PowerTutor 

A Power Monitor for Android-Based Mobile Platforms

How PowerTutor works?

For those who are interested in how PowerTutor gets the power/energy estimation of different hardware components and of different applications, this page intends to answer your questions. We will first explain how we construct the power model for different hardware components, then we will explain how we assign power/energy usage to applications. More technical details can be found in "Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones".

Construction of the power model

Each hardware component in a smartphone has a couple of power states that influence the phone's power consumption. For example, CPU has CPU utilization and frequency level, OLED/LCD has brightness levels. The power model in PowerTutor is constructed by correlating the measured power consumption with these hardware power states. The power states we have considered in the power model includes:

- **CPU:** CPU utilization and frequency level.

- **OLED/LCD:** For hardware with LCD screen and OLED screen without root, we consider brightness level; For hardware with OLED screen with root access, we consider brightness together with pixel information displayed on the screen.

- **Wifi:** Uplink channel rate, uplink data rate and packets transmitted per second.

- **3G:** Packets transmitted per second and power states.

- **GPS:** Number of satellites detected, power states of the GPS device (active, sleep, off).

- **Audio:** Power states of the Audio device (on, off).

Assign power/energy to applications

Power usage is assigned to applications as though it were the only app running. The reason for this is that sometimes when two applications are running at the same time they can cause some of the hardware components to transition into states they wouldn't otherwise be in if only one of the apps were running. In these sorts of cases it's not clear how to assign power utilization on a per-app basis.

For example, consider the pathological case where two apps are transmitting a small amount of data. Suppose if either app was running alone the wireless device would remain in the low power state (tens of mW). However, running together they force the wireless device into the high power state (hundreds of mW). It isn't immediately obvious how to divide up power between the apps in a fair way that is meaningful to users and developers using this app.

To solve this problem hardware states are simulated for each application as though the app was running alone. With predicted hardware states it is then possible to calculate how much power each app is would be using. In our example earlier each app would be charged for using the wireless device in the low power state as neither would have caused the transition to the high power state by itself.

Related tools and works

Related tools:

- Android built-in battery info: Settings -> About Phone -> Battery use. Easy to use, yet it does not provide fine-grained power profile.

- Trepn Profiler: A power profiler provided by Qualcomm that takes advantage of hardware sensor to provide accurate power profile. It can be used only on limited number of phones, e.g. the ones using Snapdragon MDP.

- PowerTop: PowerTop is a popular command line tool that shows the power consumption of the system including desktop and laptop. It is only available for X86-based processor and not available for mobile devices so far.

Related works:

- Mian Dong and Lin Zhong, "Self-constructive, high-rate energy modeling for battery-powered mobile systems," in Proc. ACM/USENIX Int. Conf. Mobile Systems, Applications, and Services (MobiSys), June 2011. [PDF]

Email: powertutor at umich dot edu

News!

Release PowerTutor source Oct. 9th, 2011

Updated Power Tutor Oct. 11th, 2010 (changelist)

Updated Power Tutor to 1.3 Oct. 9th, 2010 (changelist)

Updated Power Tutor Aug. 23rd, 2010 (changelist)

Updated Power Tutor to 1.2 Aug. 1st, 2010 (changelist)

Released Power Tutor Nov. 17th, 2009