Mobile Programming

Ian Batten
igb@batten.eu.org

Today’s Content

- Random Things
- Files
- Permissions
- Networking Overview
- Networking Libraries
- Networking Power / Performance Considerations

Random Things

- VM is available from:
- /home/archive/users/igb986/androidvm.zip on school machines
- http://www.batten.eu.org/~igb/androidvm.zip
- Password for “ubuntu” account is “reverse”
- GPS source code is at
- http://www.batten.eu.org/~igb/GPS.zip

Infix Operators

- All “infix” means is that operators come between numbers.
- So a calculator where you type 3+4+5= and get 14 is “infix”.

Files

- We might, especially for data-logging, want to emit output into a file, or read it back for analysis.
- These are **private** files: only our application (or other applications in the same bundle) can read the files.

Writing to Files

- Could not be easier!
- `FileOutputStream openFileOutput (String name, int mode)`
    - Mode is combination of MODE_APPEND, MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITABLE
    - Files end up in private directory local to the application.

Reading from a file

- Obvious inverse function
- `FileInputStream openFileInput (String name)`
  - Has to exist
  - Again, private to application

These are standard Java IO objects

- So you can create BufferedReader, BufferedWriter and so on.
- If you’re doing enough file I/O that it matters, you’re probably doing something that isn’t sensible, but you should create buffered streams anyway as a matter of good practice.

Things more useful than plain files

- `SQLiteDatabase openOrCreateDatabase (String name, int mode, SQLiteDatabase.CursorFactory factory)`
- Allows storage of structured data, and retrieval with SQL queries
  - SQL Lite isn't multi-user, but **is** ACID
  - Always use this rather than your own weird scheme.
  - We will look at this in more detail later in the course

File Transfer

```
adb shell "run-as com.example.gps chmod 666 \
      /data/data/com.example.gps/files/breadcrumbs.txt”
adb pull /data/data/com.example.gps/files/breadcrumbs.txt
adb shell "run-as com.example.gps chmod 600 \
      /data/data/com.example.gps/files/breadcrumbs.txt”
```

600: read/write to owner
666: read/write to owner, group, other

Three octal digits, owner, group, other.
4 = read, 2 = write, 1 = execute

Unix chmod takes “symbolic” modes (a+r, g=rw, etc), but Android wants good old fashioned 7th Edition octal modes.

Permissions

- Applications can do a lot: they are running on top of a full-functionality Unix machine, after all.
- A typical phone contains a wide range of personal data, and an application that could access all of it could probably use it to obtain yet more.
- So we need to limit what individual apps can do, so that users have to approve access.

Unix Permissions Aren’t Enough

- Standard Unix/Linux permissions aren’t enough: anything running as root can do anything, anything running as you can access all your data.
- Running individual apps as individual users is possible, but makes interprocess-communication a nightmare (possible for self-contained subsystems).
- Mechanisms such as SELinux (NSA), OSX Sandboxes and so on are configured as part of the OS, not as part of the application. And are scary complex, too.

And Unix permissions don't really work anyway

```
ians-macbook-air:MUC13 igb$ cat > somefile
hello
ians-macbook-air:MUC13 igb$ cat somefile
hello
ians-macbook-air:MUC13 igb$ ls -l somefile
-rw-r--r--  1 igb  staff  6 11 Feb 12:29 somefile
ians-macbook-air:MUC13 igb$ chmod 400 somefile
ians-macbook-air:MUC13 igb$ ls -l somefile
-r--------  1 igb  staff  6 11 Feb 12:29 somefile
ians-macbook-air:MUC13 igb$ cat > somefile
-bash: somefile: Permission denied
ians-macbook-air:MUC13 igb$ ./myecho somefile i should not be able to do this
ians-macbook-air:MUC13 igb$ cat somefile
hello
i should not be able to do this
ians-macbook-air:MUC13 igb$ ls -l somefile
-r--------  1 igb  staff  39 11 Feb 12:29 somefile
ians-macbook-air:MUC13 igb$
```

How the deed was done

```
/* this is horrible code, do not emulate it */
int
main (int argc, char **argv) {
  struct stat statbuf;
  char *file = *++argv;
  int fd = open (file, O_RDONLY);
  fstat (fd, &statbuf);
  fchmod (fd, 0600);
  FILE *f = fopen (file, "a");
  while (*++argv) { fputs (*argv, f); putc (' ', f); }
  putc ('\n', f);
  fchmod (fd, statbuf.st_mode & 0777);
}
```

So we need
something better

Example: Location and Power

- It would be bad if an application were able to leak and store your location, just because you wanted to play Flappy Angry Candy Birds (TM).
- So we want applications that need to reference location data to ask permission before they are able to do so.
- And if we are tracking location, we want to be able to lock against going to sleep: this kills the battery, so again should be checked with the user.
    - Depending on who signed the application, not all permissions are queried.

The Manifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.gps"
    android:versionCode="1"
    android:versionName="1.0" >

  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/
  <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
```

Example Permissions

- ACCESS_FINE_LOCATION – obtain device location to GPS-type accuracy
- ACCESS_COARSE_LOCATION – obtain device location to WiFi-triangulation accuracy
- WAKE_LOCK – allow an application to suppress inactivity sleep, so that it can continue to (in this case) log GPS data to a file.
- there are many, many other permissions for networking, sensors, cameras, making calls, ringers, vibration, etc: pretty much everything that isn’t a simple GUI and computation app.

Checking Time

- Permissions are only checked when applications are installed.
- OS is responsible for keeping track of which permissions have been approved for an applications, and making sure the application doesn’t change without the permissions being invalidated.
- Of course, a user installing an app can’t check its operation in any detail. But “is this the sort of thing I’d expect an app that does this to want to do?” is not totally worthless, either.

Networking

- Obviously, it’s a full Java networking stack, so you can if you want write client/server applications using standard libraries.
  - I’ve been tempted to port a network time protocol server to leverage cheap GPS hardware.
- But we’re going to talk about “recommended” Android networking.

Issues

- We have a lot of networking stacks available to us:
  - WiFi
  - 4G/LTE
  - 3G
  - 2.5G HSDPA/EDGE/etc
  - 2G GPRS

Performance

- WiFi: could be as fast as 600Mb/sec (802.11n and a following wind) and is fairly reliable and low latency.
- GPRS, if supported, could be as slow as 30Kbps, drop a lot of packets and have 100ms or worse latency.
- Very few computers operate in a regime as variable as this.
- The performance could change within one connection.

Android Libraries

- Android has network libraries that wrap complex tasks into simple functions

- For example, to get a stream reading from a remote site:

```
URL url = new URL("http://www.android.com/");
   HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
   try {
     InputStream in = new BufferedInputStream(urlConnection.getInputStream());
     readStream(in);
    finally {
     urlConnection.disconnect();
   }
 }
```

- And you then can read HTML (or whatever) straight from the socket.

You can also POST

This allows us to upload

OS Default

These are needed for performance

```
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
   try {
     urlConnection.setDoOutput(true);
     urlConnection.setChunkedStreamingMode(0);
     // or setFixedLengthStreamingMode (contentLength)

     OutputStream out = new
BufferedOutputStream(urlConnection.getOutputStream());
     writeStream(out);

     InputStream in = new BufferedInputStream(urlConnection.getInputStream());
     readStream(in);
    finally {
     urlConnection.disconnect();
   }
 }
```

Note that you need to buffer

- Here using a Buffered Stream **does** matter
- So unless you are only transferring tiny (<1kB) amounts of data in a single operation, you need to wrap a Buffered Input or Output Stream around the HttpURLConnection
- Cost of making small writes is that it creates small packets (unless kernel does aggressive aggregation, which few now do) which cost bandwidth, battery and latency.
- Cost of making small reads is less serious, although it can harm performance under some unusual circumstances. Again, however, best practice is to be consistent.

Not just http

- If you ask for https, you get a secure channel with “sensible” defaults (HttpsURLConnection)
    - Can I please complain about Android/Java capitalisation conventions?
- Mechanisms exist for providing your own root certificates, specifying or excluding particular cipher suites and key-exchange protocols, etc, etc.
    - Most people who mess with the defaults have no real idea what they are doing, and in doing so reduce security, compatibility and performance.

If designing a client/server application

- **Always** use HTTP/s unless you have a very, very clear reason otherwise and have analysed the issues in the context of a mobile data service.
- Specifically, datagrams which “always” get delivered are much less likely to, and protocols which seem OK in development will turn out to have problems with NAT / firewall traversal / etc.
- You might find yourself using HTTP solely to stream content which isn’t HTML and doesn’t require URL parsing, argument passing etc. That’s OK: better use standard facilities than roll your own.
- With my security hat on, if you can use https, so much the better. Don’t listen to people telling you it imposes unacceptable load on servers: it is not 2004 anymore.

However…

- You don’t know how long any individual network operation is going to take.
- So if you call an input function on some stream, and encounter a pause in available input (either the other end isn’t sending, or the network is slow or dropping packets), the rest of the application will hang.

How to deal with latency

- It is possible, if you have worked in low-level Unix programming for a long time, to write non-blocking network code without using threading.
    - Either use select() on non-blocking file descriptors or use SIGIO and FIOASYNC (seriously, **do not do this**) to be woken up when file descriptors are ready to read.
- SIGIO (**do not do this**) is not usually exposed in Java, but select() is, via java.nio.channels.Selector
- But if you do this on Android, you need to provide your own event loop, and you are almost certainly going to end up using a second thread anyway. In which case…

…Android has you covered

- AsyncTask is a simple wrapper around Thread and Handler
- You subclass ("extend") AsyncTask to perform your work
  - You must provide doInBackground() (which is executed on a background thread).
  - You can provide onPreExecute(), onProgressUpdate() and onPostExecute(), all executed on the UI thread.
  - onProgressUpdate() is executed on the UI thread when publishProgress() is called on the background thread within doInBackground ().
- AsyncTasks are expected to complete within a few seconds; if you want to run tasks for longer, there are more complete solutions available in java.util.concurrent for running Runnable tasks (Executor, ThreadPoolExecutor, FutureTask).

Example AsyncTask

```
new DownloadWebpageTask().execute(stringUrl);
```

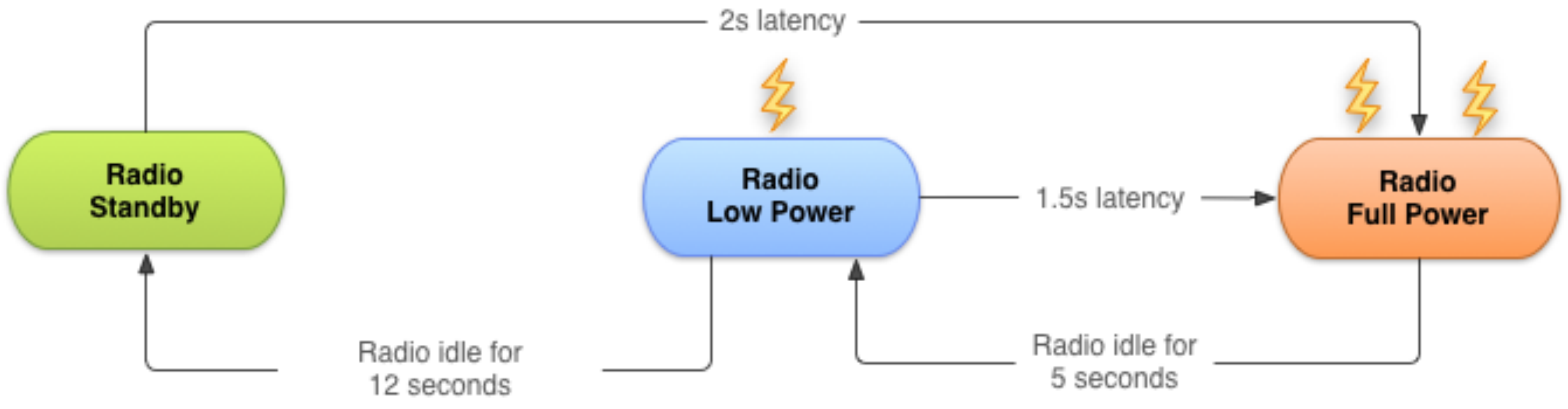
1. `Params`, the type of the parameters sent to the task upon execution.
2. `Progress`, the type of the progress units published during the background computation.
3. `Result`, the type of the result of the background computation.

```
private class DownloadWebpageTask extends AsyncTask<String, Void, String> {
        @Override
        protected String doInBackground(String... urls) {

            // params comes from the execute() call: params[0] is the url.
            try {
                return downloadUrl(urls[0]);
            } catch (IOException e) {
                return "Unable to retrieve web page. URL may be invalid.";
            }
        }
        // onPostExecute displays the results of the AsyncTask.
        @Override
        protected void onPostExecute(String result) {
            textView.setText(result);
       }
    }
```

Runs on UI Thread

Power Management



One byte transferred: 17 seconds (at least) of elevated battery use

You should batch transfers to occur less often than once every 20s: the larger the batches, the lower the amount of wasted battery

You can get details

```java
ConnectivityManager cm =
 (ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);

TelephonyManager tm =
  (TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);

NetworkInfo activeNetwork = cm.getActiveNetworkInfo();

int PrefetchCacheSize = DEFAULT_PREFETCH_CACHE;

switch (activeNetwork.getType()) {
  case (ConnectivityManager.TYPE_WIFI):
    PrefetchCacheSize = MAX_PREFETCH_CACHE; break;
  case (ConnectivityManager.TYPE_MOBILE): {
    switch (tm.getNetworkType()) {
      case (TelephonyManager.NETWORK_TYPE_LTE |
            TelephonyManager.NETWORK_TYPE_HSPAP):
        PrefetchCacheSize *= 4;
        break;
      case (TelephonyManager.NETWORK_TYPE_EDGE |
            TelephonyManager.NETWORK_TYPE_GPRS):
        PrefetchCacheSize /= 2;
        break;
      default: break;
    }
    break;
  }
  default: break;
}
```

But it can change under your feet

- If you switch from WiFi to mobile, or vice versa, your application will find out the hard way.
- IP number of device’s default interface will change, which will break all running connections.
  - When you restart the transfers, you will know the new networking context.
- BUT switching between HSDPA, LTE and 3G can occur transparently: the network provider will usually keep your apparent IP number the same (no guarantees, depends on operator).
- Hence if you are tuning behaviour to suit, you might need to periodically re-check or deal with changes asynchronously if you have long-lived connections.
  - This requires intents, and BroadcastReceiver(), which we will look at later.

Power, Price and Performance

- Power management says “once the radio is on, run it flat out and then turn it off for as long as possible”.
- Dealing with data costs (high in some locations, especially when roaming) says “minimise unnecessary transfers”.
- Making applications responsive says “make sure data is always available to the user ahead of time”.
- Trade-offs will be specific to your application (he said, helpfully).

Networking Example

- Consider a streaming MP3 player
- Minimising battery use would suggest you read ahead and fetch all the data associated with an album / playlist / etc on the assumption the user will play it to the end.
- Minimising network use would suggest you read a few seconds ahead (to avoid skips and pauses) but no more.
- Best UI performance would suggest you read perhaps one track ahead, so the listener can skip songs they don’t like without waiting for it to rebuffer.