CSE 486/586 Distributed Systems
Android Programming --- 1

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap 🎯

- What to put on top of physical networks?
  - Layers providing survivability
- Where to put functionalities?
  - Fate-sharing & end-to-end arguments
  - IP layer doesn't provide much
  - TCP handles most of the survivability issues
- TCP & UDP: the two transport protocols of the Internet
- What interface do applications see?
  - Socket API

Today

- Basic Android programming interleaved with a review of PA1
- Mainly programming model and components
- We will look at PA1 template code alongside.
- Caveats
  - Not really a comprehensive tutorial
  - Just touching on basics
- Will have more of these later as more PAs come out.

Three Most Important Things

- Tools that you need to be familiarized with:
  - Android APIs and constructs
  - Debugging
- Read the documentation.
  - Learn how to use the APIs and the constructs, e.g., AsyncTask, Messenger, etc.
  - Learn how to work within the Android's constraints.
- Learn how to debug.
  - Using LogCat, DDMS, etc.
  - Much time is spent on debugging, so learn how to use the tools.
- Incremental development
  - First write the minimum possible thing to execute your app.
  - Iterate: write something and debug

Android Programming Model

- Three things to keep in mind.
  - The responsibilities of the OS
  - The responsibilities of an app
  - How the OS knows the responsibilities of an app.
- App
  - No main()
  - Event-driven (reacting to events)
- OS
  - Deliver events by calling appropriate callbacks
- AndroidManifest.xml
  - An app declares its capabilities (e.g., its permissions).
  - An app registers all the callbacks.

What? No main()?

- There is a main()! It's just that it's hidden.
- Zygote starts at boot.
- Launcher sends a message to start an activity.
- Zygote forks a new VM instance that loads ActivityThread.
  - ActivityThread has the real main() for an app.
- ActivityThread calls the app's onCreate(), onStart(), etc.
- What main() does is implementing an event loop.
  - Wait for an event to happen.
  - When an event happens, look up which callback handles the event.
  - Call the callback.
  - Loop

Example - Activity

```java
public class Activity extends ApplicationContext {
    protected void onCreate(Bundle savedInstanceState);

    protected void onStart();

    protected void onRestart();

    protected void onResume();

    protected void onPause();

    protected void onStop();

    protected void onDestroy();
}
```
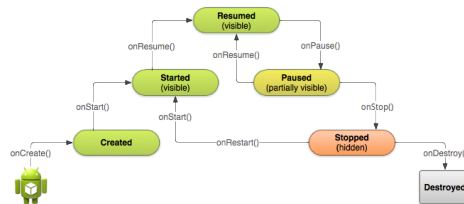
CSE 486/586

Example - Activity

CSE 486/586

Declare in AndroidManifest.xml

<manifest ... >
 ...
 <application ... >
   <activity android:name=".ExampleActivity" />
   ...
 </application>
</manifest>

CSE 486/586   9

Define Permissions

• Should define permissions (for others) in AndroidManifest.xml
• <uses-permission android:name="android.permission.INTERNET"/>

CSE 486/586   10

CSE 486/586 Administrivia

• Please use Piazza; all announcements will go there.
• Please come to my office during the office hours!
  – Give feedback about the class, ask questions, etc.

CSE 486/586   11

More

• Logging statements
• Running a terminal window per AVD
• Questions?

CSE 486/586   12