BatTrace: Android battery performance testing via system call tracing

Yeayeun Park
ypark2@swarthmore.edu

Mark Serrano
mserran2@swarthmore.edu

Craig Pentrack
cpentra1@swarthmore.edu

Abstract

The increasing number of tasks we can perform on our mobile devices feeds positively into the demand for devices with longer battery power. Since mobile devices have become integral parts of our daily lives with the number of tasks we can accomplish far outpacing battery performance improvements, consumers have increasingly encountered the issue of efficient device usage and battery life management. In this paper, we examine Android devices in particular and present BatTrace, an Android analysis tool that evaluates battery performance on the android platform by tracing system calls. BatTrace will execute different types of popular system calls, and extract the correlation between a particular system call and its influence on the battery. Subsequently, it will trace system calls made by individual Android applications and use system call performance data to profile each application. Finally, the analysis on the correlation between system calls and their battery usage, as well as the correlation between each application and system calls they initiate, will be combined to estimate battery usage of individual Android applications.

1 Motivation

Our project is motivated by an issue that we face daily: limited battery power on our mobile devices. The vast power available at our fingertips in mobile devices is tamed by the amount of battery physically available. Given that dynamic analysis executes data in real-time to evaluate and test programs, we searched for tools that would allow us to perform dynamic analysis on our mobile devices while uncovering low-level explanations as to what is really draining the battery. By profiling particular system calls in terms of their battery usage, we're hoping to derive a correlation between the two. Subsequently, with the appropriate tools (such as 'strace') we plan to trace system calls made by third party applications and, in turn, provide a good set of guidelines for mobile users to follow, when the low battery crisis hits.

2 Background

Historically much power consumption research has focused on using utilization-based methods. However, modern smartphones employ complex power strategies in device drivers and OS-level power management, sometimes rendering utilization as a poor model for representing power states and deducing battery usage (Pathak et al., 2011). While sometimes strong correlation exists between utilization and power consumption, often applications have constant power consumption while in certain states (while utilization fluctuates) or have high power consumption while low utilization (Pathak et al., 2011; Google, 2013). Additionally, measuring uti-

lization via performance counters results in accuracy loss (Pathak et al., 2011). Instead of modeling power with utilization, system calls, the only way of interacting with hardware and performing I/O, serve as a much more precise indicator of power consumption (Pathak et al., 2011). Past work and tools, such as eProf, have shown system calls to be an effective way of modeling power (Pathak et al., 2011; Yoon et al., 2012; Pathtak et al., 2012; Ding et al., 2013). Using the findings of eProf and other studies as justification, we plan to measure and classify system calls on Android smartphones in terms of their effect on battery life.

While eProf foregrounded system calls as an effective indicator of changes in power state, eProf used system calls as a means toward profiling applications' power consumption on a sub-routine level (Pathak et al., 2011; Pathtak et al., 2012). Developing models based off of system calls supplied a powerful tool, however the eProf research did not study battery drain as a result of particular system calls themselves and the frequency with which applications rely on certain system calls. Other work in smartphone battery research, including detecting energy-related bugs, correlating wireless signal strength with battery consumption, and generating battery usage information on the process or application level, has relied on system calls (Yoon et al., 2012; Pathak et al., 2012; Ding et al., 2013). We plan to supplement the research area by focusing our study on the system calls themselves, rather than using them as a means in tracking changes in power state, detecting bugs, measuring signal strength effects, or producing higher-level profilings as explored previously.

3 Our Idea

While dynamic analysis on traditional devices involves the most efficient use of finite computing resources, mobile devices introduce a new problem; finite power. The issue we immediately encounter when trying to analyze mobile software applications is that we ~~almost~~ never have access to the source code of the applications. This is especially true given the fact that most mobile software is proprietary in nature, leaving open source software to the relics that are desktop computers.

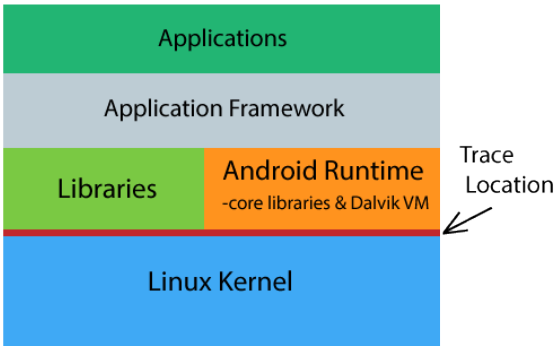

Figure 1: The Android environment stack. Trace location marks where we will be intercepting system calls

With this in mind we set out find a way of measuring mobile battery usage at very low level (software wise). We decided a good approach would involve monitoring activity at the system call level using a tool like strace. Ideally, we want to profile a variety of system calls based on how much battery is used while they are running. We intend to establish a baseline battery consumption level so we know how much battery is used by just the OS. Then, using simple programs that repeatedly make the same system call X times, we can determine how much battery was used as a result of initiating a particular system call X times.

Once system calls have been profiled, we can proceed to the last phase of the analysis. Our goal is to identify the system calls initiated by the Dalvik VM as a result of running an individual app. By identifying the types of system calls, as well as the number of calls made to an individual system call, we will be able to predict the app's impact on the battery based on what we learned about battery usage for individual system calls. While this approach may not be the most accurate, we believe it is the broadest approach that will allow us to profile any application regardless of the author or the nature of the software's license.

4 Milestones

Milestone 1
To begin the project we want to:

- Compile a small list of most used system calls by profiling a handful of popular android appli-

cations

- Compile simple C programs (on linux/android) that run the list most used system calls
- Collect data on battery usage as a result of running the C programs on the device

Milestone 2
After we have collected data on system calls, we will:

- Search for the most suitable tracing platform and trace system calls on various applications
- Identify and analyze the types of system calls and the number of each call initiated by individual applications

Milestone 3
With all this data in hand, the last step is to:

- Organize the data collected
- Attempt identification of patterns and relationships between system calls and battery usage by statistically analyzing the data
- Use established relationships to predict battery usage of a novel application

References

Ning Ding, Daniel Wagner, Xiaomeng Chen, Abhinav Pathak, Y. Charlie Hu, and Andrew Rice. 2013. Characterizing and modeling the impact of wireless signal strength on smartphone battery drain. In SIGMETRICS '13 Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems.

Inc. Google. 2013. Power profiles for android. Developer Guides.

Abhinav Pathak, Paramvir Bahl, Y. Charlie Hu, Ming Zhang, and Yi-Min Wang. 2011. Fine-grained power modeling for smartphones using system call tracing. In EuroSys '11 Proceedings of the sixth conference on Computer systems.

Abinav Pathak, Abhilash Jindal, Y. Charlie Hu, and Samuel P. Midkiff. 2012. What is keeping my phone awake? characterizing and detecting no-sleep energy bugs in smartphone apps. In MobiSys '12 Proceedings of the 10th international conference on Mobile systems, applications, and services.

Abhinav Pathtak, Y. Charlie Hu, and Ming Zhang. 2012. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In EuroSys '12 Proceedings of the 7th ACM european conference on Computer Systems.

Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulkoo Kang, and Hojung Cha. 2012. Appscope: Application energy metering framework for android smartphones using kernel activity monitoring. In USENIX ATC'12 Proceedings of the 2012 USENIX conference on Annual Technical Conference.