**Battery Optimisation for a given Citrix Mobile App**

*Project synopsis submitted*

*to*

**MANIPAL ACADEMY OF HIGHER EDUCATION**

*For Partial Fulfillment of the Requirement for the*

*Award of the Degree*

*of*

**Bachelor of Technology**

*in*

**Information Technology**

*by*

**Anuraag Palash Baishya**

**Reg. No. 140911296**

*Under the guidance of*

External Guide

**Mr. Rajeev Dixit**

Software Development Engineer (3P)

Mobile Platform Group (Automation)

Citrix R&D India Ltd

Ulsoor Road, Bangalore, India

Internal Guide

**Mrs. Aparna Nayak**

Assistant Professor

Department of I & CT

Manipal Institute of Technology

Manipal, India

January 2018

**Introduction**

In a survey of around 50,000 people across 25 countries, conducted by International Data Corporation, it was found that battery life is the most important factor when buying a smartphone, with 56% Android users, 49% iOS users and 53% of Windows phone users stating the same.[1] Mobile device manufacturers and mobile operating system developers pay great attention to battery usage. While device manufacturers are adding larger batteries to their devices, Android has been given a ‘Doze mode’ to reduce battery usage when phone is idle, along with a ‘Battery Saver’ mode that disables aggressive network usage, animations, and other elements that drain battery, and iOS comes with a ‘Low Power’ mode which essentially is similar to Android battery saver mode. Technologies such as Qualcomm Quick Charge™ and OnePlus Dash Charge have also been developed to reduce the time needed to charge a phone. Keeping such developments in mind, it has become very important for app developers to develop apps with optimized battery performance, so as to use as less battery as possible by default, i.e. even with battery life enhancing aids turned off.

**Problem Definition**

In this project, we will be working towards performing automated battery performance tests for a given Citrix Mobile app.

**Objective**

The objective of this project is as follows:

1. To gauge the current battery usage patterns of a given Citrix Mobile app
2. To determine factors within the app that drain more battery than desired

**Scope**

Beneficiaries of the work include:

1. The app development team who can utilize the results to optimize the app’s battery usage
2. The app users, who will have a more battery efficient app

**Background Theory**

There are a large number of factors that determine the battery usage of a device. Some of them are system level, including brightness, hardware power consumption and operating system battery consumption, which are beyond the control of application developers. However there exist a large number of factors such as CPU usage, network usage, wakelocks, etc which developers can look into, to optimize.

CPU usage results from performing computations. CPU usage is of two types, foreground usage, i.e when the app is running in the foreground, and background usage, for when app is running in the background (to perform tasks such as synchronisation, download, checking for mails, messages, etc). While foreground CPU usage forms a major part of battery consumption by cpu, if unchecked, a lot of unnecessary computation and processing maybe happening in the background, leading to battery drain (Unnecessary computation and processing may be taking place in the foreground as well)

Network usage, including mobile network and WiFi also form a major portion of battery usage. Like CPU usage, network usage is also of two types, background and foreground. Also like CPU usage, the app may be sending and receiving unnecessary packets in the foreground, background, or both. This is a potential candidate in making an app battery heavy.

Wakelock is a feature, which when requested by an app, lets the app to prevent the phone from going to sleep, i.e. the app will run continuously.[2] Wakelocks are handled differently in iOS and Android. While Android provides more control to the developer to invoke and use wakelocks through partial wakelocks, which lets apps run in the background, executing tasks, regardless of screen state or display timeouts, iOS permits partial wakelocks only for VOIP and location services.[3] It is pretty evident that an app requesting for unnecessary wakelocks will have a high energy consumption. This problem is greater for Android, due to availability of partial wakelocks.

There may be external factors in an app’s battery consumption, such as bluetooth usage, camera usage or some other function whose power consumption is not directly controllable by the app. Such factors cannot be handled by the developers of the app, and hence such factors will not be taken into account. There may also arise unanticipated bugs in the app which lead to battery drain.

**Methodology**

The battery consumption of an app depends a lot on the device, it’s make, operating system version, age, etc, especially for Android devices, of which there are thousands of different devices. In contrast, iOS runs only on iPhones, and thus there are very few different models. However, since we will be performing tests to determine usage patterns and battery demanding elements, we will not need the absolute battery consumption levels, and a general overview of the app’s battery consumption patterns will fulfil our requirements.

For Android, the intended test environment is as follows:

**Device:** The tests will most likely be run on a Samsung device, probably a Galaxy S7 or a Galaxy J7

**Device Profiles:** We will consider the following device profiles:

1. Device Memory (this affects the sync operation):
   1. <=70%
   2. >70%
2. Network: Cellular or WiFi. We will also simulate network fluctuations.
3. Background Processes: Apps running in the background.

**Tests to Run:**

1. Automated syncing of emails and calendar events after particular intervals.
2. Accepting or declining events.
3. Composing and sending emails (emails may contain attachments and contacts).
4. Run Build Validation Test (BVT) for running basic app operations. (Build Verification test is a set of tests run on every new build to verify that build is testable before it is released to the test teams)

NOTE: These tests will be automated through scripts

**Factors to be considered for Battery Performance Benchmarking:**

1. Checking the battery status before the test begins.
2. Enabling the location services for the application (if application requires).
3. Starting the data sync of the application.
4. Checking if the application is sending/receiving the data when in the background.
5. Observing the battery consumption while performing the above supported features by the application.

To test battery performance, we will use a tool, Battery Historian, developed by Google. Battery Historian analyses a bugreport taken from an Android phone and shows detailed battery use statistics, including CPU and Kernel uptime, running process information, mobile network, WiFi, GPS, JobScheduler, SyncManager usage details, and usage statistics for each app (such as CPU and network usage information, wakelocks, services, etc) that was running in the duration for which the bugreport was taken.[4]

For iOS, the details of the test environment, tools, and other specifications are yet to be decided.

**Timeline**

Here is the roadmap that we are planning to have for battery performance automation followed by estimated number of weeks.

1. Setting up battery historian and UI Automator automation setup - **1w**
2. Running UI Automator tests and understanding the flow- **1w**
3. Writing a shell script to send multiple emails (exchange version and exchange user will be used as arguments)- **2w**
   1. 3 iterations of 30 minute intervals each.
   2. 50 mails per interval.
4. Segregating the Test Cases (TCs) which test most commonly used UI interactions (viz. message compose, create event, contact) and add TAG as ‘BatteryPerf’- **1w**
5. Using device profile scenarios as a parameter in automation job to analyse the battery drain -**1w**
6. Creating a jenkins job and configure it to run Battery Performance automation **-1w**
7. Automating the complete flow - **2w**
   1. Select the desired profile
   2. Run Email Sync Script
   3. Run specific UI Automator TCs
   4. Generate battery stats report and dump it in jenkins archive
   5. Show the battery stats reports after
8. Performing analytics on battery consumption behaviour and representing areas of the app where majority of the battery drain is occurring - **4w**

\*w = week

**References**

[1]"IDC survey shows battery life is most important when buying smartphone", *GSMArena*. [Online]. Available: http://blog.gsmarena.com/idc-surveys-50000-people-reasons-behind-buying-smartphone-battery-life-deemed-important/. [Accessed: 17- Jan- 2018].

[2]"Keeping the Device Awake | Android Developers", *Developer.android.com*. [Online]. Available: https://developer.android.com/training/scheduling/wakelock.html. [Accessed: 17- Jan- 2018].

[3]"Is there any type of partial wake lock mechanism in ios?", *Stackoverflow*. [Online]. Available: https://stackoverflow.com/questions/20580157/is-there-any-type-of-partial-wake-lock-mechanism-in-ios. [Accessed: 17- Jan- 2018].

[4]"Analyzing Power Use with Battery Historian | Android Developers", *Developer.android.com*. [Online]. Available: https://developer.android.com/topic/performance/power/battery-historian.html. [Accessed: 17- Jan- 2018].