Demo Abstract: BatteryLab, A Distributed Power Monitoring Platform For Mobile Devices

https://batterylab.dev

Matteo Varvello†, Kleomenis Katevas⋄, Wei Hang‡, Mihai Plesa†, Hamed Haddadi†⋄,
Fabián E. Bustamante‡, Benjamin Livshits†⋄
† Brave Software, ⋄ Imperial College London, ‡ Northwestern University

ABSTRACT

There has been a growing interest in measuring and optimizing the power efficiency of mobile apps. Traditional power evaluations rely either on inaccurate software-based solutions or on ad-hoc testbeds composed of a power meter and a mobile device. This demonstration presents *BatteryLab*, our solution to *share* existing battery testing setups to build a distributed platform for battery measurements. Our vision is to transform independent battery testing setups into *vantage points* of a planetary-scale measurement platform offering heterogeneous devices and testing conditions. We demonstrate BatteryLab functionalities by investigating the energy efficiency of popular websites when loaded via both Android and iOS browsers. Our demonstration is also live at *https://batterylab.dev/*.

CCS CONCEPTS

• **Hardware → Batteries**; • **Software and its engineering → Cloud computing**.

KEYWORDS

Energy consumption, Smartphones, Distributed system

ACM Reference Format:
Matteo Varvello, Kleomenis Katevas, Wei Hang, Mihai Plesa, Hamed Haddadi, Fabián E. Bustamante, Benjamin Livshits. 2019. Demo Abstract: BatteryLab, A Distributed Power Monitoring Platform For Mobile Devices: https://batterylab.dev. In *The 17th ACM Conference on Embedded Networked Sensor Systems (SenSys '19), November 10–13, 2019, New York, NY, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3356250.3361946

1 INTRODUCTION

Advances in cloud computing have simplified the way that mobile apps are tested. Device farms [1, 12] let developers test apps across a plethora of mobile devices, in real time. To the best of our knowledge, no existing device farm offers *hardware-based* battery measurements, where the power drawn by a device is measured by connecting its battery to an external power meter. Instead, few startups [9, 13] offer *software-based* battery measurements where device resource monitoring (screen, CPU, network, etc.) are used to infer the power consumed by few devices for which a calibration was possible [5]. This suggests a demand for battery measurements, but a prohibitive cost for deploying hardware-based solutions.

In the research community, hardware-based battery measurements are quite popular [3, 4, 10]. The common research iter consists of buying the required hardware (often an Android device and a Monsoon power monitor [14]), set it up on a desk, and then use it sporadically. This is because battery testbeds are intrinsically *local*, *i.e.,* they require a researcher or an app tester to have physical access to the device and the power meter.

This demonstration presents BatteryLab, a distributed platform for high-accuracy energy measurements on Android and iOS devices. Inspired by PlanetLab [15], we developed a platform where members can both contribute and use remote resources (*e.g.,* one or more iOS / Android phones and a power monitor) in exchange of access to the hardware resources offered by other platform members. Experiments can be automated (*i.e.,* a script that automates some actions to the device) or user-based (*i.e.,* a user can control the device remotely via a web browser).

BatteryLab currently consists of two vantage points and four mobile devices. We will demonstrate how BatteryLab can be used by an experimenter to benchmark the energy efficiency of popular websites when loaded via both Android and iOS browsers.

2 SYSTEM OVERVIEW

BatteryLab consists of two main components: an *access server*, and a series of distributed *vantage points*. The access server, hosted on AWS, runs our website [17] and manage both vantage points and experiment scheduling. The access server is built atop of Jenkins [11] which enables an end-to-end test pipeline while supporting multiple users and concurrent timed session.

Vantage points (see Figure 1) are managed by a Raspberry Pi [16] which runs our software to enable remote testing. This consists of an ssh channel with the access server and *device mirroring* [6] which provides full remote control of test devices, via the browser. Next, a circuit switch connects to the Raspberry Pi's General-Purpose Input/Output (GPIO) interface and allows a programmatic selection of the phone that needs to be measured by the power monitor. The test devices also connect to the Raspberry Pi via USB, WiFi, and Bluetooth, for automation/instrumentation purpose.

Experimenters interact via BatteryLab using the Jenkins interface, where they can request access to a specific vantage point or device. Such access consists of either full remote control of the device (see Figure 2) or to run a *job*, *i.e.,* install a target application and run some tests. At any point in time the power monitor can be activated and fine-grained battery measurements can be collected.

SenSys '19, November 10–13, 2019, New York, NY, USA

ACM ISBN 978-1-4503-6950-3/19/11…$15.00
https://doi.org/10.1145/3356250.3361946

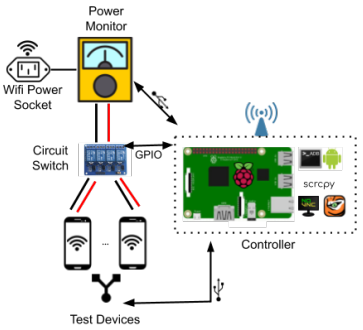

Figure 1: Vantage point architecture.

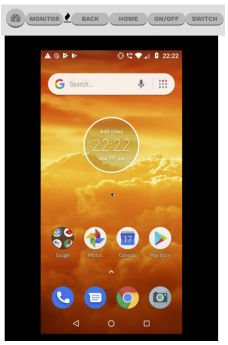

Figure 2: BatteryLab's GUI.

BatteryLab's main functionalities are available via a set of Python APIs, *e.g.,* to select a device or enable battery data collection. Device automation is instead offered via the following solutions:

Android Debugging Protocol (Android) – ADB [7] is a powerful tool/protocol to control an Android device. Commands can be sent over USB, WiFi, or Bluetooth. While USB guarantees highest reliability, it interferes with the power monitor due to the power sent to activate the USB micro-controller at the device. This is solved by sending commands over WiFi or Bluetooth. However, using WiFi implies not being able to run experiments leveraging the mobile network, and ADB-over-Bluetooth requires a rooted device. Based on an experimenter needs, BatteryLab can dynamically switch between the above automation solutions.

UI Testing (Android and iOS) – This solution uses UI testing frameworks (*e.g.,* Android's user interface tests [8] or Apple's XCTest framework [2]), to produce a separate version of the testing app, configured with automated actions. The advantage of this solution, compared with ADB, is that it does not require a communication channel with the Raspberry Pi. The main drawback is that it restricts the set of applications that can be tested since access to an app source code is required.

Bluetooth Keyboard (Android and iOS) – This approach automates a test device via (virtual) keyboard keys (*e.g.,* locate an app, launch it, and interact with it). The controller emulates a typical keyboard service to which test devices connect via Bluetooth. This approach is generic and thus works for both Android and iOS devices, with no rooting needed. Since it relies on Bluetooth, it also enables experiments on the cellular network. The main drawback is that the level of automation depends both on the OS and app support for keyboard commands.

3 DEMONSTRATION

BatteryLab currently consists of two vantage points, one located in London, UK (Imperial College University) and one in Evanston, IL (Northwestern University). Four devices are available to an experimenter: 3 Android devices and an iPhone 7. For the purpose of this demonstration, we will move the UK vantage point to the conference venue and showcase its functioning. We will leverage the other vantage point to showcase BatteryLab's remote capabilities.

The demonstration shows how an experimenter can use BatteryLab to benchmark the energy efficiency of a target website. We start by explaining how the above job is implemented, and how it gets deployed over BatteryLab. Next, we use our live tool at *https://batterylab.dev/* to benchmark the power consumption of several websites when loaded via popular web-browsers.

We will further demonstrate how Brave [18] leverages BatteryLab as part of it continuous integration testing suite. We show how a battery benchmark is automatically executed — generating an energy score — for each new browser release and pre-release.

ACKNOWLEDGMENTS

Katevas and Haddadi were partially supported by the EPSRC Databox and DADA grants (EP/N028260/1, EP/R03351X/1).

REFERENCES

[1] Amazon Inc. AWS Device Farm. https://aws.amazon.com/device-farm/.
[2] Apple Inc. XCTest - Apple Developer Documentation. https://developer.apple.com/documentation/xctest.
[3] D. H. Bui, Y. Liu, H. Kim, I. Shin, and F. Zhao. Rethinking energy-performance trade-off in mobile web page loading. In *Proc. ACM MobiCom*, 2015.
[4] Y. Cao, J. Nejati, M. Wajahat, A. Balasubramanian, and A. Gandhi. Deconstructing the energy consumption of the mobile page load. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):6:1–6:25, June 2017.
[5] X. Chen, N. Ding, A. Jindal, Y. C. Hu, M. Gupta, and R. Vannithamby. Smartphone energy drain in the wild: Analysis and implications. In *Proc. ACM SIGMETRICS*, 2015.
[6] Genymobile. Display and control your Android device. https://github.com/Genymobile/scrcpy.
[7] Google Inc. Android Debug Bridge. https://developer.android.com/studio/command-line/adb.
[8] Google Inc. Android Developers - Automate user interface tests. https://developer.android.com/training/testing/ui-testing.
[9] Greenspector team. Greenspector. https://greenspector.com/en/.
[10] C. Hwang, S. Pushp, C. Koh, J. Yoon, Y. Liu, S. Choi, and J. Song. Raven: Perception-aware optimization of power consumption for mobile games. In *Proc. ACM MobiCom*, 2017.
[11] Jenkins. The leading open source automation server. https://jenkins.io/.
[12] Microsoft, Visual Studio. App Center is mission control for apps. https://appcenter.ms/sign-in.
[13] Mobile Enerlytics. The Leader In Automated App Testing Innovations To Reduce Battery Drain. http://mobileenerlytics.com/.
[14] Monsoon Solutions Inc. High voltage power monitor. https://www.msoon.com.
[15] PlanetLab. An open platform for developing, deploying, and accessing planetary-scale services. https://www.planet-lab.org/.
[16] Raspberry Pi. Raspberry Pi 4 Model B. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.
[17] The BatteryLab team. BatteryLab. https://batterylab.dev.
[18] The Brave team. Brave: you are not a product. https://brave.com.