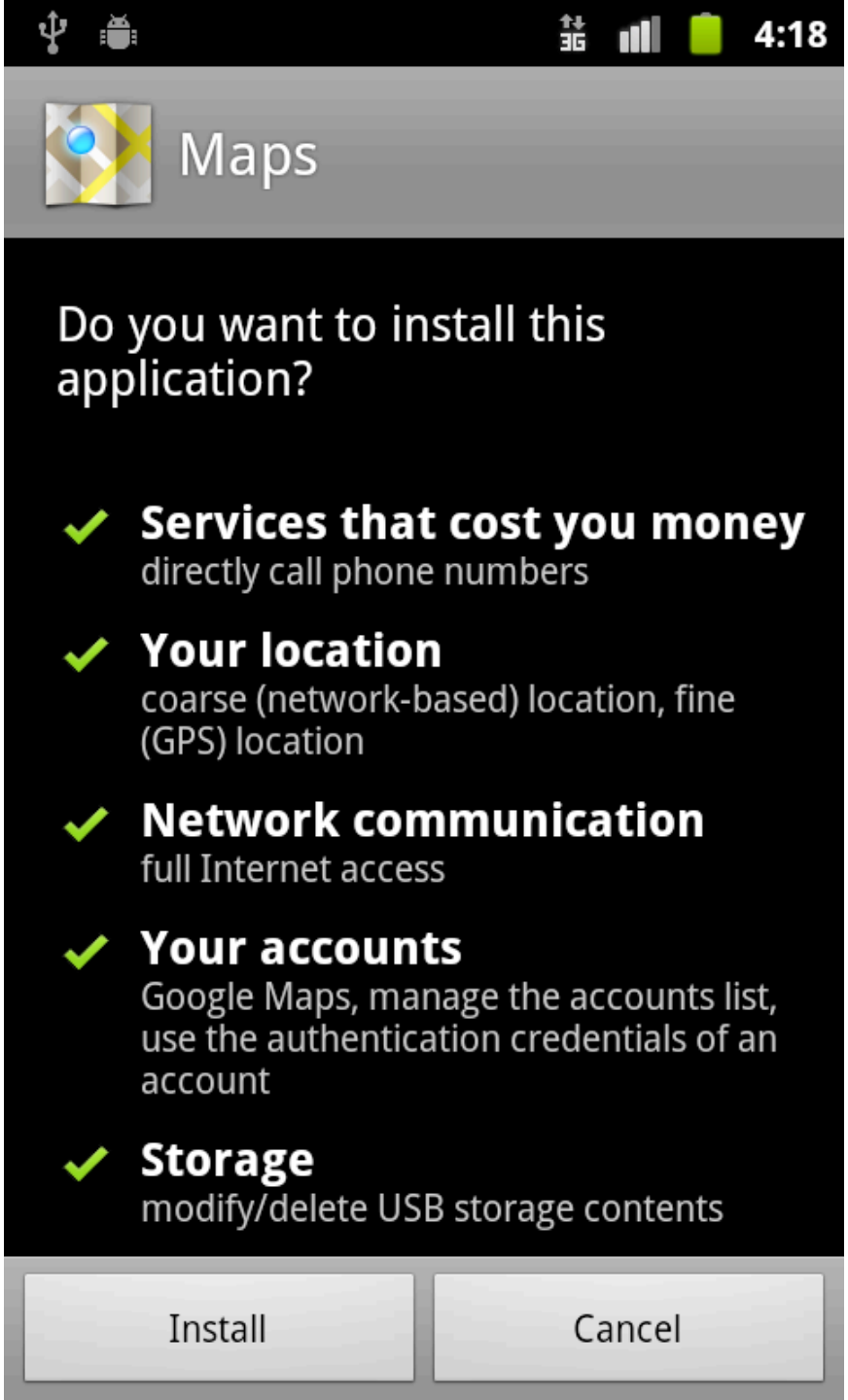
G54MDP
Mobile Device Programming

Lecture 13 - Permissions and Security, BroadcastReceivers


4:18
Maps
Do you want to install this application?
Services that cost you money
directly call phone numbers
Your location
coarse (network-based) location, fine (GPS) location
Network communication
full Internet access
Your accounts
Google Maps, manage the accounts list, use the authentication credentials of an account
Storage
modify/delete USB storage contents
Install
Cancel

Permissions

- Cost-Sensitive APIs
  - Telephony
  - SMS/MMS
  - Network/Data
  - In-App Billing
  - NFC Access
- Personal Information
  - Contacts, calendar, messages, emails
- Device Meta-data
  - System logs, browser history, network identifiers
- Sensitive Input Devices
  - Interaction with the surrounding environment
  - Camera, microphone, GPS

Using Permissions

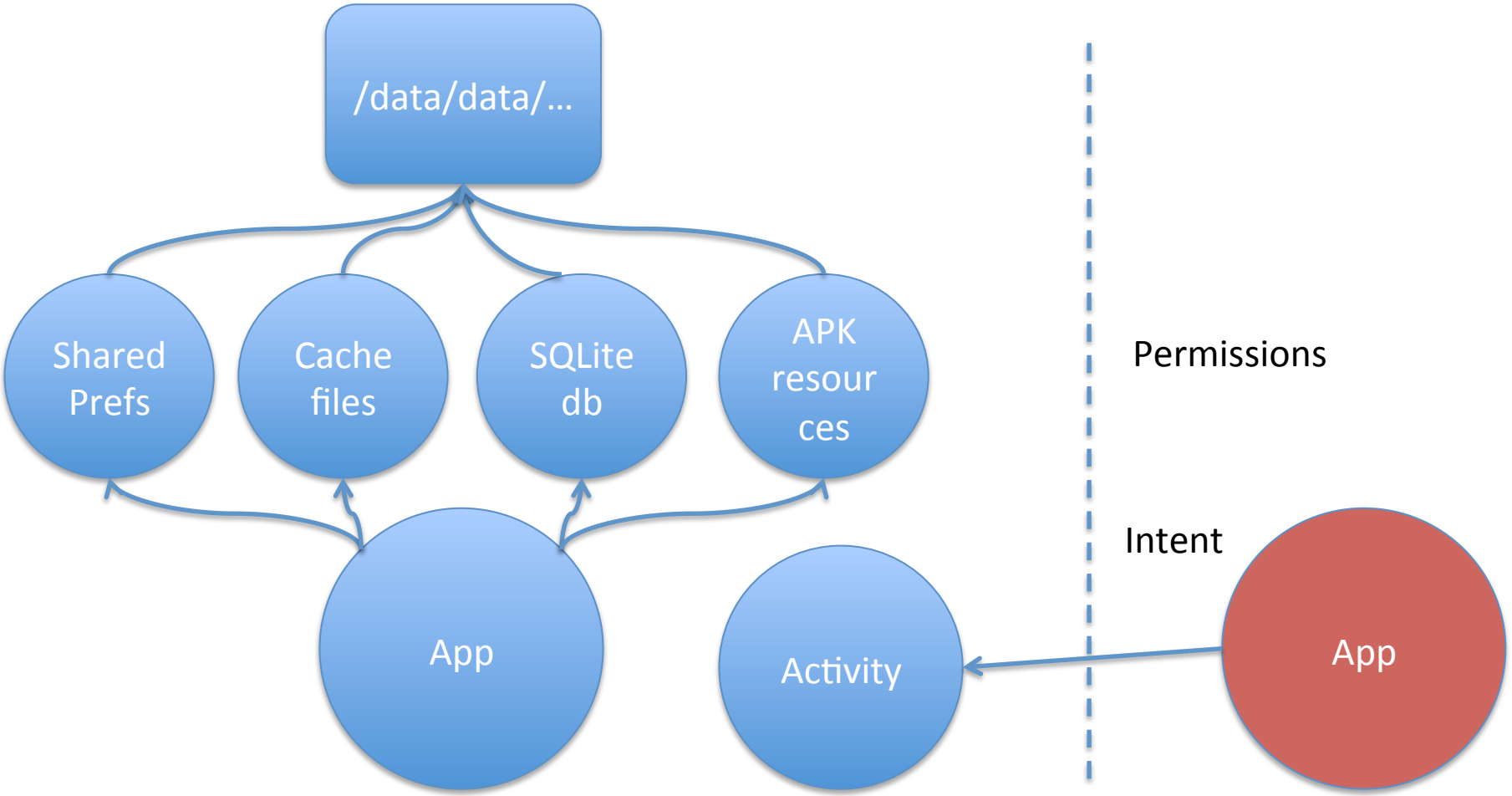
- Specify that an Application **uses** a permission

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```

- Specify that an Application **requires** a permission
  - The app must **use** permissions it **requires**

```
<provider
    android:permission="android.permission.READ_CONTACTS”
    android:authorities="com.example.martincontentprovider.MyProvider”
    android:multiprocess="true”
    android:name="com.example.martincontentprovider.MyProvider">
</provider>
```

Sharing Data – is this good enough?



Component Permissions

- Activity
  - Restricts which components can start the activity
  - Checked within execution of:
    - startActivity()
    - startActivityForResult()
- Service
  - Restricts which components can start or bind to the associated service
  - Checked within execution of:
    - Context.startService()
    - Context.stopService()
    - Context.bindService()
- ContentProvider
  - Restricts which components can read or write to a ContentProvider
- Throw SecurityException on permissions failure
  - Usually as we've forgotten to ask for permission during installation

Querying Permissions

- Cannot really ask for more permissions at runtime
- Can query permissions before we try and do something / allow an app to do something
- Does the app using my Service have a permission?
  - Context.checkCallingPermission()
- Does a process have a permission?
  - Context.checkPermission()
- Does an installed app have a permission?
  - PackageManager.checkPermission()

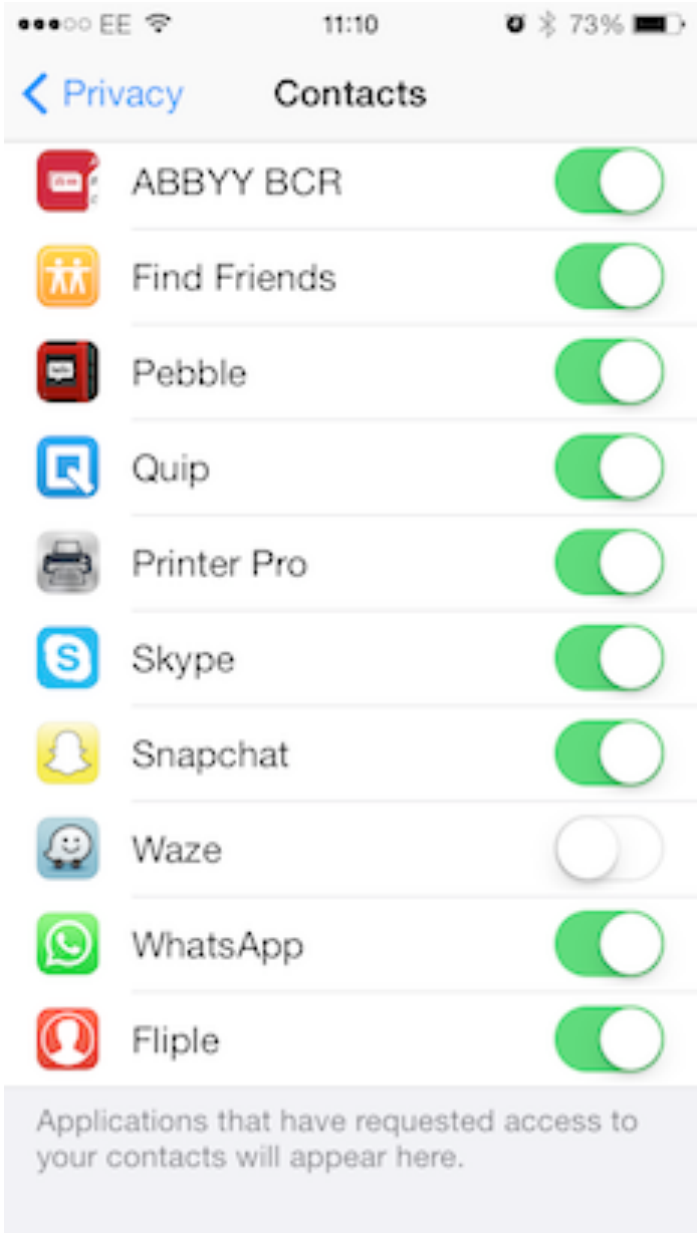
Temporary URI Permissions

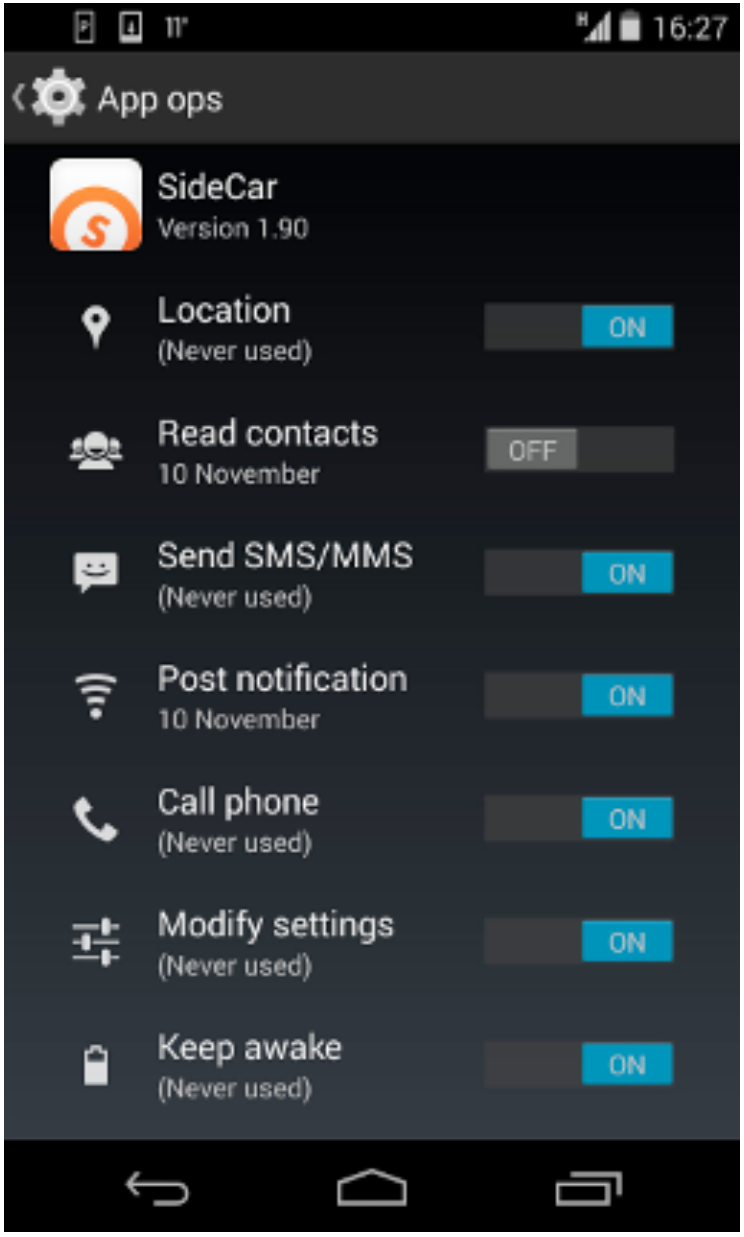
- Applications making use of multiple Activities
  - "Access to the mail should be protected by permissions, since this is sensitive user data. However, if a URI to an image attachment is given to an image viewer, that image viewer will not have permission to open the attachment since it has no reason to hold a permission to access all e-mail."
  - Allow access to specific URIs, not the whole provider

android:grantUriPermissions="true"

myIntent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

- Temporary URI permissions last while the stack of the receiving Activity is active


EE
11:10
73%
Privacy
Contacts
ABBYY BCR
Find Friends
Pebble
Quip
Printer Pro
Skype
Snapchat
Waze
WhatsApp
Fliple
Applications that have requested access to your contacts will appear here.


16:27
App ops
SideCar
Version 1.90
Location
(Never used)
ON
Read contacts
10 November
OFF
Send SMS/MMS
(Never used)
ON
Post notification
10 November
ON
Call phone
(Never used)
ON
Modify settings
(Never used)
ON
Keep awake
(Never used)
ON

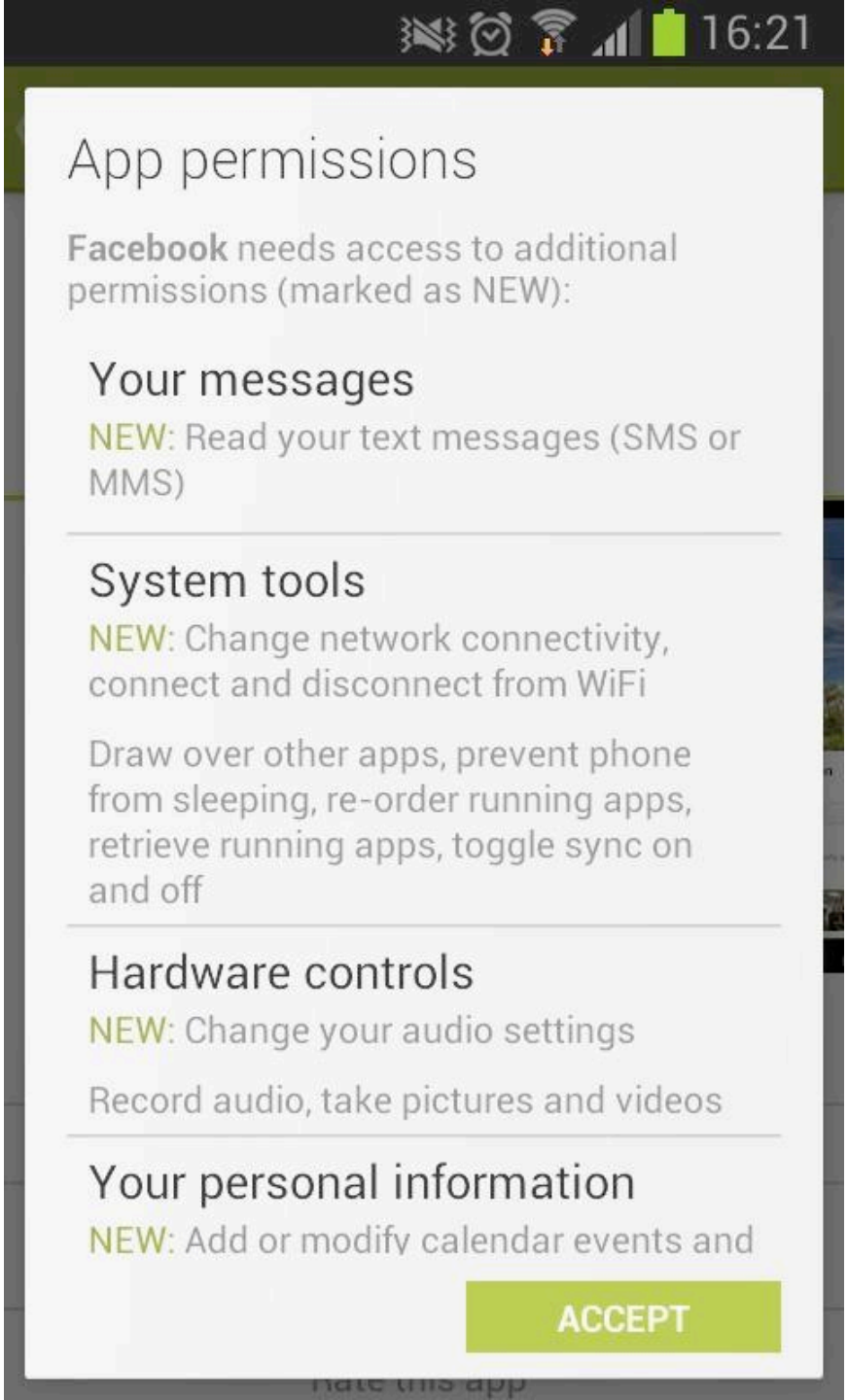
News › Technology › Android

Android torch app with over 50m downloads silently sent user location and device data to advertisers

US Federal Trade Commission charges 'deception' over app which turned on lights on Android smartphones - but also told advertisers about location and device information

Permissions in the wild

- Study participants displayed low attention and comprehension rates: both the Internet survey and laboratory study found that 17% of participants paid attention to permissions during installation, and only 3% of Internet survey respondents could correctly answer all three permission comprehension questions.
- http://dl.acm.org/citation.cfm?id=2335360


16:21
App permissions
Facebook needs access to additional permissions (marked as NEW):
Your messages
NEW: Read your text messages (SMS or MMS)
System tools
NEW: Change network connectivity, connect and disconnect from WiFi
Draw over other apps, prevent phone from sleeping, re-order running apps, retrieve running apps, toggle sync on and off
Hardware controls
NEW: Change your audio settings
Record audio, take pictures and videos
Your personal information
NEW: Add or modify calendar events and
ACCEPT

Permissions vs Use

- Read your text messages
  - To confirm your phone number via text message (if you've added it to your account)
- Read/write contacts
  - To import and sync your phone's contacts to Facebook, or vice versa (think updating contact images)
- Add and/or modify calendar events and send emails to guests without your knowledge
  - To see your Facebook events in your phone's calendar
- Read calendar events plus confidential information
  - To check your calendar for you to see if you have something already scheduled for the time of the Facebook event you're currently viewing

4 Application Components

- Activity
  - A UI for part of a task
- Service
  - A long-running background task
- ContentProvider
  - Storage and provision of data
- …all driven by user activity within an application
- BroadcastReceiver

BroadcastReceiver

- Respond to system-wide broadcast announcements
  - The user has not necessarily done something, but the OS / phone / another application has
    - The screen has turned off, the battery is low, a new SMS has arrived, the phone has booted
  - Can be sent by an application, or received by an application from the OS
- Intents are sent to specific Activities / Services
- Broadcasts are sent to anything that cares to listen
  - System-wide Intent broadcasts

Broadcasts

- Broadcasts are Intents
  - Send an Intent to start an Activity
  - Send an Intent to start a Service
  - Send an Intent to trigger Broadcast Receivers that subscribe to that particular class of Intent
    - Can define our own Broadcast Intents
    - Cannot send system Intents (battery, screen etc), as the security model prevents it

BroadcastReceiver

- Declare in AndroidManifest.xml
  - <activity>
  - <service>
  - <provider>
  - **<receiver>**
- Specify broadcast intents we are interested in
  - Listening to system intents may require certain permissions
    - i.e. phone state – why?
- Receiver registered at boot-time / install-time
  - Again, another **entry point** to our application

```
<receiver android:name="MyReceiver" >
    <intent-filter>
      <action android:name=”com.example.martinbroadcast" />
    </intent-filter>
</receiver>
```

Implementing a BroadcastReceiver

- Subclass BroadcastReciever
  - Specify which Intents we are interested in receiving
    - Permissions permitting
- Implement the onReceive() method
- Then…
  - Send a Message to our application to change our behavior
    - Reduce the audio volume, play a sound
  - Start an Activity
    - “Never start an Activity in response to an incoming broadcast Intent.”
  - Start a Service
  - Show a Notification
    - Alert the user that there is something that they need to interact with

Broadcast types

- Several broadcast methods available
- Normal or Ordered
  - Normal
    - Sent to all registered receivers
    - Undefined order
  - Ordered
    - Sequential processing in priority order
- Sticky or Non-Sticky
  - Sticky
    - Store the Intent after the initial broadcast
  - Non-Sticky
    - Discard the Intent after the broadcast
- Local or Global

Ordered Broadcasts

- sendOrderedBroadcast(intent, receiverPermission)

```
<receiver android:name="MyReceiver" >
    <intent-filter android:priority="2" >
       <action android:name=”com.example.martinbroadcast" />
    </intent-filter>
</receiver>
```

- Preferred receivers given the chance to consume the broadcast before it is delivered to less preferred receivers
  - How many things should notify you that you have a new message?
  - In onReceive()
    - abortBroadcast() to stop the chain of delivery

Sticky Broadcasts

- Sticky Intents are cached by Android
  - New Intents overwrite older Intents they match
- When BroadcastReceivers are dynamically registered
  - Cached sticky Intents matching the specified IntentFilter are broadcast to the BroadcastReceiver
  - One matching sticky Intent is returned to the caller
- isInitialStickyBroadcast()
  - How old is the broadcast?
  - ACTION_BATTERY_CHANGED

Sticky Broadcasts

- *Sticky broadcasts are **global** to the system*
- *And because of this, performing a sticky broadcast is multiple orders of magnitude slower than just implementing direct calls within your own app*
  - *IPC for each receiver to register, IPC to the system to send it, IPC from the system back to your app to deliver it, marshalling and unmarshalling of all the data within the Intent over both IPCs*
- *More than that, there is NO protection on them, so any other application can watch your sticky broadcasts, or even send their own values back to you*
- *Now I am really regretting that I made that function public. :/*

Local Broadcasts

- Like the normal BroadcastReceiver but only supports **local** broadcasts
  - Within the application
- Data broadcast will not leave the application
  - No data leakage
- Other applications cannot send broadcasts to our application
- More efficient than sending a global broadcast
  - No IPC
- Must register / sendBroadcasts using the LocalBroadcastManager
  - Programmatically rather than via the manifest

Caveats

- Either the sender or receiver of a broadcast can implement permissions
  - Control who can send broadcast intents to my application
  - Control who can receive the intents that my application broadcasts
  - Limit broadcasts to my application components only
- Lifecycle
  - A receiver handling broadcasts is considered to be running in the foreground (albeit briefly)
  - Process is aggressively killed once onReceive() has returned
    - No binding, no startActivityForResult
  - Long-running code should start a Service instead
    - As with any other Activity
- The application receiving the broadcasts must have been explicitly started / not explicitly stopped
  - Applications cannot intercept broadcasts without the user having some awareness that they have given it permission
- Can register / unregister for broadcast events programmatically
  - Most things that the manifest specifies can be done programmatically

Let’s have a look…

Facebook

- Design exercise
- Which…
  - Activities
  - Services
  - ContentProviders
  - BroadcastReceivers?

References

- http://developer.android.com/guide/topics/providers/content-providers.html
- http://developer.android.com/guide/components/fundamentals.html
- http://developer.android.com/reference/android/content/BroadcastReceiver.html