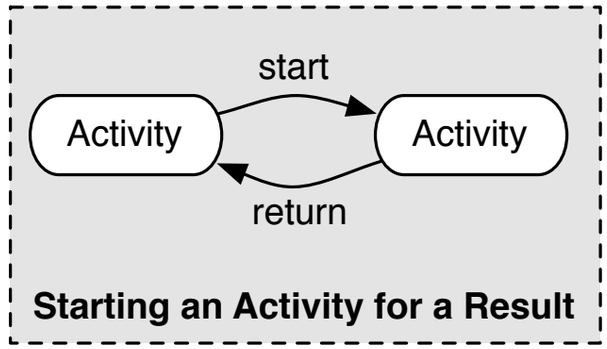
Querying a Content Provider



Receiving an Intent Broadcast

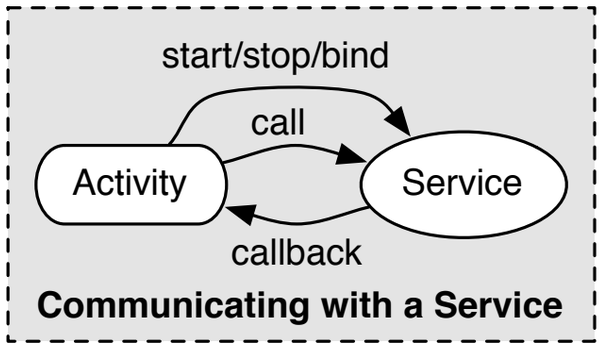
Intents

- Intents are objects used as inter-component signaling
  - Starting the user interface for an application



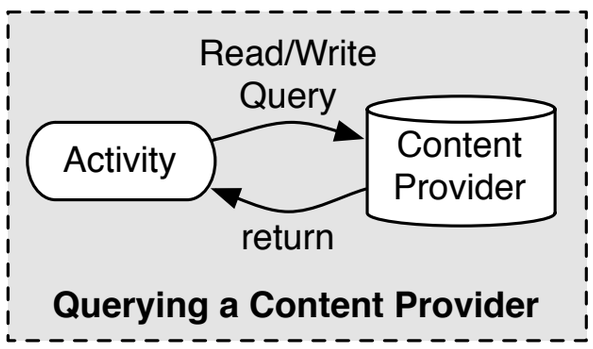
Starting an Activity for a Result

  - Starting a background service

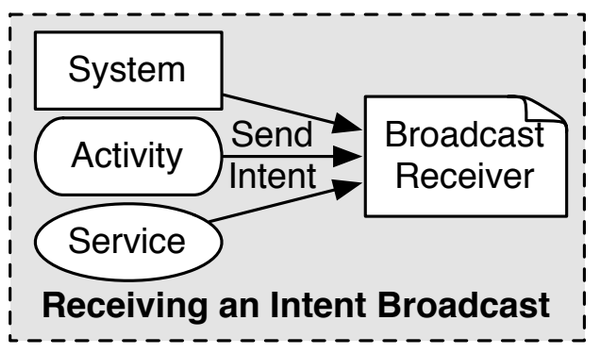


Communicating with a Service

  - Sending a message between components



Querying a Content Provider



Receiving an Intent Broadcast

Activity Component

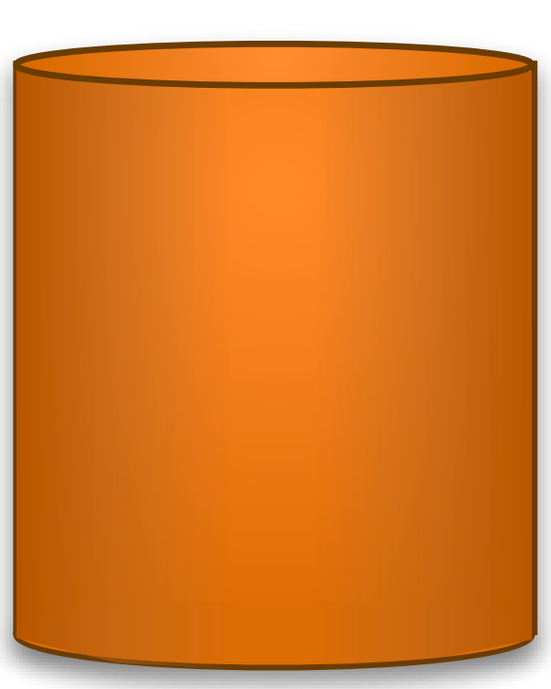
- The user interface consists of a series of *Activity* components.
- Each Activity is a "screen"
- User actions tell an Activity to *start* another Activity, possibly with the expectation of a *result*.
- The target Activity is not necessarily in the same application.
- Directly or via Intent "action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

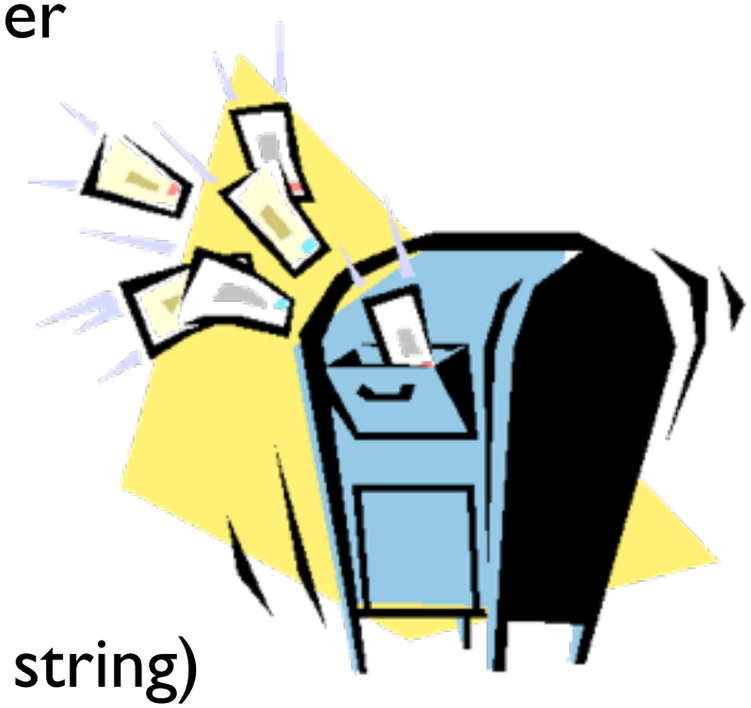
- *Content Provider* components provide a standardized interface for sharing data, i.e., content (between applications).
- Models content in a relational DB
  - Users of Content Providers can perform queries equivalent to SELECT, UPDATE, INSERT, DELETE
  - Works well when content is tabular
  - Also 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)



What's the point?

- Why are we going to all this trouble to make applications out of components?

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

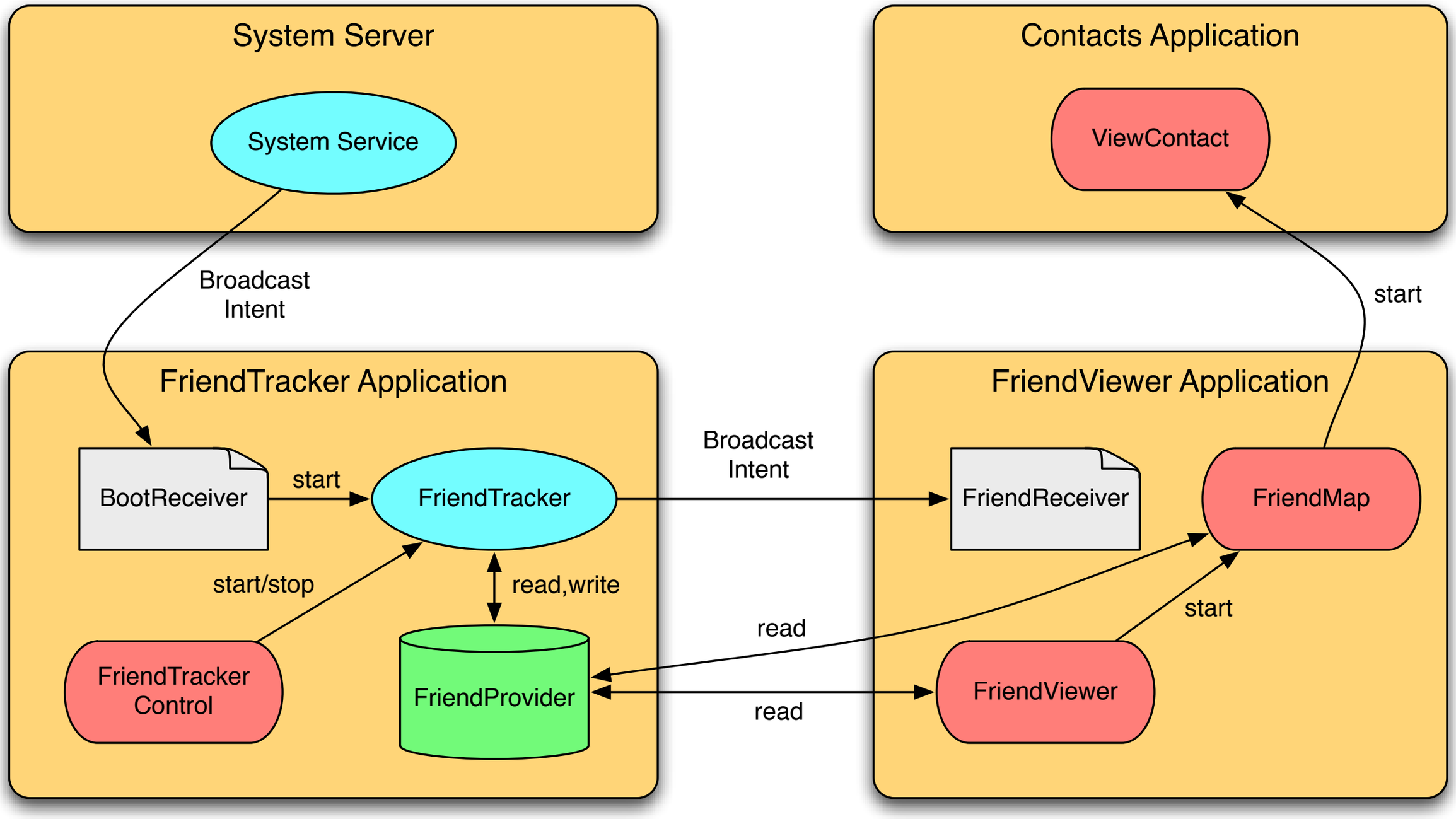
Manifest Specification

```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

Component Interaction



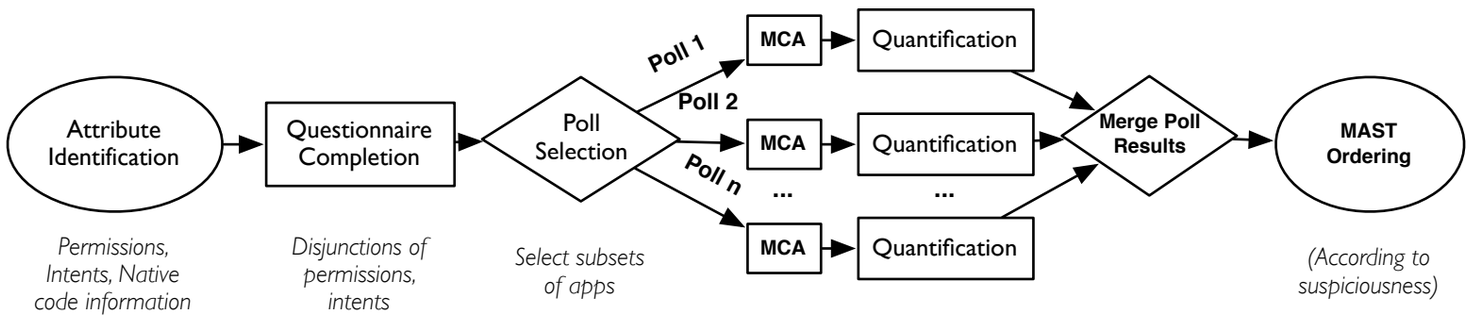
Android Security

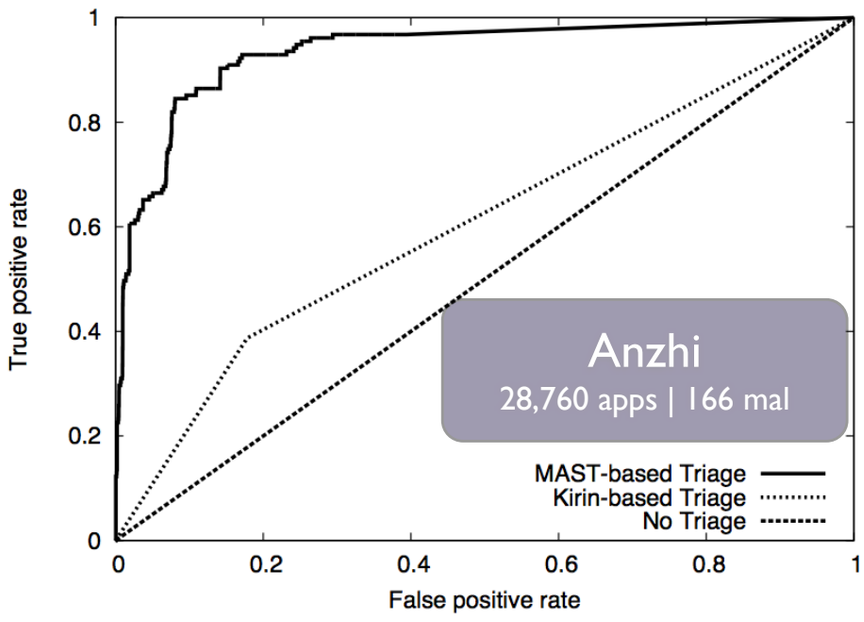
- Applications are sandboxed using the Dalvik Virtual Machine.
  - Communication can occur through the previously discussed mechanisms.

- Assuming that the underlying isolation mechanisms are sufficient, where are attacks most likely to be found in these devices?

Android Security Research at GT

- We can use high-level info about apps taken from the manifest and the APK to determine which apps in a market need security analysis.
- Only 13% of legitimate apps need to be analyzed to analyze 95% of malicious apps in the markets we studied

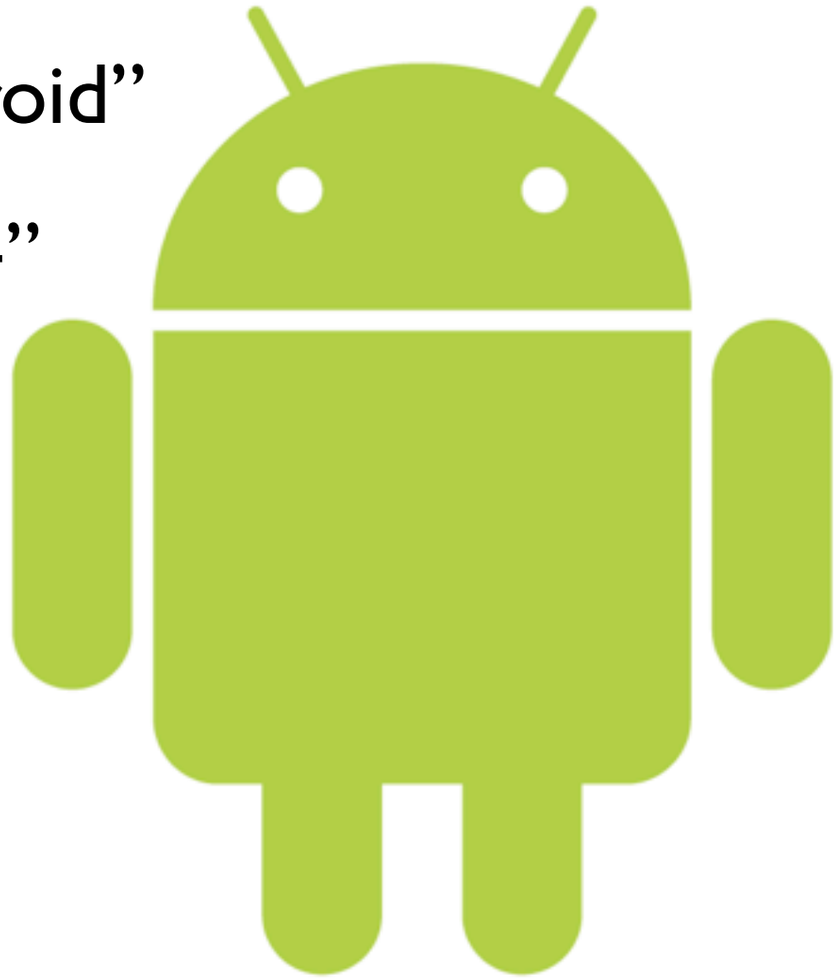




S. Chakradeo, B. Reaves, P. Traynor and W. Enck, **MAST: Triage for Market-scale Mobile Malware Analysis**, In Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec), 2013.

Android Resources

- Safari Books Online
  - Available through the GT Library
  - “Learning Android”
  - “Pro Android 4”



Preview of Thursday: A Tale of Two Networks

- Cellular Telecommunications Network

  - Network tailored for voice - very low bandwidth
  - Devices previously not suitable for Internet and computing apps

- Despite high penetration and coverage, Internet access fizzled until the second half of the decade.

Preview of Thursday: A Tale of Two Networks

- Cellular Telecommunications Network

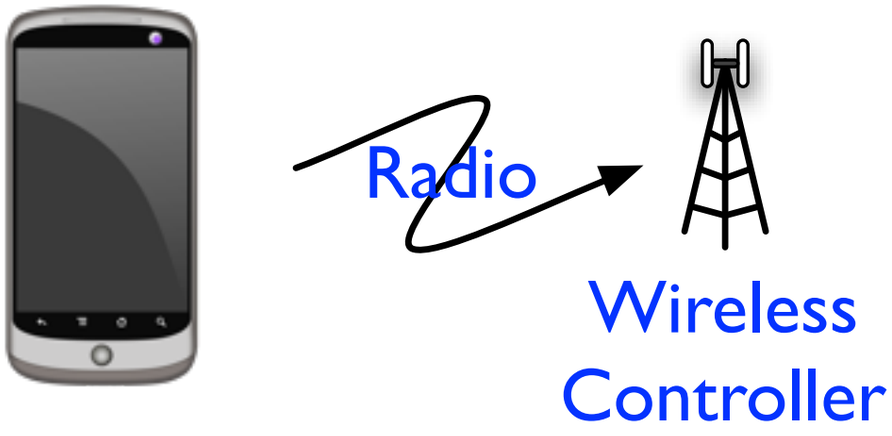
Wireless Controller

  - Network tailored for voice - very low bandwidth
  - Devices previously not suitable for Internet and computing apps

- Despite high penetration and coverage, Internet access fizzled until the second half of the decade.

Preview of Thursday: A Tale of Two Networks

- Cellular Telecommunications Network

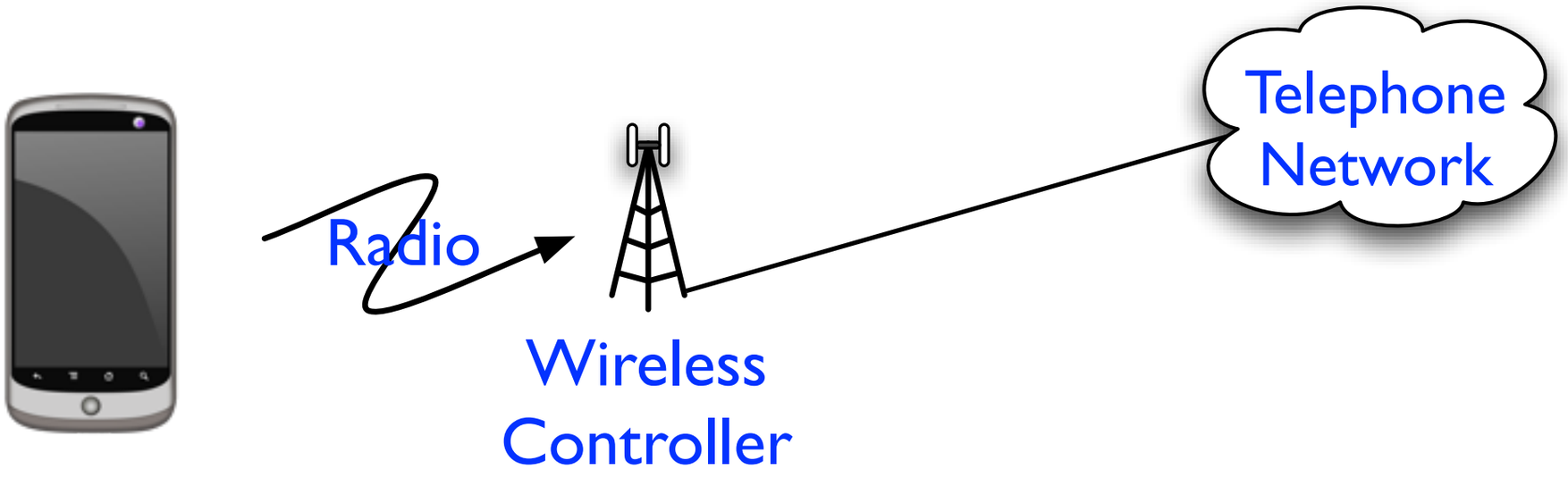


  - Network tailored for voice - very low bandwidth
  - Devices previously not suitable for Internet and computing apps

- Despite high penetration and coverage, Internet access fizzled until the second half of the decade.

Preview of Thursday: A Tale of Two Networks

- Cellular Telecommunications Network

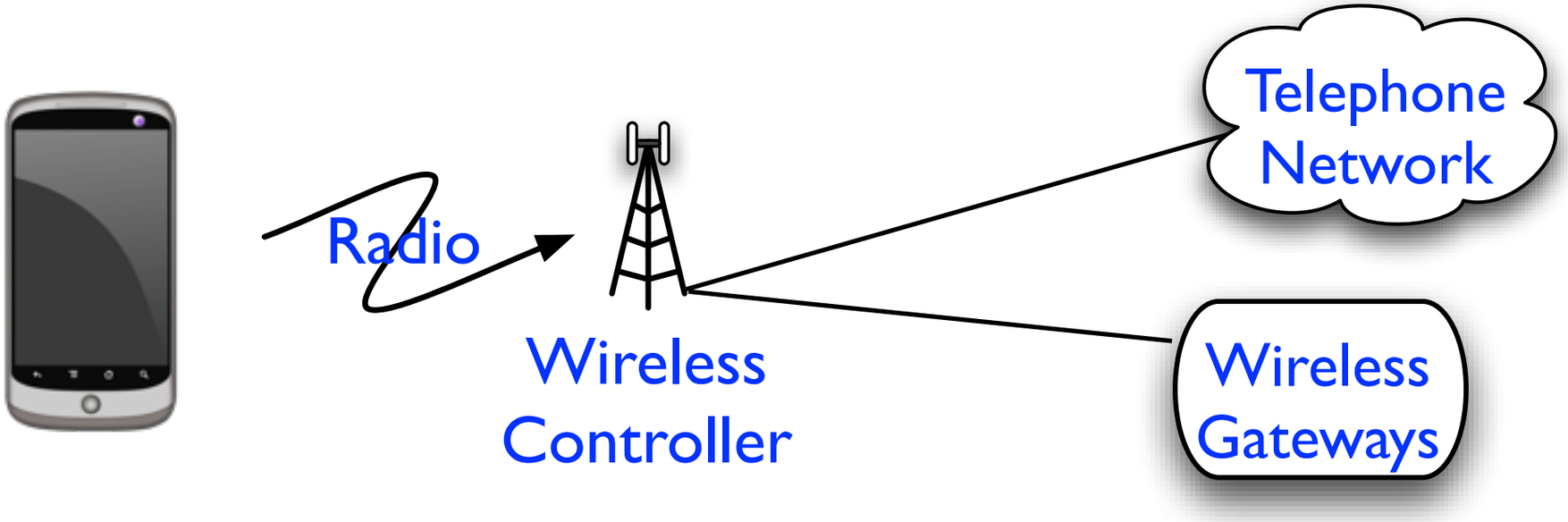


  - Network tailored for voice - very low bandwidth
  - Devices previously not suitable for Internet and computing apps

- Despite high penetration and coverage, Internet access fizzled until the second half of the decade.

Preview of Thursday: A Tale of Two Networks

- Cellular Telecommunications Network

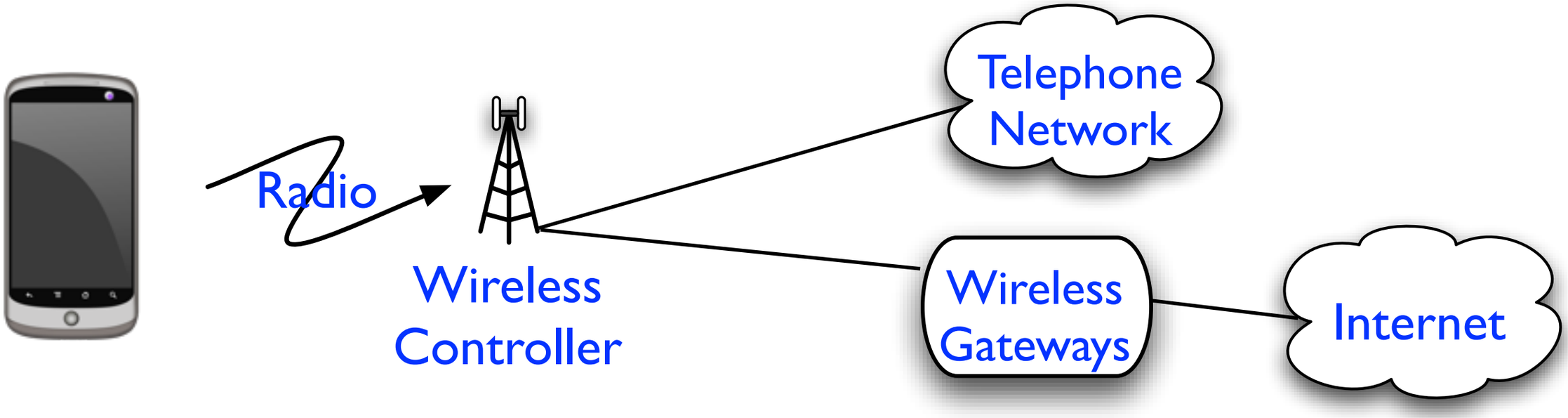


  - Network tailored for voice - very low bandwidth
  - Devices previously not suitable for Internet and computing apps

- Despite high penetration and coverage, Internet access fizzled until the second half of the decade.

Preview of Thursday: A Tale of Two Networks

- Cellular Telecommunications Network



  - Network tailored for voice - very low bandwidth
  - Devices previously not suitable for Internet and computing apps

- Despite high penetration and coverage, Internet access fizzled until the second half of the decade.

A Tale of Two Networks (2)

- The Internet - Wireless Enterprise Networks

  - Network tailored for best-effort data traffic
    - High bandwidth, no controls
  - Supports general computing and data networking applications

- Gaining high density hot-spots, but not ubiquitous coverage.

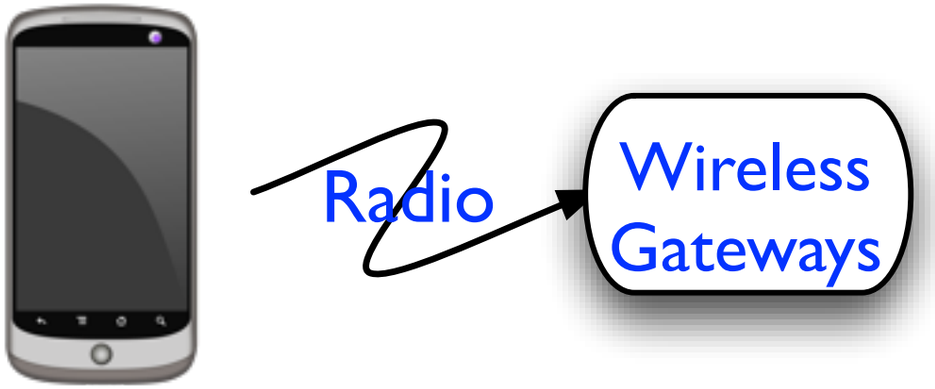
A Tale of Two Networks (2)

- The Internet - Wireless Enterprise Networks

  - Network tailored for best-effort data traffic
    - High bandwidth, no controls
  - Supports general computing and data networking applications
- Gaining high density hot-spots, but not ubiquitous coverage.

A Tale of Two Networks (2)

- The Internet - Wireless Enterprise Networks

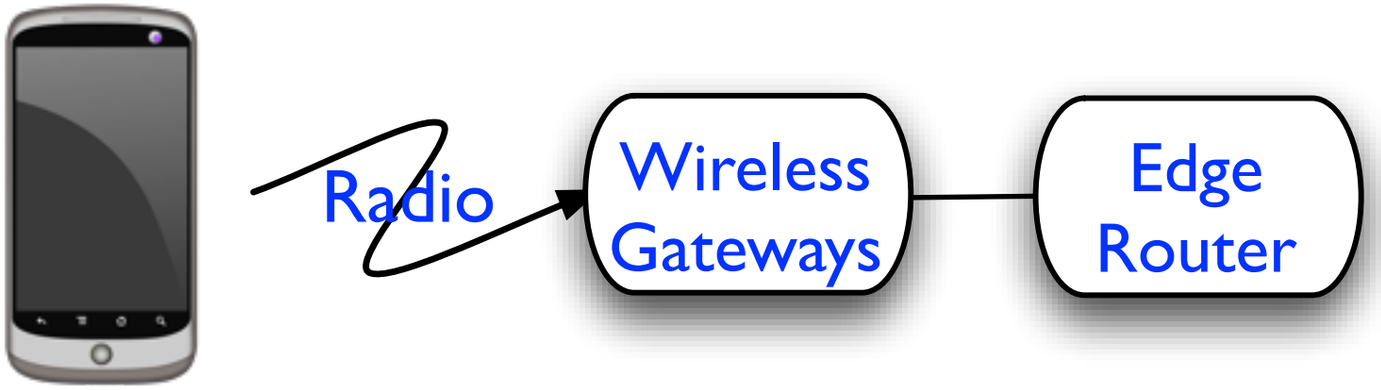


  - Network tailored for best-effort data traffic
    - High bandwidth, no controls
  - Supports general computing and data networking applications

- Gaining high density hot-spots, but not ubiquitous coverage.

A Tale of Two Networks (2)

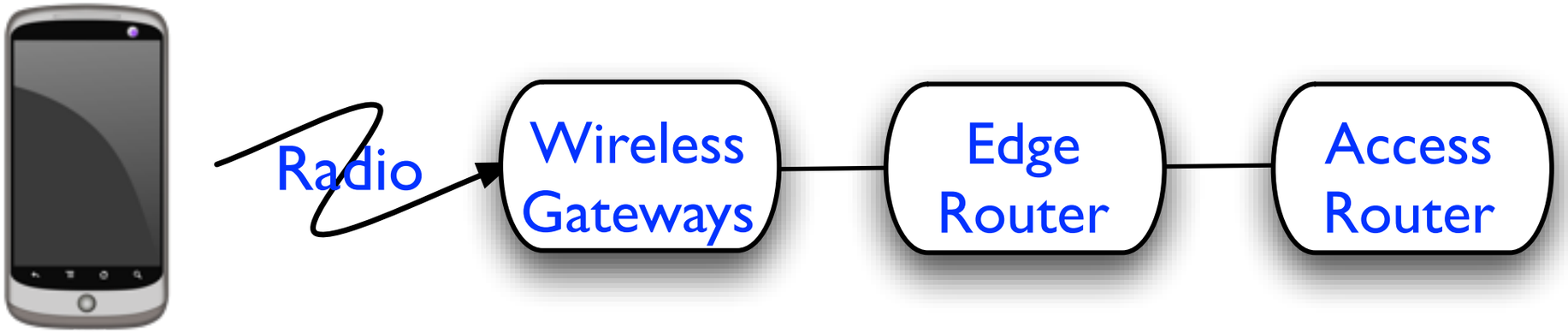
- The Internet - Wireless Enterprise Networks



  - Network tailored for best-effort data traffic
    - High bandwidth, no controls
  - Supports general computing and data networking applications
- Gaining high density hot-spots, but not ubiquitous coverage.

A Tale of Two Networks (2)

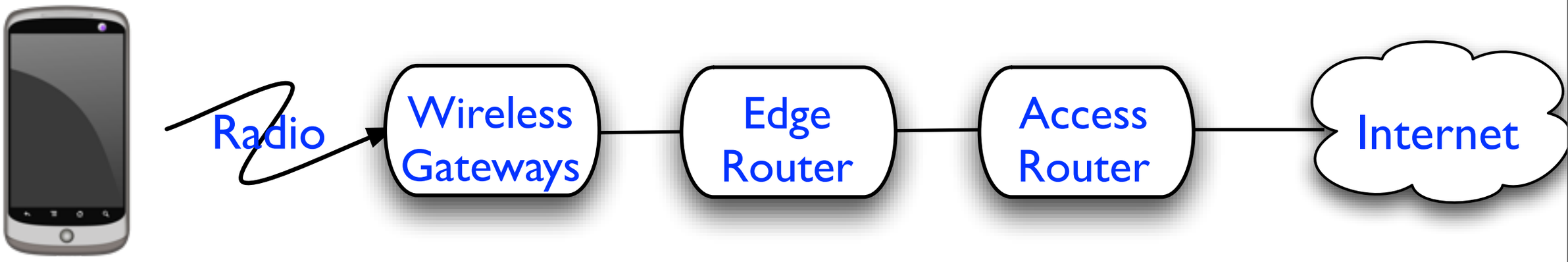
- The Internet - Wireless Enterprise Networks



  - Network tailored for best-effort data traffic
    - High bandwidth, no controls
  - Supports general computing and data networking applications
- Gaining high density hot-spots, but not ubiquitous coverage.

A Tale of Two Networks (2)

- The Internet - Wireless Enterprise Networks



  - Network tailored for best-effort data traffic
    - High bandwidth, no controls
  - Supports general computing and data networking applications

- Gaining high density hot-spots, but not ubiquitous coverage.

Questions?