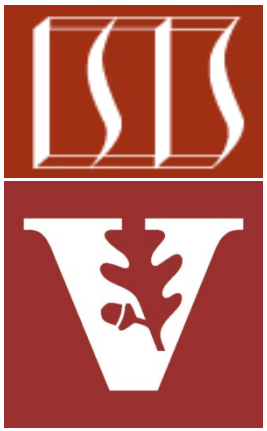
Android Services & Local IPC: The Publisher/Subscriber Pattern (Part 2)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt



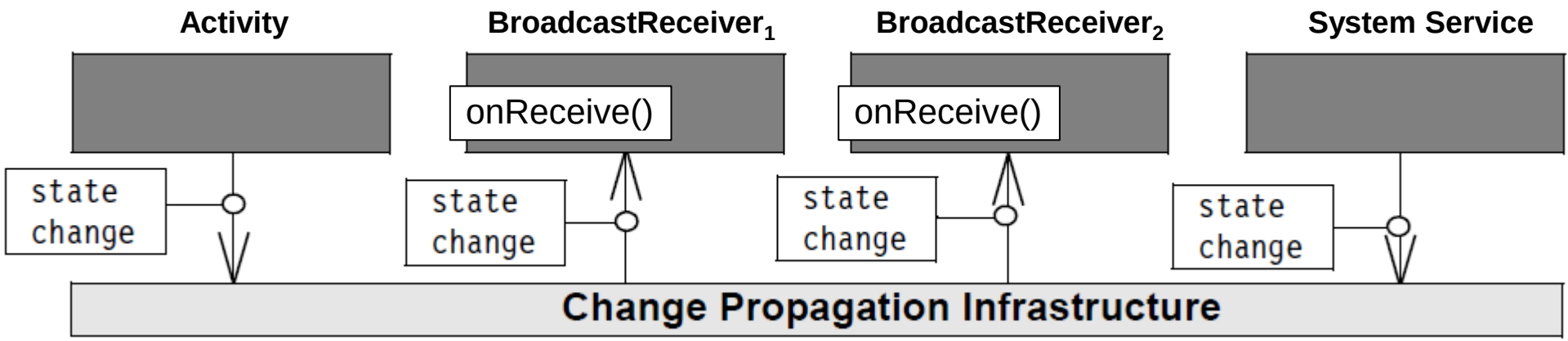
Professor of Computer Science

Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Module

- Understand how the *Publisher-Subscriber* pattern is applied in Android

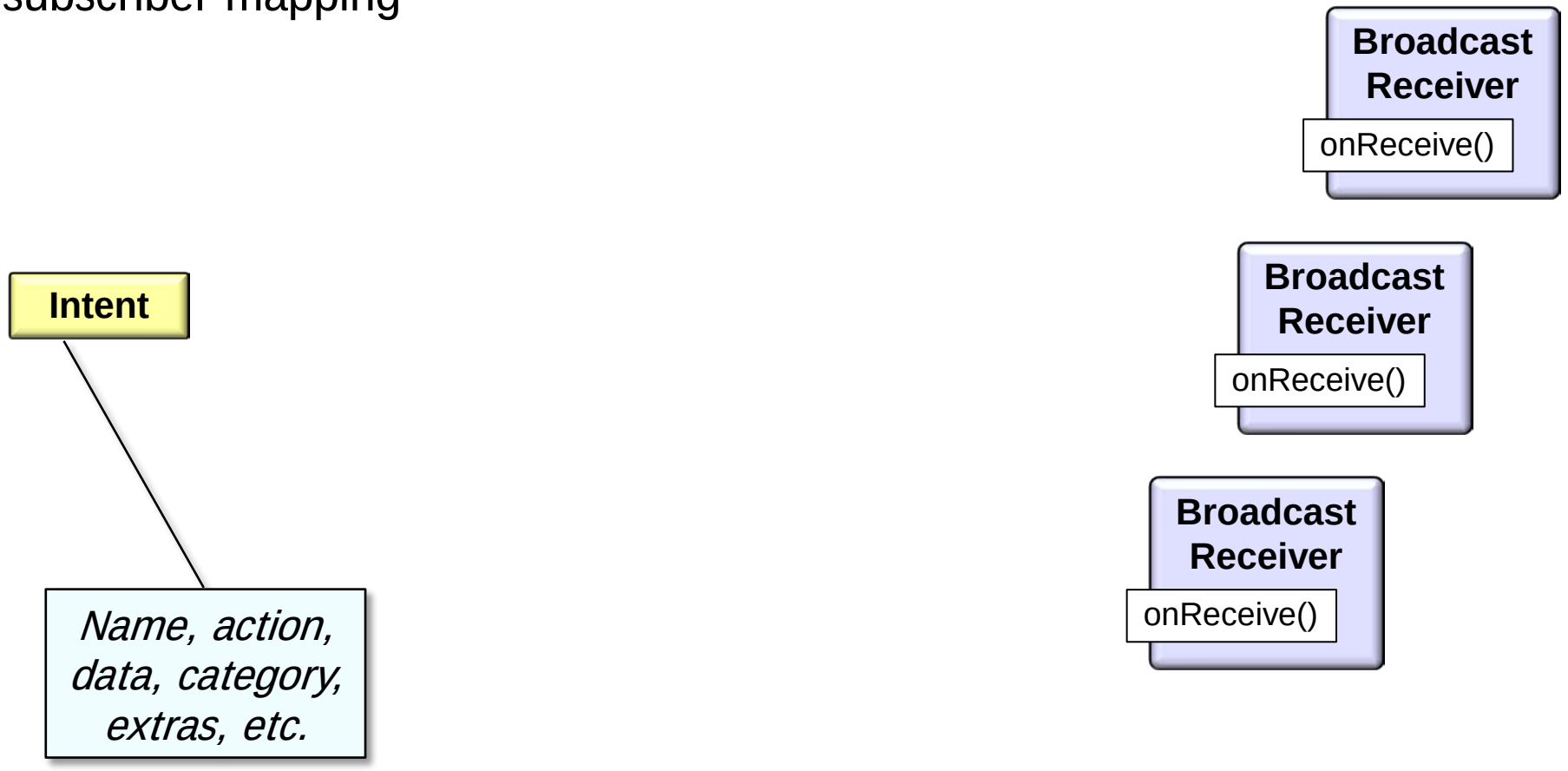


en.wikipedia.org/wiki/Publish-subscribe_pattern has more info

Publisher-Subscriber POSA1 Architectural

Implementation

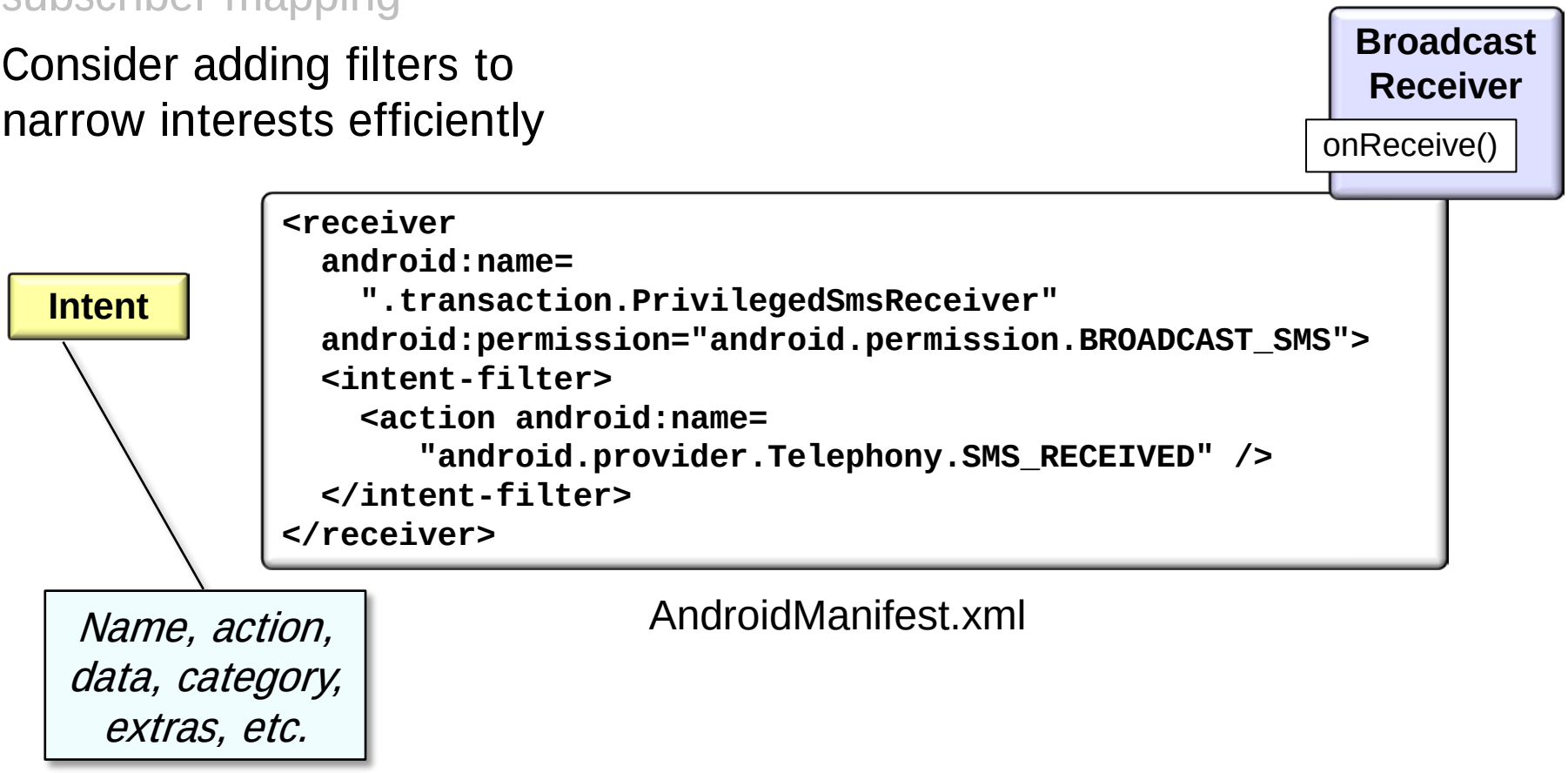
- Determine the publisher-subscriber mapping



Publisher-Subscriber POSA1 Architectural

Implementation

- Determine the publisher-subscriber mapping
- Consider adding filters to narrow interests efficiently



Publisher-Subscriber POSA1 Architectural

Implementation

- Determine the publisher-subscriber mapping
- Consider adding filters to narrow interests efficiently
- Define/implement the subscriber registration API
  - Provide method(s) for registering receives & (optionally) filters

```
public abstract class Context {
  ...

  public abstract Intent
    registerReceiver
      (BroadcastReceiver receiver,
       IntentFilter filter);

  public abstract Intent
    registerReceiver
      (BroadcastReceiver receiver,
       IntentFilter filter,
       String broadcastPermission,
       Handler scheduler);
  ...
```

frameworks/base/core/java/android/content/Context.java has source code

Publisher-Subscriber POSA1 Architectural

Implementation

- Determine the publisher-subscriber mapping
- Consider adding filters to narrow interests efficiently
- Define/implement the subscriber registration API
  - Provide method(s) for registering receives & (optionally) filters
  - Registered subscribers are typically stored in an internal data structure

```
class ActivityManagerService
    extends ActivityManagerNative ... {
  ...
  final HashMap mRegisteredReceivers
    = new HashMap();

  public Intent registerReceiver
    (IApplicationThread caller,
     String callerPackage,
     IIntentReceiver receiver,
     IntentFilter filter,
     String permission) {
    ...
    ReceiverList rl = (ReceiverList)
        mRegisteredReceivers.
          get(receiver.asBinder());
    ...
    mRegisteredReceivers.
        put(receiver.asBinder(), rl);
    ...
```

frameworks/base/services/java/com/android/server/am/ActivityManagerService.java

Publisher-Subscriber POSA1 Architectural

Implementation

- Determine the publisher-subscriber mapping
- Consider adding filters to narrow interests efficiently
- Define/implement the subscriber registration API
- Define/implement the subscriber notification API
  - Provide method(s) for controlling how notifications are delivered

```
public abstract class Context {
  public abstract void
    sendBroadcast(Intent intent);

  public abstract void
    sendOrderedBroadcast
       (Intent intent,
        String receiverPermission);
  ...
```

Publisher-Subscriber POSA1 Architectural

Implementation

- Determine the publisher-subscriber mapping
- Consider adding filters to narrow interests efficiently
- Define/implement the subscriber registration API
- Define/implement the subscriber notification API
  - Provide method(s) for controlling how notifications are delivered
  - Handle concurrent & sequential deliveries

```
class ActivityManagerService
    extends ActivityManagerNative ... {
  ...
  private final int
    broadcastIntentLocked
      (..., Intent intent, ...) {
    ...
    receivers = AppGlobals.
      getPackageManager().
        queryIntentReceivers(intent,
                             ...);
    ...
    registeredReceivers =
      mReceiverResolver.queryIntent
        (intent, ...);
    ...
  }
```

Static receivers

Dynamic receivers

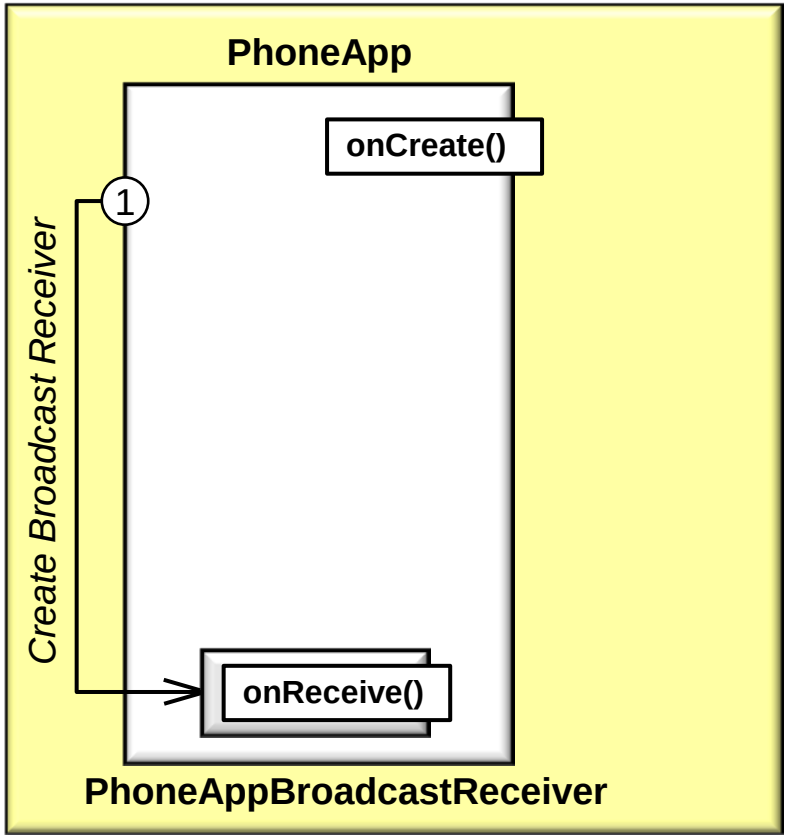
Broadcast intent to receivers

frameworks/base/services/java/com/android/server/am/ActivityManagerService.java

Publisher-Subscriber POSA1 Architectural

Applying the Publisher-Subscriber pattern in Android

- Use the Intents framework to report low battery status on an Android device

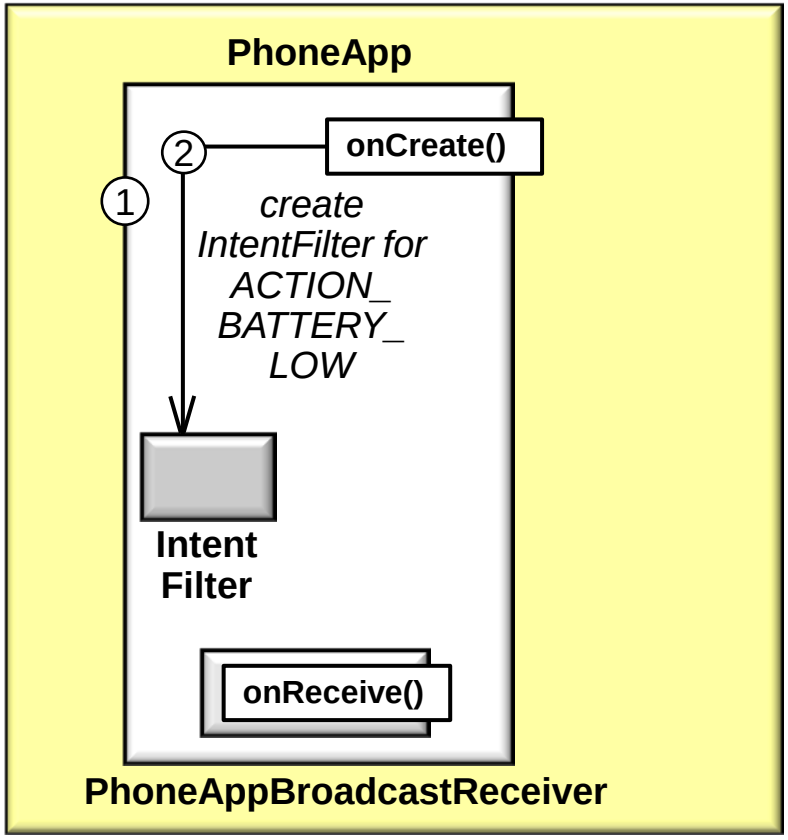


packages/apps/Phone/src/com/android/phone/PhoneApp.java has source code

Publisher-Subscriber POSA1 Architectural

Applying the Publisher-Subscriber pattern in Android

- Use the Intents framework to report low battery status on an Android device

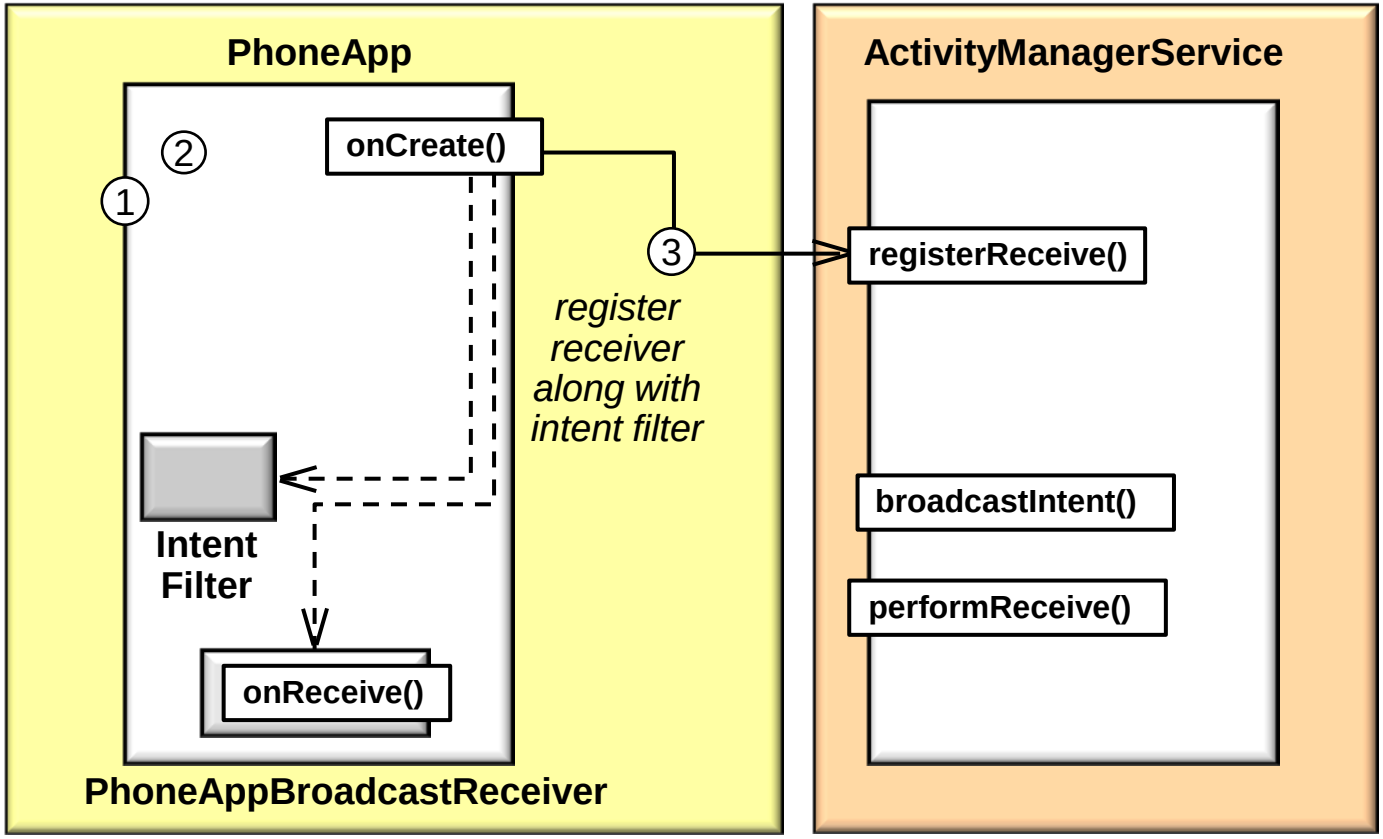


packages/apps/Phone/src/com/android/phone/PhoneApp.java has source code

Publisher-Subscriber POSA1 Architectural

Applying the Publisher-Subscriber pattern in Android

- Use the Intents framework to report low battery status on an Android device

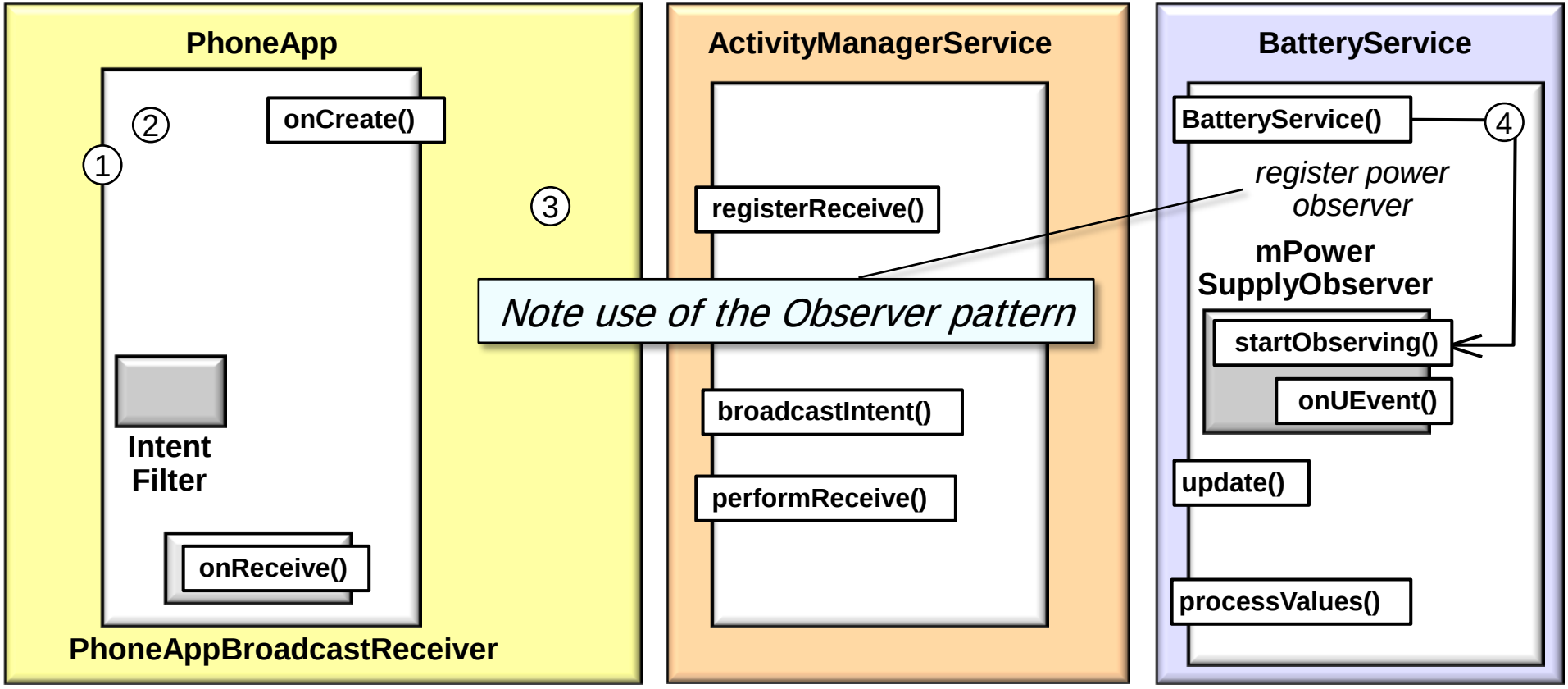


frameworks/base/services/java/com/android/server/am/ActivityManagerService.java

Publisher-Subscriber POSA1 Architectural

Applying the Publisher-Subscriber pattern in Android

- Use the Intents framework to report low battery status on an Android device

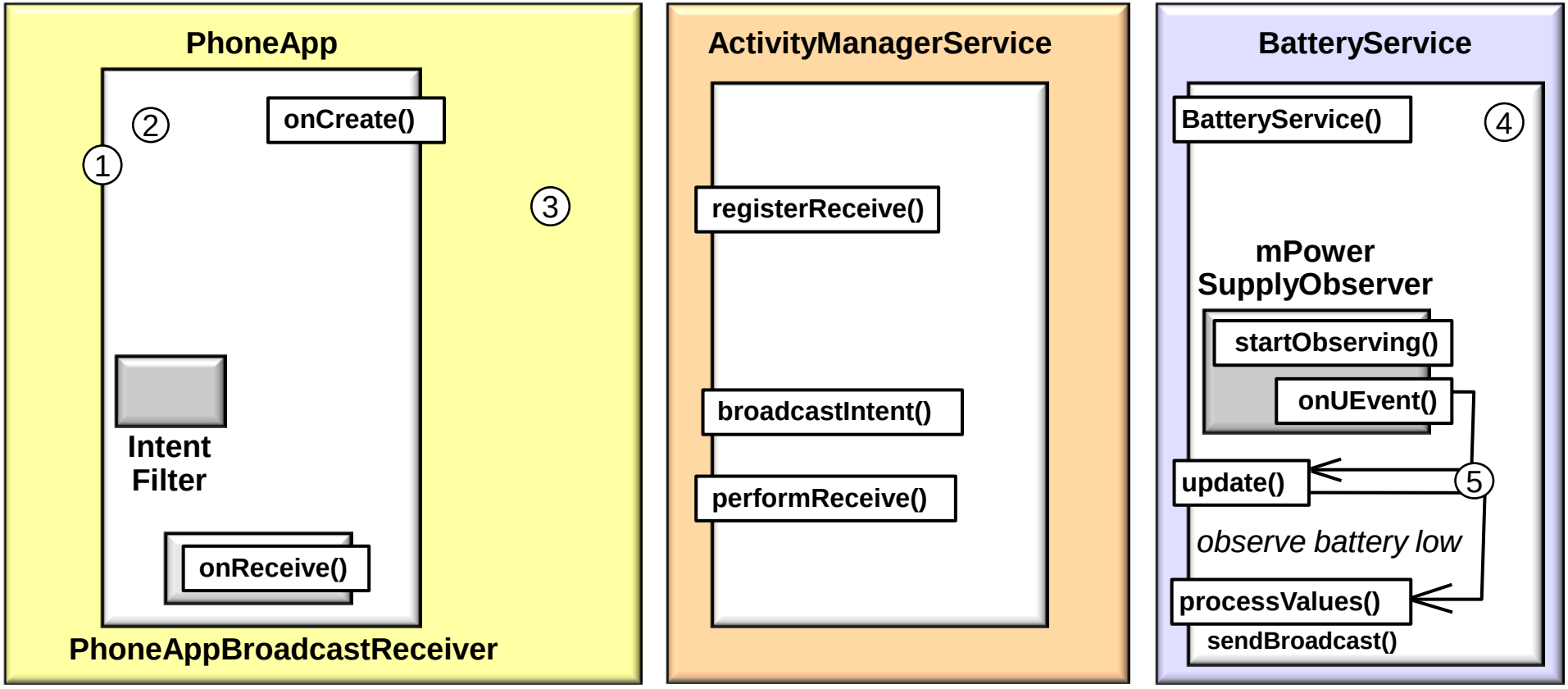


See frameworks/base/services/java/com/android/server/BatteryService.java

Publisher-Subscriber POSA1 Architectural

Applying the Publisher-Subscriber pattern in Android

- Use the Intents framework to report low battery status on an Android device

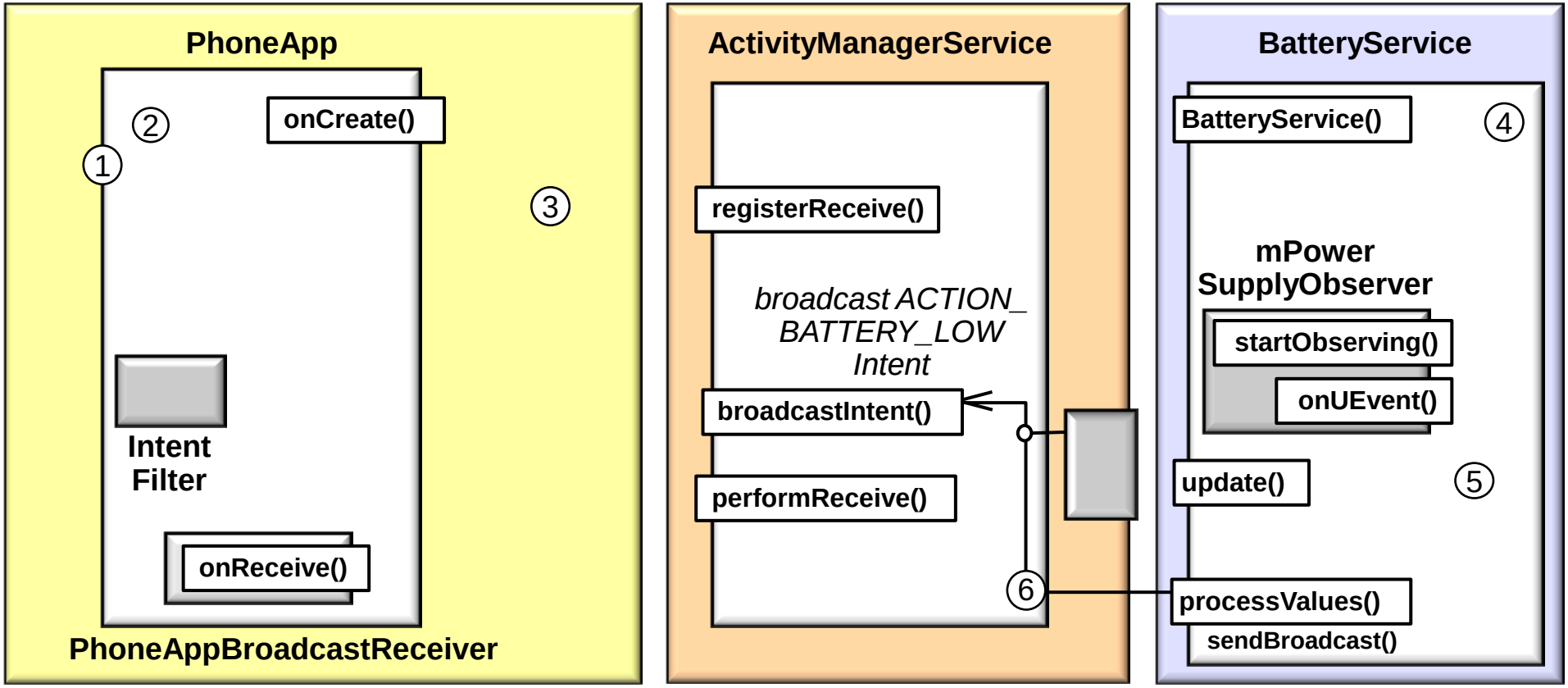


See frameworks/base/services/java/com/android/server/BatteryService.java

Publisher-Subscriber POSA1 Architectural

Applying the Publisher-Subscriber pattern in Android

- Use the Intents framework to report low battery status on an Android device

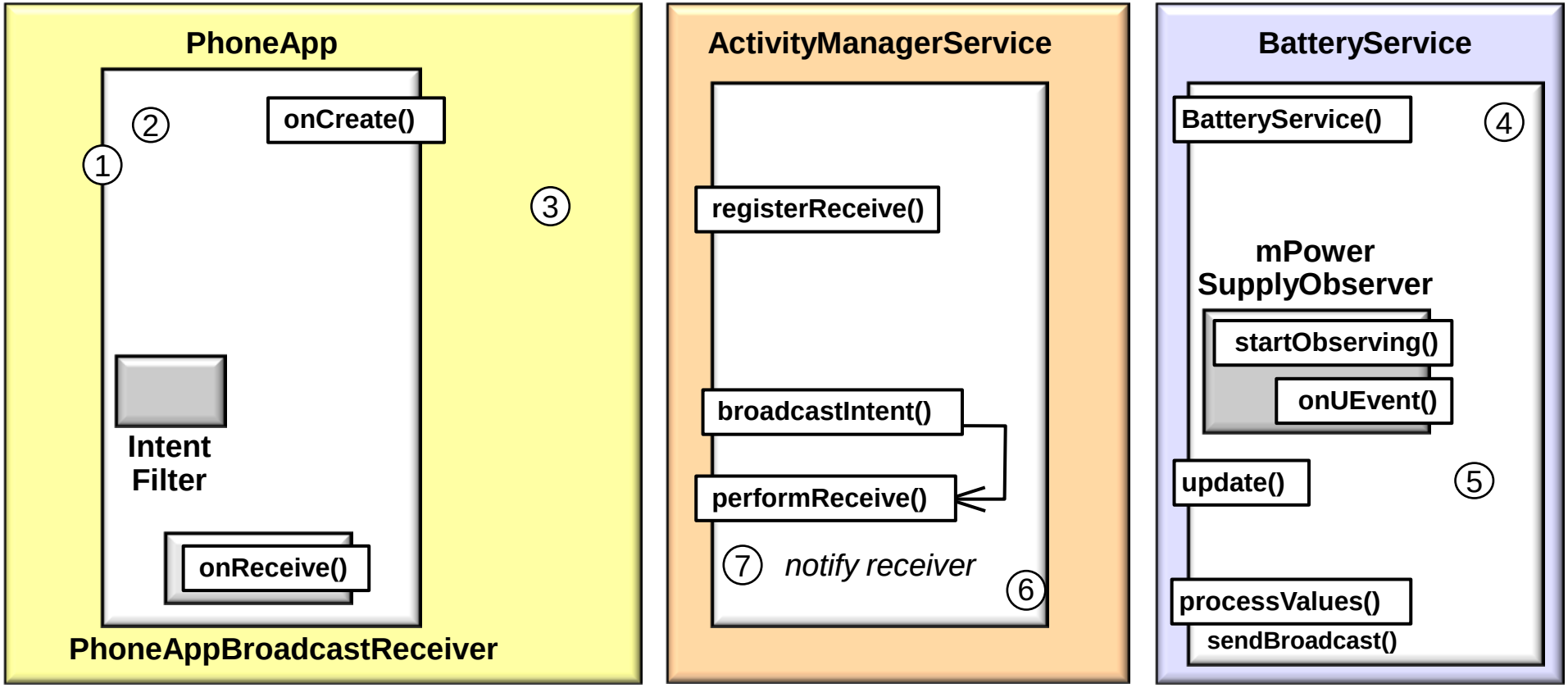


frameworks/base/services/java/com/android/server/am/ActivityManagerService.java

Publisher-Subscriber POSA1 Architectural

Applying the Publisher-Subscriber pattern in Android

- Use the Intents framework to report low battery status on an Android device

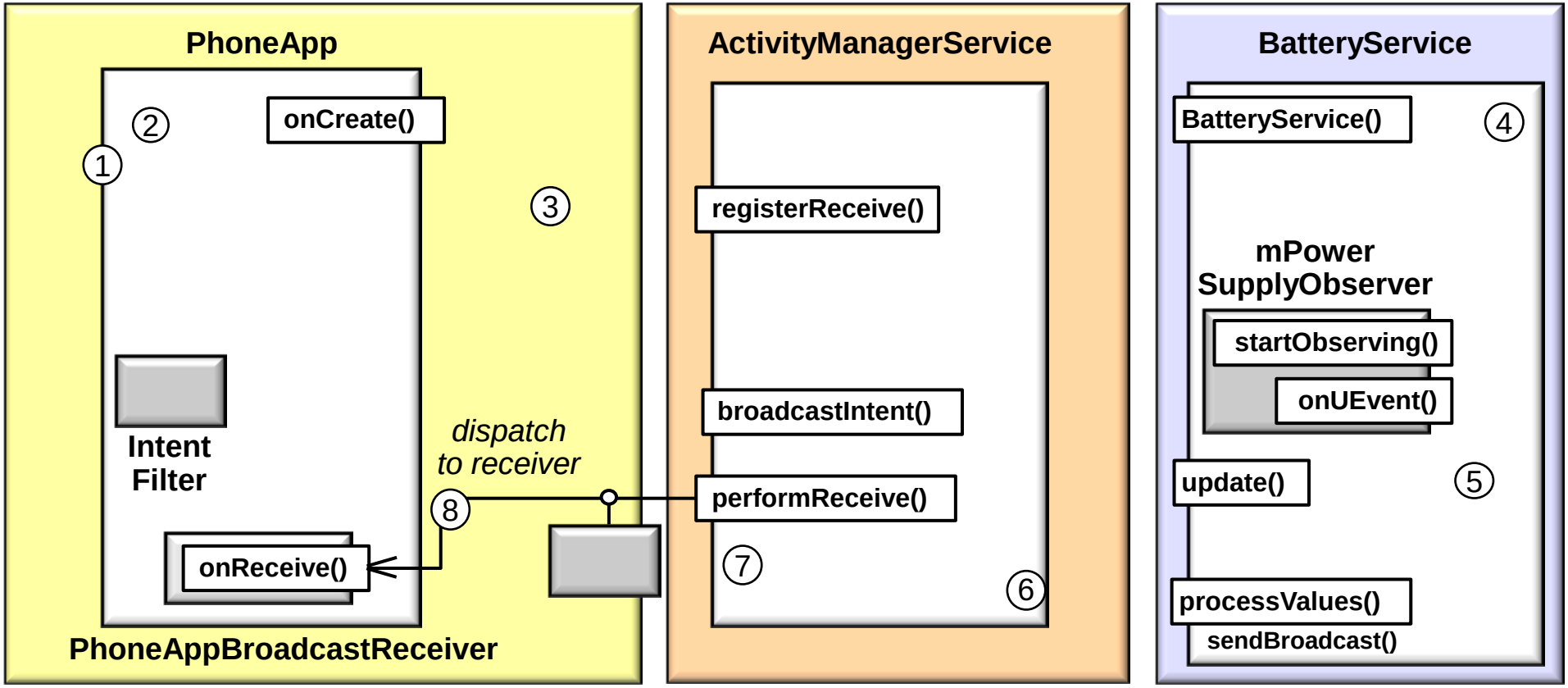


frameworks/base/services/java/com/android/server/am/ActivityManagerService.java

Publisher-Subscriber POSA1 Architectural

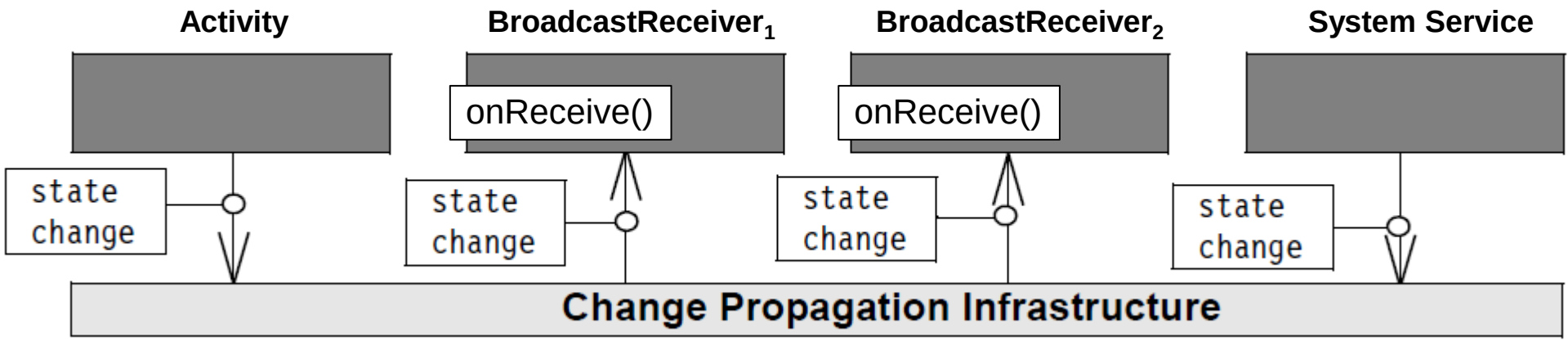
Applying the Publisher-Subscriber pattern in Android

- Use the Intents framework to report low battery status on an Android device



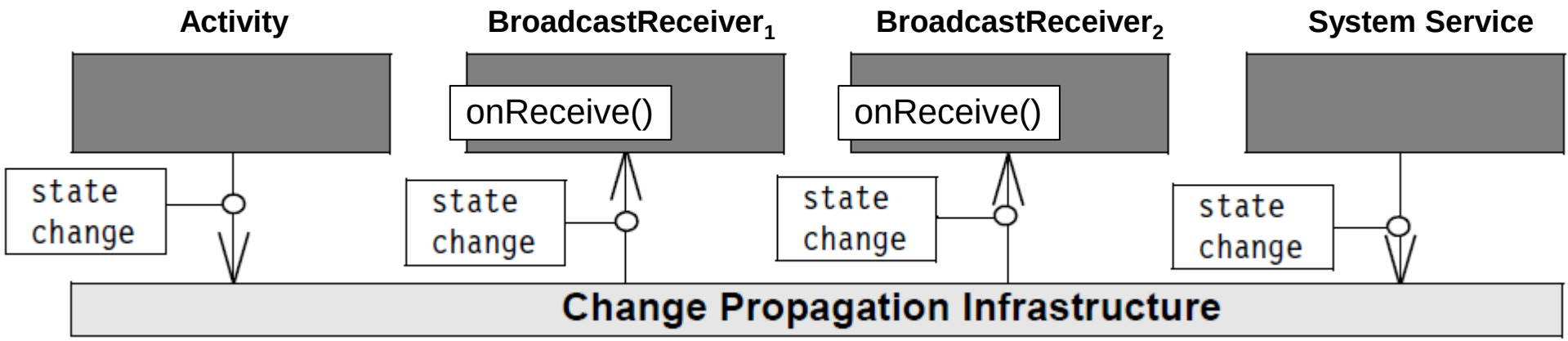
packages/apps/Phone/src/com/android/phone/PhoneApp.java has source code

Summary



- Android implements the *Publisher-Subscriber* pattern via the Intents framework to enable late run-time binding between components in the same or different Apps
  - The Intent object is a passive data structure holding an abstract description of some change that has occurred & is being announced

Summary



- Android implements the *Publisher-Subscriber* pattern via the Intents framework to enable late run-time binding between components in the same or different Apps
- Intent objects passed to any of the broadcast methods (such as Context. sendBroadcast() or Context.sendOrderedBroadcast()) are delivered to all interested broadcast receivers