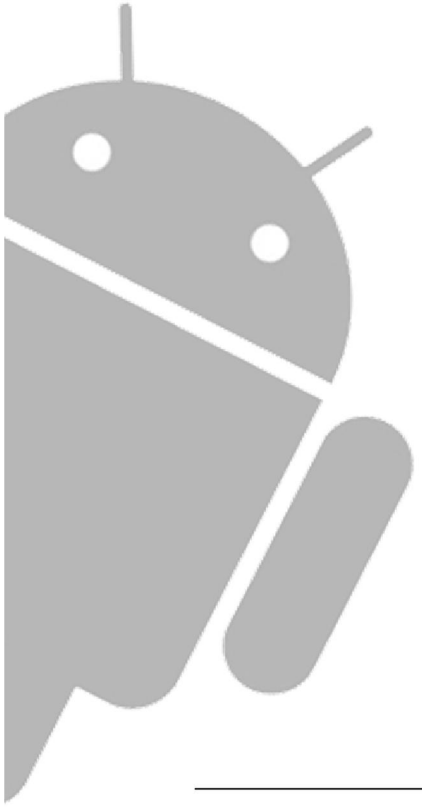


Mobility Programming

Lecture 5: Broadcast Receivers

DR. RAMEZ ALKHATIB

Broadcast Receivers

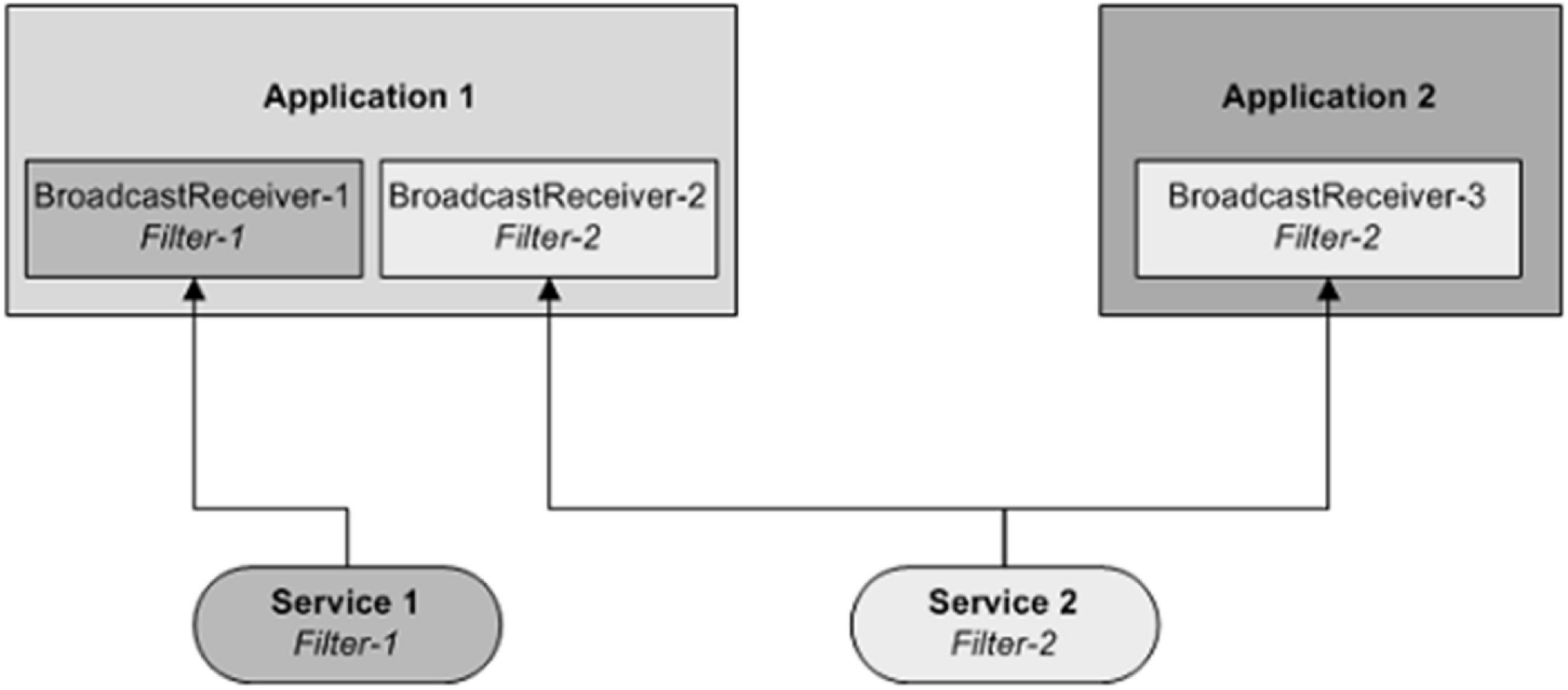
- simply respond to broadcast messages from other applications or from the system itself.
  - ✓ These messages are sometime called events or intents.
- Example system broadcasts: screen has turned off, the battery is low, user is present using phone, or a picture was captured.
- Applications can initiate broadcasts—e.g., to let other applications know that some data has been downloaded to the device and is available for them to use.
- Don't display a UI, but can create a status bar notification to alert the user when a broadcast event occurs.

Broadcast Receivers

- Usually, a broadcast receiver is just a "gateway" to other components and is intended to do a very minim amount of work. For instance, it might initiate a service to perform some work based on the event.
  - Important: you must complete tasks in a Broadcast Receiver in <10s. If you have a task that will take longer, you must start a new thread to avoid app assassin OS.

Lifecycle

The system delivers a broadcast Intent to all interested broadcast receivers, which handle the Intent sequentially.



Lifecycle

A broadcast receiver has a single callback method:

```
void onReceive(ContextcurContext, IntentbroadcastMsg)
```

1) When a broadcast message arrives for the receiver, Android calls its onReceive() method and passes it the Intent object containing the message.
2) The broadcast receiver is considered to be active only while it is executing this method.
3) When onReceive() returns, it is inactive.

- There are following two important steps to make BroadcastReceiver works for the system broadcasted intents:
  - Creating the Broadcast Receiver.
  - Registering Broadcast Receiver

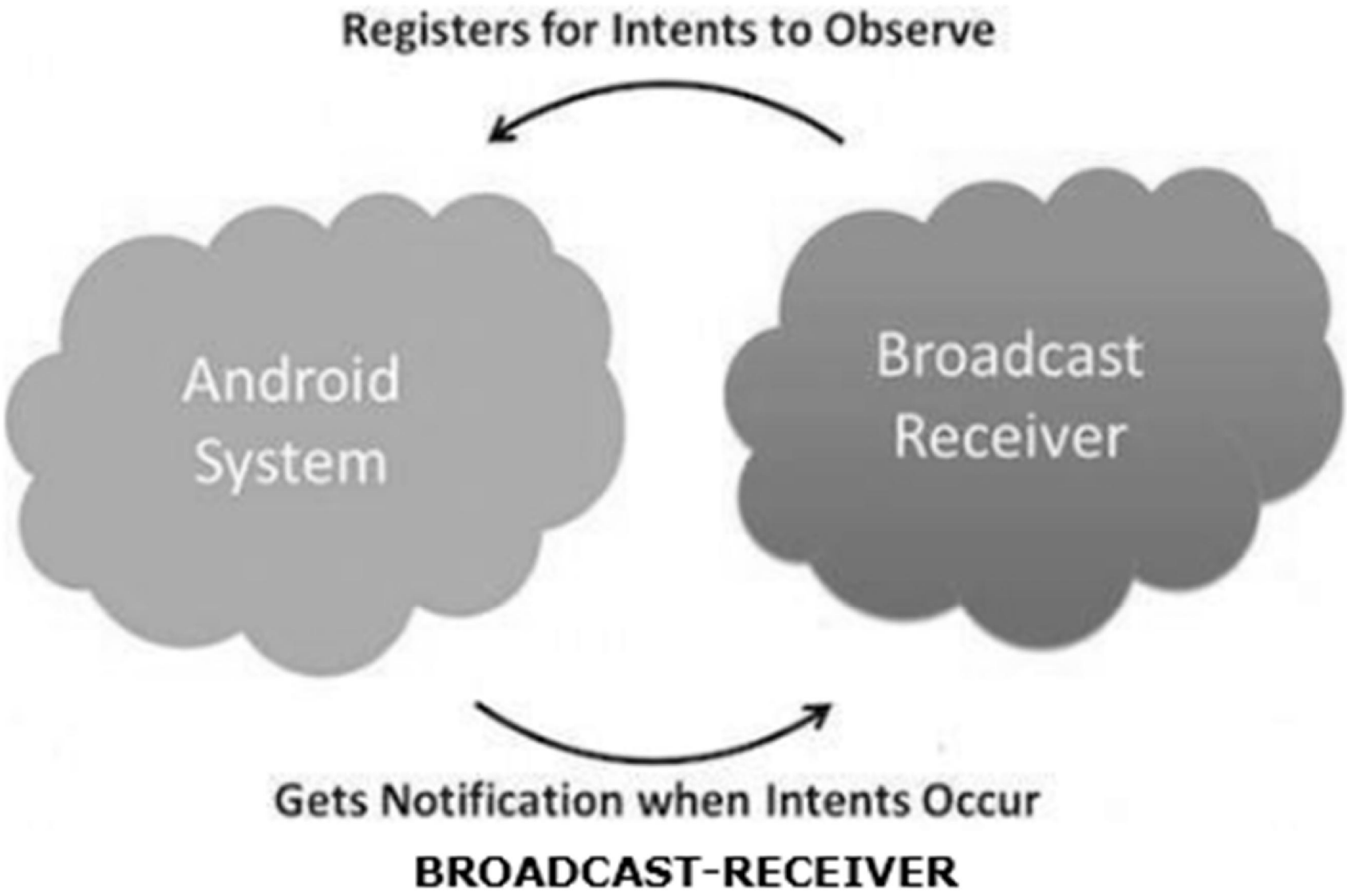
Creating the Broadcast Receiver

A broadcast receiver is implemented as a subclass of **BroadcastReceiver**class and overriding the onReceive() method where each message is received as a **Intent** object parameter.

```
package com.example.BroadcastDetector;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class MyReceiver extends BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent) {
                  // Implement code here to be performed when
            // broadcast is detected
        }
}
```

Registering Broadcast Receiver

An application listens for specific broadcast intents by registering a broadcast receiver in *AndroidManifest.xml* file. Consider we are going to register *MyReceiver* for system generated event ACTION_BOOT_COMPLETED which is fired by the system once the Android system has completed the boot process.



BROADCAST-RECEIVER

Registering a Broadcast receiver

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.BroadcastDetector"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <receiver android:name="MyReceiver" >
            <intent-filter>
                <action android:name="com.example.Broadcast" >
                </action>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

System Events

- Now whenever your Android device gets booted, it will be intercepted by BroadcastReceiver *MyReceiver* and implemented logic inside *onReceive()* will be executed.
- There are several system generated events defined as final static fields in the **Intent** class. The following table lists a few important system events.

System Events

Sr.No	Event Constant & Description
1	**android.intent.action.BATTERY_CHANGED** Sticky broadcast containing the charging state, level, and other information about the battery.
2	**android.intent.action.BATTERY_LOW** Indicates low battery condition on the device.
3	**android.intent.action.BATTERY_OKAY** Indicates the battery is now okay after being low.
4	**android.intent.action.BOOT_COMPLETED** This is broadcast once, after the system has finished booting.
5	**android.intent.action.BUG_REPORT** Show activity for reporting a bug.

System Events

Sr.No	Event Constant & Description
6	**android.intent.action.CALL** Perform a call to someone specified by the data.
7	**android.intent.action.CALL_BUTTON** The user pressed the "call" button to go to the dialer or other appropriate UI for placing a call.
8	**android.intent.action.DATE_CHANGED** The date has changed.
9	**android.intent.action.REBOOT** Have the device reboot.

Broadcasting Custom Intents

- If you want your application itself should generate and send custom intents then you will have to create and send those intents by using the *sendBroadcast()* method inside your activity class.
- If you use the *sendStickyBroadcast(Intent)* method, the Intent is **sticky**, meaning the *Intent* you are sending stays around after the broadcast is complete.

```
public void broadcastIntent(View view) {
   Intent intent = new Intent();
   intent.setAction("com.tutorialspoint.CUSTOM_INTENT");
   sendBroadcast(intent);
}
```

This intent *com.tutorialspoint.CUSTOM_INTENT* can also be registered in similar way as we have regsitered system generated intent.

```
<application
   android:icon="@drawable/ic_launcher"
   android:label="@string/app_name"
   android:theme="@style/AppTheme" >
   <receiver android:name="MyReceiver">

      <intent-filter>
         <action android:name="com.tutorialspoint.CUSTOM_INTENT">
         </action>
      </intent-filter>

   </receiver>
</application>
```

EXAMPLE

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends Activity {

   /** Called when the activity is first created. */
   @Override

   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }


   // broadcast a custom intent.

   public void broadcastIntent(View view){
      Intent intent = new Intent();
      intent.setAction("com.tutorialspoint.CUSTOM_INTENT"); sendBroadcast(intent);
   }
}
```

MyReceiver.java

```
public class MyReceiver extends BroadcastReceiver{
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();
    }
}
```

AndroidManifest.xml file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.tutorialspoint7.myapplication">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">

      <activity android:name=".MainActivity">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>

      <receiver android:name="MyReceiver">
         <intent-filter>
            <action android:name="com.tutorialspoint.CUSTOM_INTENT">
            </action>
         </intent-filter>

      </receiver>
   </application>

</manifest>
```

res/layout/activity_main.xml file

```
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"
   tools:context=".MainActivity">

   <TextView
      android:id="@+id/textView1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Example of Broadcast"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:textSize="30dp" />

   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials point "
      android:textColor="#ff87ff09"
      android:textSize="30dp"
      android:layout_above="@+id/imageButton"
      android:layout_centerHorizontal="true"
      android:layout_marginBottom="40dp" />

   <ImageButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageButton"
      android:src="@drawable/abc"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true" />
```

Questions?