Android Best Practices

Dominic Duggan
Stevens Institute of Technology
Based on materials by Reto Meier

1

Latest APIs and Hardware

- SDKs backward compatible
- Detect platform version at runtime
- Parallel activities for backwards compatibility

2

Example: Printing

- Printing API introduced in Kitkat

```
public class Printer implements IPrinter {

public void print(Activity context, WebView view) {
      PrintManager printManager = (PrintManager)
         context.getSystemService(Context.PRINT_SERVICE);

      PrintDocumentAdapter printAdapter =
         webView.createPrintDocumentAdapter();

      // Create a print job with name and adapter instance
      String jobName = context.getString(R.string.app_name);
      printManager.print(jobName, printAdapter,
             new PrintAttributes.Builder().build());
   }
}
```

3

Parallel Activity Pattern

- Printing API introduced in Kitkat

```
public interface IPrinter {
   public void print(Activity context, WebView view);
}

public class PrinterFactory {
   public IPrinter createPrinter() {
      IPrinter printer = null;
         if (Build.VERSION.SDK_INT
                  < Build.VERSION_CODES.KITKAT) {
            printer = new PrinterDummy();
         } else {
            printer = new Printer();
      }
      return printer;
   }
}
```

4

Parallel Activity pattern

- Same layouts and fragments
- Encapsulate functionality within fragments
- Activity for animations and action bar
- Res folders

```
res/layout-por
res/layout-land
res/layout-xlarge-port-v11
res/layout-xlarge-land
```

5

Interfaces for backward compatibility

```
private static boolean newSensorAPIsSupported =
  Build.VERSION.SDK_INT >= Build.VERSION_CODES.CUPCAKE;

boolean gyroExists =
  getPackageManager().hasSystemFeature(
  PackageManager.FEATURE_SENSOR_GYROSCOPE);

IOrientationSensorListener myListener;
if (gyroExists)
  myListener = new GyroOrientationSensorListener();
else if (newSensorAPIsSupported)
  myListener = new AccOrientationSensorListener();
else
  myListener = new AccOldOrientationSensorListener();

myListener.setOrientationChangeListener(myOCListener);
```

6

Track Installation, not Device

- `TelephonyManager.getDeviceId()`?
- MAC address?
- Device swiped and resold?
- `Settings.Secure.ANDROID_ID`
  - Reset on wipe
  - Unreliable pre Android 2.2

7

Detecting Unique Installations

```
private static String uniqueID = null;
private static final String PREF_UNIQUE_ID = "PREF_UNIQUE_ID";

public synchronized static String id(Context context) {
  if (uniqueID == null) {
    SharedPreferences sp = context.getSharedPreferences(
      PREF_UNIQUE_ID, Context.MODE_PRIVATE);
    uniqueID = sp.getString(PREF_UNIQUE_ID, null);
    if (uniqueID == null) {
      uniqueID = UUID.randomUUID().toString();
      Editor editor = sp.edit();
      editor.putString(PREF_UNIQUE_ID, uniqueID);
      editor.commit();
    }
  }
  return uniqueID;
}
```

8

BEST PRACTICES: FRESH DATA

9

Fresh

- Never having to wait
- Always knowing where you are
- Always up to date

- Best time to update
  - Immediately before looked at
  - Battery
  - Connectivity & bandwidth

10

Passive Location Provider

- Location updates if app requests
- ACCESS_FINE_LOCATION permission
- Location.getProvider() for underlying provider

```
String passiveProvider = LocationManager.PASSIVE_PROVIDER;
locationManager.requestLocationUpdates(passiveProvider,
                                        minTime, minDistance,
                                        myLocationListener);
```

11

Intents to monitor location changes

- Pending intent
- Intent key KEY_LOCATION_CHANGED
- Multiple activities/services tracking location

```
final int resultCode = 0;
final String locAction = "com.ioApp.LOCATION_UPDATE_RECEIVED";
int flags = PendingIntent.FLAG_UPDATE_CURRENT;

Intent intent = new Intent(locAction);
PendingIntent pi = PendingIntent.getBroadcast(this,
  resultCode, intent, flags);

locationManager.requestLocationUpdates(provider, minTime, minDistance, pi);
```

12

Intents to monitor location changes

- Pending intent
- Intent key `KEY_LOCATION_CHANGED`
- Multiple activities/services tracking location

```
BroadcastReceiver locReceiver = new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {
    String key = LocationManager.KEY_LOCATION_CHANGED;
    Location location = (Location)intent.getExtras().get(key);
     // [... Do something with the new location ...]
  }
};
IntentFilter locIntentFilter = new IntentFilter(locAction);
registerReceiver(locReceiver, locIntentFilter);
```

13

Passively Detect Location Changes

- Background service for location updates

```
<receiver android:name=".locReceiver" android:enabled="true">
  <intent-filter>
    <action android:name="com.ioApp.LOCATION_UPDATE_RECEIVED"/>
  </intent-filter>
</receiver>
```

14

Check Last Known Location

- Go through ALL providers

```
List<String> providers = lm.getProviders(criteria, false);
for (String provider: providers) {
  Location location = lm.getLastKnownLocation(provider);
  location.getAccuracy();
  location.getTime();
  // Is this the best previously known location?
}
```

15

Monitor inactive providers

- When better option becomes available

```
locationManager.getBestProvider(criteria, false);

public void onProviderEnabled(String provider){
  // Switch providers!
}
```

16

WAKING UP

Worth waking up for?

- Set wake alarm for minimum frequency
- Set non-waking alarm for optimal frequency
- Reset minimum trigger on each update

```
int wake  = AlarmManager.ELAPSED_REALTIME_WAKEUP;
int sleep = AlarmManager.ELAPSED_REALTIME;
long minInt  = AlarmManager.INTERVAL_HALF_DAY;
long bestInt = AlarmManager.INTERVAL_HALF_HOUR;
long trigger = SystemClock.elapsedRealtime() + bestInt;

alarms.setInexactRepeating(wake, trigger, minInt, alarmIntent);
alarms.setInexactRepeating(sleep, trigger, bestInt, alarmIntent);
```

Vary refresh rate based on state

- Update without connectivity?
- More updates on WIFI?
- More updates when charging?
- Suspend updates on low battery?
- More updates when docked?
- Suspend updates in car dock?

19

Connectivity

- Disable receivers if not connected
- Scale up with WIFI, down with 2G etc

```
ConnectivityManager cm = (ConnectivityManager)context.
  getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo activeNW = cm.getActiveNetworkInfo();
boolean isConnected = activeNW.isConnectedOrConnecting();
boolean isMobile =
  activeNW.getType() == ConnectivityManager.TYPE_MOBILE;
```

20

Docking state

- Sticky broadcast intent
- Dock status
- Dock type
  - Compare with app purpose

```
IntentFilter dFilter = new IntentFilter(Intent.ACTION_DOCK_EVENT);
Intent dock = context.registerReceiver(null, dFilter);
int dState = battery.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
boolean isDocked = dockState != Intent.EXTRA_DOCK_STATE_UNDOCKED;
```

21

Monitor Device State

- Update without connectivity?
- More updates on WIFI?
- More updates when charging?
- Suspend updates on low battery?
- More updates when docked?
- Suspend updates in car dock?

22

Monitor State-Change Broadcasts

```
<receiver android:name=".UpdateRateMonitorReceiver">
 <intent-filter>
  <action
  android:name="android.intent.action.ACTION_DOCK_EVENT"/>
  <action
  android:name="android.intent.action.ACTION_BATTERY_LOW"/>
  <action
  android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
  <action
  android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
  <action
  android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
 </intent-filter>
</receiver>
```

23

Toggle Manifest Receivers

- Enable and disable for state changes
- Receiver as passive alarm
- Stop when not needed

```
ComponentName receiver = new ComponentName(this, myReceiver.class);

PackageManager pm = getPackageManager();

pm.setComponentEnabledSetting(receiver,
  PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
  PackageManager.DONT_KILL_APP);
```

24

Services

- Asynchronous
- Die as quickly as possible
- Control restart
  - Time since last success

25

Services

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
  startAsynchServiceWorker();
  return forceRefresh() ? START_STICKY : START_NOT_STICKY;
}

private boolean forceRefresh() {
  int failCount = sp.getInt(FAIL_COUNT, 0);
  long lastSuccess = sp.getLong(LAST_SUCCESS, 0);
  Editor editor = sp.edit();
  editor.putInt("FAIL_COUNT", failCount+1);
  editor.commit();

  return ((System.currentTimeMillis()-lastSuccess > maxSuccessLatency) ||
          (failCount > maxFailCount));
```

26

EFFICIENCY

27

Invisible Internet

- Work offline
- Be consistent but creative
- Know that less is more
- Understand not all devices are the same

28

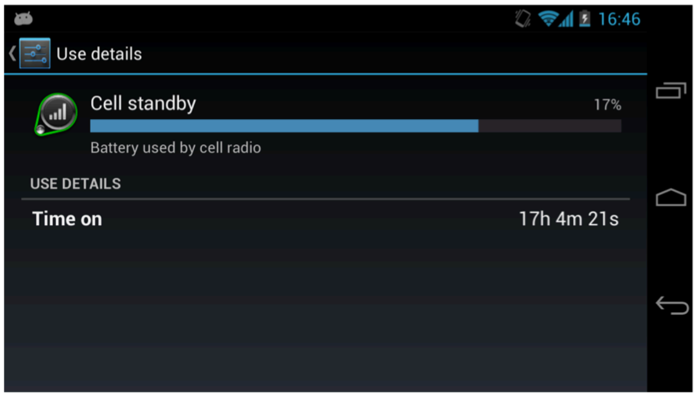
Work Offline

```
if (!isConnected) {
  alarms.cancel(retryQueuedCheckinsPendingIntent);
  pm.setComponentEnabledSetting(connectivityReceiver,
       PackageManager.COMPONENT_ENABLED_STATE_ENABLED, PackageManager.DONT_KILL_APP);
  addToQueue(timeStamp, reference, id);
} else {
  if (!checkin(timeStamp, reference, id))
    addToQueue(timeStamp, reference, id);
  // Retry each of the queued checkins
  // Delete the queued checkins that were successful.
  // If there are still queued checkins then set a non-waking alarm to retry them.
  if (queuedCheckins.getCount() > 0) {
    long trigger = SystemClock.elapsedRealtime() + RETRY_INTERVAL;
    alarms.set(AlarmManager.ELAPSED_REALTIME, trigger, retryQueuedCheckinsPendingIntent);
  }
}
```

MyService.java

29

Airplane Mode



16:46

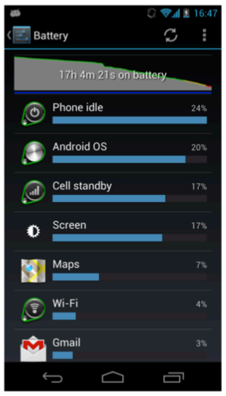
Use details

Cell standby 17%

Battery used by cell radio

USE DETAILS

Time on 17h 4m 21s



16:47

Battery

17h 4m 21s on battery

Phone idle 24%

Android OS 20%

Cell standby 17%

Screen 17%

Maps 7%

Wi-Fi 4%

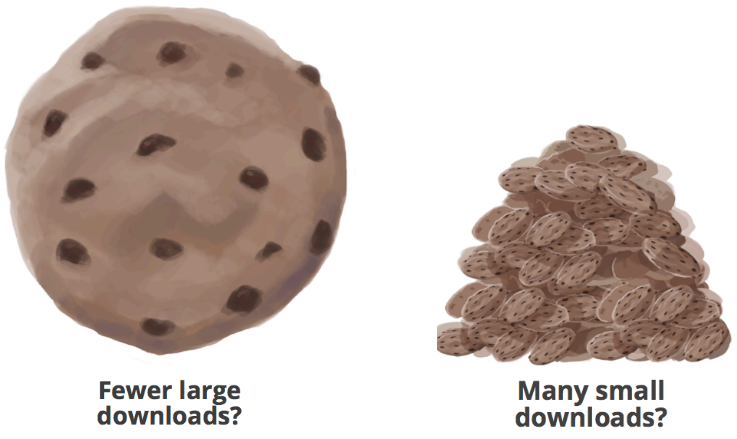
Gmail 3%

30

Use Mobile Radio Less

- Smaller payloads
- Transfer less often
- Cache your results

Big vs Little Cookie

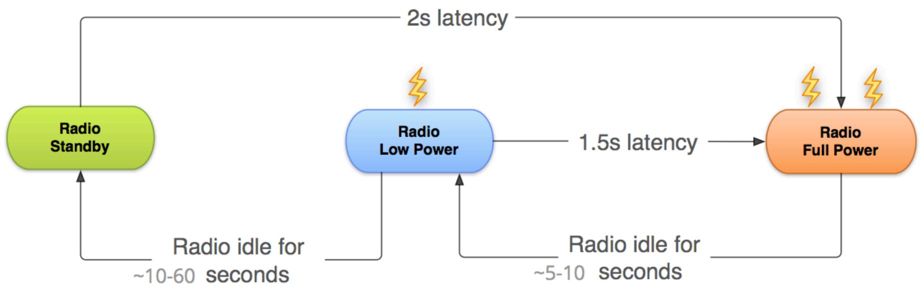


False Economy of the Little Cookie

- Transfer less data across the network
- Store and process less data on the device
- Use less memory / storage / bandwidth

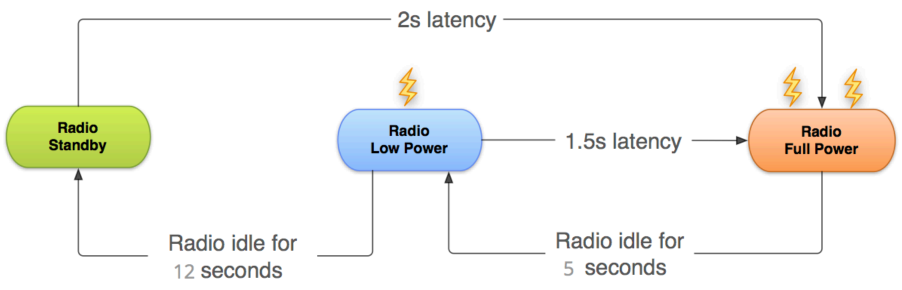
33

Mobile Radio State Machine



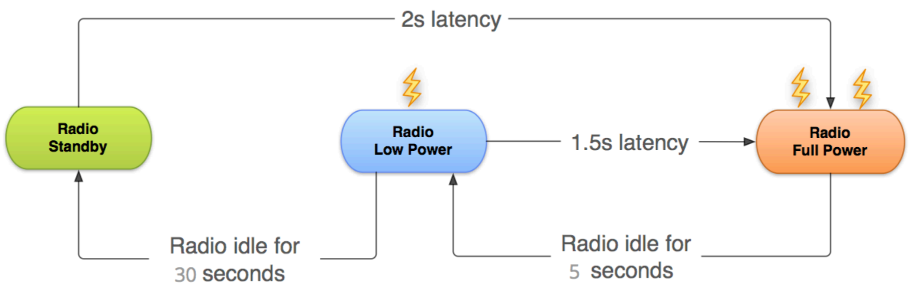
34

AT&T 3G



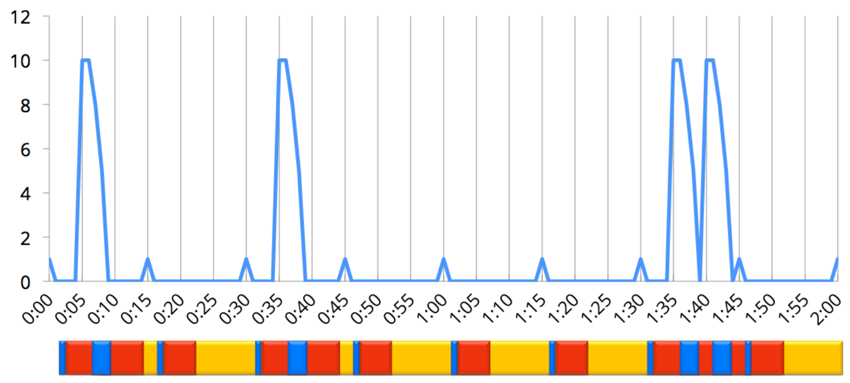
35

Vodafone



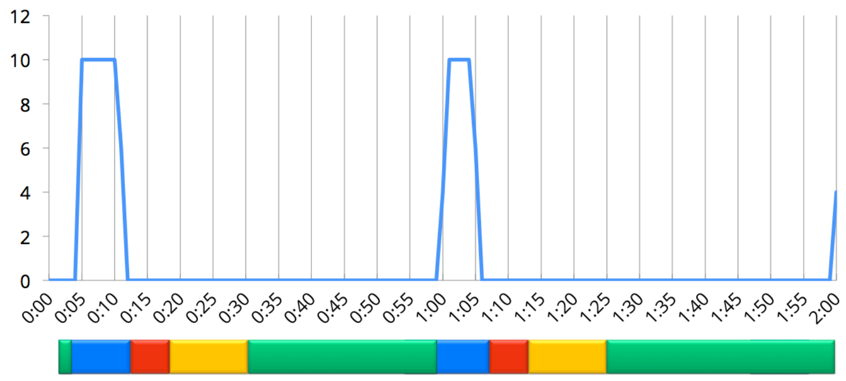
36

Fragmented Network Traffic



37

Defragmented Network Traffic

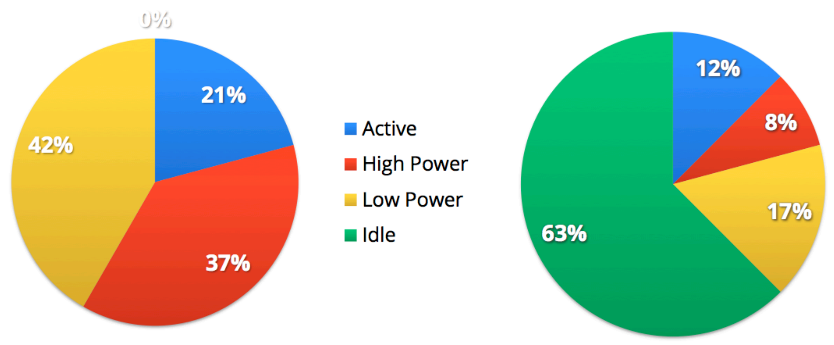


38

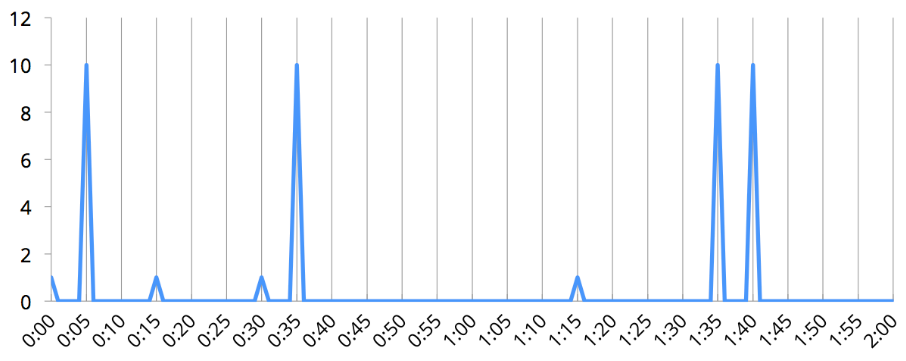
Defragmenting Network Traffic

- Prefetching
- Batching, bundling, pre-empting
- Reducing number of connections

Power Profiles

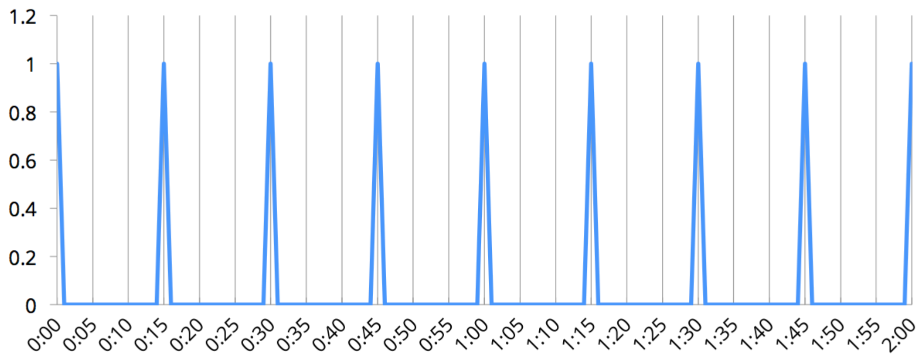


Short Spikes

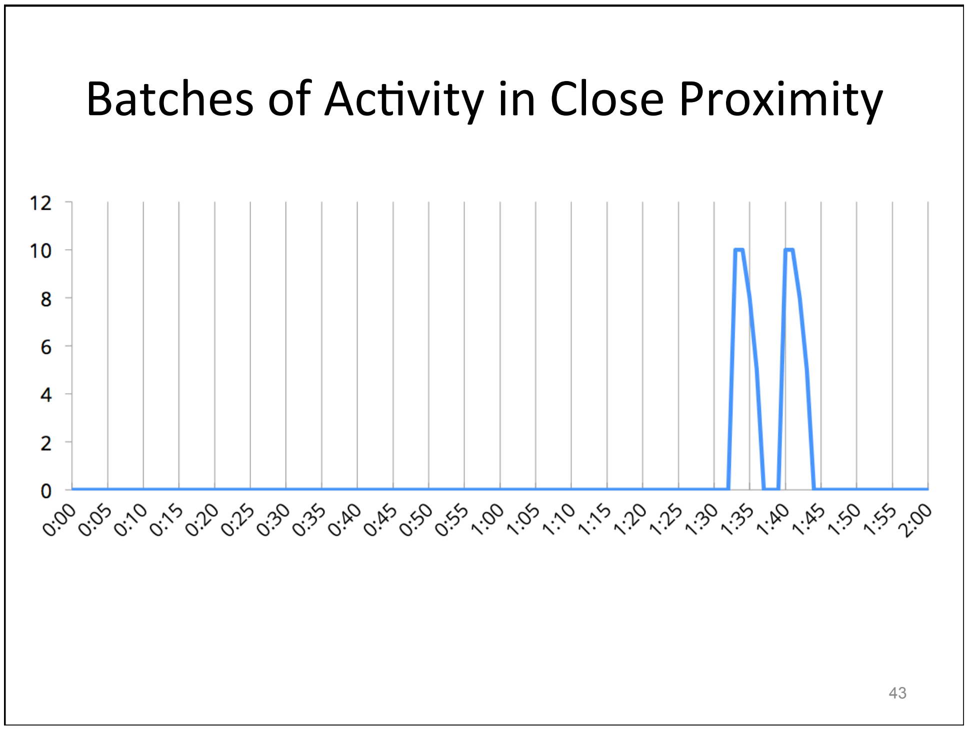


41

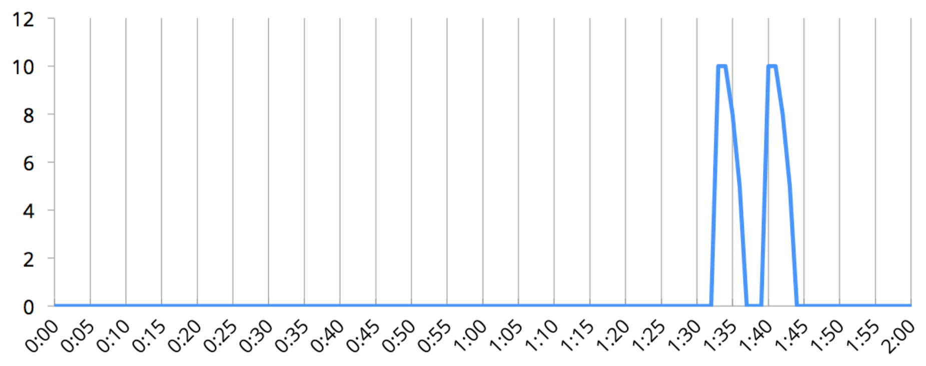
Regular / Periodic Transfers



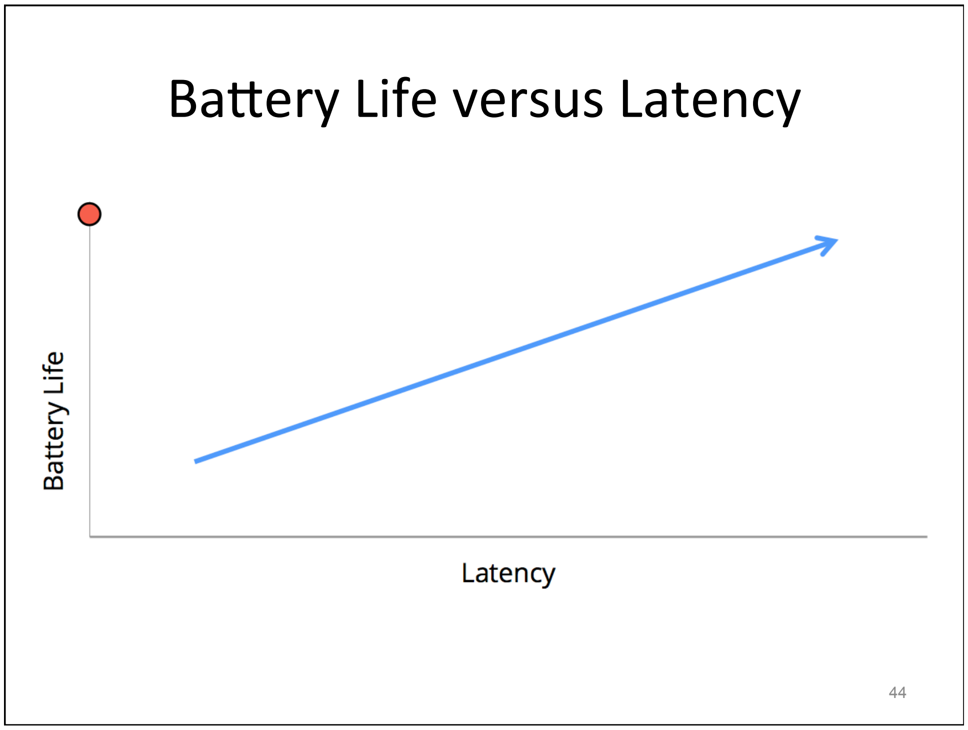
42



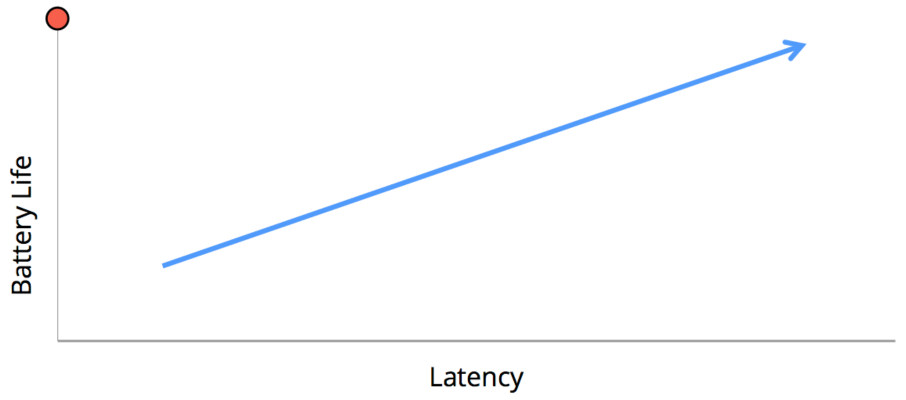
Batches of Activity in Close Proximity



43



Battery Life versus Latency



44

Prefetching

- Forecast need
- 2-5 minutes of app usage
- 1-5MB data (3G)

Prefetching

```
int prefetchCacheSize = DEFAULT_PREFETCH_CACHE;
switch (activeNetwork.getType()) {

  case ConnectivityManager.TYPE_WIFI:
    prefetchCacheSize = MAX_PREFETCH_CACHE; break;

  case ConnectivityManager.TYPE_MOBILE): {
    switch (telephonyManager.getNetworkType()) {
      case TelephonyManager.NETWORK_TYPE_LTE:
      case TelephonyManager.NETWORK_TYPE_HSPAP:
        prefetchCacheSize *= 4; break;
      case TelephonyManager.NETWORK_TYPE_EDGE:
      case TelephonyManager.NETWORK_TYPE_GPRS:
        prefetchCacheSize /= 2; break;
      default: break;
    } break;
  default: break;
  }
}
```

Batching and Pre-empting

- Transfer as much as possible during each session
- Minimize # of sessions
- Delay time-insensitive transfers
- Pre-empt scheduled transfers

47

Batch Queue for Periodic Transfers

```
private Queue<MyPeriodicTransfer> updateQueue;

public synchronized void
  enqueuePeriodicTransfer(MyPeriodicTransfer periodicTransfer) {
  updateQueue.add(periodicTransfer);
}
public void executeBatchedPeriodicTransfers() {
  // Execute the batched periodic update queue.
  executeBatchedPeriodicTransfersOnly();
  // Preempt scheduled update
  executeNextPrefetch();
}
private synchronized void executeBatchedPeriodicTransfersOnly() {
  // TODO Bundle received updates / requests into single transfer.
  updateQueue.clear();
  // TODO Upload / download the periodic transfer
}
```

48

Batch Queue for Periodic Transfers

```
public void executeOnDemandDownload
    (DownloadDetails details) {
  // TODO Execute an on demand download.
  executeNextPrefetch();
}

public void executeNextPrefetch() {
  // TODO Execute the next planned prefetch.

  // Execute the batched periodic update queue
  executeBatchedPeriodicTransfersOnly();
}
```

49

REGULAR UPDATES

50

Inexact Repeating Alarms

```
int alarmType = AlarmManager.ELAPSED_REALTIME;
long interval = AlarmManager.INTERVAL_HOUR;
long start = SystemClock.elapsedRealtime() +
             interval;

alarmManager.setInexactRepeating(alarmType,
start, interval, pi);
```

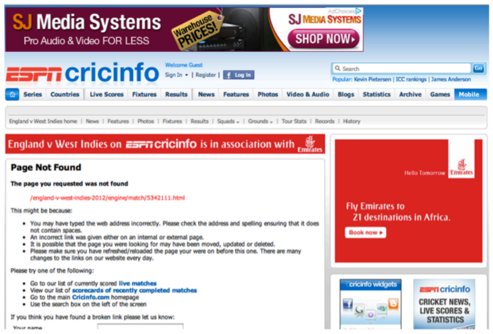
51

Inactivity Back-Off

```
boolean appUsed = prefs.getBoolean(PREF_APPUSED, false);
long updateInterval =
  prefs.getLong(PREF_INTERVAL, DEFAULT_REFRESH_INTERVAL);

if (!appUsed)
  if ((updateInterval *= 2) > MAX_REFRESH_INTERVAL)
    updateInterval = MAX_REFRESH_INTERVAL;

reschedulePeriodicUpdates(updateInterval);
                      // Save interval & reschedule alarm.
executeUpdate();      // Execute data transfer.
```

52

Failure Back-Off


SJ Media Systems Pro Audio & Video FOR LESS

Page Not Found

The page you requested was not found

/england-v-west-indies-2012/engine/match/5342111.html

This might be because:

- You may have typed the web address incorrectly. Please check the address and spelling ensuring that it does not contain spaces.
- An incorrect link was given either on an internal or external page.
- It is possible that the page you were looking for may have been moved, updated or deleted.
- Please make sure you have refreshed/reloaded the page your were on before this one. There are many changes to the links on our website every day.

Please try one of the following:

- Go to our list of currently scored **live matches**
- View our list of **scorecards of recently completed matches**
- Go to the main **Cricinfo.com** homepage
- Use the search box on the left of the screen

If you think you have found a broken link please let us know:

Your name

```
404 = 131kb
      *60
      *24
      *7
    = 1320480kb
    = 1.26 GIGABYTES
```

53

Failure Back-Off

Google

404. That's an error.

The requested URL /1231 was not found on this server. That's all we know.

```
404 = 4kb
      *60
      *24
      *7
    = 40320kb
    = 39mb
```

54

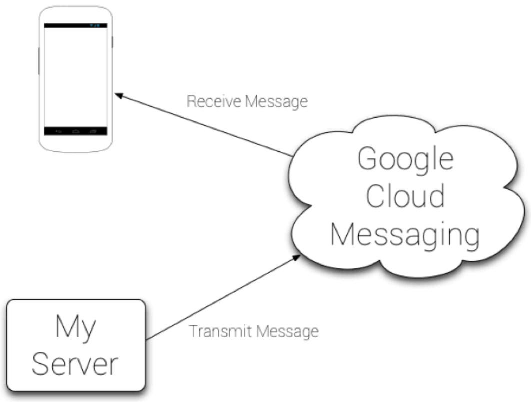
Failure Back-Off

```
private void retryIn(long interval) {
  Thread.sleep(interval);
  boolean success = attemptTransfer();

  if (!success) {
    retryIn(
      interval*2 < MAX_RETRY_INTERVAL ?
      interval*2 : MAX_RETRY_INTERVAL);
  }
}
```

55

Google Cloud Messaging (GCM)

- Avoid polling



56

Reduce Payloads

- Filter on the server
- Rescale images on the server
- Cache everything!

```
// Non-sensitive data
Context.getExternalCacheDir();

// Sandboxed application data
Context.getCacheDir();
```

57

Don’t Download again until necessary

```
long expires = httpURLConnection
   .getHeaderFieldDate("Expires", currentTime);

long lastModified = httpURLConnection
   .getHeaderFieldDate("Last-Modified", currentTime);

// Don’t refresh until at least the expiry time
setDataExpirationDate(expires);

if (lastModified > lastUpdateTime) {
   // Parse update
}
```

MyService.java

58

Don’t Download again until necessary

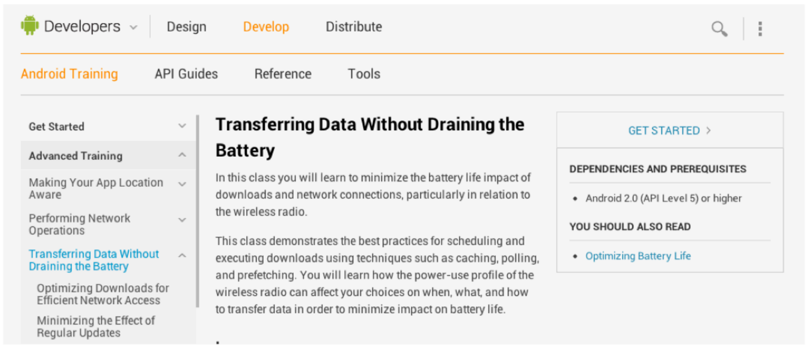
```
private void enableHttpResponseCache() {
  try {
  long httpCacheSize = 10 * 1024 * 1024; // 10 MiB
  File httpCacheDir = new File(getCacheDir(), "http");

  Class.forName(“android.net.http.HttpResponseCache")
    .getMethod(“install", File.class, long.class)
    .invoke(null, httpCacheDir, httpCacheSize);

  } catch (Exception httpResponseCacheNotAvailable) {
    Log.d(TAG, "HTTP response cache is unavailable.");
  }
}
```

MyService.java

59

Reading

http://developer.android.com/training/efficient-downloads


Developers
Design
Develop
Distribute
Android Training
API Guides
Reference
Tools
Get Started
Advanced Training
Making Your App Location Aware
Performing Network Operations
Transferring Data Without Draining the Battery
Optimizing Downloads for Efficient Network Access
Minimizing the Effect of Regular Updates
GET STARTED

Transferring Data Without Draining the Battery

In this class you will learn to minimize the battery life impact of downloads and network connections, particularly in relation to the wireless radio.

This class demonstrates the best practices for scheduling and executing downloads using techniques such as caching, polling, and prefetching. You will learn how the power-use profile of the wireless radio can affect your choices on when, what, and how to transfer data in order to minimize impact on battery life.

DEPENDENCIES AND PREREQUISITES

- Android 2.0 (API Level 5) or higher

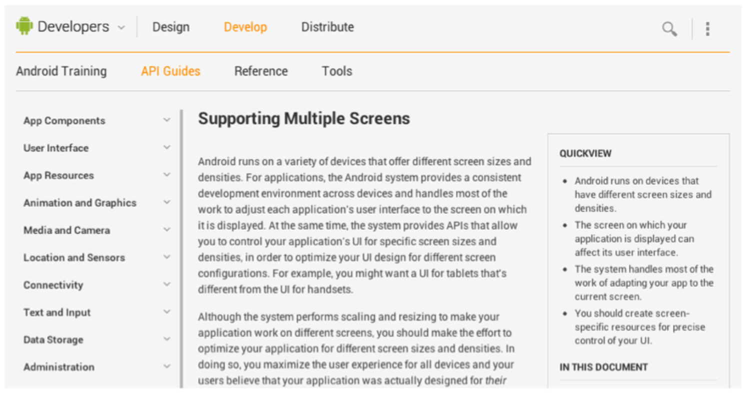
YOU SHOULD ALSO READ

- Optimizing Battery Life

60

Reading

http://developer.android.com/guide/practices/screens_support.html



Supporting Multiple Screens

Android runs on a variety of devices that offer different screen sizes and densities. For applications, the Android system provides a consistent development environment across devices and handles most of the work to adjust each application's user interface to the screen on which it is displayed. At the same time, the system provides APIs that allow you to control your application's UI for specific screen sizes and densities, in order to optimize your UI design for different screen configurations. For example, you might want a UI for tablets that's different from the UI for handsets.

Although the system performs scaling and resizing to make your application work on different screens, you should make the effort to optimize your application for different screen sizes and densities. In doing so, you maximize the user experience for all devices and your users believe that your application was actually designed for *their*

QUICKVIEW

- Android runs on devices that have different screen sizes and densities.
- The screen on which your application is displayed can affect its user interface.
- The system handles most of the work of adapting your app to the current screen.
- You should create screen-specific resources for precise control of your UI.

IN THIS DOCUMENT

61