A Light Weight Energy Profiling Approach on Android Devices

Peixing Yang [1, a], Di Zhou [2, b]

[1] College of Computer Science and Technology , Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

[2] College of Computer Science and Technology , Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

[a]email: pxyang115@163.com, [b]email: dzhou0216@163.com

Keywords: Energy Consumption; Energy Measurement; Mobile Apps

Abstract. In recent years, the popularity of smartphones increases rapidly. Although the smartphone market is a sharp increase, its utility is still limited to the battery life. Therefore, much research effort has been made in order to understand the power consumption of the application to run on mobile devices. Energy conservation can extend battery lifetime, however, any energy management policy requires accurate prediction of energy consumption, which is impossible without reliable energy measurement and estimation methods and tools. We present an analysis of the energy measurement methodologies and proposed a light weight approach to profile smartphones power consumption which can integrate energy data into the development and testing process. Research results are expected to solve some problems in the development of energy saving for Android applications in order to make contribution to the realization of green computing.

Introduction

In portable electronic devices, such as laptop, tablet or smartphone, energy is an important restriction factor, because 1) mobile devices are not always connected to a fixed power supply, but rather are supplied from batteries; 2) complex hardware components and mobile applications need more energy; 3) the portability requirement imposes severe constraints on the size and weight of a hand-held system, including its batteries [1]. In terms of computation capacity, most of the current generation smartphones are double or even three orders of magnitudes better than they are ten years age. Modern intelligent mobile phone battery life, however, only two or three times increased. In the case of the battery's physical storage capacity gradually tends to the limit, how to save energy from the software is an urgent problem to be solved. Energy saving allows to extend battery life and to include more services that can be provided by a device for the same battery capacity. However, any energy management policy requires accurate prediction of energy consumption, which is impossible without reliable energy measurement and estimation methods and tools.

Energy measurement can be performed at hardware, instruction-level, OS, algorithmic, data, application software and user levels. Previous studies on the measurement of energy consumption in mobile devices are based on 1) hardware based measurements, 2) energy profiling software running on the smartphone, and 3) external measurement software running on a computer connected to a mobile device.

In this paper, we analyze the methods of energy consumption measurement and summarized. According to the advantages and disadvantages, we proposed a light weight approach to measure smartphones power consumption which can integrate energy data into the development and testing process considering the measurement effect, difficulty to get started, conditions of use etc. This new approach can help developer and test personnel more convenient and easier to get the energy consumption data of Android smartphones.

Energy measurement methodologies

Hardware based methodology. The mobile device is treated as a black box and energy measurement is simplified to obtain the supply voltage and current on the battery terminals. The measurement system consists of a device under test, a hardware data acquisition system (usually a measurement resistor, a digital multimeter and a data acquisition board), and a computer [2].

Internal profiling software based methodology. The external equipment is not needed; however, energy profiling software, such as PowerTutor, is necessary, which may only be applied to some certain mobile devices.

External software based methodology. The mobile device is connected to a computer as its dedicated software queries a device for its energy related characteristics. This allows to measure energy consumption without interacting directly with the smartphone, which can avoid pressing any keys and having backlight of the display turned on and reduce misleading results.

The advantages and disadvantages of the energy models are summarized in Table 1.

Model	Advantage	Disadvantage
Hardware	Most reliable measurement	Additional hardware is required; Loss of noise and power measurement; only be applied to some certain types of batteries;
Internal profiling software	No need for additional hardware device computer	Additional power consumption caused by energy logging application; restriction to certain devices;
External software	Direct interaction with smartphone and additional applications are not required	Restricted by functions provided by OS API; Loss of data communication;

Table 1. Evaluation of advantages and disadvantages of energy measurement models

For most Android software developers, they are very difficult to have the condition, and will not purchase a dedicated meter specifically for the monitor of mobile phone energy consumption. What's more, some professional knowledge is needed when they want to connect smartphone and the dedicated meter because of the good sealing of modern devices. Although the hardware based measurement can get a reliable result, it is much less practical than the energy internal (profiling) software. While the external software based methodology do not need special device but only computer, the number of computer may not be one. For example, Flinn and Satyanarayanan [3] use PowerScope for measuring energy consumption. The measurement system consists of two computers: the profiling computer and the data-collection computer. Thus the external software based methodology may suffer the same problem like hardware-based methodology.

The internal energy profiling software does not require an external device, which is a very lightweight method for developers and testing personnel. Although there have been a lot of research on the energy consumption, but developers have not yet attracted enough attention to it. In the small part of developers who are aware of the energy consumption issue, if they are trapped in the absence of external equipment to obtain energy consumption, they are likely to give up energy optimization. Therefore, internal energy profiling software is not only the most widely used method, but also the most likely approach to make contribution to the optimization of software energy consumption.

Our Light Weight Energy Profiling Approach

Although the internal energy profiling software has advantages such as no need of external equipment and high practicality, it will be limited by hardware and operation system. Due to the open source nature of Android system, almost all Android mobile phone manufacturers customize Android system on their own hardware devices, which makes it difficult to have generic internal energy profiling software that can be adapted to various brand models of mobile phones.

PowerTutor [4] is an online power estimation system that has been implemented for Android

platform smartphones. This tool has been supported by Google, but it stopped update recent years due to unknown reasons. PowerTutor cannot provide accurate, real-time power consumption estimates for power-intensive hard-ware components on the modern smart phone although it ever had this ability according to the official introduction.

In this paper, we choose TrepnProfiler [5] as our internal energy profiling software. The TrepnProfiler is a diagnostic tool designed to measure and profile the power consumption of Android applications on mobile devices. The profiler can track power, CPU, memory, and network usage to help better understand and optimize application power performance. When used with Mobile Development Platform devices based on the Qualcomm Snapdragon Processor, the profiler is capable of displaying power consumption of individual hardware components.

Since the profiler is also an Android application, it will generate extra energy consumption. So we need to filter out the extra part of energy consumption when analyze the target application. Baselining Interval is the amount of time TrepnProfiler collects baseline data before the start of profiling. The baseline timer only runs if Show Deltas is enabled and a power statistic is selected in the Data Points preferences. This improves how accurately Trepn can estimate the actual power usage, if available, of a profiled application. A larger baselining interval provides a more accurate estimate. This option is disabled if Show Deltas is not turned on.

We measure a given event tracking application of power consumption. However, it is impossible to detect defects in an application entirely by analyzing its power tracking. For example, consider a scenario: two programs P1 and P2 have similar power consumption. Program P1, however, have a higher system resources (such as CPU), compared to P2.In this case, the procedure P1 more energy-saving than P2 plan. Therefore, in order to accurately detect the low efficiency of energy, it is important to define appropriate measurement system resource utilization.

Most Android smartphones are shipped with a XML file (usually named as power_profile.xml) containing the average power consumption ratings for the hardware components in the device. The example is shown in Figure 1.

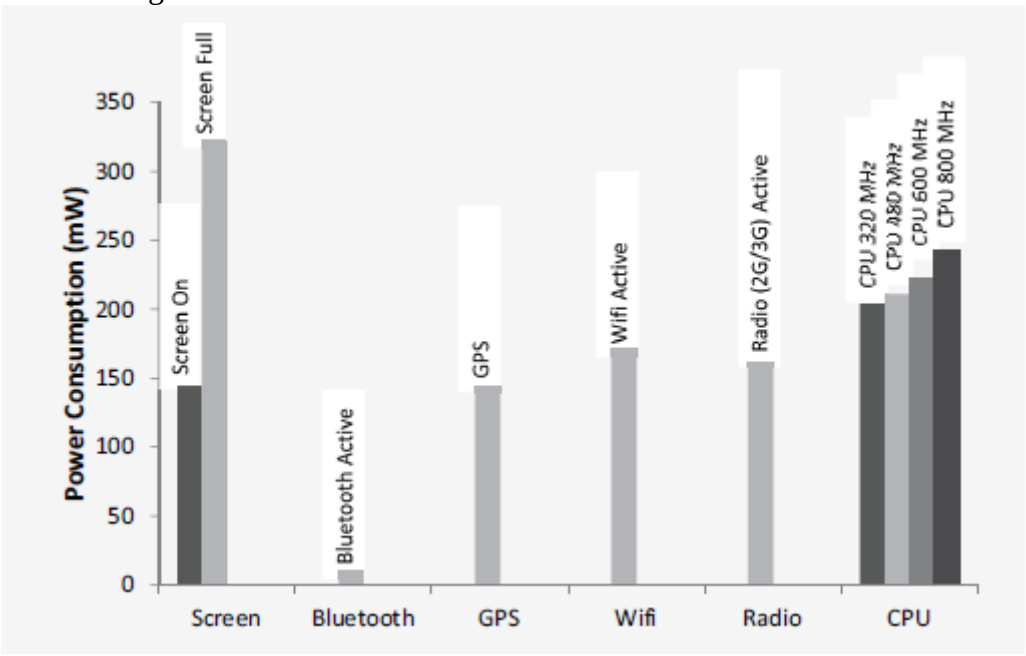

Figure 1.Power profile for a smartphone

So that Utilization (U) can be defined as the weighted sum of utilization rates of all major power consuming hardware components in a device, over a given period of time.

$$\text{Utilization} = U_{Screen} + U_{CPU} + U_{WiFi} + U_{Radio} + U_{GPS} \quad (1)$$

$$U_x = W_x \bullet Load_x, \quad x \in \{\text{Screen, CPU, WiFi, Radio, GPS}\}$$

For a hardware component x, the $Load_x$ represents the average amount of computational work performed by the hardware component x over a given period of time. U_x represents the utilization of hardware component x. Utilization of a component x is directly proportional to its $Load_x$. For any component x, the value of W_x is computed from the power profile. Specifically, the value of

W_x is normalized such that W_x for the most power consuming component is 1.

Energy Utilization (E/U) ratio is the measure of energy-inefficiency of an application for a given time period. Energy utilization rate can be judged from the E/U ratio and then guide the optimization of energy consumption. For instance, if E/U ratio of an application is high, it implies that the energy consumption is high, while utilization is low. Therefore, a high E/U ratio indicates that the application is energy-inefficient [6].

Then a key problem is that a test case may have a lot of operations, how can we correspond different energy consumption data to its operation? For example, say the application being profiled is communicating with a web server and a local SQLite database to get/send data. If the application is performing several requests and a spike in power consumption appears, it may not be clear which task in the application caused this consumption spike. The profiler is ability to capture Android intents from an application and allows developers and testing personnel to view the application state with the other selected data points (power values, CPU loads, etc.), which can be extremely valuable in diagnosing a problem in the application. Example for energy consumption data with application state in the profile is shown in Figure 2.

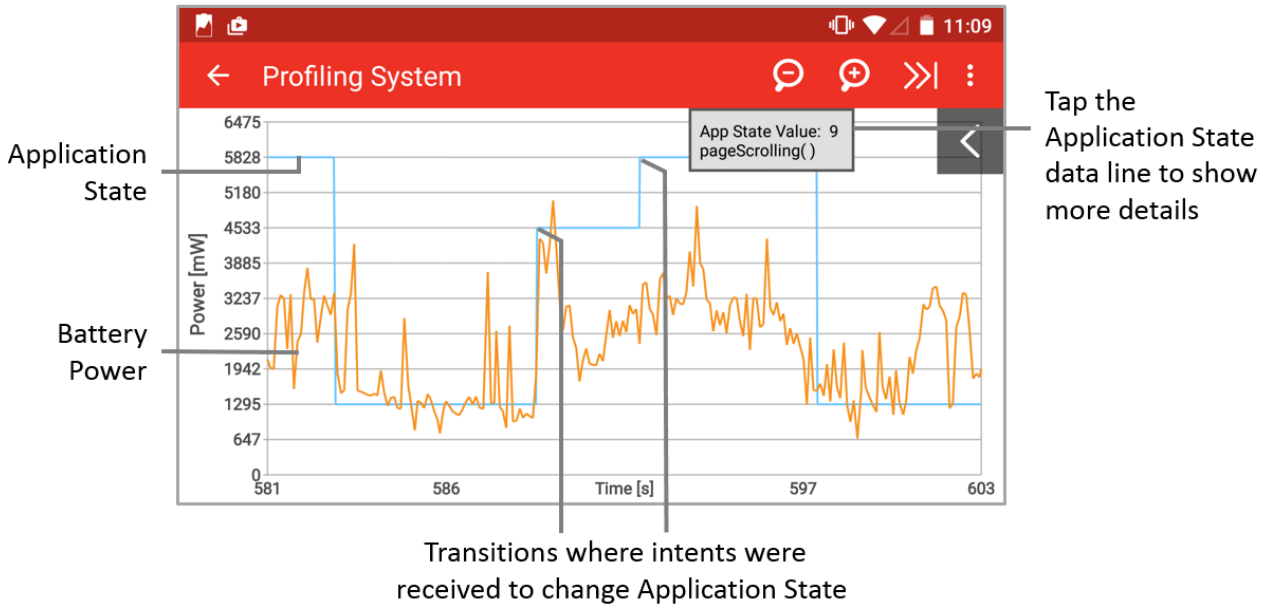

Figure 2. Energy consumption data with application state in the profile

For developers, they can add the code snippet directly in the key operation code of their own program. For testing personnel, they can use soot to instrument the program under test, or add code to the test script such as MokeyRunner and Robotium. The applicable code for a developer-assigned state with a state description is shown as Figure 3:

```
1   Intent stateUpdate = new Intent("com.quicinc.Trepn.UpdateAppState");
2   stateUpdate.putExtra("com.quicinc.Trepn.UpdateAppState.Value", <int_value>);
3   stateUpdate.putExtra("com.quicinc.Trepn.UpdateAppState.Value.Desc", <string_value>);
4   sendBroadcast(stateUpdate);
```

Figure 3. Android intent for Trepn to capture

For example, we modify a Robotium test script for an application so that it can fit our proposed approach. We insert the code fragment is just like follows in Figure4:

```
1     private Intent stateUpdate;
2     private static int state=1;
3     public BroadCast() {
4         this.stateUpdate = new Intent("com.quicinc.Trepn.UpdateAppState");
5     }
6     public void injectInteraction(Activity theActivity, View v, String interactionType, String value) {
7         ComponentName componentName = theActivity.getComponentName();
8         String activityDesc = "activity=" + componentName.getClassName();
9         String eventDesc = "";
10        if (v == null) {
11            eventDesc = "ViewId=null" + ", interactionType=" + interactionType  + ", value=" + value;
12        } else {
13            eventDesc = "ViewId=" + v.getId() + ", interactionType=" + interactionType + ", value=" + value;
14        }
15        String desc = activityDesc + "," + eventDesc;
16        stateUpdate.putExtra("com.quicinc.Trepn.UpdateAppState.Value", state++);
17        stateUpdate.putExtra("com.quicinc.Trepn.UpdateAppState.Value.Desc", desc);
18        theActivity.sendBroadcast(stateUpdate);
19    }
```

Figure 4. Our instrument code in an Android application

Test result can be stored in the form of a table to facilitate future analysis. Energy consumption data in the table as shown in Figure5:

A	B	C	D	E	F	G	H	I	J	K
App	Package	Start Tir	Duration	Filename						
System		Oct 4, 20	2 min 33	aagtl_26_30.db						
Time [ms	Battery S	Time [ms	Battery F	Battery F	Time [ms	Applicati	Description			
21711	0	21821	922261	71171	21419	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4
21814	0	21920	733578	0	21510	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4
21916	0	22020	1266066	414976	21611	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4
22016	0	22122	796141	0	21712	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4
22117	0	22223	793639	0	21816	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4
22219	0	22324	726731	0	21916	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4
22321	0	22428	644736	0	22016	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4
22423	0	22559	1606660	755570	22118	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4
22555	0	22625	1650059	798969	22220	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4
22623	0	22727	1189346	338256	22321	4	activity=com.zoffcc.applications.aagtl.aagtl	ViewId=16908916	interactionType=longClickListItem	value=4

Figure 5. Energy consumption data in the table

Conclusion

There is no widely applicable method to obtain the energy consumption of the smart phone device is mainly because of the wide variability of mobile device platforms, operating systems, models, ways to connect to external devices/computer and APIs. In this paper, we have discussed different methods to measure energy consumption and summarized the advantages and disadvantages. We have also proposed a light weight approach to profile smartphones power consumption which can integrate energy data into the development and testing process.

References

[1] Damaševičius R, Štuikys, Vytautas, Toldinas J. Methods for Measurement of Energy Consumption in Mobile Devices[J]. Metrology & Measurement Systems, 2013, 20(3):419-430.

[2] Krejcar O. Testing the Battery Life of Mobile Phones and PDAs[C]// Proceedings of International Conference on Software and Computer Applications（ICSCA 2011）2011.

[3] Flinn J., Satyanarayanan M. (1999). PowerScope: a tool for profiling the energy usage of mobile applications. In Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications (WMCSA '99). IEEE Computer Society, Washington, DC, USA, 2.

[4] Zhang L, Tiwana B, Qian Z, et al. Accurate online power estimation and automatic battery

behavior based power model generation for smartphones[C]// Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesisACM, 2010:105 - 114.

[5] Trepn Power Profiler. http://developer.qualcomm.com/Trepn

[6] Banerjee A, Chong L K, Chattopadhyay S, et al. Detecting energy bugs and hotspots in mobile apps[C]// Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software EngineeringACM, 2014.