Precise Power Characterization of Modern Android Devices

Wei Lin, Joshua Wise

September 23, 2010

Abstract

We propose to perform a novel precise characterization of the power performance of modern Android cell phones. Finding the battery life on phones such as HTC's EVO 4G inadequate, we intend to answer the question: where is the power going? We demonstrate how existing tools (Intel's PowerTOP and the Android built in power meter) are inadequate for this end, and propose a mechanism of accurately measuring the power of each independent device on the system. We propose easily-buildable hardware to perform the precise power measurement as well as a series of analyses that can be performed given the recorded data to develop a model of the power consumption of such a device.

1 Motivation

With society's increasing dependency upon mobile cellular devices to perform functions beyond simply making a telephone call, the cell phone itself has evolved to into an entirely new device–the smartphone. The desire for speed and more integrated functionality has led these devices to sport bigger and more vibrant screens, higher resolution cameras equipped with flash, and support for the hundreds of thousands of little applications (apps) that a user can dream of. To keep up with communication, the modern device also comes equipped with a wide assortment of radios including, 802.11n ("Wi-Fi"), Bluetooth, WiMAX, and LTE – all in addition to the basic CDMA/GSM stack. However, all this comes at a price; what used to last a week detached from a wall outlet now barely survives the better half of a day. This phenomenon is especially apparent with the Android line of cell phones; on many of these devices, thirty hours of standby time is almost unheard of. It's true that battery density has failed to scale linearly with performance, but that doesn't explain the standby performance – where is all this power going?

In its current state, power consumption modeling for the Android platform is in its infancy. There exists the official application that ships with the operating system, and there also exist a few third party power instrumentation tools. The current state of some of these tools is detailed below; in short, though, they simply paint an incomplete picture of the power consumption of an Android phone.

The goal of this project, then, is to more accurately model the power draw of the Android platform, and in doing so, distinguish between the power required by the software stack (by waking the processor from idle via interrupts) and the devices and components integrated within the hardware stack. This will allow the attribution of the poor battery life of the platform to a source that is disproportionally drawing power. This will give the community and industry a direction for exploration which will hopefully lead to an improvement in battery life on the Android platform.

2 Prior work

2.1 PowerTOP

In the early days of energy-aware Linux optimizations, Intel developed a wakeup-analysis tool called PowerTOP. In Linux kernel version 2.6.21, support was introduced for a "tickless kernel", in which the CPU is not to be regularly woken up; prior to that version, the kernel would cause the CPU to wake from a sleep state on every tick of the system timer. With the "tickless kernel", the system timer is *disabled* while the system is blocked on I/O (i.e., there are no processes ready to run). In early iterations of the tickless kernel, drivers were poorly optimized for a system in which wakeups cost power, and would perform inefficient tasks such as busy-waiting for hardware.

Intel's PowerTOP is a tool designed to monitor the causes of wakeups on a tickless system; by narrowing down the sources of wakeups, a user (or developer) can modify his system to avoid drivers and behaviors that result in poor power performance. PowerTOP, as a secondary function, can monitor various metrics of a CPU's sleep state ("P-states" and "C-states"); in the case of the Android platform, these apply less, as the ARM CPUs do not have this functionality.

PowerTOP gives a clearer picture of what is consuming power than a simple CPU graph, but no actual estimate of current power being consumed is available; the view that it provides is incomplete to fully diagnose a system with poor battery life.

2.2 Android built-in tools

The built-in Android application is by far the most ubiquitous power monitoring application. Unfortunately, its utility is substantially limited. It shows graphs detailing devices and significant applications that have been run on the phone and its contribution to the total energy consumed since the last charge cycle in percentage form; however, it does not show any indication of the current power draw of the device or even a prediction of the amount of battery life remaining.

The built-in application can be customized by vendors to tweak the constants for each device, but there is evidence that the usage of this feature is imprecise at best. (Indeed, the data presented itself to be somewhat suspect; on the HTC Incredible, it showed a 4.3" AMOLED screen to be contributing only 7% to the total energy consumed.) We intend to build on the themes of this utility, but bring the mechanisms used to a higher degree of accuracy and calibration.

3 Proposed research

We anticipate building a mechanism to analyze current flowing into the phone. The present thoughts for this device involve using a lab supply to provide a "virtual battery" for the phone (to avoid having a physical battery buffer current spikes); current can be measured over time using a sense resistor and a small microcontroller (ATTINY) to log readings over time.

To analyze the power requirements of the various radios present on the phone, we plan on taking an initial measurement, and incrementally turning on each radio. For our baseline, we will measure the device's power draw with the device idling with all antennas disabled. We will then proceed to measure consumption with each radio on individually on, but idle (i.e., not actively transmitting or receiving data). We will then repeat this experiment, but with different data rates over each data radio. For the CDMA/GSM radio, we plan on doing one additional measurement with the phone electromagnetically isolated to observe the power requirements of the radio actively seeking a cellphone tower signal.

The display's backlight can be analyzed in a similar fashion; we can set the system to idle, and modify different backlight brightnesses, while measuring the current consumed. Similar data can be taken for the flashlight LEDs on the camera.

We intend to analyze the power consumed by the CPU in various states by tabulating the power consumption versus the CPU frequency, CPU load, and number of wakeups per second. We also intend to take long-term measurements of the software stack running in various "natural states" (i.e., uncontrolled by our research). Given these data points, we can hopefully create a model of CPU power consumption given various system load parameters.

With data collection complete, our analysis should proceed by creating a model of the power draw of the EVO 4G. In theory, the total power dissipation of the system should be the summation of all previous terms; the final stage in the analysis will be to validate these results. Providing this model will pave a pathway for future research in runtime analysis of software-caused power usage above and beyond that provided by the existing Android stack.

3.1 Deliverables

At the conclusion of this project, we intend to have produced a power model equation calibrated for the HTC EVO 4G; this equation will detail the power requirements for each radio present on the phone, the screen, and the software stack running at various CPU frequencies. We will also have produced an easy-to-build power measurement apperatus customized for this phone.

4 Schedule and milestones

Over the course of this project, we intend to meet the following milestones:

- **Oct. 11, 2010** – Hardware analyzer built.
- **Oct. 28, 2010** – Data collection complete; tables produced showing idle and load power consumption for all radios on EVO 4G, and power draw of CPU under various operating frequencies. (checkpoint 2)
- **Nov. 12, 2010** – Analysis underway.
- **Nov. 16, 2010** – Presentation given.
- **Dec. 3, 2010** – Written report written.