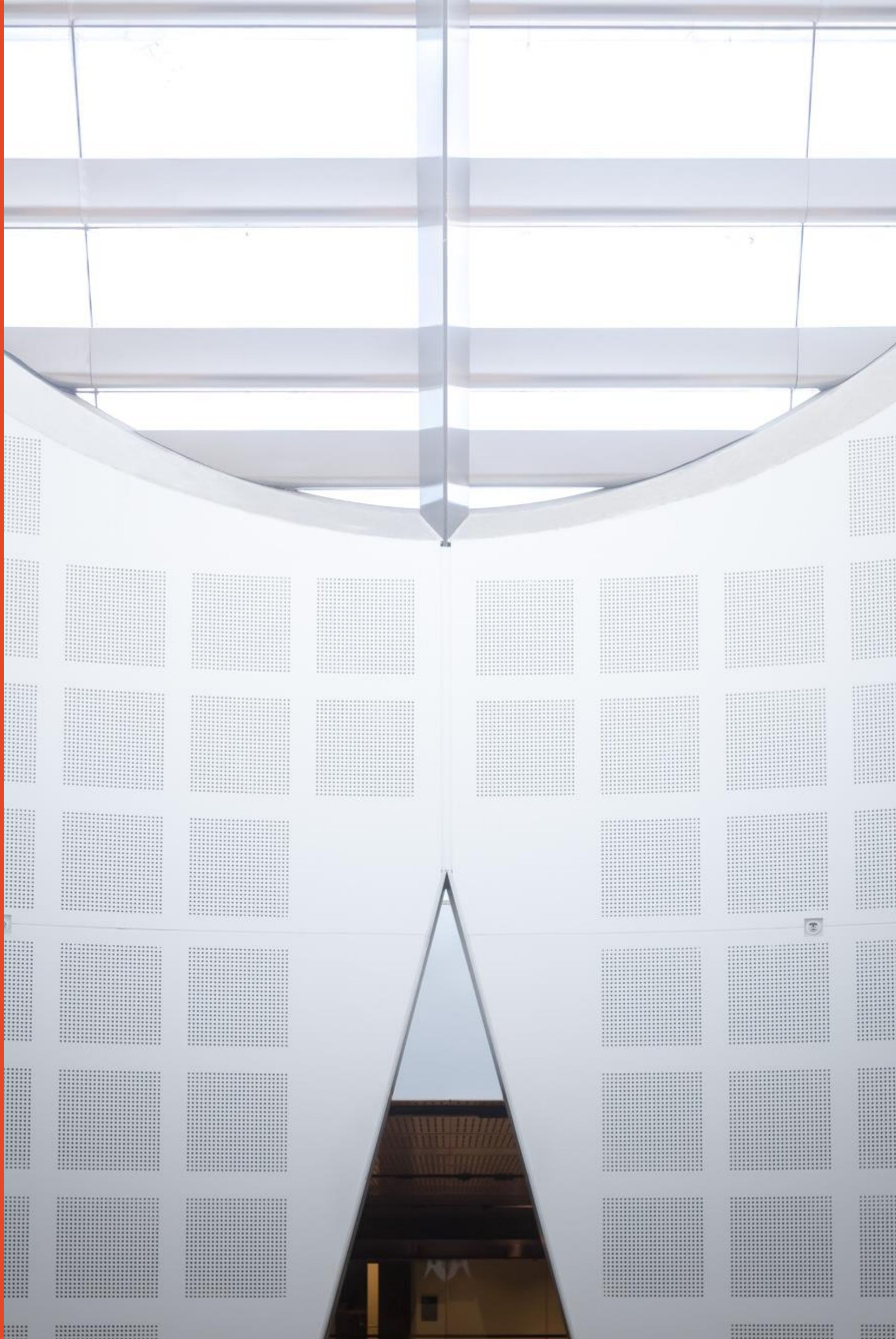
Mobile Computing
COMP5216/COMP4216

Week 08

Semester 2, 2023

Dr. Thilina Halloluwa

School of Computer Science



Announcements

- Feedback on helpdesk
  - Did you attend the helpdesk?
  - Rate the effectiveness of the helpdesk.
  - Should we organize another session?
  - Would you attend if we organize a session for project?

Outline

- Recap of Week 7

- Computation
  - Managing Computation tasks
  - Monitoring the load
  - Cloud computing support

- Energy Management
  - Energy consumption
  - Best practices for energy management
  - Platform supported energy management

Muddy Card – W6

– “I am having difficulty completing Tutorial 6: Media Access. I don't have access to an Android device and thus I have to use the emulator. I followed all of the lab instructions, and used the provided source code zip file. I confirmed that the images and videos are indeed being saved. However, I cannot get images and videos to appear in the gallery”

Recap of Week 7

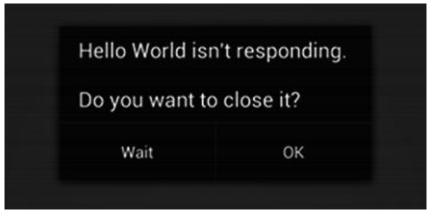
1. Using Bluetooth for local data transferring is always energy efficient than WiFi. True or False ?

2. It is better to use one compression algorithm/library to minimize complexity and loading time for all content within an app. True or False>?

3. You started a new job at a startup who owns one of the public transport app in Sydney that leverages Transport NSW public APIs to access real-time updates for the status of public transport network. Customers have complained that this app is unusually at the top of bandwidth usage and battery usage lists. As a student who followed this course, what are the checks/verifications you suggest to perform ?

Computing Challenge

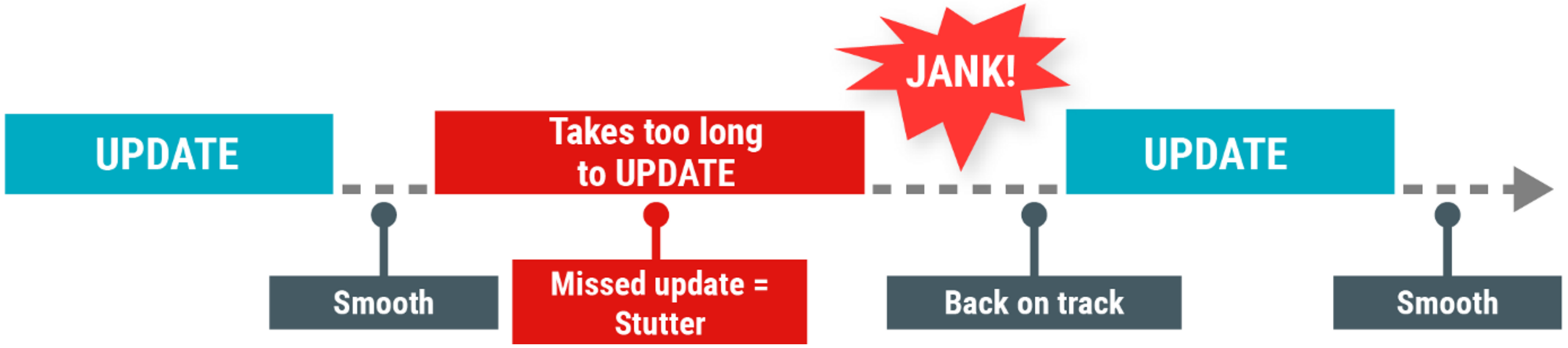
- Slow Rendering
  - Hardware updates screen every 16 milliseconds
  - UI thread has 16ms to do all its work
  - If it takes too long, app stutters or hangs



Hello World isn't responding.

Do you want to close it?

Wait OK



Why 60 fps?

Computing Challenge

- What are long running tasks ?
  - Downloading/Uploading files
  - Image processing, e.g. object detection
  - Loading data
  - Complex calculations

- How to check whether your app does well ?

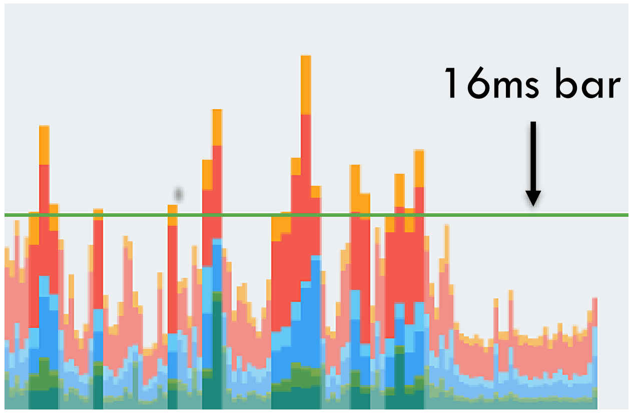
Checking your frame rate

- What are long running tasks ?
  - Downloading/Uploading files
  - Image processing, e.g. object detection
  - Loading data
  - Complex calculations

- How to check whether your app does well ?
  - Settings > Developer options > Monitoring section > Profile GPU rendering > On screen as bars
  - How to find Developer options ?

Checking your frame rate

- Settings > Developer options > Monitoring section > Profile GPU rendering > On screen as bars
- How to find Developer options ?
  - Hidden by default.



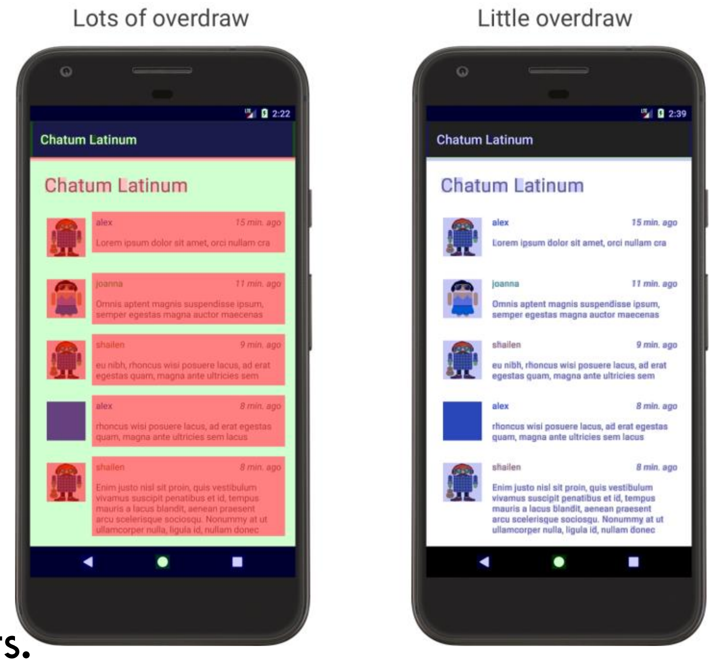
- One bar represents one frame of rendering
- Taller the bar, the longer it takes to render
- The horizontal green line represents 16 milliseconds.
- **Each frame needs to stay below this line → 60 frames/s**

Misc	Input	Anim.	Measure	Draw	Upload	Issue	Swap

Visualize GPU overdraw

– Settings > Developer options > Hardware accelerated rendering> Debug GPU Overdraw > overdraw areas

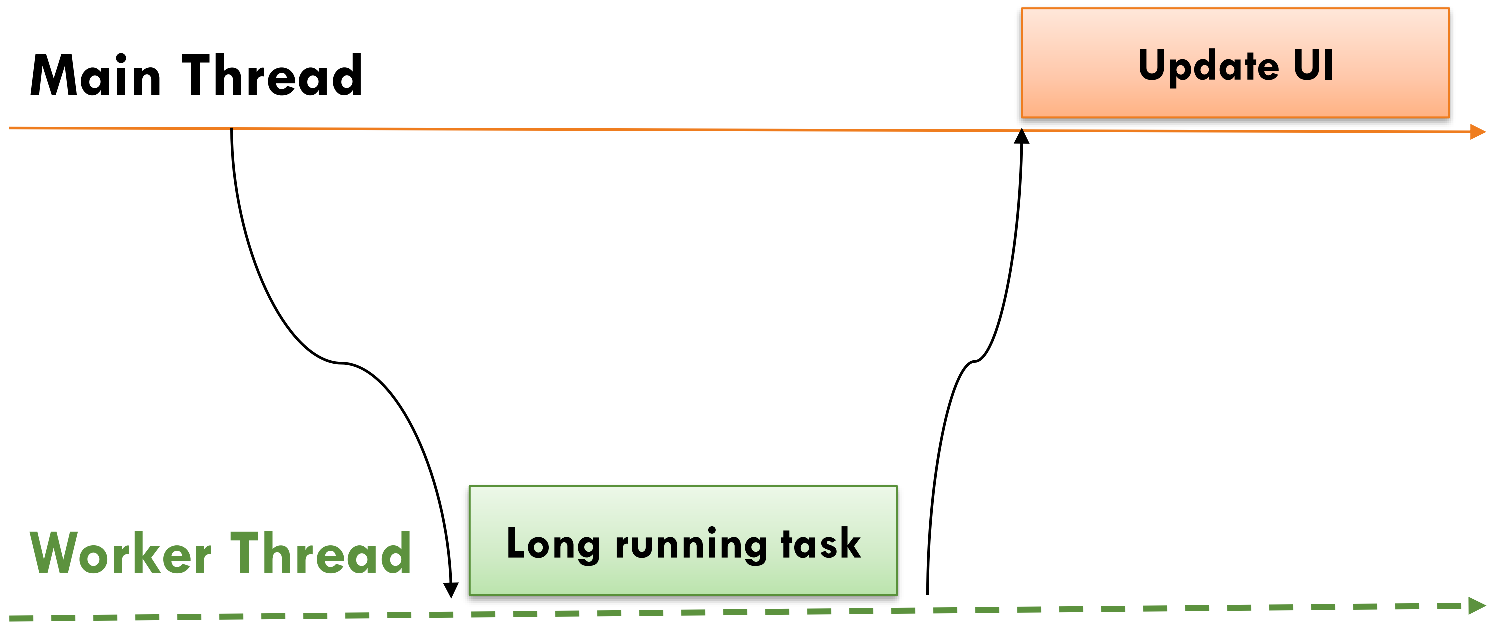
- **True color:** No overdraw
- 🟦 **Blue:** Overdrawn 1 time
- 🟩 **Green:** Overdrawn 2 times
- 🟥 **Pink:** Overdrawn 3 times
- 🟥 **Red:** Overdrawn 4 or more times



– How to reduce overdraw ?
  – Removing unwanted backgrounds in layouts.
  – Flattening the view hierarchy.
  – Reducing transparency.
  – https://developer.android.com/topic/performance/rendering/overdraw

Best Practice Computing

1. Complete tasks in less than 16ms
2. Move non-UI tasks to background thread



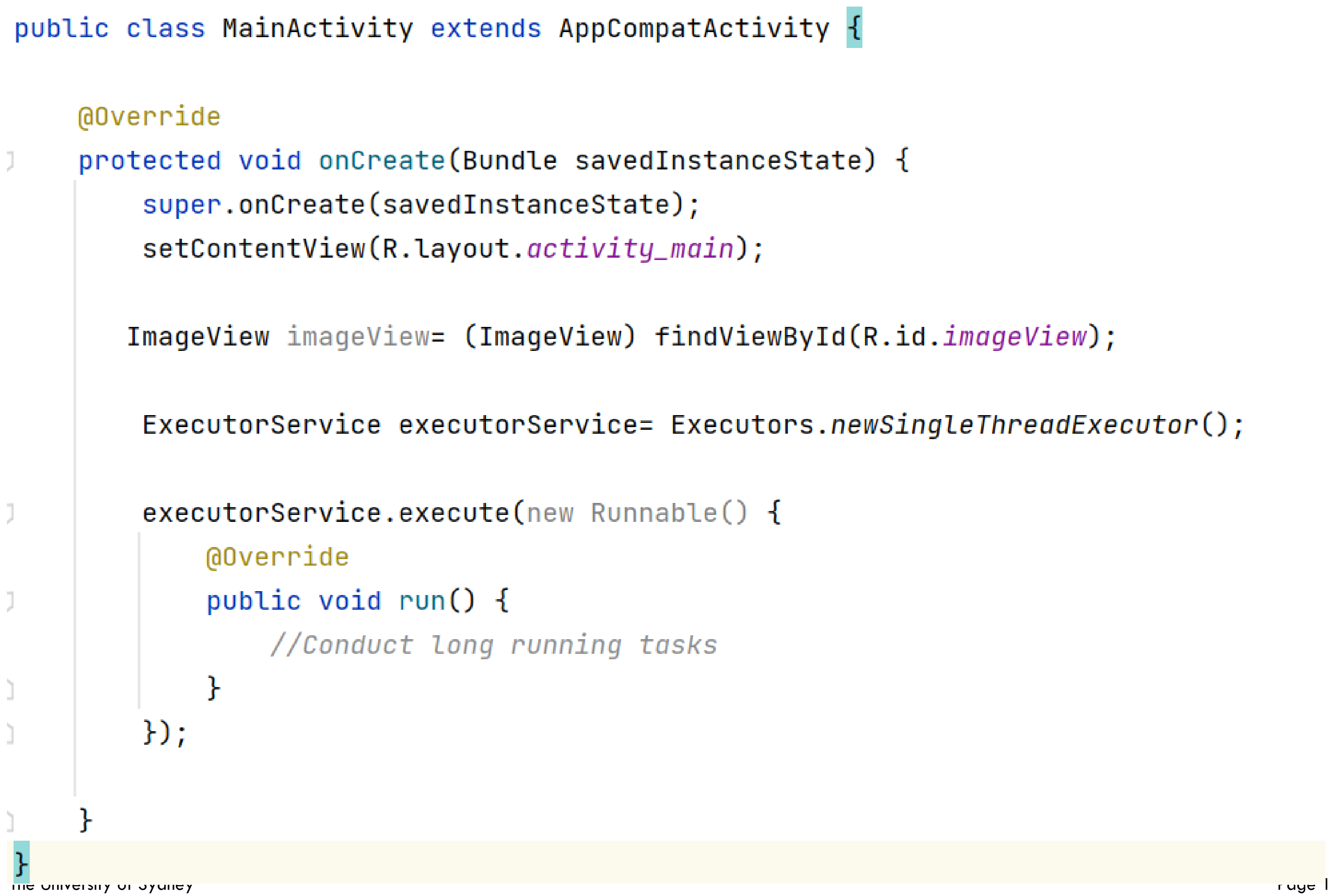
– In Android
  – AsyncTask, Background Services (Executer Service)

Best Practice Computing- Threads

- When an application is launched, the system creates a thread of execution for the application, called the *main thread*.
  - it is in charge of dispatching events to the appropriate user interface widgets, including drawing events.
- It is also almost always the thread in which your application interacts with components from the Android UI toolkit's android.widget and android.view packages. For this reason, the main thread is sometimes called the *UI thread*.
- Performing long operations in the UI thread, such as network access or database queries, blocks the whole UI.
  - If you have operations to perform that aren't instantaneous, make sure to do them in separate *background* or *worker* threads.

https://developer.android.com/reference/java/util/concurrent/Executors

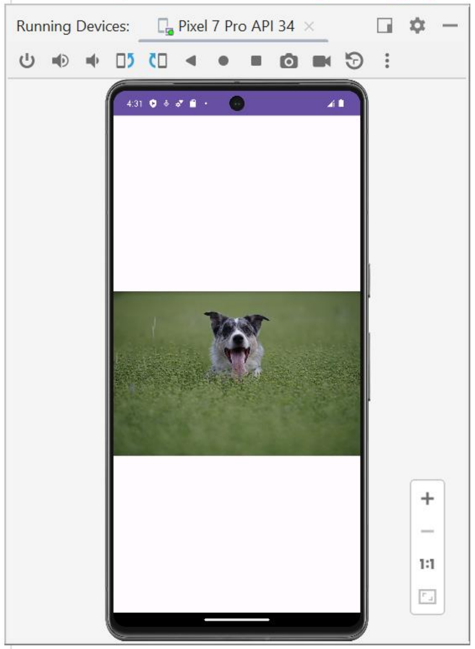
Example



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

       ImageView imageView= (ImageView) findViewById(R.id.imageView);

        ExecutorService executorService= Executors.newSingleThreadExecutor();

        executorService.execute(new Runnable() {
            @Override
            public void run() {
                //Conduct long running tasks
            }
        });

    }
}
```

```java
public class NetworkTask {
    no usages
    public  static Bitmap loadImage(String url){
        Bitmap bitmap=null;
        try{
            InputStream inputStream=new URL(url).openStream();
            bitmap= BitmapFactory.decodeStream(inputStream);

        }catch (Exception ex){
            ex.printStackTrace();
        }
        return bitmap;
    }
}
```

```java
public class MainActivity extends AppCompatActivity {

    1 usage
    String mUrl= "https://images.pexels.com/photos/17819194/pexels-photo-17819194/free-photo-of-adorable-white-poodle-with-its-tongue-out.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=2";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

       ImageView imageView= (ImageView) findViewById(R.id.imageView3);


        ExecutorService executorService= Executors.newSingleThreadExecutor();
        Handler handler=new Handler(Looper.getMainLooper());
        executorService.execute(new Runnable() {
            @Override
            public void run() {
              Bitmap bitmap= NetworkTask.loadImage(mUrl);
              runOnUiThread(new Runnable() {
                  @Override
                  public void run() {
                      if(bitmap!=null)
                          imageView.setImageBitmap(bitmap);
                  }
              });
            }
        });

    }
}
```



Best Practice Computing

1. Complete tasks in less than 16ms
2. Move non-UI tasks to background thread
3. Offload to a location with enough resources

- **Mobile Computing + Cloud Computing**
  - Provides mobile application developers a way to connect their application to backend cloud storage and processing

What is Mobile Cloud Computing ?

- **Mobile Computing + Cloud Computing**
  - Provides mobile application developers a way to connect their application to backend cloud storage and processing

- **"Anything-as-a-Service"**
  - Software-as-a-Service
  - Platform-as-a-Service
  - Infrastructure-as-a-Service
  - Mobile Backend as-a-Service

- Multiple monetization models
  - Pay per use
  - Subscriptions
- Pros and Cons??

https://www.back4app.com

Mobile Cloud Computing

- **Why ?**
  - Limited resources on mobile devices
    - Battery, Computation, Network, etc.
  - Abstract away complexities of app development
    - E.g. Google Play Services
  - Minimize launching and managing own infrastructure
    - Enable enterprises to treat IT as a **utility** than a **capital** expenditure
  - Focus more on front-end development instead of backend functions
  - Integration of multiple developers, apps, services
  - To enable data sharing
  - For permanent storage, backup
  - Easy app analytics
  - Security

- **Examples from current popular apps ?**

Examples

- Apple Siri
  - Speech recognition → Too complex for a mobile device

- Dropbox, Google Drive, Apple iCloud
  - Unlimited storage → Not enough storage on mobile devices
- Single Sign-on Authentication
  - Focus more on the front-end development

- Social Networking
  - Data storage and sharing, Push notifications

MBaaS Providers

- Not a comprehensive list
  - Literary every company has a MBaaS Cloud Service

Google Cloud Firebase amazon web services™

Microsoft Azure

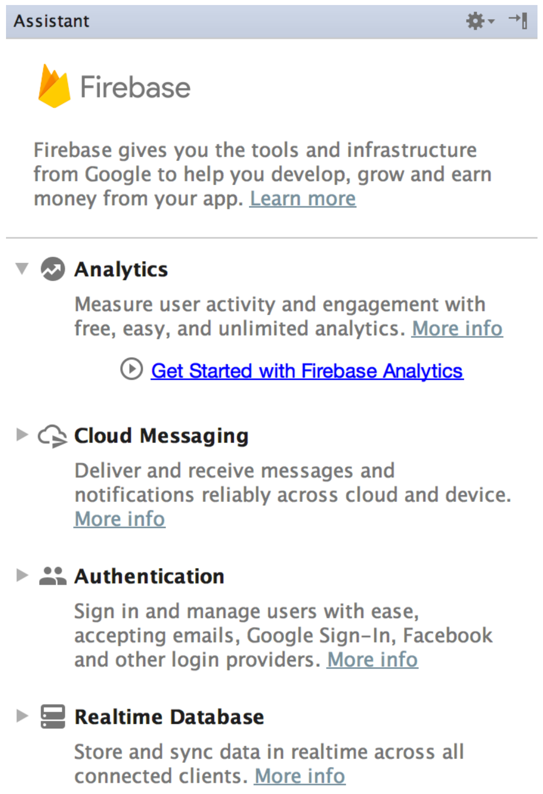
kinvey Sencha anypresence Kii Back4App

Firebase

- Google MBaaS Solution
  - Not only for Android
  - Most Features supports

- Offer a number of features
  - Build better apps
  - Improve app quality
  - Grow your business

- Click **Tools >Firebase** to open the **Assistant** window



Assistant

Firebase

Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. Learn more

Analytics

Measure user activity and engagement with free, easy, and unlimited analytics. More info

Get Started with Firebase Analytics

Cloud Messaging

Deliver and receive messages and notifications reliably across cloud and device. More info

Authentication

Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. More info

Realtime Database

Store and sync data in realtime across all connected clients. More info

Firebase

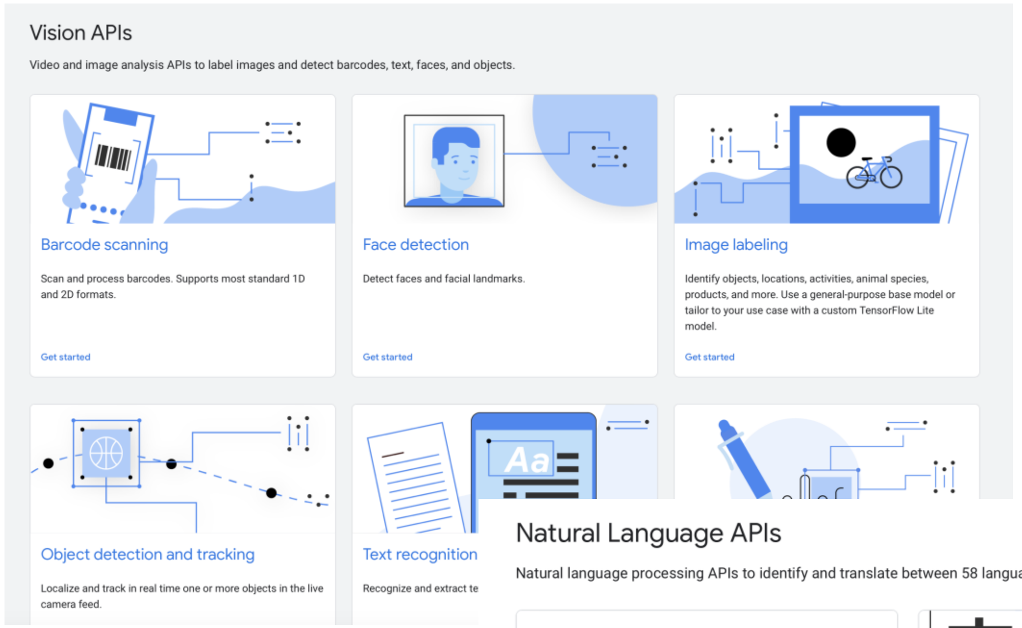
- Firebase ML - https://firebase.google.com/docs/ml

  - Still a beta version
  - APIs that work either in the in the cloud or on the device
  - Convenient APIs to use deep learning capabilities
  - For both iOS and Android
  - Text recognition
  - Face detection
  - Barcode reading
  - Label images
  - Landmark recognition

- Firebase Cloud Messaging - https://firebase.google.com/docs/cloud-messaging/
  - FCM provides a **single, persistent connection** to the cloud
  - All apps needing real-time messaging can share this connection

ML Tool kit

- ML Tool Kit – https://developers.google.com/ml-kit
  - Released on Jun 2020



Vision APIs

Video and image analysis APIs to label images and detect barcodes, text, faces, and objects.

Barcode scanning

Scan and process barcodes. Supports most standard 1D and 2D formats.

Get started

Face detection

Detect faces and facial landmarks.

Get started

Image labeling

Identify objects, locations, activities, animal species, products, and more. Use a general-purpose base model or tailor to your use case with a custom TensorFlow Lite model.

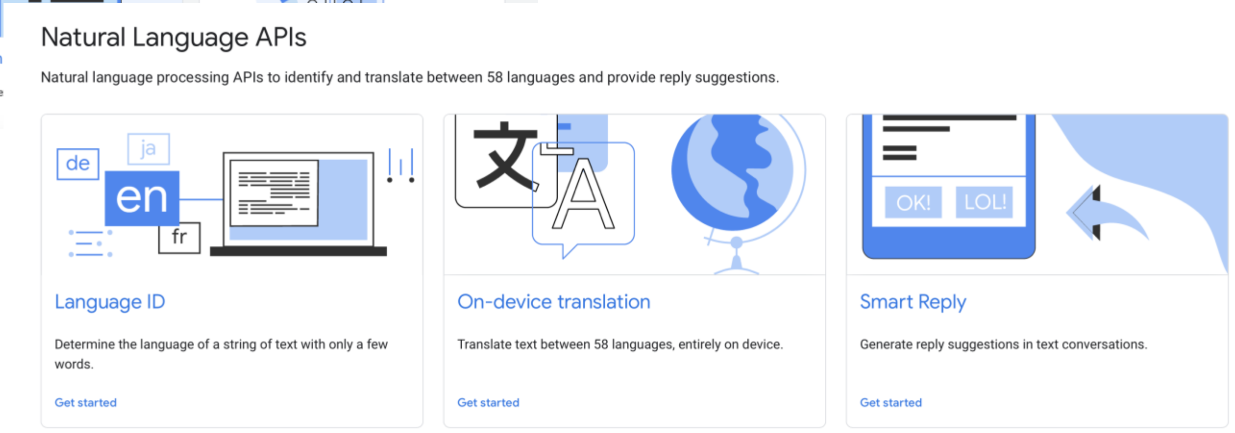
Get started

Object detection and tracking

Localize and track in real time one or more objects in the live camera feed.

Text recognition

Recognize and extract te



Natural Language APIs

Natural language processing APIs to identify and translate between 58 languages and provide reply suggestions.

Language ID

Determine the language of a string of text with only a few words.

Get started

On-device translation

Translate text between 58 languages, entirely on device.

Get started

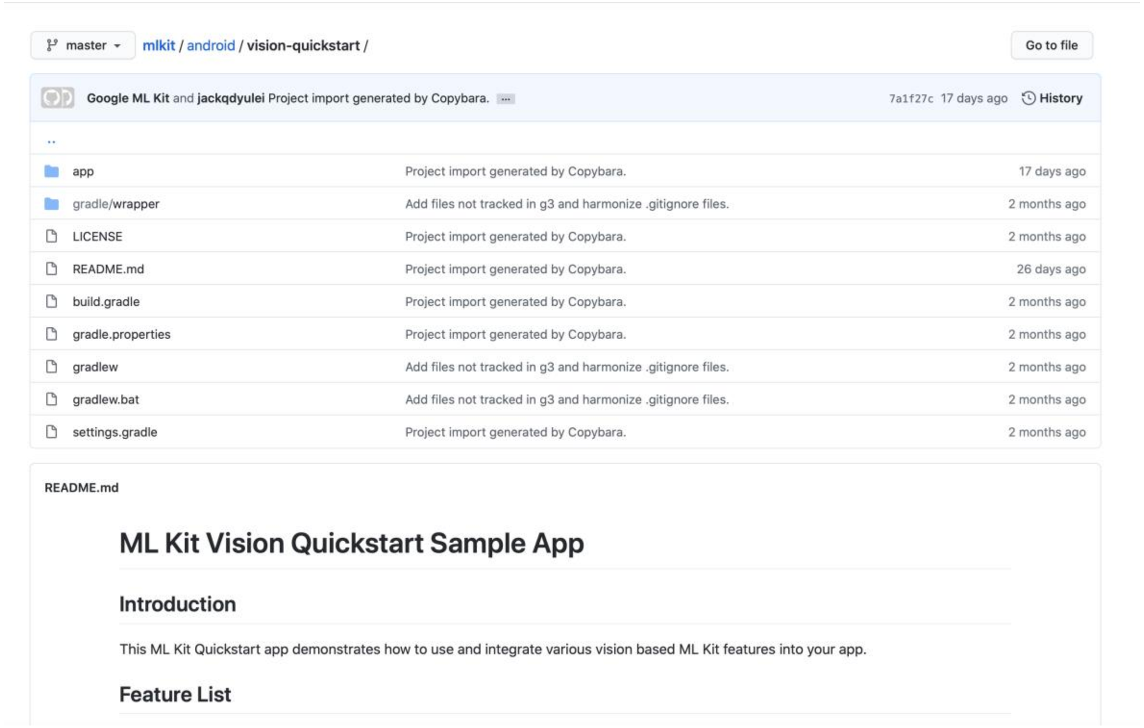
Smart Reply

Generate reply suggestions in text conversations.

Get started

ML Tool kit

- ML Tool Kit – https://developers.google.com/ml-kit
  - Released on Jun 2020
- Samples: https://github.com/googlesamples/mlkit/tree/master/android/vision-quickstart


master mlkit / android / vision-quickstart /   Go to file

Google ML Kit and jackqdyulei Project import generated by Copybara. ...   7a1f27c 17 days ago   History

..		
app	Project import generated by Copybara.	17 days ago
gradle/wrapper	Add files not tracked in g3 and harmonize .gitignore files.	2 months ago
LICENSE	Project import generated by Copybara.	2 months ago
README.md	Project import generated by Copybara.	26 days ago
build.gradle	Project import generated by Copybara.	2 months ago
gradle.properties	Project import generated by Copybara.	2 months ago
gradlew	Add files not tracked in g3 and harmonize .gitignore files.	2 months ago
gradlew.bat	Add files not tracked in g3 and harmonize .gitignore files.	2 months ago
settings.gradle	Project import generated by Copybara.	2 months ago

README.md

ML Kit Vision Quickstart Sample App

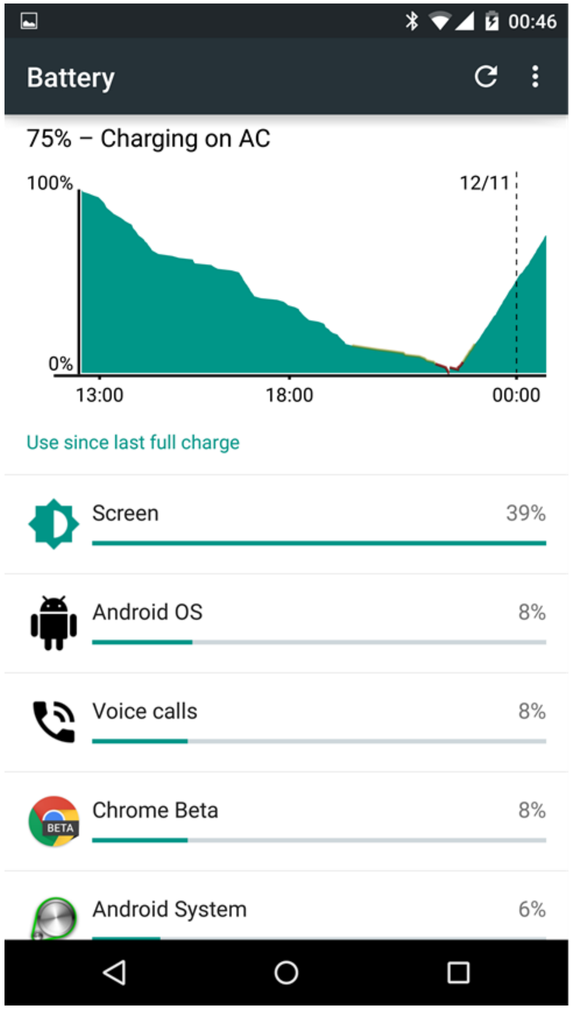
Introduction

This ML Kit Quickstart app demonstrates how to use and integrate various vision based ML Kit features into your app.

Feature List

Energy Management

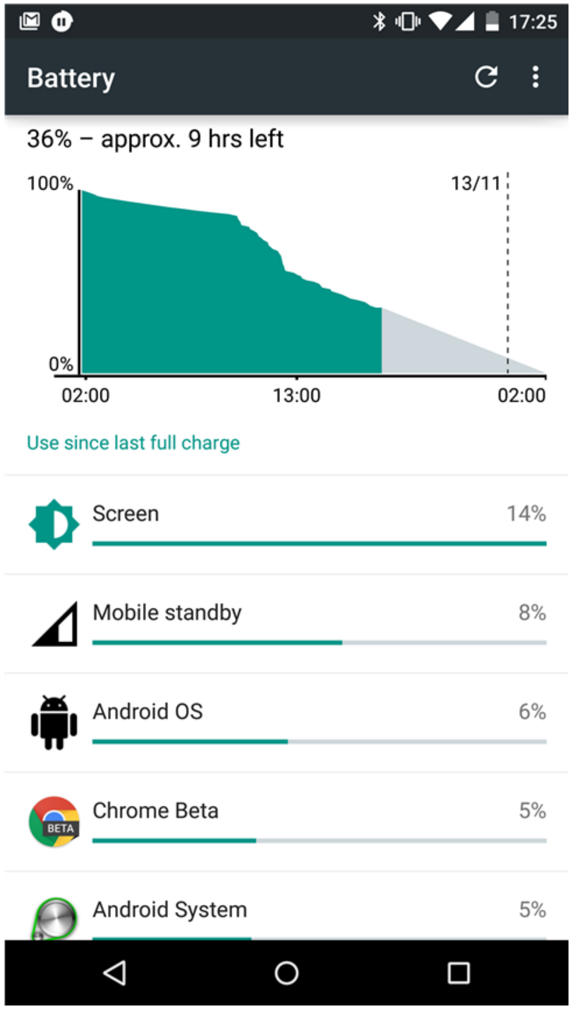
Battery consumption



75% – Charging on AC

Use since last full charge

Screen	39%
Android OS	8%
Voice calls	8%
Chrome Beta	8%
Android System	6%



36% – approx. 9 hrs left

Use since last full charge

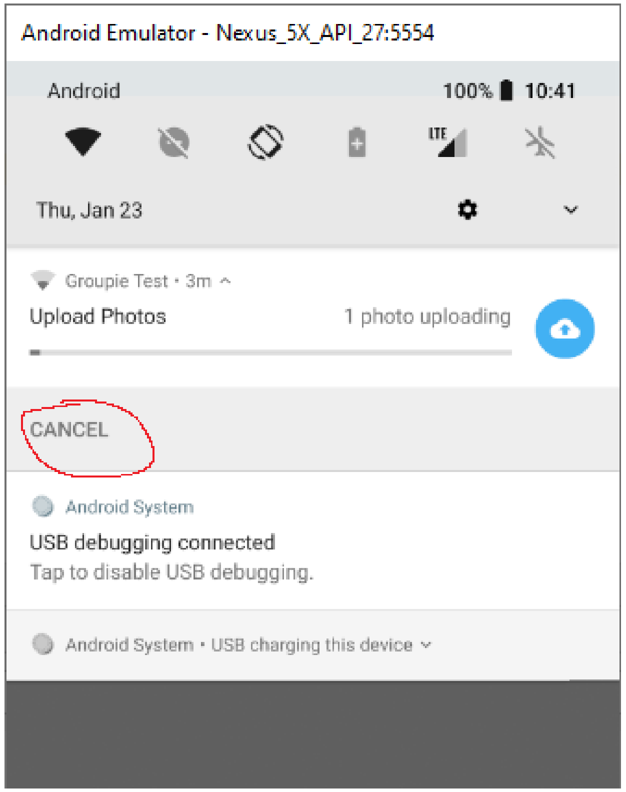
Screen	14%
Mobile standby	8%
Android OS	6%
Chrome Beta	5%
Android System	5%

Battery consumption

- **Screen & CPU**
  - Foreground vs Background
  - Activity vs Services
  - Sleep vs Active
- **Input modalities**
  - Type, Talk or Swype
- **Sensing**
  - Location (GPS vs Network)
  - Activity monitoring sensors - Accelerometer, Gyroscope, Magnetometer, etc.
  - Camera
- **Network interface**
  - Cellular>WiFi>Bluetooth in general
  - Network protocols

Screen & CPU

- Be cautious on the use of Services
  - Background vs foreground
- Move the app to background, if the user interaction is not required.
  - Activity vs Services in Android
- Good programming practices
  - Efficient algorithms
  - Reduce disk access frequency


Android Emulator - Nexus_5X_API_27:5554

Android 100% 10:41

Thu, Jan 23

Groupie Test • 3m

Upload Photos 1 photo uploading

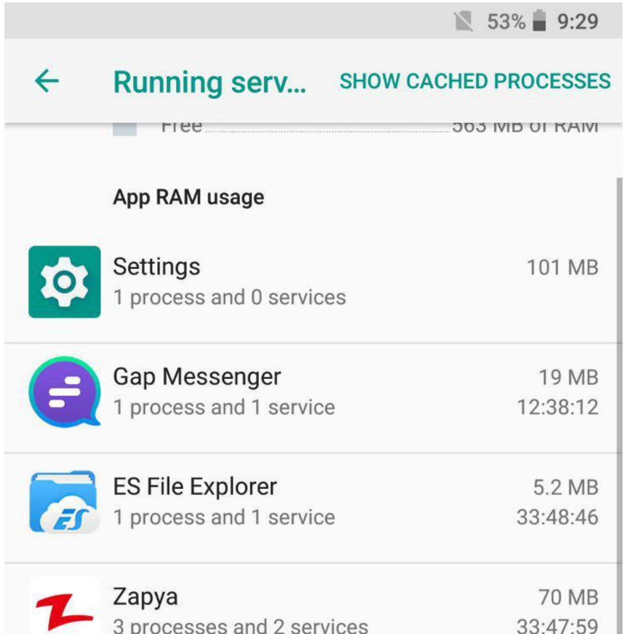
CANCEL

Android System

USB debugging connected

Tap to disable USB debugging.

Android System • USB charging this device


53% 9:29

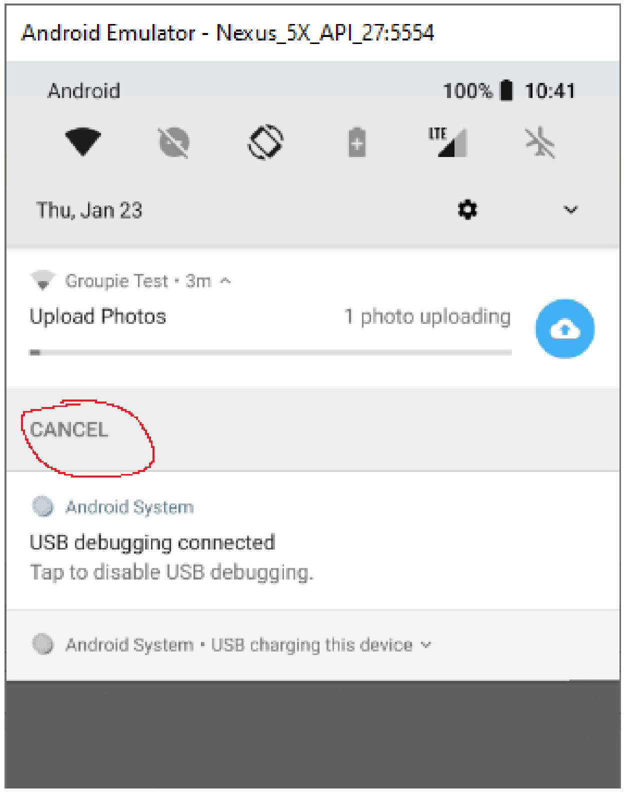
Running serv... SHOW CACHED PROCESSES

App RAM usage

Settings 1 process and 0 services	101 MB
Gap Messenger 1 process and 1 service	19 MB 12:38:12
ES File Explorer 1 process and 1 service	5.2 MB 33:48:46
Zapya 3 processes and 2 services	70 MB 33:47:59

Screen & CPU

- Be cautious on the use of Services
  - Background vs foreground
- Move the app to background, if the user interaction is not required.
  - Activity vs Services in Android
- Good programming practices
  - Efficient algorithms
  - Reduce disk access frequency



Android Emulator - Nexus_5X_API_27:5554

Android 100% 10:41

Thu, Jan 23

Groupie Test • 3m

Upload Photos 1 photo uploading

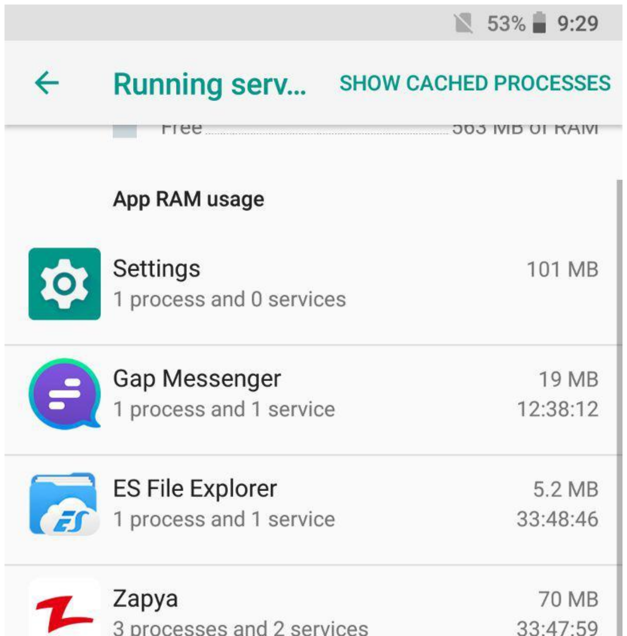
CANCEL

Android System

USB debugging connected

Tap to disable USB debugging.

Android System • USB charging this device



53% 9:29

Running serv... SHOW CACHED PROCESSES

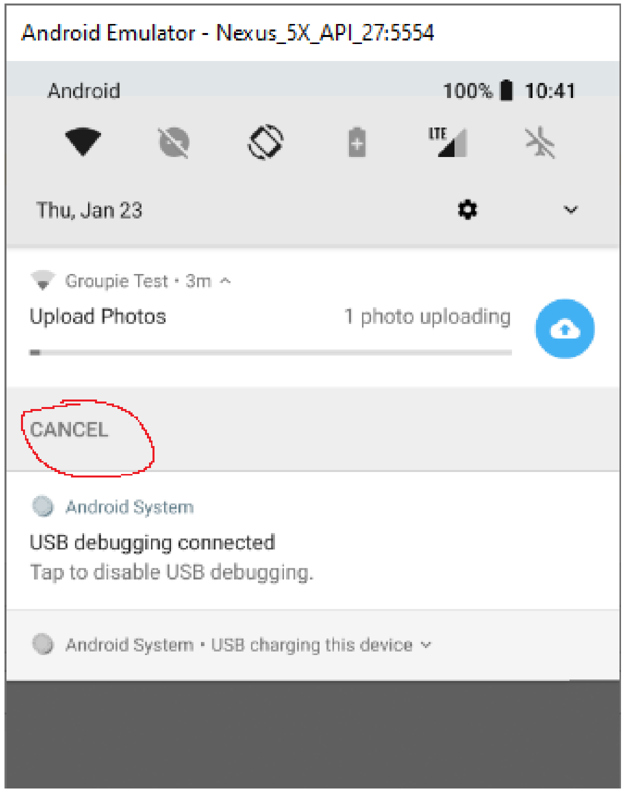
Free 563 MB of RAM

App RAM usage

Settings 1 process and 0 services	101 MB
Gap Messenger 1 process and 1 service	19 MB 12:38:12
ES File Explorer 1 process and 1 service	5.2 MB 33:48:46
Zapya 3 processes and 2 services	70 MB 33:47:59

Screen & CPU

- Be cautious on the use of Services
  - Background vs foreground
- Move the app to background, if the user interaction is not required.
  - Activity vs Services in Android
- Good programming practices
  - Efficient algorithms
  - Reduce disk access frequency


Android Emulator - Nexus_5X_API_27:5554

Android 100% 10:41

Thu, Jan 23

Groupie Test · 3m

Upload Photos 1 photo uploading

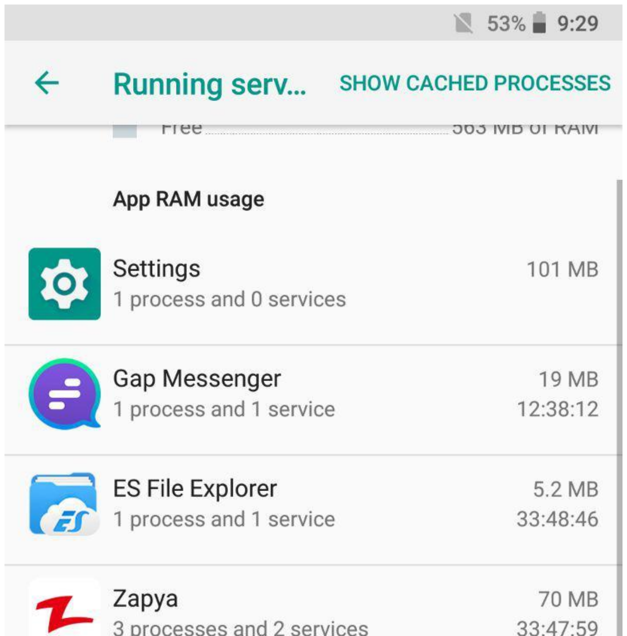
CANCEL

Android System

USB debugging connected

Tap to disable USB debugging.

Android System · USB charging this device


53% 9:29

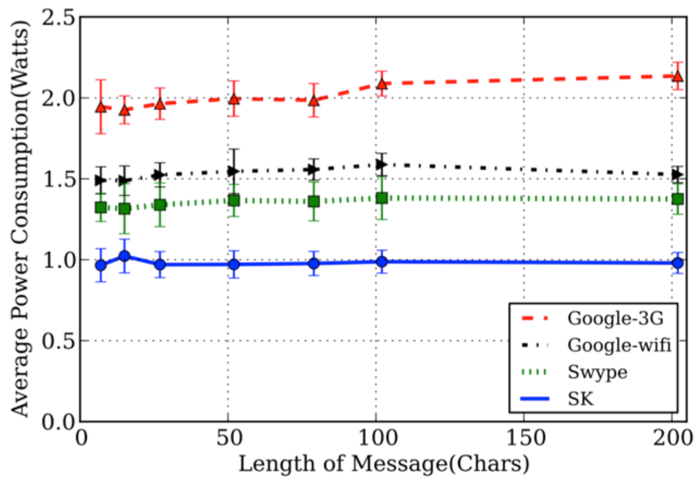
Running serv... SHOW CACHED PROCESSES

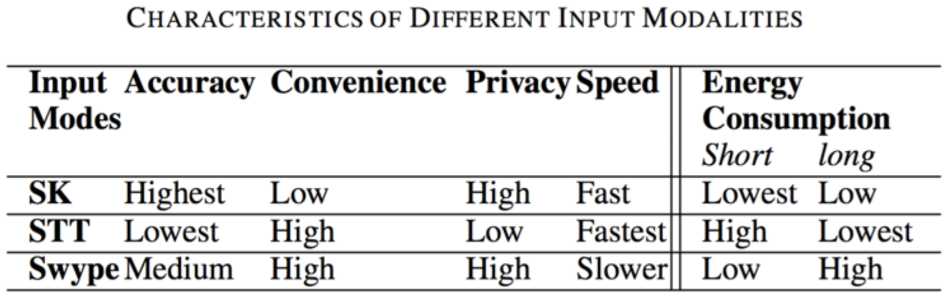
App RAM usage

Settings 1 process and 0 services	101 MB
Gap Messenger 1 process and 1 service	19 MB 12:38:12
ES File Explorer 1 process and 1 service	5.2 MB 33:48:46
Zapya 3 processes and 2 services	70 MB 33:47:59

Input Modalities

– Talk (Speech to Text), Type (Soft Key) and Swipe




CHARACTERISTICS OF DIFFERENT INPUT MODALITIES

Input Modes	Accuracy	Convenience	Privacy	Speed	Energy Consumption *Short*	*long*
SK	Highest	Low	High	Fast	Lowest	Low
STT	Lowest	High	Low	Fastest	High	Lowest
Swype	Medium	High	High	Slower	Low	High

Reference: Jiang, Fangzhou, et al. "When to type, talk, or Swype: Characterizing energy consumption of mobile input modalities." Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on. IEEE, 2015.

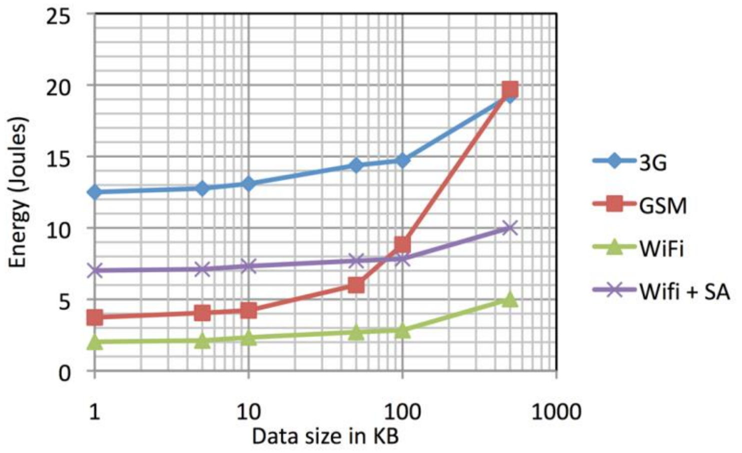
Sensing

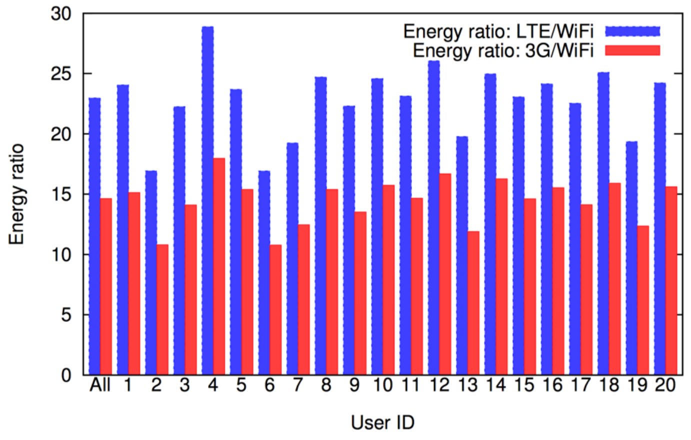
- Sensor API are primarily used;
  - Identifying sensors and sensor capabilities
  - Monitor sensor events

- **Identify Sensor features**
  - getResolution() for sensor resolution
  - getMaximumRange() for maximum range of measurement
  - **getPower() for sensor's power requirements**
  - getVendor() and getVersion() to optimize for different sensors or different versions of sensor
  - **getMinDelay() to determine maximum rate at which sensor can acquire data**

Network Power Usage

LTE vs 3G vs WiFi



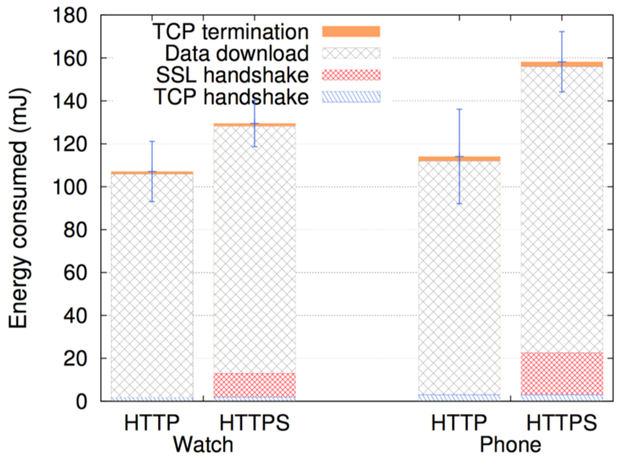


- In general, LTE>3G>WiFi.
  - Dependent on the network conditions, e.g. available bandwidth, signal strength, interference, etc.

Balasubramanian, Niranjan, Aruna Balasubramanian, and Arun Venkataramani. "Energy consumption in mobile phones: a measurement study and implications for network applications." Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference. ACM, 2009.

Privacy and Security

- Energy cost of secure protocols is not negligible.
  - More data due to adding noise, encryption, channel coding, etc.
  - Extra communication due to key exchanges



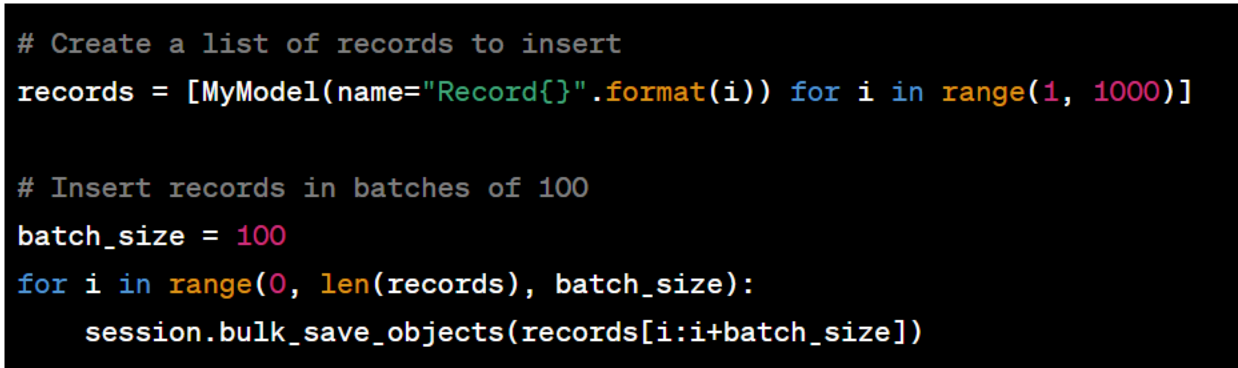
- Security is important in today's "mobile" world.
  - **It is worth the cost of energy**

H. Kolamunna, J. Chauhan, K. Thilakarathna, D. Perino, D. Makaroff and A. Seneviratne, "Are Wearables Ready for Secure and Direct Internet Communication?", ACM GetMobile: Mobile Computing and Communications, vol. 21, no. 3, pp. 5-10, Sep 2017.

Best Practices for Energy Management

- "**Lazy First**" design strategy
  - emphasizes deferring work until it is necessary or until the last possible moment.

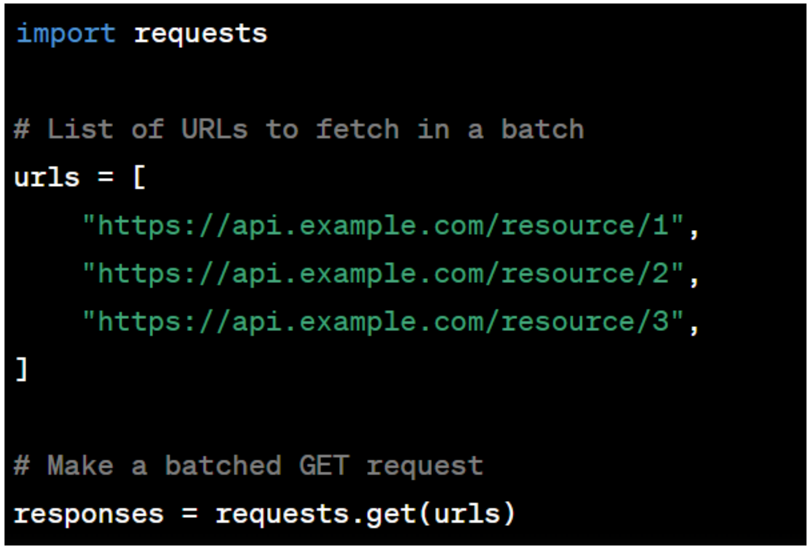
1. **Reduce**
   - Can we cache data with out re-downloading?
2. **Defer**
   - Can we wait until the device is charging ?
3. **Batch**
   - Can we batch downloads together?



```
# Create a list of records to insert
records = [MyModel(name="Record{}".format(i)) for i in range(1, 1000)]

# Insert records in batches of 100
batch_size = 100
for i in range(0, len(records), batch_size):
    session.bulk_save_objects(records[i:i+batch_size])
```

Best Practices for Energy Management

- Reduce
  - Upload/Download only necessary data [Week 7]
  - Reduce – Caching [Week 7]
  - Compressing [Week 7]
- Defer
  - Offloading (reactive/predictive) to energy efficient networks [Week 7]
- Batch
  - Reduce the frequency of communication [Week 6 – Location updates]



```python
import requests

# List of URLs to fetch in a batch
urls = [
    "https://api.example.com/resource/1",
    "https://api.example.com/resource/2",
    "https://api.example.com/resource/3",
]

# Make a batched GET request
responses = requests.get(urls)
```

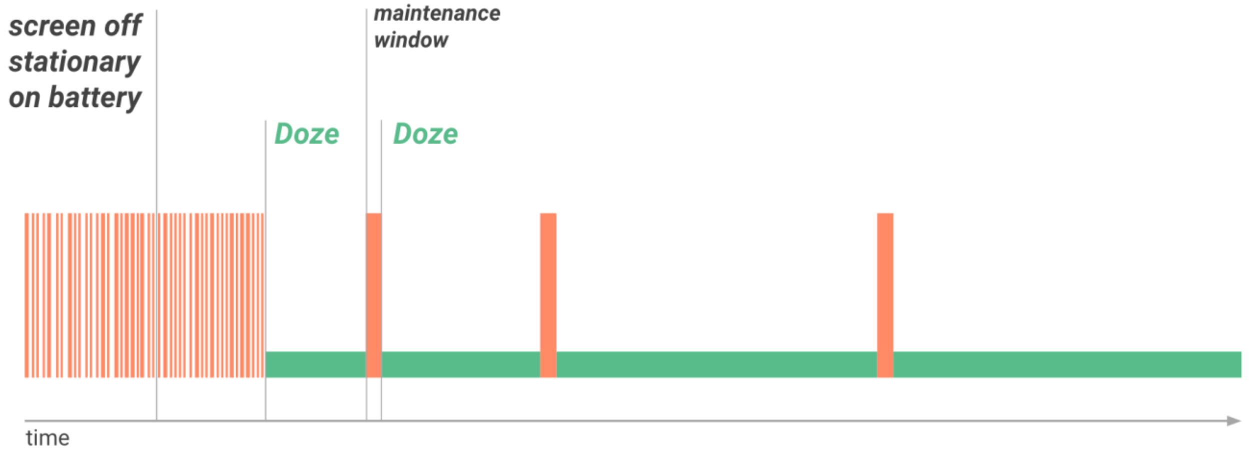
https://pypi.org/project/requests/

Best Practices for Energy Management

Android Platform Battery Management

Understanding Doze and Standby mode (HW)

- Deferring background CPU and network activity for apps when the device is unused for long periods of time
- If the device is **unplugged** and **stationary for a period of time**, with the **screen off**, the device enters Doze mode.
- After that, system provides **periodic maintenance windows**



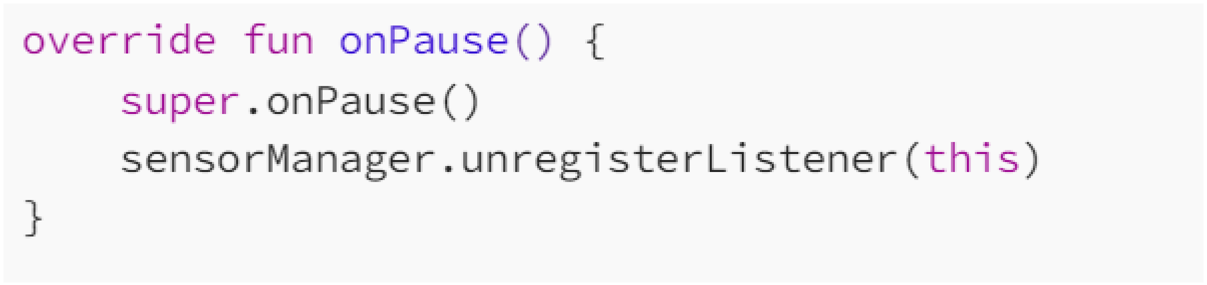
https://developer.android.com/training/monitoring-device-state/doze-standby

Best Practices for Energy Management

- Detecting “no-sleep” defects
  - Getting access to wake-lock and forgetting to release

```
wakeLock.acquire()
// Perform long-running tasks
wakeLock.release()
// Release the wake lock when the task is finished
```

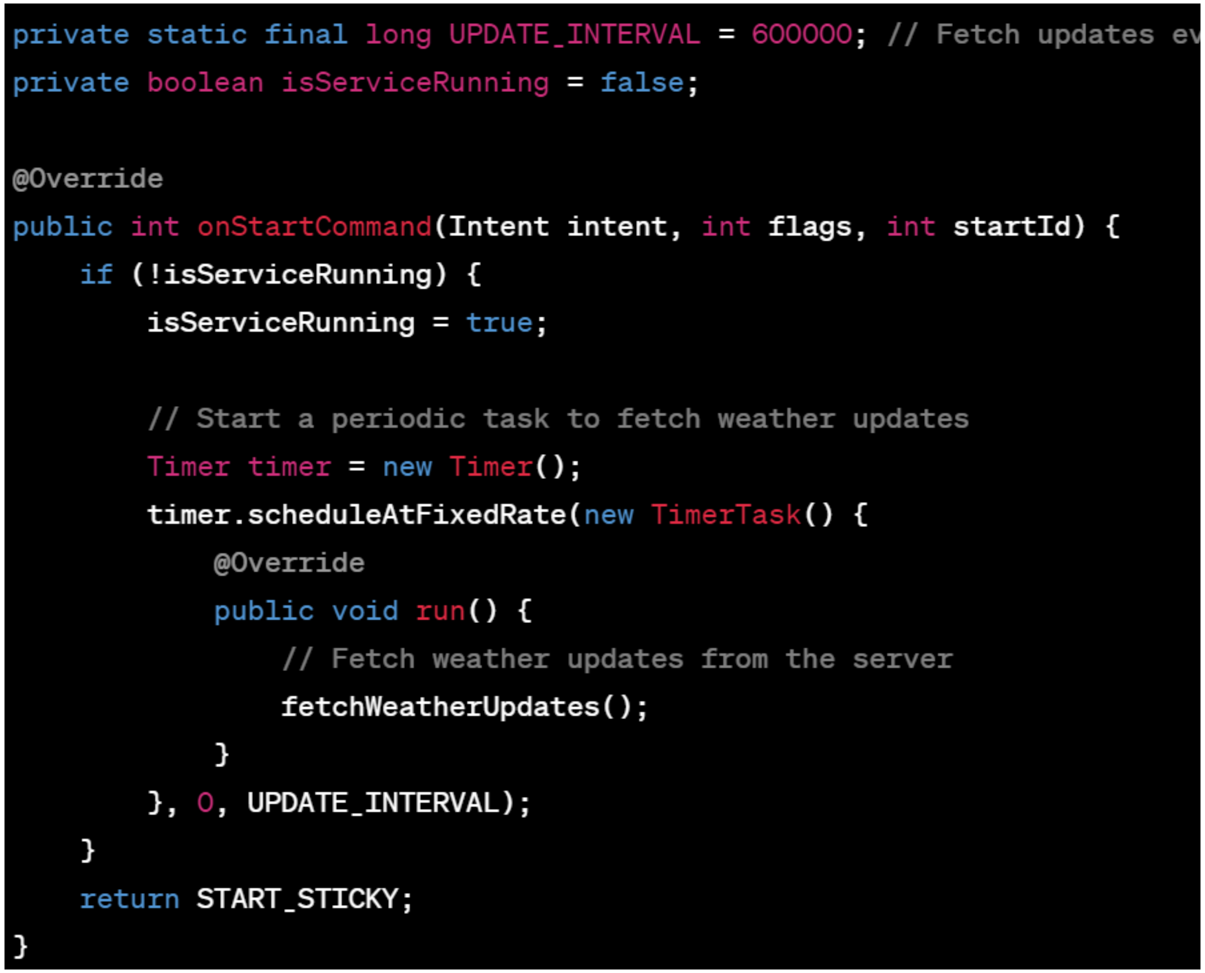
- Prevent resource Leaks
  - an app might register a sensor listener but forget to unregister it



```
override fun onPause() {
    super.onPause()
    sensorManager.unregisterListener(this)
}
```

Best Practices for Energy Management

- Optimize background services
  - E.g. Avoid polling server to see whether there is an update

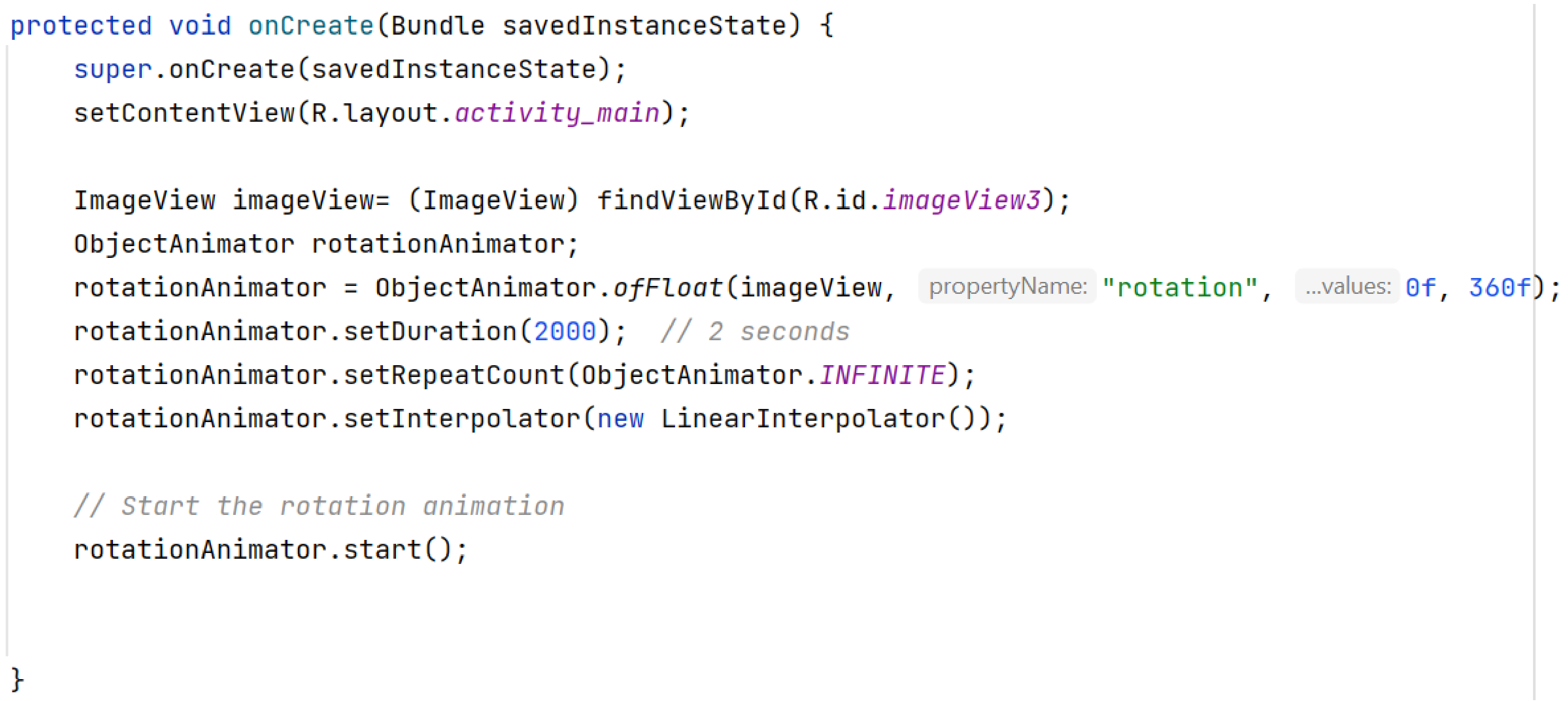


```java
private static final long UPDATE_INTERVAL = 600000; // Fetch updates ev
private boolean isServiceRunning = false;

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    if (!isServiceRunning) {
        isServiceRunning = true;

        // Start a periodic task to fetch weather updates
        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                // Fetch weather updates from the server
                fetchWeatherUpdates();
            }
        }, 0, UPDATE_INTERVAL);
    }
    return START_STICKY;
}
```

Best Practices for Energy Management

- Preventing GUI defects
  - a continuously animating UI element ?

```
<ImageView
android:id="@+id/myImageView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/my_image"
android:rotation="0"
android:animateRotationBy="360"
android:animationDuration="2000"
android:repeatCount="infinite" />
```



```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ImageView imageView= (ImageView) findViewById(R.id.imageView3);
    ObjectAnimator rotationAnimator;
    rotationAnimator = ObjectAnimator.ofFloat(imageView,  propertyName: "rotation",  ...values: 0f, 360f);
    rotationAnimator.setDuration(2000);  // 2 seconds
    rotationAnimator.setRepeatCount(ObjectAnimator.INFINITE);
    rotationAnimator.setInterpolator(new LinearInterpolator());

    // Start the rotation animation
    rotationAnimator.start();


}
```

https://developer.android.com/topic/performance/power/setup-battery-historian

Profile battery usage in Android

- **Batterystats** – a tool in Android framework
- **Battery Historian** – visualize data collected from Batterystats

Install Battery Historian

  - **https://github.com/google/battery-historian**

- Using Docker
  - Install Docker.
  - Choose a port number and replace <port> with that number in the commands below:
  - `docker -- run -p<port>:9999 gcr.io/android-battery-historian/stable:3.0 --port 9999`

  - For Linux and Mac OS X:
    - Historian will be available at `http://localhost:<port>`.

  - For Windows:
    - You may have to enable Virtualization in your BIOS.
    - Find the IP address of docket.
    - Historian will be available at `http://<ip address>:<port>`.

Profile battery usage in Android

- Using Android Debug Bridge (ADB)
  - **https://developer.android.com/studio/command-line/adb**

  - Follow this walkthrough - https://developer.android.com/studio/profile/battery-historian
    - Connect the device
    - Restart battery stats collection
      - `adb shell dumpsys batterystats -reset`
    - Disconnect the device
    - Play with your app
    - Reconncet the devoce
    - Dump battery stats
      - `adb shell dumpsys batterystats>[path/]batterystats.txt`
      - `adb bugreport>[path/]bugreport.txt`
    - Run Battery Historian and open bugreport file

Profile battery usage in Android

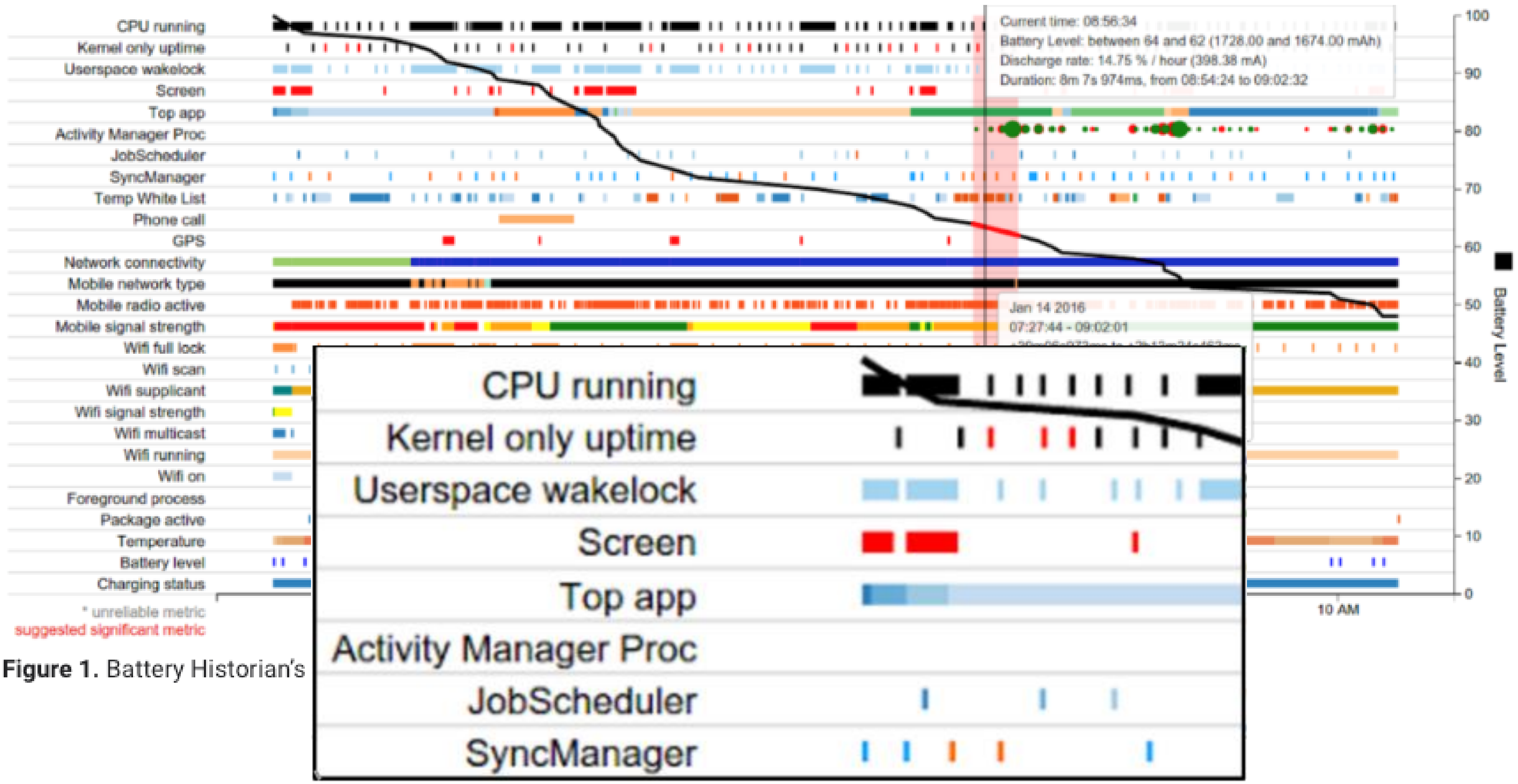
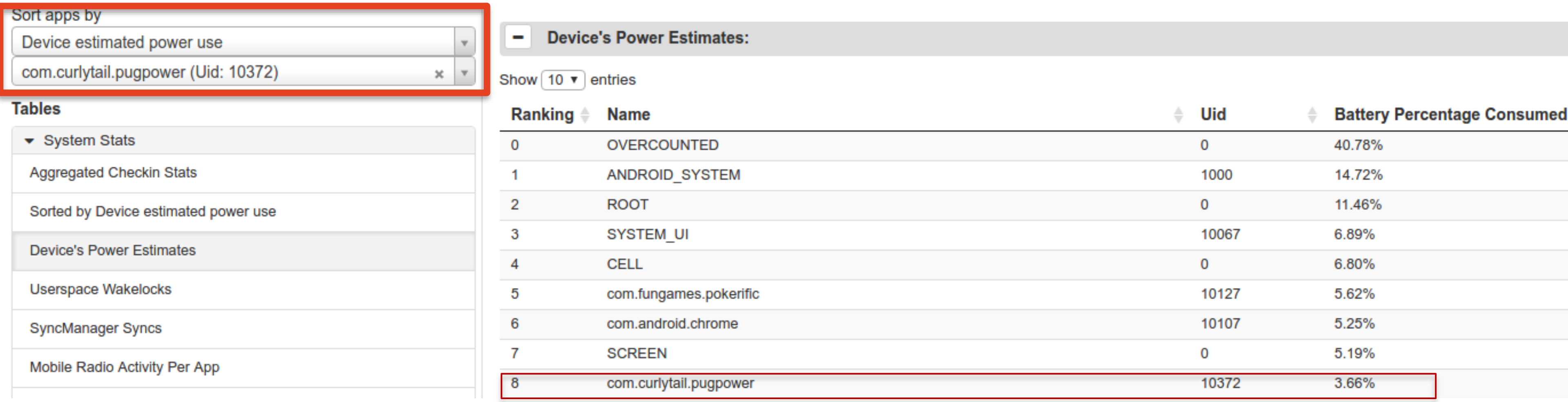


Figure 1. Battery Historian's

Profile battery usage in Android



Sort apps by

Device estimated power use

com.curlytail.pugpower (Uid: 10372)

Tables

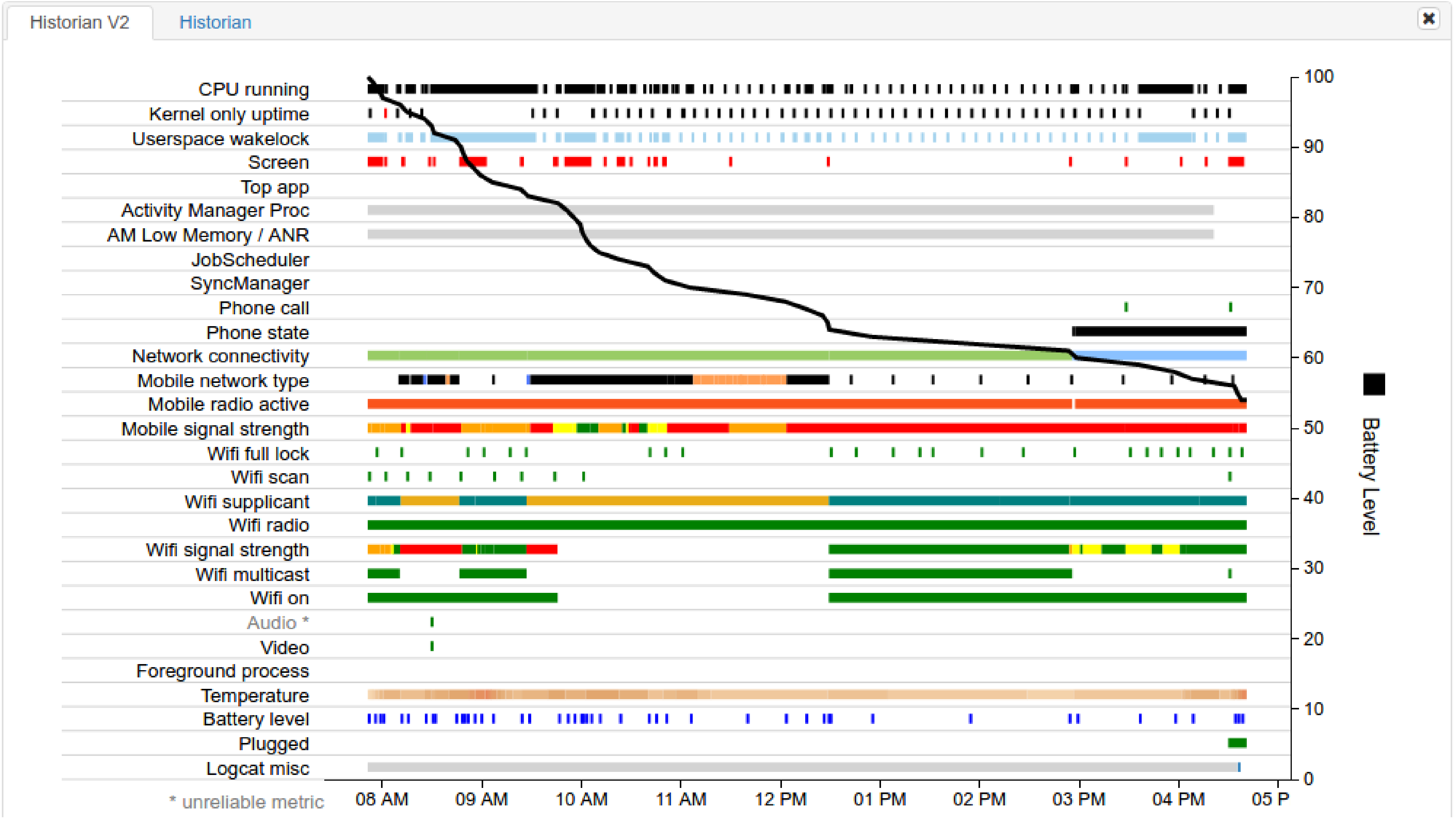
- System Stats
  - Aggregated Checkin Stats
  - Sorted by Device estimated power use
  - Device's Power Estimates
  - Userspace Wakelocks
  - SyncManager Syncs
  - Mobile Radio Activity Per App

Device's Power Estimates:

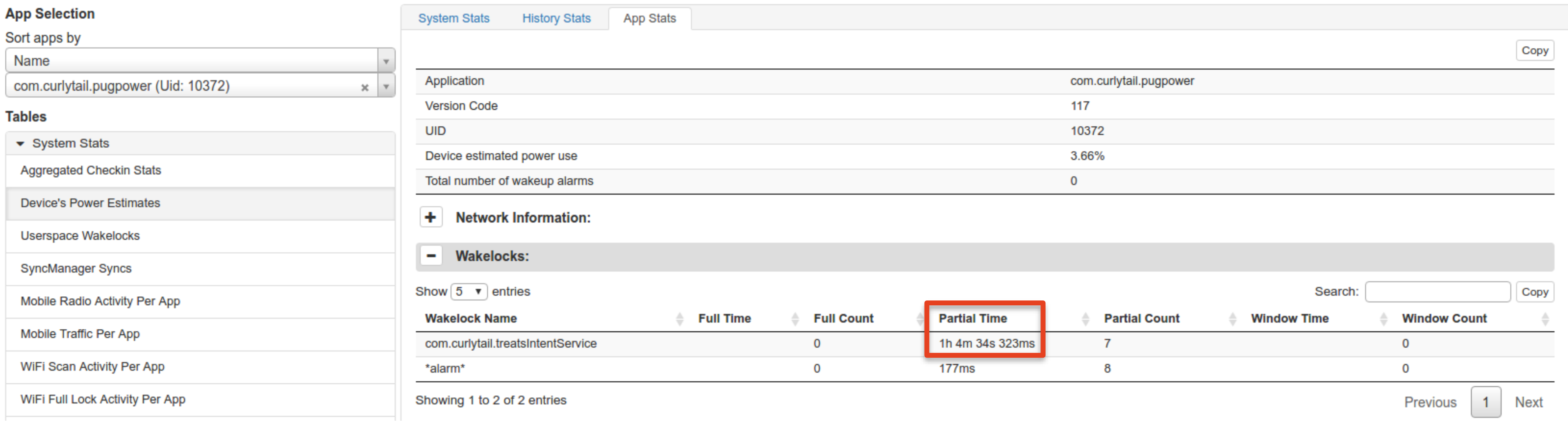
Show 10 entries

Ranking	Name	Uid	Battery Percentage Consumed
0	OVERCOUNTED	0	40.78%
1	ANDROID_SYSTEM	1000	14.72%
2	ROOT	0	11.46%
3	SYSTEM_UI	10067	6.89%
4	CELL	0	6.80%
5	com.fungames.pokerific	10127	5.62%
6	com.android.chrome	10107	5.25%
7	SCREEN	0	5.19%
8	com.curlytail.pugpower	10372	3.66%

Profile battery usage in Android – App specific visualaization



Profile battery usage in Android



Application	com.curlytail.pugpower
Version Code	117
UID	10372
Device estimated power use	3.66%
Total number of wakeup alarms	0

Wakelock Name	Full Time	Full Count	Partial Time	Partial Count	Window Time	Window Count
com.curlytail.treatsIntentService		0	1h 4m 34s 323ms	7		0
alarm		0	177ms	8		0

Showing 1 to 2 of 2 entries

Other usages of Battery Historian

- Firing wakeup alarms overly frequently (every 10 seconds or less).
- Continuously holding a GPS lock.
- Scheduling jobs every 30 seconds or less.
- Scheduling syncs every 30 seconds or less.
- Using the cellular radio more frequently than you expect.

What's Next?

- You can reach out to be to setup one-to-one meetings if you require feedback about your proposals.
- Have a great mid-semester break and the following public holiday !
- There will be a recorded lecture on Security Practices
- There will be a guest lecture in Week 9 – 3rd October.
- Happy Learning ☺

Thank you

THE UNIVERSITY OF SYDNEY