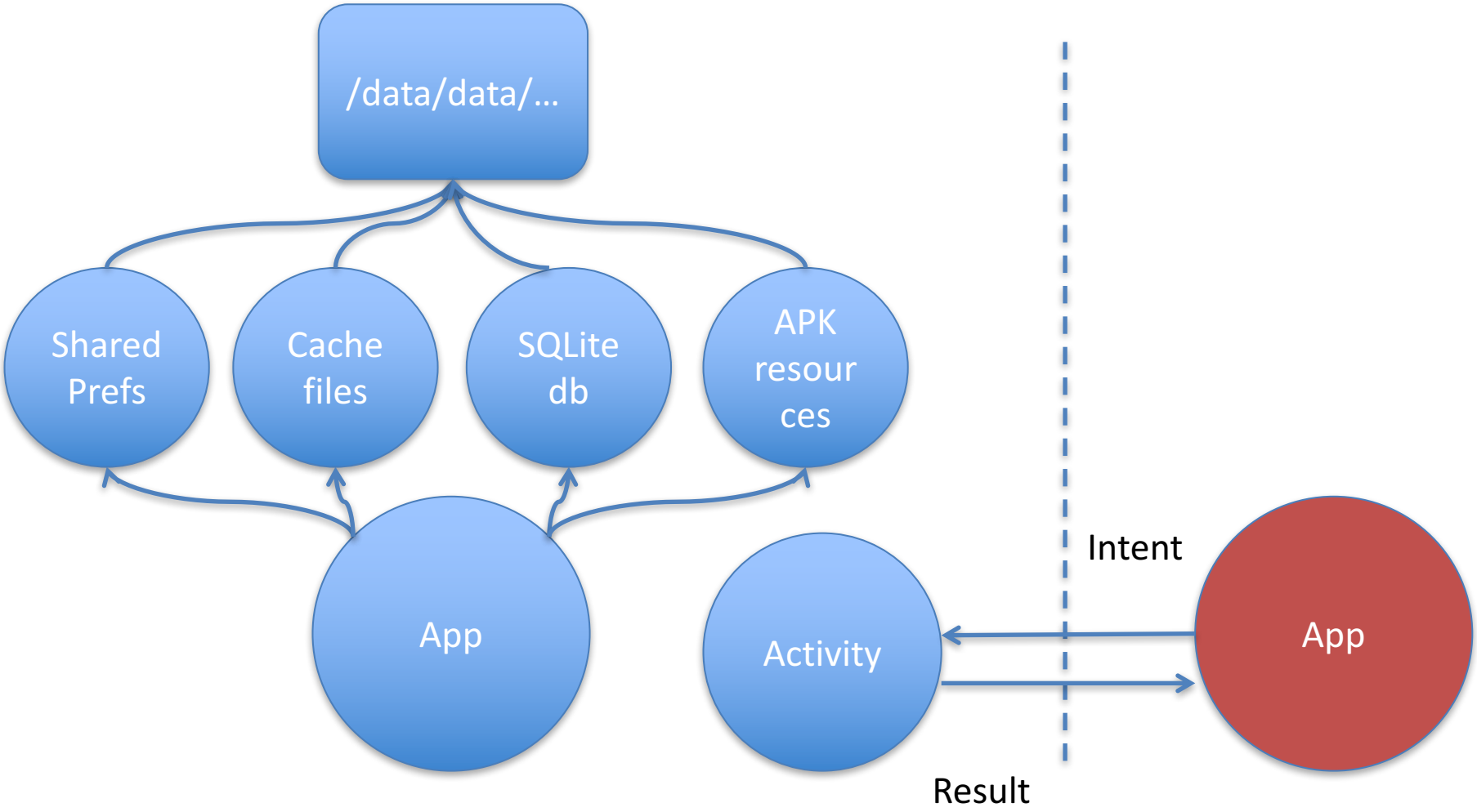
G53MDP
Mobile Device Programming

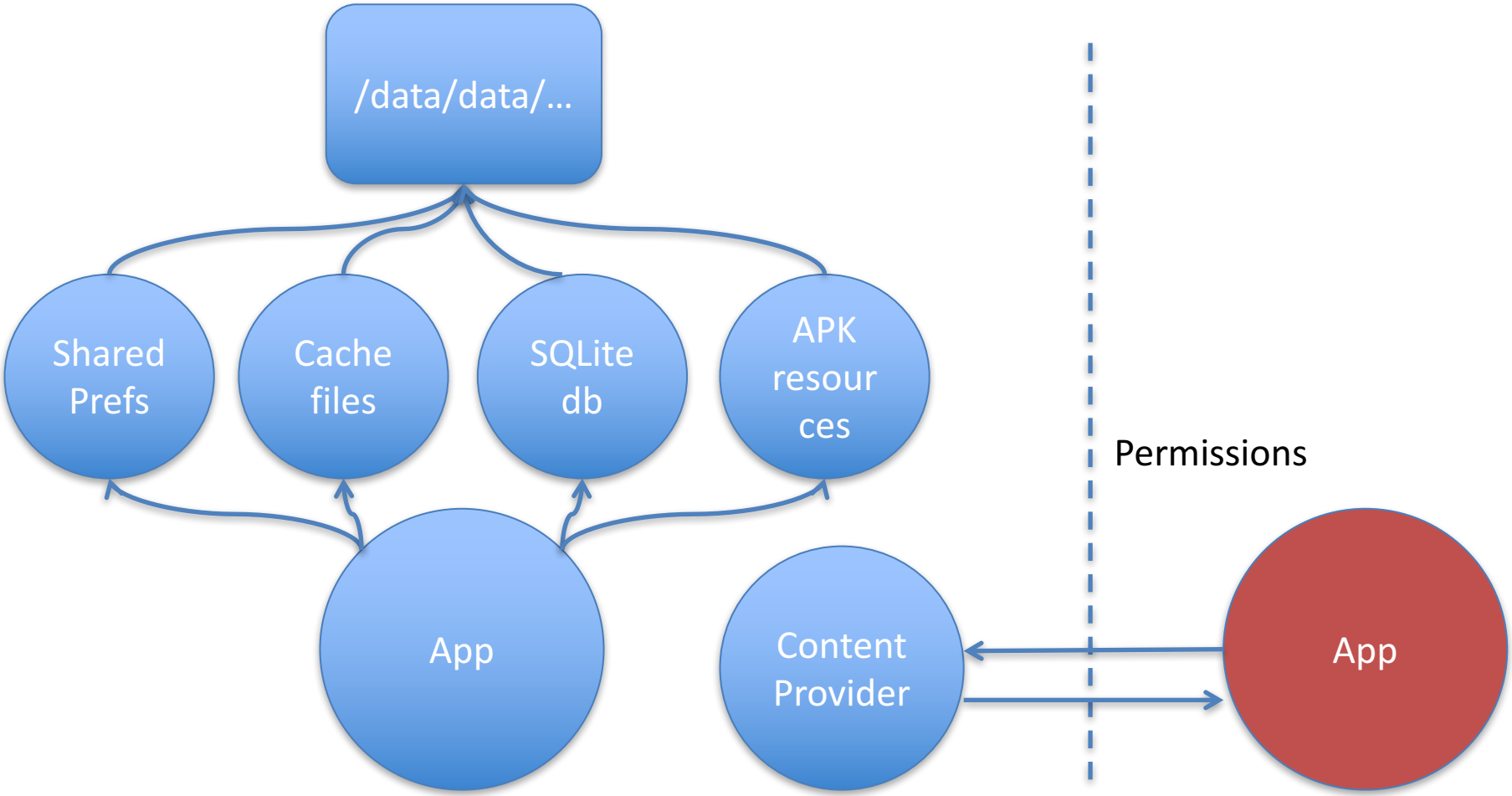
Lecture 13 –Content Providers

Broadcasts

Sharing Data – is this good enough?



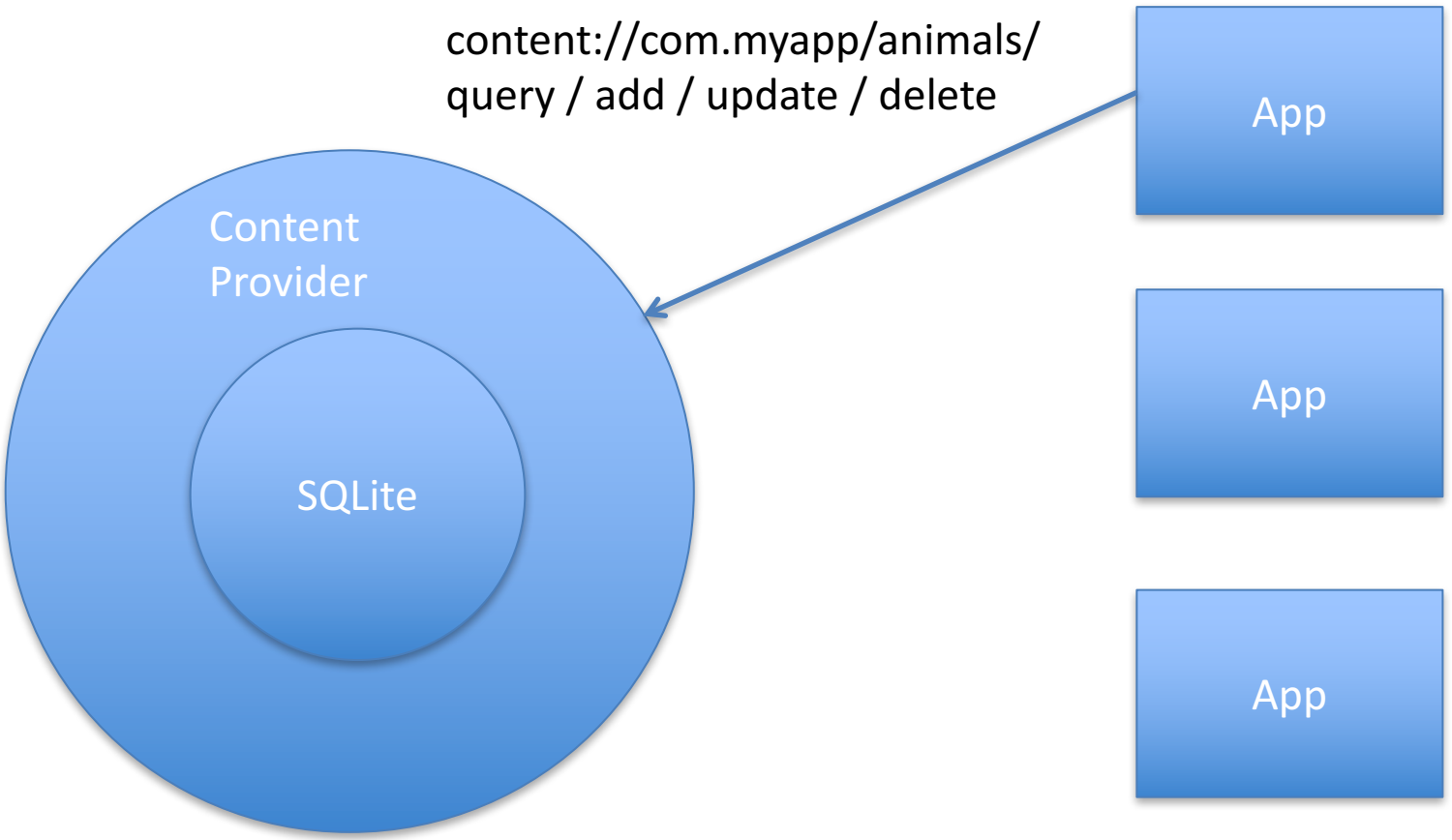
Sharing Data – if not



ContentProvider

- Access to data is restricted to the app that owns it
  - Database is located in *internal* app-specific storage
    - Inaccessible by other applications
  - If we want other apps to access our data, or we want to access other apps' data, or we want to be notified when data has changed
- Provide or make use of a **ContentProvider**
    - Application component number 3
    - Exposes data / content to other applications in a structured manner
    - Fundamentally IPC via Binder (again) + ashmem with a well defined (database-like) interface

Data Model


content://com.myapp/animals/
query / add / update / delete
Content
Provider
SQLite
App
App
App

Modifying a ContentProvider

- Uri insert(Uri uri, ContentValues values)
  - Get back the URI of the newly inserted item
- int update(Uri uri, ContentValues values, String where, String[] selectionArgs)
- Uri
  - The “table” that we wish to update / insert into
- ContentValues
  - Values for the new row
  - Key/value pairs
    - Key is the “column” name
- where
  - SQL WHERE clause

Creating a Content Provider

- Implement a storage system for the data
  - Structured data / SQLite
    - Values, binary blobs up to 64k
    - Database
  - Large binary blobs
    - Files
  - Photos / media manager
- Implement a ContentProvider
  - query, add, update, insert etc
  - onCreate
  - getType
    - What type of data are we providing?
      - Single item, multiple items, mime types
  - ParcelFileDescripter openFile()
- Tell Android we are a provider
  - Declare in the AndroidManifest
    - **android:multiprocess**
      - “Don't use this, it is some old cruft from pre-1.0 design that doesn't work and should be ignored these days. Just pretend like the attribute doesn't exist. :}”

Contract

- Defines metadata pertaining to the provider
- Constant definitions that are exposed to developers via a compiled .jar file
  - Authority
    - Which app is responsible for this data
  - URI
  - Meta-data types
  - “Column” names
    - Abstraction of database architecture
      - Why?

URI Matching

- All of these methods (except onCreate()) take a URI as the first parameter
  - The object will need to parse it to some extent to know what to return, insert or update
  - Android provides android.content.UriMatcher to simplify this
    - Provides mapping between abstraction of contract class to concrete db implementation
  - Does the calling application want all data from a table, or just a row, or a specific table?
    - content://authority/tablename/
    - content:// authority/tablename/#
    - Or a “virtual” table
      - Joins across various underlying resources

Let’s have a look…

Network

- Cloud storage
  - Use the network to push data off the device
    - The cloud provides the illusion of “infinite” storage
- Implement a SyncAdapter
  - Synchronizes a local database / content provider with a remote server
  - Components
    - A content provider
      - To locally store the data
    - An authenticator
      - Appears in the “Accounts and Sync” menu, stores network credentials
      - Wrapped in a service
    - A SyncAdapter
      - Performs the network communication
      - Wrapped in a service
        - » So external processes can bind to it, triggering the synchronisation
  - Make use of a Service to push data in the background
- http://developer.android.com/training/sync-adapters/creating-sync-adapter.html

4 Application Components

- Components are *entry points*
  - Activity
    - To a user interface as part of a task
  - Service
    - To perform a long-running background task or computation, or access to a specific system resource
  - ContentProvider
    - To storage and provision of data
- What else?
  - How else to drive activity within an application?
  - What is a missing entry point?

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
  - Explicitly or implicitly via Intent filters, URI provision
- 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
      - Why?
  - Again a messaging wrapper around Binder

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

References

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