Energy Consumption Measurements as a Basis for Computational Offloading for Android Smartphones

Quang-Huy Nguyen, Johannes Blobel and Falko Dressler
Dept. of Computer Science and Heinz Nixdorf Institute, Paderborn University, Germany
{nguyen,blobel,dressler}@ccs-labs.org

***Abstract*—Computational offloading has been shown being a promising approach to prolong the battery life of smartphones. To come up with energy-efficient offloading strategies, it is crucial to understand how much energy is used not only for local computation but also for network communication. Therefore, reliable methods to measure the energy consumption of the phones are needed as a fundamental basis. In this paper, we evaluate the accuracy of the Android smart battery interface. As a second contribution, we also designed a new, portable and cheap microcontroller-based power monitoring device. We conduct energy measurement experiments for a set of typical applications including local computations as well as WiFi and 3G operations. Our results show that both our device and smart battery interface can be used to estimate the energy consumed by the applications running on the phone. The smart battery interface, however, clearly underestimates in idle status. We see our measurements as an important step towards the development of more accurate offloading strategies for smartphones.**

I. Introduction

In the last decade, we have seen an explosion in the development of smartphones and their applications. Today, modern smartphones are equipped with high-performance processors, large memory, high-resolution displays, and a multitude of sensors and network interfaces. All these capabilities enable smartphones to execute complex tasks like video playback, games playing, image processing, and Internet services. However, one problem remains: the batteries of smartphones have not evolved as fast as their capabilities. This is mainly due to the constraints on size and weight of mobile devices. Therefore, most smartphones cannot operate for a long time and users normally have to charge their phones everyday. And also because of short battery life, the computational capacity, and memory of smartphones are still limited [1].

The development of cloud computing in computational power and storage makes smartphones more scalable [2]. It reduces the hardware limitations and the burden on battery capability of smartphones. Along with a more powerful server, a program for power consumption management on smartphones is necessary. Offloading intensive tasks to the cloud is a straightforward solution to save energy on smartphones and to make use of the capabilities of cloud computing [3], [4]. Even though computational offloading provides high potentials for energy-efficient mobile systems, the feasibility and benefits of this technique depend on each application and network parameters. In general, the problem of offloading tasks involves two questions that need to be answered: (1) can the system benefit from offloading local computations to a more resourceful server, and (2) which tasks or computations should be migrated.

The offloading decisions are usually made by estimating the energy consumed by local part and communication part of the target application. The goal of saving energy only holds if the energy used to send the processing data to the cloud and download the results is less than the energy for local execution. Hence, it is obvious that good understanding of energy consumption on smartphones plays a vital role in computational offloading. In our previous work [5], we presented an overview of a new smart battery interface of Android smartphones for the energy measurements. In this paper, we further investigate its capability with various load characteristics and whether it can be used for accurate decision making. We also developed a portable Microcontroller Unit (MCU)-based measurement device that provides the same accuracy as a fully featured oscilloscope.

Our main contributions can be summarized as follows:

- We investigate the use of Android smart battery interface for energy consumption monitoring;
- we design a new MCU-based power monitoring device, which is flexible, portable, and inexpensive;
- we experimentally examine the accuracy of the smart battery interface and our new device by comparing the results with the measurement data from an oscilloscope.

II. Related Work

A. Computational Offloading

Many researches have been investigating different aspects of computational offloading. One of the first frameworks for offloading was presented in [6]. In the paper, the authors focused on the effects of the Round Trip Time (RTT) in WiFi and 3G networks. Offloading decisions are made based on the profiling of the device, application program and network.

In [7], the authors came up with a toolkit called SmartDiet, which analyzes the application code at the method level. The statistic results help determining which methods have possibilities to offload to the cloud for saving energy. Moreover, the results also provide good information for developers to improve their applications in terms of energy-efficiency.

Energy consumption of each network interface is an essential factor in order to make offloading decision. In [8], the authors experimentally investigated the power consumption of three common network connections on smartphones, namely 2G,

3G, and WiFi. With the same idea, the energy consumption of these network interfaces have been further analyzed in [9]. In these two papers, the authors only relied on some network parameters, such as the transfer size, the packet size and the transmission intervals, to measure the energy consumed by each of interface. In [10], the authors concentrated only on the power consumption of an IEEE 802.11g WLAN, the most popular network interface of smartphones at that time.

Another approach for estimating the energy consumption of GSM, 3G, and WiFi was presented in [11]. According to the paper, the power consumption of 3G and GSM can be characterized by three power states: high power, intermediate power, and idle state. The phone remains in the high power and intermediate power states not only during the data transfer but also as long as the connection is maintained. It only switches to idle after releasing the connection. This helped explaining the high energy consumption of cellular network. The authors also discussed three power states of the WiFi interface including idle, Power Saving Mode (PSM), and active state.

Current works mainly address offloading as a distributed scheduling problem [12]. The fundamental basis, however, remains accurate energy measurements and predictions.

B. Energy Measurement

In general, the power traces can be classified into two groups: hardware-based and software-based. The hardware-based methods utilize external devices connected to smartphones to measure the power consumption. Several existing devices are suggested to measure the voltage across the phone battery. Monsoon's PowerMonitor[1] is a popular power monitoring device used in many works [6], [11], [13]. In [14], [15], the National Instruments sampling boards are used to measure the voltages. In [9], the authors use the Voltech PM1000+ with the sampling frequencies up to 1 Hz. In this paper, we use a Tektronix TDS1012B oscilloscope for sampling the voltage.

Besides the commercial devices, custom-made measurement boards are also suggested. In [16], Battor, a portable power monitor, is presented. In [17], the authors use an Arduino Duemilanove board to sample the voltage. These two approaches utilize the built-in Analog-to-Digital Converter (ADC) in the microcontroller, normally 10 bit, which only provides coarse-grained values of voltage signal. In this paper, we also present our own MCU-based power monitoring device, which supports an external ADC with higher-precision.

The software-based methods use energy-profiling to read the battery statistics. Additionally, some other battery information, such as the remaining battery capacity and the temperature are also logged through the Application Program Interfaces (APIs) provided by the operating system. In [1], the Nokia Energy Profiler is used to trace the power, current, temperature, signal strength, and CPU usage of Nokia smartphone. In [13], the authors develop a power modeling tool called DevScope that is based on the Battery Monitoring Unit (BMU). Similarly, in this paper, we investigate the accuracy and capabilities of Android smart battery interface in energy measurement.

[1]https://www.msoon.com/LabEquipment/PowerMonitor/

III. Measurement Setup

A. Methodology

In theory, the energy consumption E over time T is calculated based on the following equation:

$$E = \int_0^T P(t)\,\mathrm{d}t = \int_0^T U(t)I(t)\,\mathrm{d}t,$$

where $P(t)$, $U(t)$, and $I(t)$ represent instant power, voltage, and current at time t, respectively. In practice, we have to sample voltage and current and numerically integrate their product to achieve the accumulated energy consumption.

B. Android Smart Battery Interface

Most modern Android smartphones support the smart battery interface, which facilitates the measurement of power consumption. This interface can provide a number of power-related properties in /sys/class/power_supply/battery/ on the Android file system, such as status, capacity, voltage_now, current_now, current_average, energy_counter, charge_counter. The actually available properties, however, are platform-specific.

Normally, the biggest difference between battery interfaces lies in the capability of current measurement. This functionality of the battery interface is closely related to a special fuel gauge IC integrated in the hardware of the smartphone. Some fuel gauges only support the battery State Of Charge (SOC) while the others can offer a coulomb counter, which enables to estimate the current. The updating rate of the battery status also varies from different fuel gauges. Table I lists some of common chips that are used in Android devices.

Generally, the intervals for updating power properties are quite coarse, from hundreds of milliseconds to few seconds or even longer. Therefore, the accuracy of the power estimation of this approach is not as high as hardware-based approaches. However, the smart battery interface is still a useful method to measure the energy consumption of smartphones due to its simplicity and availability in most modern Android smartphones. This approach is often the only solution for smartphones with non-removable battery, where it is non-trivial to attach external devices for energy measurements.

C. Oscilloscope

We use a Tektronix TDS1012B oscilloscope for observing the changes of voltage signal of the phone battery over time. This device has three main different operation modes: auto, trigger, and scan mode. For energy measurement, we need to continuously sample the voltage signal in a long period of time.

Table I
Fuel gauge chips of Android devices

Device	Fuel Gauge	Measurements	Current Updating Rate
Galaxy W	MAX17043	SOC	-
Galaxy S3	MAX17047	Current	175.8 ms
LG G3, G4	MAX17048	SOC	-
Nexus 6	MAX17050	Current	175.8 ms
Nexus 10	DS2784	Current	3.5 s

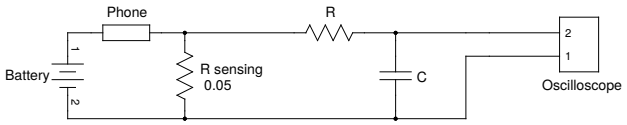


Figure 1. Schematic of the oscilloscope measurement setup.

Therefore, only the scan mode is suitable for our experiments. In this mode, the oscilloscope supports a sampling frequency up to 2.5 kHz. Additionally, the TDS1012B can work with the minimum vertical sensitivity of 2 mV, which enables to distinguish a very small difference in voltage signal level. This allows the oscilloscope to sample the data without the use of additional amplifiers.

In order to measure the current, we insert a high-precision sensing resistor of 0.05 Ω between the terminals of the battery and the smartphone. The current is then calculated by the Ohm's Law according to the voltage drop across the sensing resistor. We intercept the connection between the phone and the battery by two cooper tapes. We utilize both channels of the oscilloscope: One channel is used to acquire the voltage of the battery and the other one is used to sample the voltage drop across the sensing resistor. Figure 1 depicts the schematic of our circuit for current measurement using the oscilloscope.

Typically, there are always short-term fluctuations in voltage signal. With the limitation of the sampling frequency, these quick high spikes could be lost during the acquisition process. Hence, there is a risk of measurement errors on the average values of voltage and current. One possible solution to this problem is to use a RC low pass filter to remove the instantaneously changes in voltage and provide a smoother form of the signal. The cutoff frequency of the RC low pass filter is selected based on the sampling frequency in order to minimize the aliasing effect while not smoothing the voltage signal too much. The relationship between the resistor R, capacitor C, and the cutoff frequency f_c relies on the equation:

$$f_c = \frac{1}{2\pi RC} \ .$$

The TDS1012B is connected to a PC to record the sampling data. We automate the measurement process by developing a *python* program that interfaces with the USBTMC driver on the PC. Our program first sends a series of commands to configure the operation modes for the oscilloscope. Then it periodically sends queries to collect the measurement data.

D. Microcontroller-based Measurement Device

While the measurements with the oscilloscope give precise results, this technique has some drawbacks. First of all, a good oscilloscope is rather expensive. If simultaneous measurements across multiple devices should be conducted, multiple oscilloscopes would have to be available. It is also rather big and requires a mains power supply which makes it unsuitable for mobile measurements. But especially in mobile scenarios like smartphones or sensor networks, energy consumption and therefore energy measurements are of great importance. Finally, it is also necessary to have an oscilloscope with a PC interface in order to record the measured data over a longer period.

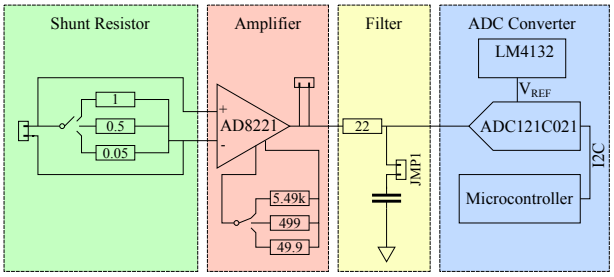


Figure 2. Architecture of the MCU-based measurement board.

Since energy measurements for mobile devices are a recurring technique in many scientific researches, we created a versatile and inexpensive platform to perform current measurements. It's main design goals are: low cost and simple design so it can be rebuild by everyone interested; flexibility to support a wide range of energy measurements; portability and mobility.

Figure 2 shows the architecture of our board. It has a separate power source so the phone battery is not affected. The process of measuring current can be divided into four stages:

1) convert current to voltage using a shunt resistor;
2) amplify the voltage signal;
3) filter the signal (optional);
4) sample the signal.

To make our platform versatile we made each step configurable so that the board can be used for a wide range of measurements. In the following we explain the four stages in more detail.

1) Shunt Resistor: As explained before, we measure current indirectly by measuring the voltage drop across a shunt resistor. The resistance used depends on the current that should be measured and also has an influence on the circuit under test. It should be chosen rather small, so that the additional resistance does not affect the behavior of the analyzed circuit. But especially for very small currents, the shunt resistor must be large enough, so that a measurable voltage falls off. To allow a wide range of measurements, we placed multiple shunt resistors on our board, which can be chosen via a switch. If need arises for a different resistor, the board also has two female pin headers, where a custom resistor can be placed.

2) Amplifier: For small currents the voltage across the shunt resistor can be very small, but the resistance can not be made arbitrarily high, since this can negatively affect the working of the circuit under test. Therefore, we added a high-precision, low power instrumentation amplifier (AD8221) with variable gain to our board, to amplify the voltage to a more useful value. To ensure a high flexibility of our board, the gain can also be selected with a switch. Apart from fixed resistors that determine the gain of the amplifier, the user can also add a custom gain resistor, to select an appropriate gain.

3) Filter: The integration of the amplified signal with an RC filter can be used to filter out short spikes in the signal. But

not in all scenarios this filtering is desired: our board has an inbuilt RC filter that can be enabled with a jumper if desired.

4) ADC Converter: The last step in current measurements is the sampling of the (amplified) voltage with an ADC. While microcontrollers typically come with built in ADCs, the resolution and precision is not sufficient for our needs. Therefore, we added a 12 bit ADC (ADC121C021) to our board and a high precision voltage reference (LM4132) to get accurate readings.

The intermediate signals can be accessed via pin headers if needed. The board can then also be used just to amplify a given signal and sample it with an oscilloscope or to read in a given signal without amplifying it. To make the board easy to build we used only relatively large SMD components and made the PCB one-sided. This makes it possible to etch and solder the PCB by hand. Overall, the board was designed in a way that allows the user to use it in a very flexible way, therefore making it a good tool for many application scenarios.

5) Calibration: The ADC outputs raw values $x_{raw} \in [0, 4095]$. This can be transformed to a voltage U_{adc} by

$$U_{adc} = \frac{x_{raw}}{4095} \times U_{ref} \ , \tag{1}$$

where $U_{ref} = 2.5\,\mathrm{V}$ is the reference voltage from the LM4132. Since this is the amplified voltage, we have to calculate the original voltage U_{shunt} dropped at the shunt resistor.

$$U_{shunt} = \frac{U_{adc}}{G_{amp}} - V_{offset} \ , \tag{2}$$

where V_{offset} is the input offset voltage of the amplifier and G_{amp} is the amplification factor. With this value and the resistor R_{shunt} we can calculate the current by:

$$I = \frac{U_{shunt}}{R_{shunt}} \tag{3}$$

The main error source is the value of the shunt resistor and the resistor used to determine the gain and the input offset voltage of the amplifier. To get precise measurements it is therefore necessary to calibrate the measurement device before using it. This has been done prior to the measurements with the help of the oscilloscope as a baseline.

IV. Experiment Description

In this section, we present our experiments in order to evaluate the accuracy of three measurement methods that are mentioned in Section III. The setup of the experiments is shown in Figure 3. In this setup, we utilize a PANTECH VEGA S5 smartphone running Android v4.1.2 which supports a smart battery interface with the capability of current estimation. We perform our measurements with two groups of applications: local computation and network communication. In detail, four different applications are used for the experiments including:

1) compute matrix multiplication locally on the phone,
2) play a video stored on the phone memory,
3) upload data to a server, and
4) download data from a server.

For local computations, first, we developed an application to perform the multiplication of two matrices with the dimension 500×500 and 500×100 (using a rather naïve implementation). The matrix input data is read from two text files saved in the local storage. The result matrix is written again into a text file and stored in the phone memory. During the execution, the screen of the smartphone is turned off to avoid side effects on the power trace. Secondly, we used a local video application, which plays an MP4 file. The screen is kept on at 50 % brightness during the video playing. We also performed experiments for video streaming over WiFi and 3G. However, because the measurement results depend on many factors including network parameters and caching effect, we do not further discuss the results in this paper.

For the file transfer, we developed an application for downloading and uploading a file of 3.5 MB from and to our server. This file has the same size as the input data file that we use in the matrix multiplication application. The phone display is switched off again for this type of applications. We performed the measurements using both WiFi and 3G – only one of network interfaces is enabled while the application is running. To minimize the effect of network parameters to the energy measurement, we configure the smartphone to connect to a dedicated WiFi Accsess Point (AP), which guarantees perfect link quality. We, however, have no control on network parameters for 3G applications.

For all experiments, we added delays at the beginning and ending of the applications to track the power consumption behavior in idle state. We repeated each application 10 times for improving statistical confidence. All three measurement methods were used at the same time for each application. For the software-based approach that employs the battery interface, we develop an Android service running in the background of smartphones which reads the current status of the phone battery from Android file system every 0.5 s. The target application starts this service at the beginning of the execution and stops it after finishing the test. The service stores the readings in

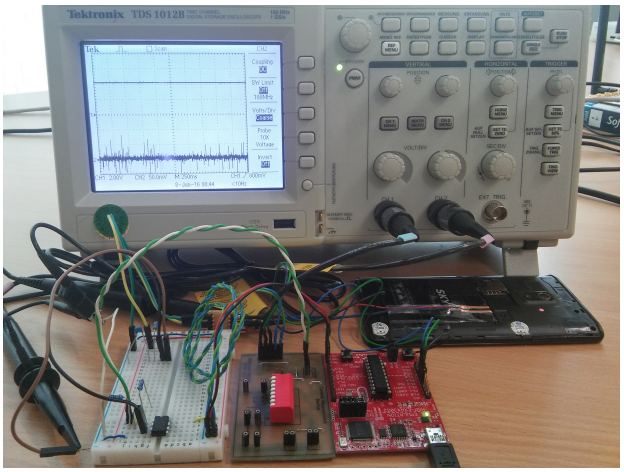


Figure 3. Photograph of the used measurement setup.

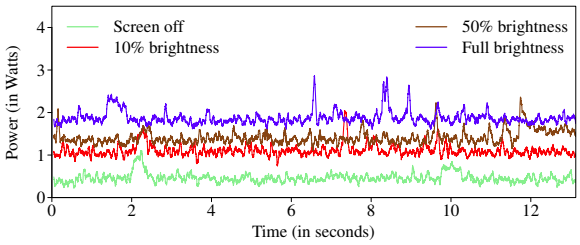
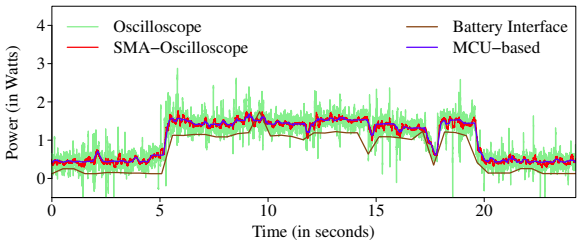
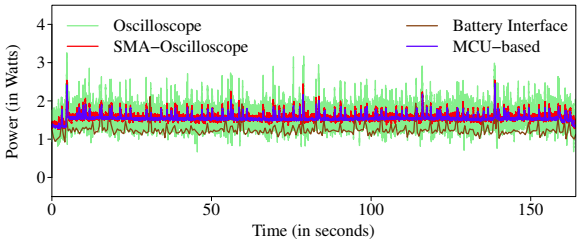


Figure 4. Power trace of smartphones in idle state.



(a) Matrix multiplication



(b) Local video

Figure 5. Power trace for local computations.

the local memory of the smartphones as log files. The logging data includes the following parameters: time points of data reading, voltage_now, and current_now.

For the hardware-based methods, we use both the oscilloscope and our own measurement hardware to sample the voltage signal of the phone battery and the voltage drop across the sensing resistor. Since the input impedances of the amplifier and the oscilloscope are very large, there is no problem when we use both the oscilloscope and our own hardware to measure the same voltage signal. The sampling frequency of the oscilloscope and our board is 1 kHz and 0.25 kHz, respectively. Both of these hardware-based techniques are started at the same time using a script running on the PC. We also perform the calibration for our power monitoring device using the oscilloscope before running experiments.

Obviously, Android is a highly complex system with a lot of activities and background services. In order to improve the accuracy of our measurement, we disable the power management and all unnecessary programs, services and unused network interfaces on the smartphone during the execution of the main task. For the same reason, we also deactivated the network connections while performing local computations. In certain conditions, the smartphone could switch to standby mode or sleep mode, which leads to the instability of measurement results – we applied a wake lock mechanism to keep the CPU of smartphone staying ON during the operation. And, most importantly, we automated the measurement process so that no interaction with the smartphone is needed.

V. Results and Discussion

We investigated the accuracy of three measurement methods based on their experimental results. The results from the oscilloscope are used to provide the ground trough baseline for the other two methods due to its higher sampling rate and lower error of sampled data values.

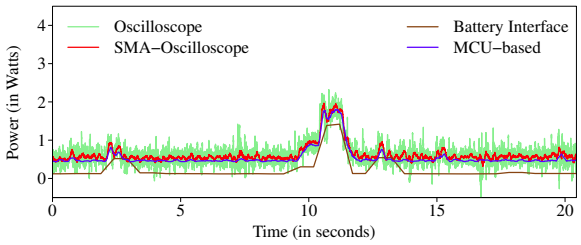
A. Power Trace

To assess the similarity among these power traces, we apply the Simple Moving Average (SMA) to the raw data from oscilloscope. The SMA creates a series of average voltage values from the sampling data, which can more easily be compared to the power traces provided by the battery interface and our new platform. For the majority of the applications, most of energy is consumed by the phone screen. Figure 4 gives the overview of power consumption when the phone is in idle state. It is shown that switching the display of the device from off to full brightness increases the power consumption of the phone by nearly 1.5 W.

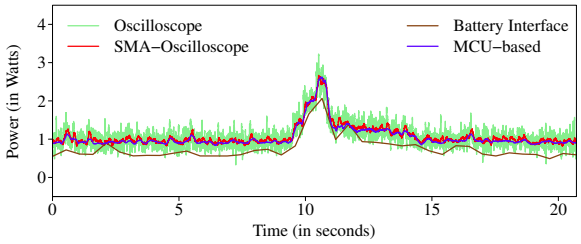
Figure 5 shows the power traces for local computations on the smartphone. The result for matrix multiplication application is shown in Figure 5a. The average executing time for computing matrix multiplication is approximate 14.7 s. The observation of power consumption for video playing application is depicted in Figure 5b. As can be seen, the screen is the main source of power drain and the playback algorithm does not add much to the energy consumption.

In Figure 6, we show the power consumed for downloading and uploading file from and to a server over WiFi, respectively. Even though there is a slight difference in download and upload speed, the overall execution time is nearly the same. The reason is that the size of transfer files is quite small, which only requires a little time for sending and receiving data. Most of the time is spent on the connection setup for communication. The average time for file transfer using WiFi in our applications is approximate 1.3 s.

Finally, Figure 7 shows the power consumption behavior of 3G communication. We notice that the phone only spends approximate 5 s to download a file of 3.5 MB from server over 3G (cf. Figure 7a), while it requires more than 20 s for uploading the same file to server (cf. Figure 7b). This is due to the difference in download and upload speed of 3G connections. Besides that, we also can see that the phone does not switch directly to idle state after finishing the operation, but remains in an intermediate power state in a specified period of time. This is handled by Radio Resource Controller (RRC) in cellular network [18]. In the 3G communication, the RRC idle mode has the lowest power consumption. The RRC connected mode

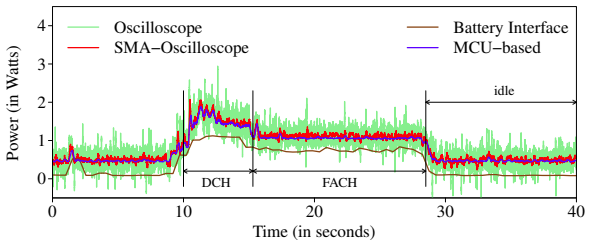


(a) Download

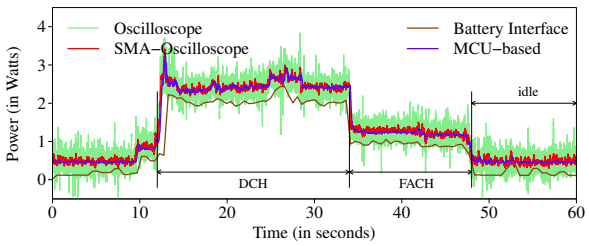


(b) Upload

Figure 6. Power trace for file transfer over WiFi.



(a) Download



(b) Upload

Figure 7. Power trace for file transfer over 3G.

is divided into two sub-states: the high-power state CELL_DCH and the intermediate power state CELL_FACH in which data are carried through a Dedicated Channel (DCH) and a Forward Access Channel (FACH), respectively. This observation also gives us more accurate estimation of energy consumption.

Overall, we can see that our MCU-based measurements are matching the SMA data from the oscilloscope very well. However, in all the shown power traces, the Android battery interface seems to underestimate the power consumption of the phone compared to oscilloscope and our monitoring device. The measurements on different smartphones need to be done to

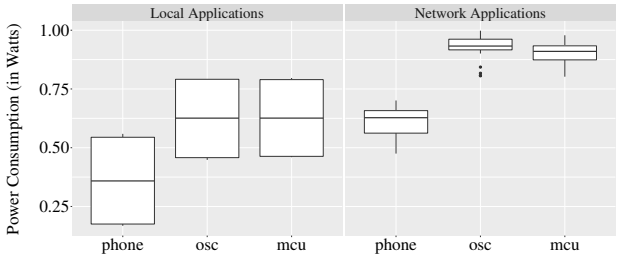


Figure 8. Statistics of power consumption in idle mode.

validate this finding. We will report on the estimation quality under load later but first examine the idle state more closely.

B. Idle State Power Consumption

As a base line, we determine the substantial power consumption of the phone in idle state by tracking the power during the delayed period at the beginning of the testing application. We did so for each experiment individually. When no application is running, the power consumed by the phone is around 0.3 W and increases to 0.55 W when the logging service is enabled.

In Figure 8, we can see the difference in power consumed by the phone in idle mode among various applications. The first observation is that the MCU-based approach is exactly matching the results obtained by the oscilloscope.

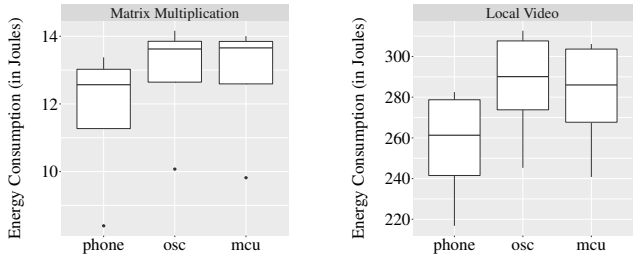
We further have to distinguish between local computations and communication experiments. For the local computations, all of network interfaces are disabled, which clearly impacts the idle power as shown in Figure 8. The average idle power is 0.36 W and 0.62 W measured by the battery interface and the MCU/oscilloscope, respectively. This quantity is estimated to be approximate 0.6 W by the battery interface and 0.9 W by the MCU/oscilloscope when the WiFi or 3G network is activated.

For all further investigations, we simple subtracted the idle power from the total power consumed by the smartphone when the applications are running. This mechanism provides higher accuracy of energy consumption estimation for each computation and network operation on smartphones.

C. Total Energy Consumption

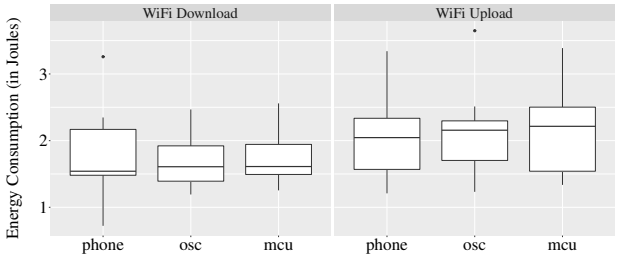
We now explore the total energy consumption as measured by all three methods. As a first result, the distribution of the measurements for local computation is depicted in Figure 9. Comparing the results in Figures 9a and 9b, it is clear that the energy consumption of video playing application is higher than the matrix multiplication due to the use of the phone screen. The average energy consumed by the phone display with 50 % brightness in 154 s is 215.6 J, which is nearly 75 % of overall energy consumption of the video playing application. The smart battery interface reports less energy being consumed compared to the MCU and the oscilloscope.

Figure 10 shows the experimental statistics on energy consumption of our file transfer applications using WiFi and 3G communication. Generally, smartphones use much more

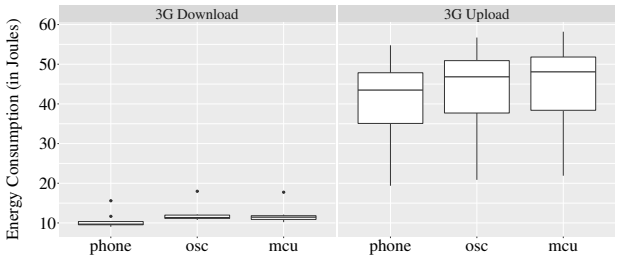


(a) Matrix multiplication (b) Local video

Figure 9. Statistics of energy consumption for local computations.



(a) WiFi



(b) 3G

Figure 10. Statistics of energy consumption for network communication.

energy (at least one order of magnitude) for 3G communication compared to WiFi. The average energy consumption for sending and receiving a file of 3.5 MB over WiFi is 2.12 J and 1.62 J, respectively. These quantities are 43.09 J and 12.13 J for the 3G connection. Moreover, it can be seen that downloading a file from a server is less expensive than uploading for both network interfaces, however, the difference in WiFi communication is not so significant. In our measurement, the WiFi connection quality is maintained in very good condition so we do not see the effect of this parameter on energy consumption of the smartphone.

Another interesting observation is that the measurement results are more stable for short-time running applications such as the matrix multiplication, WiFi communication, and 3G download. For long-time running applications like video playing or 3G upload, the energy evaluation suffer higher ratio of interference from the background services and the impacts of operating conditions. Therefore, the differing variation in measurement results of these two classes of application is straightforward.

Figures 9 and 10 also reveal some noticeable quantitative aspects. It is obvious that smartphones spend much more time and energy on local heavy tasks. In the opposite, it only needs a little time for sending and receiving data to and from the server over the WiFi, so the energy required for executing these operations is quite small. This brings the possibility of saving energy using computational offloading. But there are still many issues related to the network communication that need to be considered in detail, especially, the network selection algorithm.

D. Overall Accuracy Comparison

In order to assess the accuracy of our power monitoring device and the Android battery interface method, we compute the errors between the results of these two measurement techniques with the ones from the oscilloscope for each application. Table II summarizes the average energy measurement errors of the two methods in all experimental applications.

The results indicate that our MCU-based device provides very high accuracy in energy consumption estimation. For the smart battery interface, even though there is a limitation in updating rate, the measurement results are still acceptable in most scenarios. The high average error rates in local computation and 3G communication come from the long-running time of these applications.

Table II
ENERGY MEASUREMENT ACCURACY.

Application	Error rate (avg.)	
	Our device	Battery interface
Matrix multiplication	0.80 %	9.43 %
Local video	1.77 %	10.47 %
WiFi download	2.51 %	3.23 %
WiFi upload	1.55 %	2.37 %
3G download	1.33 %	6.25 %
3G upload	2.53 %	8.13 %

Still, the smart battery interface provides an excellent anchor for on the fly measurements and decision taking on the smartphone. More accurate measurements, e.g., using our portable device, can help calibrating the system for different hardware architectures.

VI. CONCLUSION AND FUTURE WORK

In this paper, we studied the use of Android smart battery interface and our own MCU-based power monitoring equipment for measuring the energy consumption of Android smartphones. We employed an oscilloscope in parallel with these two approaches to validate the measurement results. Using the power traces provided by three methods, we were able to analyze the power consumption behavior and determine the energy used by many common applications running on smartphones. Based on the experimental results, we examined the correlation between local computation and network communication in terms of energy dissipation. We also compared the energy cost for data transfer over two popular wireless technologies, namely WiFi and 3G, which demonstrates the advantages of WiFi technology.

We believe that our work forms a fundamental basis for better understanding of energy consumption of smartphones, which is essential for developing energy-efficient offloading algorithms. It is, however, necessary to deal with more general applications in different operation conditions in order to enhance the accuracy of our methods. Additionally, one of the most important aspects in task offloading is the network communication. The technologies, protocols and conditions of the network have great impacts on the power dissipation of smartphones. So, energy consumption models are needed (particularly for upcoming LTE Advanced and 5G technologies) to present the relationship between these parameters and the energy consumption during the offloading process, and to assist making offloading decision.

REFERENCES

[1] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer, "Survey on Energy Consumption Entities on the Smartphone Platform," in *75th IEEE Vehicular Technology Conference Fall (VTC2011-Spring)*, Budapest, Hungary: IEEE, May 2011, pp. 1–6.

[2] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84 –106, 2013.

[3] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, 129–140, 2013.

[4] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, "When mobile terminals meet the cloud: computation offloading as the bridge," *IEEE Network*, vol. 27, no. 5, pp. 28–33, Sep. 2013.

[5] Q.-H. Nguyen and F. Dressler, "The Accuracy of Android Energy Measurements for Offloading Computational Expensive Tasks," in *17th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2016), Poster Session*, Paderborn, Germany: ACM, 2016, pp. 393–394.

[6] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making Smartphones Last Longer with Code Offload," in *8th International Conference on Mobile Systems, Applications and Services (MobiSys 2010)*, San Francisco, CA: ACM, Jun. 2010, pp. 49–62.

[7] A. Saarinen, M. Siekkinen, Y. Xiao, J. K. Nurminen, M. Kemppainen, and P. Hui, "SmartDiet: offloading popular apps to save energy," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 42, no. 4, pp. 297–298, Aug. 2012.

[8] L. Wang and J. Manner, "Energy Consumption Analysis of WLAN, 2G and 3G interfaces," in *IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing (GreenCom-CPSCom 2010)*, Hangzhou, China: IEEE, Dec. 2010, pp. 300–307.

[9] M. Segata, B. Bloessl, C. Sommer, and F. Dressler, "Towards Energy Efficient Smart Phone Applications: Energy Models for Offloading Tasks into the Cloud," in *IEEE International Conference on Communications (ICC 2014)*, Sydney, Australia: IEEE, Jun. 2014, pp. 2394–2399.

[10] Y. Xiao, P. Savolainen, A. Karppanen, M. Siekkinen, and A. Ylä-Jääski, "Practical Power Modeling of Data Transmission over 802.11g for Wireless Applications," in *1st International Conference on Energy-Efficient Computing and Networking (e-Energy 2010)*, Passau, Germany: ACM, Apr. 2010, pp. 75–84.

[11] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in *9th ACM SIGCOMM Conference on Internet Measurement (IMC 2009)*, Chicago, IL: ACM, Nov. 2009, pp. 280–293.

[12] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-Efficient Dynamic Offloading and Resource Scheduling in Mobile Cloud Computing," in *35th IEEE Conference on Computer Communications (INFOCOM 2016)*, San Francisco, CA: IEEE, Apr. 2016.

[13] W. Jung, C. Kang, C. Yoon, D. Kim, and H. Cha, "DevScope: A Nonintrusive and Online Power Analysis Tool for Smartphone Hardware Components," in *8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '12)*, Tampere, Finland: ACM, Oct. 2012, pp. 353–362.

[14] A. Rice and S. Hay, "Decomposing Power Measurements for Mobile Devices," in *IEEE International Conference on Pervasive Computing and Communications (PerCom 2010)*, Mannheim, Germany: IEEE, Apr. 2010, pp. 70–78.

[15] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," in *USENIX Annual Technical Conference 2010 (USENIX ATC 2010)*, Boston, MA: USENIX, Jun. 2010, pp. 21–21.

[16] A. Schulman, T. Schmid, P. Dutta, and N. Spring, "Phone Power Monitoring with BattOr," in *17th ACM International Conference on Mobile Computing and Networking (MobiCom 2011), Demo Session*, Las Vegas, NV: IEEE, Sep. 2011.

[17] R. Trestian, A.-N. Moldovan, O. Ormond, and G.-M. Muntean, "Energy consumption analysis of video streaming to Android mobile devices," in *18th IEEE/IFIP Network Operations & Management Symposium (NOMS 2012)*, Maui, HI: IEEE, Apr. 2012, pp. 444–452.

[18] P. H. J. Perälä, A. Barbuzzi, G. Boggia, and K. Pentikousis, "Theory and Practice of RRC State Transitions in UMTS Networks," in *IEEE Global Telecommunications Conference (GLOBECOM 2009): 5th IEEE Broadband Wireless Access Workshop*, Honolulu, HI, Nov. 2009.