Measure Energy Consumption on Various Network Conditions

Panagiotis Spentzouris, Dimitris Mendrinos

Athens University of Economics and Business Athens, Greece

Abstract. For the present project we developed an android application to measure energy consumption given different network interfaces and conditions (e.g 3G, LTE, Wifi etc). We extended an open source tool named Augemented Traffic Control (ATC)[**?**], built by Facebook, in order to build an application that simulates various network conditions and interfaces, parses battery statistics"reference" from android phones and records energy consumption after some event, e.g. play an online video.

1 Introduction

Initially we wanted to make some experiments to report the trade off of energy consumption and connection speed. We wanted to capture this trade-off for different interfaces and network conditions. The idea was to formulate an optimization problem the result of which would return the best possible choice of interface(between wifi or 3G/4G connection), i.e. that minimized the trade-off we mentioned earlier, on the fly. But along the way we came face to face with many challenges that made us pivot our early thoughts and turn into an alternative objective. The main problem, which we will mention later in more detail in this paper, was the lack of tools, commands, ways to get energy consumption related data, and simulate network conditions and different interfaces, along with the very large amount of time needed to complete even a single meaningful experiment.

1.1 We Pivoted!

All the above made us decide that we could contribute to the whole open source community by building an android app and an extended documentation that will make the procedure of getting energy consumption data and simulate various network conditions an easy task.

2 Research and Available Tools

To begin with, we searched the web for available methods, tools etc. that we could use in order to retrieve battery consumption related data from android phones. The only solution that exist, at least for now, is by using "adb" commands in

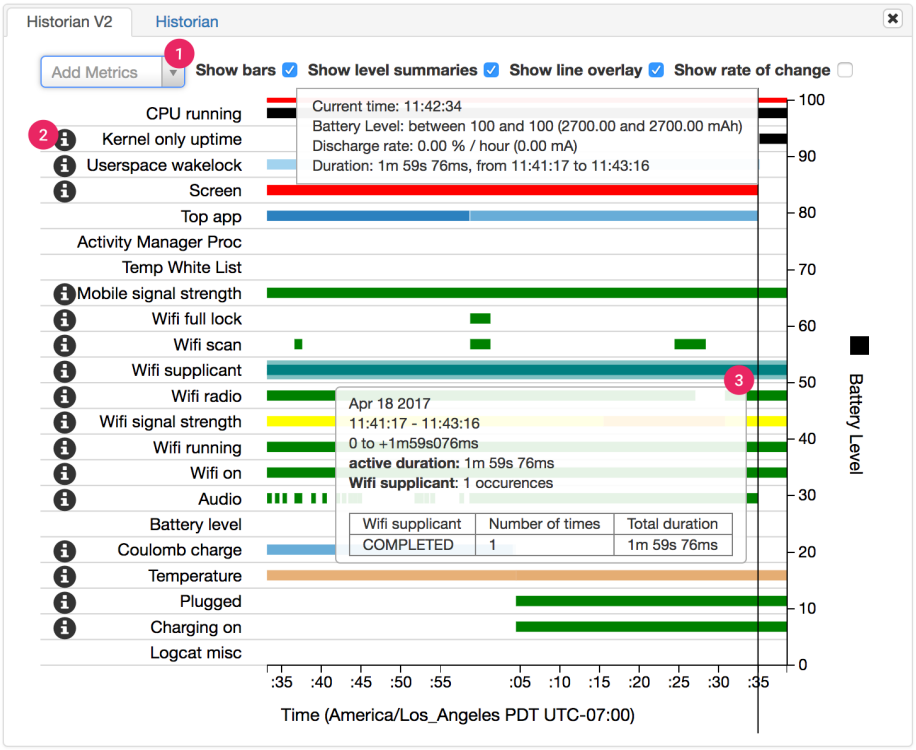

Fig. 1. Example of a Battery Historian chart.

android, i.e. execute commands as a "super user" from an android phone, that meaning that the device needs to be rooted [4]. After we rooted our android phones we executed the appropriate commands to get a file containing all the information we needed about the battery consumption of the phone, with data available even per application.

2.1 Battery Historian

Battery Historian [2] is a tool that converts the report from Batterystats into an HTML visualization that you can view in your browser. The whole procedure of how to install and use this tool can be found here [2].

3 Difficulties

At this stage of our efforts we had to deal with two main problems. The first one was how time demanding was to complete a single experiment. In order to run a meaningful experiment you have to first connect an android phone to your linux based" computer, reset the battery statistics on the phone, disconnect the phone

	WiFi		3G/4G	
	speed	battery consumption	speed	battery consumption
Youtube	14.15	0.17%	2.10	0.47%
Instagram	14.15	0.23%	2.10	0.53%
Gazzeta	14.15	0.3%	2.10	0.17%

	WiFi		3G/4G	
	speed	battery consumption	speed	battery consumption
Youtube	13.22	0.13%	72.53	0.47%
Instagram	13.22	0.20%	72.53	0.53%
Gazzeta	13.22	0.3%	72.53	0.43%

	WiFi		3G/4G	
	speed	battery consumption	speed	battery consumption
Youtube	0.70	0.13%	61.09	0.57%
Instagram	0.70	0.20%	61.09	0.60%
Gazzeta	0.70	0.3%	61.09	0.33%

Fig. 2. Connection speed and battery consumption for different apps, interfaces and network conditions.

from the computer, so that it won't charge while you run the experiment, run an app for a period of time, e.g. 5-15 minutes, and then reconnect the phone to the computer, download the battery statistics and use a tool like battery historian in order to read the battery statistics file and get the data that we want.

The second and more important problem was how to simulate the network conditions, i.e. 3g/4g or wifi. Naturally we couldn't do this manually, i.e. we couldn't find a place that has lets say good/bad connection to wifi/4g. But even if we could this methodology couldn't scale and allow us to do a significant number of experiments. Below, we present some experiments we did using the above methodology for different apps, interfaces, and network conditions.

4 Solution

4.1 Facing Battery Stats Issue

In order to cope with the complicated methodology of fetching the energy consumption related data, we built our own algorithm. First, we use a set of necessary commands on terminal in order to get the file with the battery stats and then we feed that file to our java implemented parser that returns us the battery consumption data we want.

4.2 Facing NEtwork Conditions Issue

This particular problem was a very tough to solve. After a lot of search on the web we managed to find a tool that we could use in order to simulate network

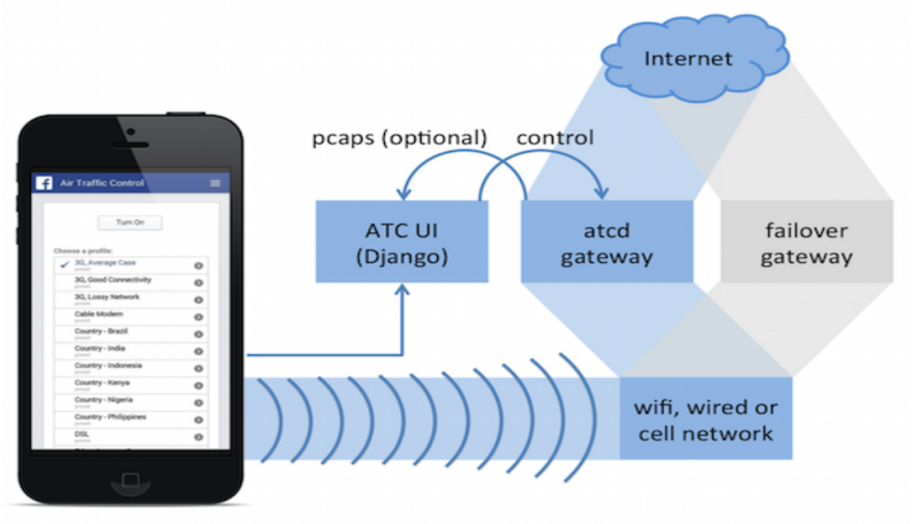

Fig. 3. How ATC works.

conditions. Nevertheless, the documentation available on the web is bad-written and with severe omissions.

Augmented Traffic Control: An Open-Source tool by Facebook. Augmented Traffic Control (aka ATC) is a project that allows developers to control the connection that a device has to the internet. Developers can use ATC to test their application across varying network conditions, easily emulating high speed, mobile, and even severely impaired networks. Aspects that can be controlled by ATC include (i) Bandwith, (ii) Latency, (iii) Packet loss, (iv) Packet corruption, (v) Packet reordering.

Because ATC runs on a gateway, any devices that are connected to a network routing through it, can use it to shape its traffic. Traffic can be shaped/unshaped using a web interface allowing any devices that have a web browser to use ATC without the need for a client application.

Augmented Traffic Control is made of a set of applications that inter-operate together. At the core of it is atcd, the ATC daemon, that is responsible for setting shaping characteristics. atcd provides a Thrift interface in order to interact with it. A django REST API, django-atc-api, is also provided in order to make it easier to build web applications that can interact with atcd using the well known REST paradigm. A simple Web UI is provided: django-atc-demo-ui.

5 Our Application

In our android application we managed to integrate the atc tool and use it in order to shape various network conditions. First, we had to set the server in a

computer running linux Ubuntu 16.04 operating system. Then, we turned the computer into a wifi hotspot. By doing that every device that connects to this hotspot can access the localhost atc server.

Then from an adnroid phone we start our application. The application starts with the communication webpage with the ATC Server. We connect with the Server to shape the network conditions. In the Server we created various profiles to simulate various types of networks. When the user shapes the network can press a button and be directed to another page of the app where an action takes place. In our case, the projection of a 10 minutes youtube video. After the video is over the user can go back to the atc server communication page and shape again the network conditions etc. For each projection of the video we measured and stored the battery consumption data as well as the network conditions. A number of screenshots follow to demonstrate the different application's activities. At this point it is important to say that our app is built in such a fashion that someone could instead of playing a viedo like in our case, start another application or do whatever else suits better his needs.

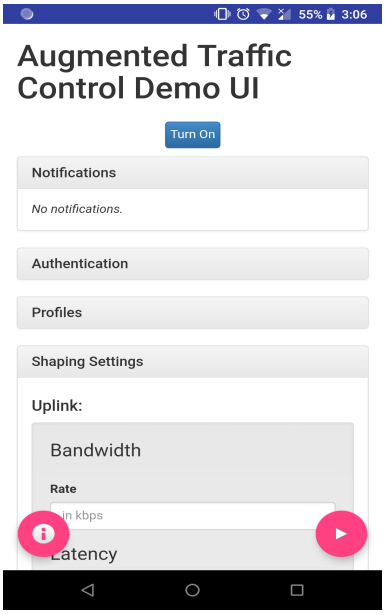

Fig. 4. Start page.

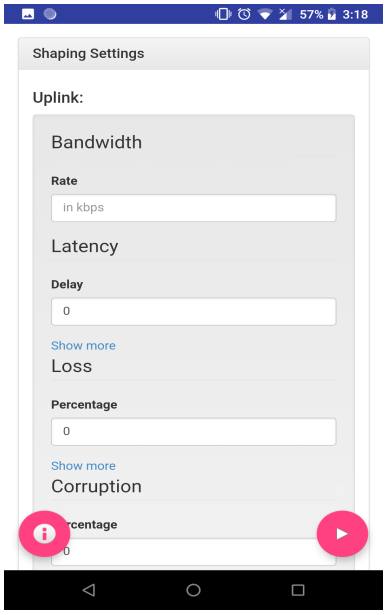

Fig. 5. Example of shaping uplink.

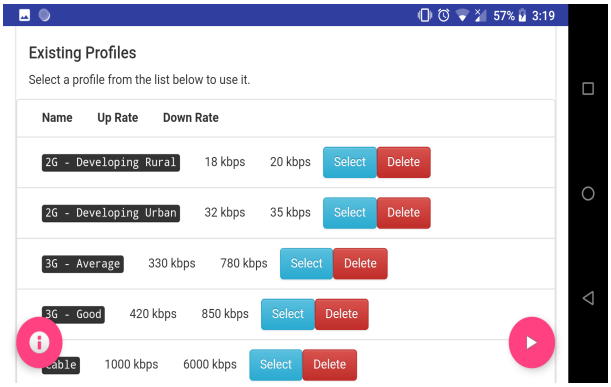

Fig. 6. Profiles we created.

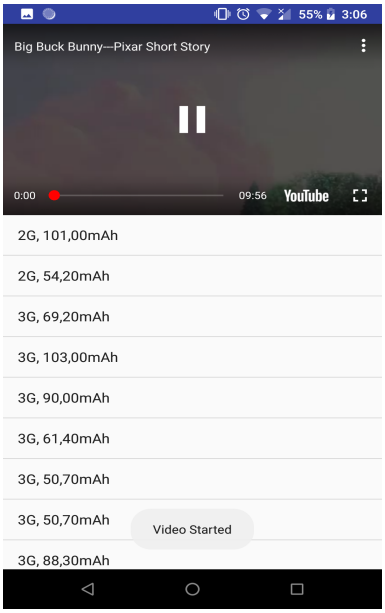

Fig. 7. Video playing after hitting the floating button with the play icon. Below the video player we can see data from previous experiments.

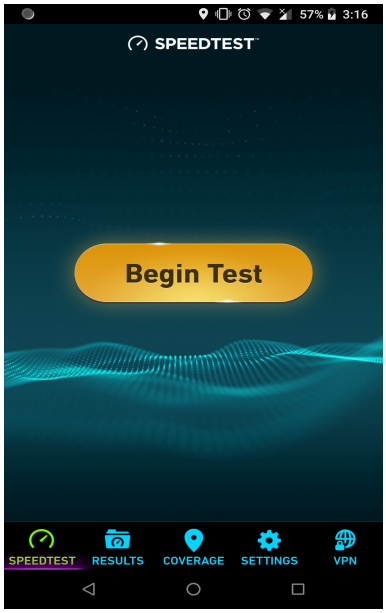

Fig. 8. Launch of speedtest after hitting the floating button with the exclamation mark.

6 Conclusion

In this project we developed an Android application to make experiments about energy consumption in different network conditions and record the statistics of the consumption to a file. This application can be extended in many ways but there some restrictions. All the procedures of the application including the reseting and downloading of the battery stats file, require the mobile device to be rooted. Also, one could run pretty much anything, i.e. the projection of a video is not restrictive since it can be switched with e.g. the execution of another application. The project is already up on github [3] and we hope that it will be a great guide and a significant assistance for developers that want to deal with similar tasks and projects.

At this point we want to mention that there are no available ways to change the signal strength of a phone's antennas, at least without hardware. As a result we weren't able to do meaningful experiments and explore the relation between energy consumption and signal strength for different interfaces. Also, we weren't able to find a way to dynamically measure the congestion or network conditions. Thus, we weren't able to build a tool that will make decisions on choosing the best possible interface, in terms of speed or energy consumption, in a dynamically fashion.

To conclude, we strongly believe that our work will make a big contribution to the open source community as a ready and easy to use tool in order to measure energy consumption of any android app.

7 Experiments

In figure 9 we can see that wifi is by far the most battery friendly interface on average, while 3G and 4G are very close.

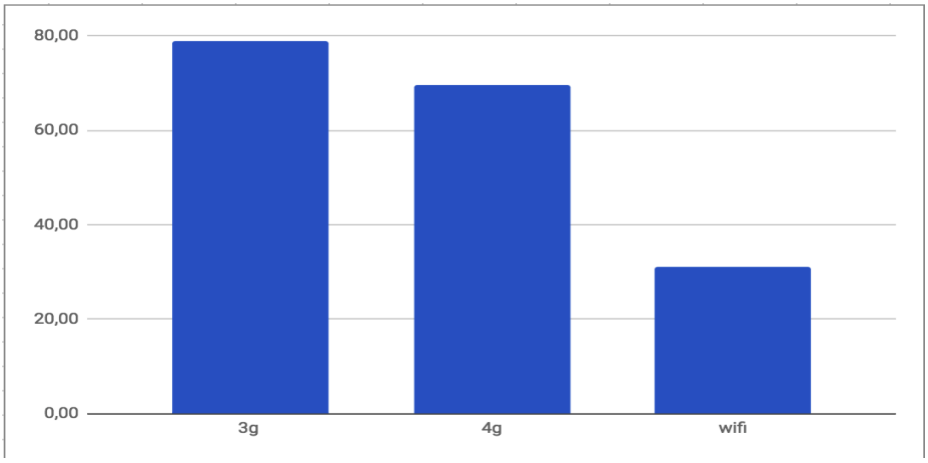

Fig. 9. Average battery consumption for different interfaces in mAh.

In figure 10 we can see that as the signal strength gets higher the energy consumption seems to drop.

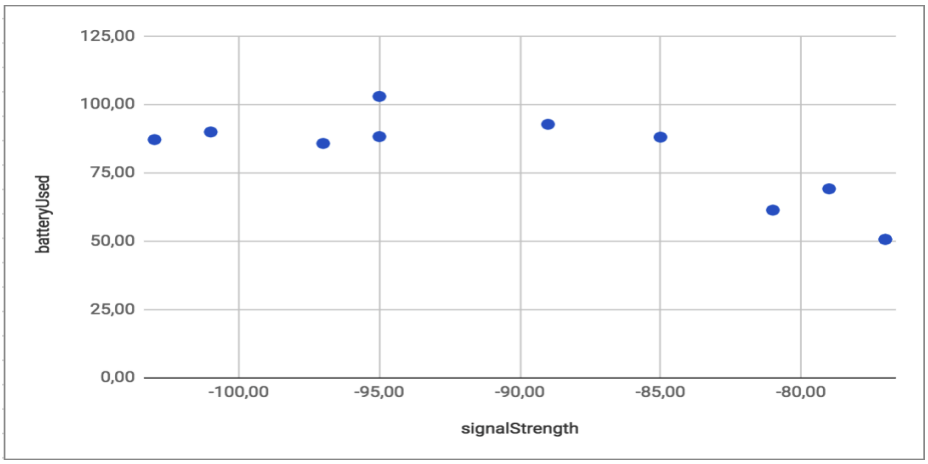

Fig. 10. Battery consumption for different signal strength values.

References

1. *ATS documentation*, Available on github page: *http://facebook.github.io/augmented-traffic-control/*.
2. *Battery Historian documentation*, Available on android developers page: *https://developer.android.com/studio/profile/battery-historian.html*.
3. *Our Application*, Available on github page: *https://github.com/HackArrow/WMC_Project*.
4. *Guide on how to root an Android device*, Available on page: *https://www.digitaltrends.com/mobile/how-to-root-android/*.
5. Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Z. Morley Mao, Lei Yang, "Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones".
6. Aaron Carroll, Gernot Heiser, "An Analysis of Power Consumption in a Smartphone".
7. Kirti Keshav, Venkata Raju Indukuri, Pallapa Venkataram, "Energy efficient scheduling in 4G smart phones for Mobile Hotspot Application".
8. Le Wang, Jukka Manner, "Energy Consumption Analysis of WLAN, 2G and 3G interfaces".
9. Daquan Feng, Chenzi Jiang, Gubong Lim, Leonard J. Cimini Jr, Gang Feng, Geoffrey Ye Li, "A Survey of Energy-Efficient Wireless Communications".
10. Shalini Prasad, S. Balaji, "Energy Dissipation Model for 4G and WLAN Networks in Smart Phones".
11. Zeeshan Hameed Mir and Fethi Filali, "On the Performance Comparison between IEEE 802.11p and LTE-based Vehicular Networks".