College of Science and Technology

Royal University of Bhutan

Unit II:
Broadcast Receiver

CTE308- AS2025

Tutor: Pema Galey
#17682761

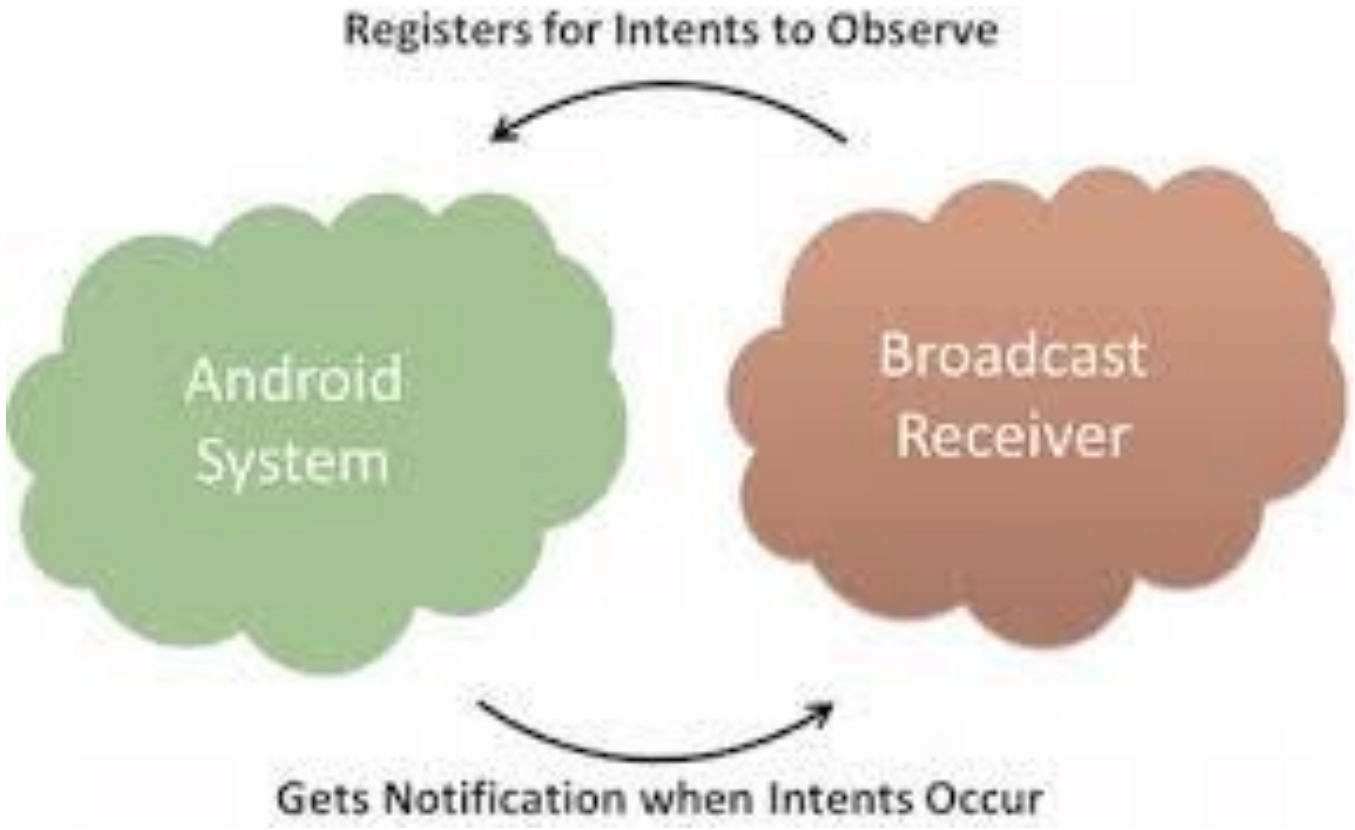
OUTLINES

- Introduction
- Broadcast Intents
- Broadcast Receivers (BR)
- Implementing BR
- Custom Broadcast
- Local Broadcast

Introduction

- **Broadcast Receiver**
  - Some Android application would do anything until a certain types of message is broadcasted by android OS or any other application.
  - Suppose you want your application to react in some way when system events occurs.
    - Example, you may want to play music when ear phone is connected and stop when it is disconnected.
  - **How can your app tell when this event occurs?**

Introduction



Introduction

- **How can your app tell when this event occurs?**
  - For this we have Broadcast Receiver.
- Broadcast Intent and Broadcast receiver helps to send and receive certain events.
- Use implicit intents to send broadcasts or start activities

Introduction

Broadcast Intents: Broadcast and Activity

- Use implicit intents to send broadcasts or start activity

Sending broadcasts

- Use sendBroadcast()
- Can be received by any application registered for the intent
- Used to notify all apps of an event

Starting activities

- Use startActivity()
- Find a single activity to accomplish a task
- Accomplish a specific action

Types of broadcast intent

- Systems Broadcast Intents
  - System delivers a system broadcast intents

- Custom Broadcast Intents
  - App delivers custom broadcast intents

System Broadcast Intents

- System automatically delivers a system broadcast intents when system events occurs
- Examples:
  - ACTION_BOOT_COMPLETED
    - Intent constant values: "android.intent.action.ACTION_BOOT_COMPLETED"
  - ACTION_POWER_(DIS)CONNECTED
    - Intent constant values: "android.intent.action.ACTION_POWER_DISCONNECTED"
    - "android.intent.action.ACTION_POWER_CONNECTED"
  - ACTION_DEVICE_STORAGE_LOW
    - Intent constant values: "android.intent.action.DEVICE_STORAGE_LOW"

Custom Broadcast Intents

- Use custom broadcast intents when you want your app to take an action without launching an activity. For instance, file downloaded successfully.
- Use custom event action to create custom broadcast intents.
- To deliver the custom broadcast to other apps, pass the intent to:
  - sendBroadcast() - asynchronous
  - sendOrderedBroadcast() - synchronously
  - sendStickyBroadcast()
    - android.example.com.CUSTOM_ACTION

sendBroadcast()

- All receivers of the broadcast are run in an undefined order
- Can be at the same time
- Efficient
- Use to send custom broadcasts

sendOrderedBroadcast()

- Delivered to one receiver at a time
- Receiver can propagate result to the next receiver or abort the broadcast
- Control order with **android:priority** of matching intent filter
- Receivers with same priority run in arbitrary order

Custom broadcast intents

- Example of creating intents and broadcasts to all interested broadcast receiver.

```
public void sendBroadcastIntent() {
  Intent intent = new Intent();
  intent.setAction("com.example.myproject.ACTION_SHOW_TOAST");
  sendBroadcast(intent);
}
```

- **NOTE:** Use unique package name to not conflict the intent with other intent that is broadcast from a different app/system

Broadcast Receiver

- Broadcast intents aren't targeted at specific recipients. Instead, interested apps register a component to "listen" for these kind of intents. This listening component is called a **broadcast receiver**.
- Listens for incoming intents sent by sendBroadcast()
  - In the background
- Intents can be sent
  - By the system, when an event occurs that might change the behavior of an app
  - By another application, including your own

Broadcast Receiver always responds

- Responds even when your app is closed
- Independent from any activity
- When a broadcast intent is received and delivered to onReceive(), it has 5 seconds to execute, and then the receiver is destroyed

Implementing Broadcast Receiver

Steps for creating Broadcast Receiver:

1. Subclass `BroadcastReceiver`
2. Implement `onReceive()` method
3. Register to receive broadcast
   - Statically, in `AndroidManifest`
   - Dynamically, with `registerReceiver()`

Register in AndroidManifest.xml

- <receiver> element inside <application>
- <intent-filter> registers receiver for specific intents

```
<receiver
    android:name=".CustomReceiver"
    android:enabled="true"
    android:exported="true">
        <intent-filter>
 <action android:name="android.intent.action.BOOT_COMPLETED" />
        </intent-filter>
</receiver>
```

Register Dynamically

- In onCreate() or onResume()
- Use registerReceiver() and pass in the intent filter
- Must unregister in onDestroy() or onPause()

```
registerReceiver(mReceiver, mIntentFilter)
unregisterReceiver(mReceiver)
```

Available Intents

- ACTION_TIME_TICK
- ACTION_TIME_CHANGED
- ACTION_TIMEZONE_CHANGED
- ACTION_BOOT_COMPLETED
- ACTION_PACKAGE_ADDED
- ACTION_PACKAGE_CHANGED
- ACTION_PACKAGE_REMOVED
- ACTION_PACKAGE_RESTARTED
- ACTION_PACKAGE_DATA_CLEARED
- ACTION_PACKAGES_SUSPENDED
- ACTION_PACKAGES_UNSUSPENDED
- ACTION_UID_REMOVED
- ACTION_BATTERY_CHANGED
- ACTION_POWER_CONNECTED
- ACTION_POWER_DISCONNECTED
- ACTION_SHUTDOWN

Implement onReceive()

```
@Override
public void onReceive(Context context, Intent intent) {
   String intentAction = intent.getAction();
   switch (intentAction){
      case Intent.ACTION_POWER_CONNECTED:
            break;
      case Intent.ACTION_POWER_DISCONNECTED:
            break;
   }
}
```

Local Broadcast Manager

- For broadcasts only in your app
- No security issues since no cross-app communication

- `LocalBroadcastManager.sendBroadcast()`
- `LocalBroadcastManager.registerReceiver()`

Register Local Broadcast Manager

```
LocalBroadcastManager.getInstance(this)
  .registerReceiver( mReceiver,
     new IntentFilter(ACTION_CUSTOM_BROADCAST));
```

Thank you!