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
- 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.

Five Most Important Things

- Tools that you need to be familiarized with:
  - Android APIs and constructs as well as the command line interface
- Read the documentation.
  - Learn how to use the APIs and the constructs, e.g., AsyncTask, etc.
  - Learn how to work within the Android’s constraints.
- Learn how to use a terminal.
  - Setting up the environment
  - Using LogCat, etc. (for debugging)
- Incremental development
  - First write the minimum possible thing to execute your app.
  - Iterate: write something and debug
- Trace your execution

Execution Tracing

- One of the most important things when you write your code.
- With a distributed system, you need to trace across different machines.
- My suggestion: draw a diagram for tracing

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 ApplicationContent {
    protected void onCreate(Bundle savedInstanceState) {
        protected void onStart();
        protected void onRestart();
        protected void onResume();
        protected void onPause();
        protected void onDestroy();
        protected void onLowMemory();
    }
}
```

Example - Activity

Declare in AndroidManifest.xml
```xml
<manifest ... >
...
<application ... >
    <activity android:name="ExampleActivity" />
    ...
</application>
</manifest>
```

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

CSE/586 Administirivia
- Please use Piazza; all announcements will go there.
More

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