Energy Estimation of Android Applications

Anshu Rathor, Fadi Wedyan, and Yaser Jararweh
Department of Mathematics and Computer Science
(Duquesne University, Pittsburgh, PA)
https://www.duq.edu
rathora@duq.edu
Department of Software Engineering
The Hashemite University
Zarqa, 13315, Jordan
https://hu.edu.jo
fadi.wedyan@hu.edu.jo

Abstract

Smart phones in general have high shading contrast, the splendor of light, which can handle by the client, products applications working all the while, big screen, these all require superior processors to meet the intricacy of these applications. However, all these technologies drain the battery quickly, which has become a significant worry for Cell phone clients. Nowadays we have so numerous energy assessments apparatuses accessible in the application store. Aside from this, we have applications that save battery if applications are not dynamic being used. In this paper, We propose the presentation of some applications and devices that records the energy use by the applications at some given measure of time and number of cycles. Cell phone applications energy effectiveness is fundamental, however numerous Android applications experience the ill effects of genuine energy shortcoming issues

The research of energy utilization is acquiring significance because of the broadly expanding utilization of compact gadgets that sudden spike in demand for batteries, notwithstanding the monetary and natural concerns. Exploration on the energy utilization of uses requires exploring the measure of energy, distinguishing the holes which are answerable for tops in energy devoured by an application. Here, We explored tools like PETrA, Android Profiler to gauge the energy devoured by the applications. We have utilized 10 genuine well-known Android applications from various classifications.

Here, We explored tools like PETrA, Android Profiler to measure the energy consumed by the apps. Wehave used 10 real-world popular Android applications from different categories.

Index Terms: Smartphone, Applications, Power usage, Energy, Energy Consumption, Mobile Apps, Estimation.

1 Introduction

There has been a developmental change seen since recent years in the cell phones. Ongoing cell phones use amazing equipment and sensors which give more noteworthy offices to the clients. Advancement in cell phone innovation expanded the energy necessity of these gadgets. Cell phones are battery headed to permit portability to the client. Numerous clients would be satisfied if their cell phones just went on for a long time on a solitary charge. Lamentably, battery should be re-

energized a few times each day under ordinary utilization. Force usage is restricting the advancement of cell phones as the improvement in battery limit is moderate contrasted with the expansion in the intricacy because of new equipment and administrations. Force utilization has consistently been an issue in cell phones and applications composed by beginner software engineers are making it considerably more troublesome. As batteries can store fixed measure of force, the operational time a client can utilize its telephone inside one charging cycle is restricted. The operational time is one of the significant variables for purchasers when they purchase another telephone, along these lines, the cell phone industry is sharp in discovering answers for broaden the operational time.

Smartphones now tend to have high color contrast, the brightness of light, which can control by the user, multiples apps working simultaneously, big screen, these all require the high-performance processors to meet the complexity of these apps. However, all these technologies drain the battery quickly, which has become a major concern for Smartphone users. These days we have so many energy estimations tools available in the apps stores. Apart from this, we have apps that save battery if apps are not active in use. In this paper, We propose the introduction of some apps and tools that records the energy use by the apps at some given time. Smartphone applications energy efficiency is vital, but many Android applications suffer from serious energy inefficiency problems.

2 Motivation

Last year, I had an iPhone, and its battery drains very quickly. For instance: if it is 40-percent charged and We are using the camera for taking pictures from my phone. It suddenly switched off and I had so many events like that while I'm using a Google map, Camera, or surfing some website. So what I did I bought a portable battery to charge that phone anywhere which I used to carried it with me, and whenever the phone battery goes out, I plugged it simple. So basically I'm carrying two devices all the time expect at home, which were really painful whenever I am forgetting that portable battery I always live in fear that any time my phone will be switch off.

Since then I am exploring that which component in my phone consuming more energy. After researching it came out that Apple is one of the worst phones in the battery. but this is largely due to the site not having a version optimized for mobile use.

3 Battery Devouring Components of Smart Phone

In this section, we first describe about the components those drain the energy A study published in 2010 by Aaron Carroll and Gernot Heiser measured various key points of a smartphone. For example, they measured each component, such as the GPS, all under different power modes and usage levels. The power drain of each component varies greatly on usage. For example, calls drained 834mW and GPS 143- 166mW on the average. Screen back light amounts to between 7-8mW to 404mW depending on brightness. It should also be noted that the testing device is a 2.5G phone and 3G drains more power[**?**].

Mobile devices sensors majorly contribute to the total power utilization of the device. The most common today is the Accelerometer, Camera, GPS, Compass, Gyroscope, Gravitation sensors, Light sensor, Proximity sensors, Pressure sensors etc. Mobile hardware such as Network Module, Flash Card, RAM, Processor, Audio and Video Codecs are also consumes lot of battery power. Figure 1 shows the major components of the smartphone that utilizes battery power. Based on the study by Carroll and Heiser [1] the power utilizations of smartphone components are discussed below.

3.1 Screen

The power taken by the screen light over the scope of a whole number worth between 1 and 255. This is constrained by power management module. An ordinary Brightness control UI gave the value somewhere in the range of 30 and 255. The base back light power is approximately7.8m W and the maximum 414mW. The backdrop illumination devours negligible power when incapacitated

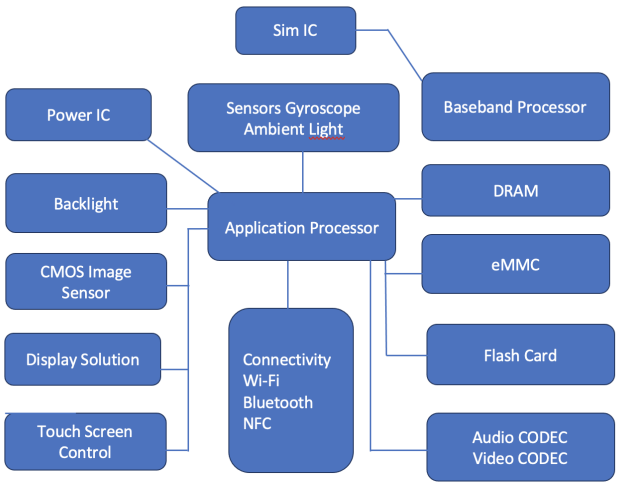

Figure 1: Power consuming components of Smartphone

3.2 CPU and RAM

CPU and RAM matter more than probably anything else here because it basically runs the whole phone. They both uses a lot of power in overall system approximately around 15-percent and it increases with usage. It is core part and needs to be awaken at all the time as long as the phone is switched on. Certain processes such as video streaming, internet browsing, CPU utilization rises it starts consuming more battery. RAM SD Card Both are memory units and uses around 7-10-Percent each [27] of the total battery power while performing memory read/write operations

3.3 Network

Wi-Fi and GPRS are biggest contributor in utilization of power. Wi-Fi showed a throughput of 660.1 ± 36.8 KiB/s, and GPRS 3.8 ± 1.0 KiB/s. The increased CPU and RAM power for Wi-Fi reflects the cost of processing data with higher throughput. The effect of signal strength on power resulted in an increase of GSM power of30-percent, but no effect on throughput.

3.4 GPS

The power consumed by the GPS module in three situations as shown below in Table 1 using only the internal antenna, with an external active antenna attached, and when idle.

State	Power(mW)
Enabled (Internal Antenna)	143.1 ± 0.05%
Enabled (External Antenna)	166.1 ± 0.04%
Disabled	0.0

Table 1: GPS Energy Consumption

3.5 Bluetooth

To measure Bluetooth power utilization, the audio output to a Bluetooth stereo headset was played. The power difference between this and the baseline audio benchmark should yield the utilization of the Bluetooth module. The headset was placed approximately 30 cm from the phone, and 10m in the far benchmark as shown in following Table.

Benchmark	Power (mW)	Power (mW)
-	Total	Bluetooth
Audio baseline	459.7	-
Bluetooth (near)	495.7	36.0
Bluetooth (far)	504.7	44.9

Table 2: Bluetooth power under the audio benchmark

4 USAGE SCENARIOS

4.1 Audio playback

The audio subsystem typically consumes 33.1mW. Approximately 58% of this energy is consumed by the codec, with the remaining 42% used by amplifier. Overall, the audio subsystem accounts for less than 12% of energy consumed. The additional energy consumed in the high-volume benchmark is less than 1mW compared with the low-volume case

4.2 Video playback

A video file of 5 minutes was recorded without audio and played on the same platform to measure the power usage. The brightness levels with backlight power on were 30, 105, 180 and 255. GSM power was included in the measurements. The CPU is biggest single contributor of power, the display subsystems still account for 38% of aggregate power, up to 68% with maximum backlight

brightness. Negligible power requires to load the video from the SD card.

4.3 Phone call

The measurement of power utilization of a GSM phone call includes loading the dialer application, dialing a number, and making a 57-second call. Thus the time spent in the call was approximately 40 seconds, assuming a 7-second connection time. The total benchmark runs for 77 seconds. GSM power clearly dominates in this benchmark at 832:4 ± 99:0 mW. Android disables the backlight during the call.

4.4 Web Browsing

The power utilization for web-browsing workload using both GPRS and Wi-Fi connections. The benchmark ran for a total of 490 seconds which consisted of loading the browser application, selecting a website and browsing several pages.

4.5 Taking Photo Cost Energy

* The camera is quite possibly the main bits of equipment on a smart-phone. Notwithstanding, it additionally has the ability to deplete the battery a lot. The first and most clear explanation is that it is a different piece of components. Although, by far most of camera battery channel comes from screen and processor utilization. Your presentation is required as a viewfinder and some OEMs even knock up the brilliance of the display when in camera mode. Moreover, every advanced smart phone has probably some post-processing and that likewise requires additional handling power.

5 CAUSES OF BATTERY DRAINING

There are many aspects to look at the problem of high energy utilization in smartphones. Some of the major causes of battery draining are discussed below.

5.1 Missing sensor Deactivation Bugs

To use a sensor, an application needs to register a listener with OS, and specify a sensing rate [9].When use of sensors is finished, its listener should be unregistered in time. Forgetting to unregister sensor listeners can cause unnecessary sensing operations.

5.2 Wake lock Registration Bugs

To keep a smartphone awake for operations, Applications need to acquire a wake lock from OS and specify a wake level. A full wake lock can keep a CPU awake and its screen at full brightness. Acquired wake lock must be released as soon as the computation finishes execution. Forgetting to un-register wake locks can quickly drain the battery.

5.3 Sensory Data Under utilization

Sensory data are acquired at the cost of battery power. These data must be effectively used by applications to produce benefits to phone user. When an application's program logic becomes complex, sensory data may be "underutilized" in certain executions. In such executions, the power cost for acquiring sensory data may outweigh the actual usages of these data.

5.4 No-Sleep power bugs in applications

No-sleep bugs are power bugs resulting from misshandling power control APIs in an app, which keeps smartphone battery usage active.

5.4.1 Causes from Event-driven Programming

In programming, Event driven computer programs is a programming in which the progression of the program is dictated by events, for example, client activities (mouse clicks, key presses), sensor yields, or message passing from different projects or threads. Event driven computer programs is

the prevailing utilized in graphical UIs and different applications (e.g., JavaScript web applications) that are focused on playing out specific activities in light of client input.[10] This is likewise valid for programming for smart phones drivers.

An ordinary client confronting smart phone application is composed as a bunch of events controllers with events being client or outer exercises.[11] The programmer needs to monitor every conceivable event and when it triggered and manipulate the wake locks likewise.

The Dialer app in smart phones application executes the dialing function of the smart phone. The application is set off when the client gets an approaching call or when the client taps the telephone symbol to settle on an outgoing calls. To execute this call or function, the application unequivocally primary tains three wake locks: FULL-WAKE-LOCK for keeping the screen on (e.g., in circumstances like when the client is dialing the numbers to call), PARTIAL WAKE LOCK for keeping the CPU on (e.g., if there should arise an occurrence of an approaching consider when the telephone is turned off), and PROXIMITY SCREEN OFF WAKE LOCK which turns the vicinity sensor on and off (to recognize client's closeness to the telephone).

To deal with the three wake locks, the application expressly keeps a state machine where the states address the lock conduct, i.e., which lock should be obtained and which should be delivered, and the "condition" of the telephone addresses the state advances. The conditions are different and incorporate occasions, for example, (a) if the telephone gets a call, (b) if the telephone is squeezed against the client's ear wherein case the closeness sensor triggers the screen to go off, (c) if the call closes, (d) if a wired or bluetooth headset is connected (e.g., in a call), (e) if the telephone speaker is turned on, (f) if the telephone slider is opened in the middle of calls, and (g) if the client clicked home catch in a call. For every one of these setting off occasions, the telephone changes the condition of wake lock state machine, gaining one and delivering another.

6 Energy estimation tools used in this research

6.1 PETrA

PETrA [1] is a software-based solutions are easier to use, but they are possibly less precise than hardware-based solution. In this demo, we present PETRA, a novel software-based tool for measuring the energy consumption of Android apps. With respect to other tools, PETRA is compatible with all the smartphones with Android 5.0 or higher, not requiring any device specific energy profile. We also provide evidence that our tool is able to perform similarly to hardware based solutions.

6.2 Android Profiler

The Android Profiler[15] in Android Studio 3.0 and higher replaces the Android Monitor tools. The Android Profiler tools provide real-time data to help you to understand how your app uses CPU, memory, network, and battery resources. The Energy Profiler appears as a row in of the Profiler window when you run your app on a connected device or Android Emulator running Android 8.0 (API 26) or higher.

6.3 Test Environment Setup for PETrA and Android Profiler

- Selected applications from different criteria
- Download those applications
- Run the selected Android applications on the selected tool(Petra/Android Profiler)
- Analyzing the measurements
- Report the results of applications

6.4 Implementation of PETrA

PETRA shows the configuration view. Using this window, the developer can customize PETRA with respect to (1) the location of the apk file, (2) the ANDROID MONKEY options, (3) the ANDROID MONKEYRUNNER script location, (4) the number of times (i.e., runs) the energy measurements

ID	Name	Category
1	Newson	News/Magazine
2	Notepad	Tool
3	Oxford Dictionary	Tool
4	Angry Bird	Game
5	YouTube	Video
6	Bubble Blast	Game
7	Sniper Shooter	Game
8	Accu Battery	Tool
9	Google Map	Navigation Tool
10	Calculator	Tool

Table 3: Applications tabel and their category

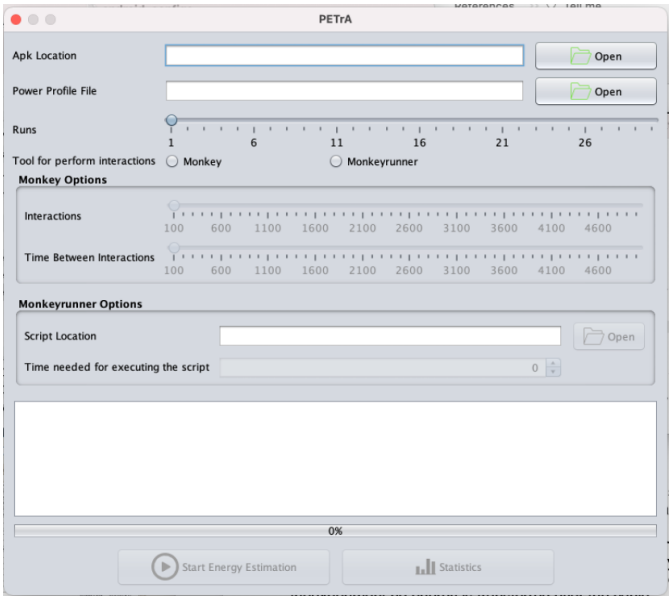

Figure 2: PETrA tool

must be computed, (5) the location of the ANDROID SDK, and (6) the location of the XML file containing the power profile of the device. Regarding the ANDROID MONKEY configuration it is possible to set the number of interactions (e.g., keystrokes, gestures) to send to the app and the time that must elapse between an interaction and the next one.

PETRA's workflow can be observed in Figure 3. PETRA starts a pre-processing phase, which entails installing the app (step 1), cleaning the app cache and resetting the Android tools that PETrA needs in order to make the estimations (step 2).[1] Such pre-processing phases are needed to create an adequate test environment not influenced by the previous app executions. Subsequently, in step 3 of Figure 1, it exercises the app using (1) ANDROID MONKEY tool4, or (2) a ANDROID MONKEY RUNNER [24] script. ANDROID MONKEY generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events with the aim of performing stress-testing of the app under analysis. ANDROID MONKEY RUNNER is an API for controlling an Android device. It allows to write programs

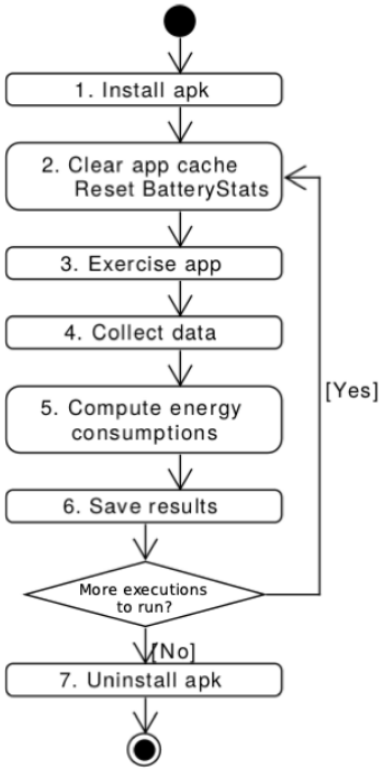

Figure 3: PETrA work flow

6.5 Implementation of Android Profiler

The Android Profiler in Android Studio 3.0 and higher replaces the Android Monitor tools. The Android Profiler tools provide real-time data to help you to understand how your app uses CPU, memory, network, and battery resources. The Energy Profiler appears as a row in of the Profiler window when you run your app on a connected device or Android Emulator running Android 8.0 (API 26) or higher.

If prompted by the Select Deployment Target

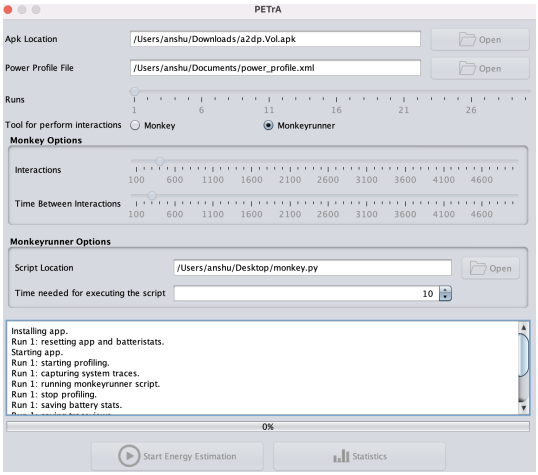

Figure 4: PETrA tool, installing app and resetting inner process

dialog, choose the device to which to deploy your app for profiling. If you've connected a device over USB but don't see it listed, ensure that you have enabled USB debugging. Click anywhere in the Energy timeline to open the Energy Profiler. When you open the Energy Profiler, it immediately starts displaying your app's estimated energy usage. You should see something similar to figure 1. To open the Energy Profiler, follow these steps: Select View > Tool Windows > Profiler or click Profile in the toolbar.

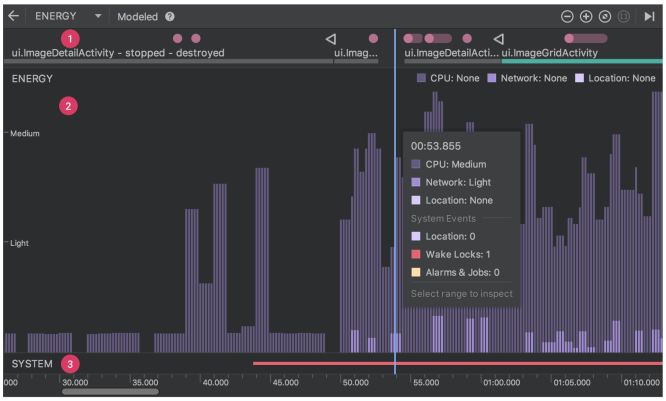

Figure 5: Android Energy Profiler

7 Inspect network traffic with Network Profiler

Why you should profile your application's network movement?

When your application makes a solicitation to the organization, the samrt phone should utilize

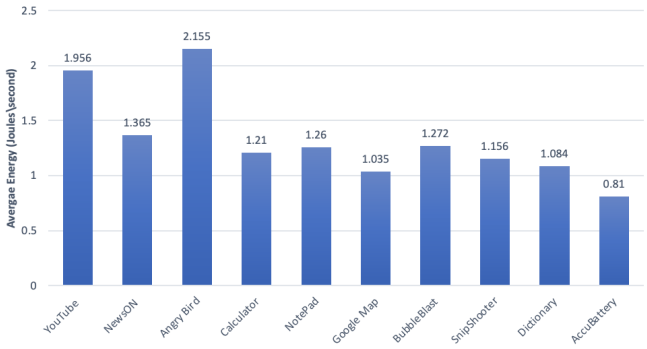

Figure 6: Average chart of five test run of each app

the power-hungry mobile or WiFi radios to send and get packets. The radios use power to move information or packets, however it use addition power to remain alert.[19]

Utilizing the Network Profiler, we can see for spikes, short spikes of network profiler, which imply that your application requires the radios to turn on as often as possible, or to remain alert for extensive stretches to deal with many short demands near one another. This example shows that we might have the option to advance your application for improved battery execution by batching network demands, consequently decreasing the occasions the radios should go on to send or get information. This likewise permits the radios to switch into low-power mode to save battery in the more drawn out holes between grouped solicitations.

7.1 Network Profiler overview

Android Studio 3.1 recently came out of beta. It has a lot of features such as kotlin lint checks, D8 compiler and also a revamped Network Profiler.[18] From early days of DDMS, we could always check how network data was being consumed but the current iteration of profiler has added a whole new set of features. Let's check them out First of all, network graph looks pretty neat.

I have used OkHttp to estimate the network usage sending and receive HTTP-based network requests.

7.2 OkHttp

*OkHttp is an library created by Square for sending and getting HTTP-based net-work demands. It is based on top of the Okio library, which attempts to be more proficient about perusing data by composing information than the standard Java I/O libraries by making a common memory pool. It is likewise the under lying library for Retrofit library that provides type security for burning-through REST-based APIs.

The OkHttp library actually provides an implementation of the Http Url Connection interface, which Android 4.4 and later versions now use. Therefore, when using the manual approach described in this section of the guide, the underlying Http Url Connection class may be leveraging code from the OkHttp library. However, there is a separate API provided by OkHttp that makes it easier to send and receive network requests, which is described in this guide.

In addition, OkHttp v2.4 also provides a more updated way of managing URLs internally. Instead of the java.net.URL, java.net.URI, or android.net.Uri classes, it provides a new HttpUrl class that makes it easier to get an HTTP port, parse URLs, and canonicalizing URL strings.

[20]For Request and responceWehave used following website.

`https://reqres.in`

7.3 Energy Estimation while using Network

We can see in figure 8 that when network usage is increasing then energy usage is also increasing.

8 Discussion

The findings blamed built-in systems for retrieving and displaying advertisements with the app. However, these more recent tests on mobile websites blamed poor coding choices for the higher-than-necessary consumption.Research showed that free mobile apps were also overly draining battery power.The researchers revealed that by analyzing and tweaking the design of Wikipedia, energy

Figure 7: Network Profiler

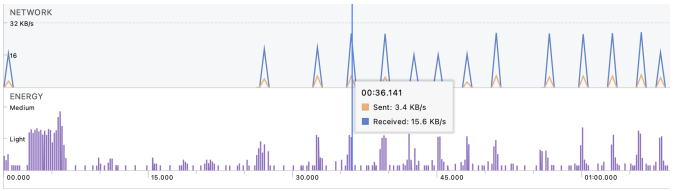

Figure 8: Energy Network Profiler

consumption could be reduced by 30% - without affecting the user experience.

Other tweaks suggested by the report include using the .jpeg image format instead of other file types like .gif and .png. Google's email offering Gmail was declared the most efficient site in the findings, a fact credited to its use of HTML for key functions rather than the more power-hungry Javascript. "Our experiments suggest that using links instead of Javascript greatly reduces the rendering energy for the page," the researchers said. "Thus, by designing the mobile version of the site differently than its desktop version, Gmail was able to save energy on the phone." The research is to be presented at the World Wide Web 2012 conference in Lyon this week.

Tests were run using an Android smartphone, with Google Mail declared as the "greenest" mobile site tested.After researching it came out that Apple is one of the worst phone in battery. but this is largely due to the site not having a version optimized for mobile use.

9 Conclusions and Future Work

Advancement in power use of uses in smart phone has become a significant field for research in the present IT world. The most important purposes behind battery depleting in cell phones are Network Data Communication like Multimedia Streaming, GPS, Wi-Fi, and Signal Dead Spots. Utilization situations like significant degree of Back light, high-goal Video Playbacks, Graphics Rich Gaming, and Heavy Computing Processes are the fundamental wellsprings of influence utilization. Different reasons for power wastage are Application Energy Bugs, No-Sleep Bugs, Unnecessary utilization of Sensors and consistently running Background Processes. Wake Locks and Sensors can likewise rapidly deplete the battery if the software engineers neglect to un-register it on schedule. This examination featured the issues and the answers for advancement of energy utilization in cell phones.

We presented PETRA, and Android Profile software-based tools for the estimation of the energy consumption of Android apps at method level granularity. The measures average energy consumption for 10 applications from various categories are studied. Our result show that the application have the significant variation in the amount of consumed energy. To understand their energy pattern, we analyzed the most energy consuming methods in the studies application.

We hope this paper demonstrated the importance of building a mobile application and site optimized for mobile devices.

References

[1] Dario Di Nucci; Fabio Palomba; Antonio Prota; Annibale Panichella; Andy Zaidman; Andrea De Lucia:
”PETrA: A Software-Based Tool for Estimating the Energy Profile of Android Applications”

[2] Dario Di Nucci1, Fabio Palomba13, Antonio Prota1, Annibale Panichella2, Andy Zaidman3, Andrea De Lucia1 University of Salerno — 2University of Luxembourg — 3Delft University of Technology:
”A Software-Based Tool for Estimating the Energy Profile of Android Applications”

[3] Balaji A. Naik, CSE Department SGGSIET, Nanded-431606 R.K. Chavan CSE Department SGGSIET, Nanded-431606 PowerUsage
”Optimization in Power Usage of Smartphones”
http://large.stanford.edu/courses/2017/ph241/park-j1/docs/naik.pdf

[4] Yepang Liu; Chang Xu; S.C. Cheung; Jian Lü, ”GreenDroid: Automated Diagnosis of Energy Inefficiency for Smartphone Applications”

[5] XXiao Ma, University of Illinois at Urbana-Champaign, Peng Huang and Xinxin Jin, University of California, Pei Wang, Peking University; Soyeon Park, Dongcai Shen, Yuanyuan Zhou, Lawrence K. Saul, Geoffrey M. Voelker, University of California:
”eDoctor: Automatically Diagnosing Abnormal Battery Drain Issues on Smartphones”

[6] Shutong Song, Fadi Wedyan and Yaser Jararweh ”Empirical Evaluation of Energy Consumption for Mobile Applications”

[7] Aaron Carroll NICTA and University of New South Wales, Gernot Heiser NICTA, University of New South Wales and Open Kernel Labs, ”An Analysis of Power Consumption in a Smartphone”

[8] Mohammad A. Hoque; Matti Siekkinen; Jukka K. Nurminen ”On the energy efficiency of proxy-based traffic shaping for mobile audio streaming”

[9] Abhinav Pathak, Y. Charlie Hu, Purdue University, Ming Zhang, Microsoft Research. ”Where is the energy spent inside my app? Fine Grained Energy Accounting on Smartphones with Eprof”

[10] Event-driven programming: ”Wikipedia”

[11] Samuel P. Midkiff, Y. Charlie Hu, Abhinav Pathak, Abhilash Jindal Purdue University:
”What is keeping my phone awake?: characterizing and detecting no-sleep energy bugs in smartphone apps”

[12] BBC
https://www.bbc.com/news/technology-17811557

[13] Android Profiler
https://developer.android.com/studio/profile/android-profiler

[14] Android Network Profiler:
https://developer.android.com/studio/profile/network-profiler?utm_source=android-studio#network_connection_troubleshooting

[15] Advanced Profiling:
https://developer.android.com/studio/profile/android-profiler#advanced-profiling

[16] OkHttp:
https://square.github.io/okhttp/

[17] OkHttp Git:
https://github.com/square/okhttp

[18] Network Profiler:
https://medium.com/android-news/quick-tip-network-profiler-in-android-studio-3--491e530a

[19] Android Developers-User guide Inspect network traffic with Network Profiler:

[20] Request and Response:
https://reqres.in

[21] GreenDroid:
Automated Diagnosis of Energy Inefficiency for Smartphone Applications
https://ieeexplore.ieee.org/document/6815752

[22] Green Droid:
http://greendroid.ucsd.edu

[23] Manage Network Usage:
basics/network-ops/managing

[24] Monkey Runner:
https://developer.android.com/studio/test/monkeyrunner

[25] 9 Battery consuming applications on Android:
https://phoneia.com/en/9-devouring-battery-applications-on-android/