Publication II

Yu Xiao, Rijubrata Bhaumik, Zhirong Yang, Matti Siekkinen, Petri Savolainen, and Antti Ylä-Jääski. A System-Level Model for Runtime Power Estimation on Mobile Devices. In *Proceedings of 2010 IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing*, Hangzhou, China, 27-34, December 2010.

2010 IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing

A System-level Model for Runtime Power Estimation on Mobile Devices

Yu Xiao[1], Rijubrata Bhaumik[1], Zhirong Yang[1], Matti Siekkinen[1], Petri Savolainen[2], Antti Ylä-Jääski[1]
[1]School of Science and Technology, Aalto University
[2]Helsinki Institute for Information Technology (HIIT)/University of Helsinki
Espoo, Finland
e-mail: {yu.xiao, rbhaumik, zhirong.yang, matti.siekkinen}@tkk.fi, petri.savolainen@hiit.fi, antti.yla-jaaski@tkk.fi

Abstract—**The growing popularity of mobile internet services, characterized by heavy network transmission, intensive computation and an always-on display, poses a great challenge to the battery lifetime of mobile devices. To manage the power consumption in an efficient way, it is essential to understand how the power is consumed at the system level and to be able to estimate the power consumption during runtime. Although the power modeling of each hardware component has been studied separately, there is no general solution at present of combining them into a system-level power model. In this paper we present a methodology for building a system-level power model without power measurement at the component level. We develop a linear regression model with nonnegative coefficients, which describes the aggregate power consumption of the processors, the wireless network interface and the display. Based on statistics and expert knowledge, we select three hardware performance counters, three network transmission parameters and one display parameter as regression variables. The power estimation, based on our model, exhibits 2.62% median error on real mobile internet services.**

Keywords-regression; power model; system-level; mobile device

I. INTRODUCTION

In recent years, mobile internet services have become increasingly popular. Compared with the traditional mobile services such as voice call and short messaging service, mobile internet services such as video-on-demand and mobile gaming are often characterized by heavy network transmission, intensive computation and an always-on display. These features imply a heavy workload on the processors, the wireless network interface and the display in performing these services, which causes high power consumption. Because the battery technology is not advancing as fast as the mobile computing and networking technology, the usage of mobile internet services poses a big challenge to the battery lifetime of mobile devices. It is therefore critical to manage the power consumption in a more efficient way, a core requirement of which is to understand how the power is consumed at the system level and to be able to estimate the power consumption during runtime.

In this paper we present a system-level power model, which includes the power consumption of the processors, the wireless LAN interface (WNI), and the display. Unlike the system-level power model proposed by A. Carroll et al. [1], our methodology does not require a physical power measurement at the component level. Instead, we use the total power consumption of a mobile device, which can be measured through a power meter or energy profiling software like Nokia Energy Profiler[1]. It addresses the issue that the power measurement could not be done at the component level on most of the off-the-shelf devices.

Our model is based on linear regression with nonnegative coefficients. We select the variables that reflect the activity levels of each hardware component such as the hardware performance counters (HPCs) for processors, the downlink and uplink data rates for the WNI, and the brightness level for the display.

Linear regression has been widely used for modeling the power consumption of processors [2, 3, 4, 5], with a set of HPCs as regression variables. A challenge to the HPC-based modeling is that only a small proportion of the HPC set can be monitored simultaneously during runtime. For instance, only 3 out of 17 HPCs can be simultaneously observed on ARM 1136, a mobile processor, which is deployed in many commercial mobile devices such as the Nokia N810, Motorola Z6 and T-Mobile G1. To the best of our knowledge, no prior research has introduced the nonnegative coefficient feature into regression-based power modeling. In this paper, we show that this feature makes the variable selection efficient.

We extend the usage of linear regression to system-level power modeling. In addition to the HPC-based regression variables that are selected through statistical methods, we select other regression variables including the network transmission parameters and the display parameter based on the knowledge gained from a previous study [6]. The variables we use can be obtained from OS or application, without monitoring the power status of each hardware component and the network traffic underlying the transport layer.

We aim at a model that can be independent of the usage scenarios of a mobile device. Thus, we define our test cases in ways that they can be used to explore the possible activity levels of each hardware component, and use different test

[1] http://www.forum.nokia.com/info/sw.nokia.com/id/324866e9-0460-4fa4-ac53-01f0c392d40f/Nokia_Energy_Profiler.html

978-0-7695-4331-4/10 $26.00 © 2010 IEEE
DOI 10.1109/GreenCom-CPSCom.2010.114

cases for model fitting and evaluation. The power estimation, based on our model, exhibits a median error of 2.62% in real mobile internet services.

In summary, we report on a regression-based approach for modeling the system-level power consumption of a mobile device. Our work makes the following specific contributions:

- We demonstrate the feasibility of building a system-level power model that is still able to reflect the activity levels of each hardware component without using component-level power measurement.
- We show the advantage of using nonnegative coefficients in regression-based power modeling for reducing the size of the variable set more efficiently.
- We provide a power model that is independent of usage scenarios and that can be used for runtime power estimation with reasonable accuracy.

We present our methodology and show its practicality in the remainder of this paper. Section 2 introduces the related research on power measurement and modeling. Section 3 presents the theoretical approach including the model definition, the variable selection and the benchmark design. Section 4 describes our experiments and shows the results. The potential usage of our model and future work is discussed in Section 5. Finally, Section 6 concludes the paper.

II. Related Work

Dynamic power management on mobile devices requires rich knowledge about how and where the power is consumed. Power measurement is a must for quantitative analysis of power consumption, although it is not always feasible. For example, the power measurement at the component level requires information about the power distribution network at the circuit level, which is not publicly available except for a few devices like Openmoko Neo Freerunner[2]. In addition, power measurement itself does not give information about how the energy is consumed. To solve this problem, power modeling is proposed as a complementary method for power analysis. It uses the information gained from hardware, OS and/or applications to describe how the power is consumed.

Power modeling of mobile devices has been under study in the last decade. Some models are based on the physical design of the hardware, such as the intellectual property-level power model that was proposed by M. Onouchi et al. [7]. These physical-level models can be used for power analysis during the design stage, but not for the power estimation during runtime. V.Tiwari et al. [8] proposed an instruction-level model for processors, which was based on the power measurement of instructions. It is useful for the power analysis of a software component, but it has its shortcoming in supporting multi-threaded applications. To address the above issues, some researchers [2, 3, 4, 5, 9] proposed building a power model for processors based on HPCs, because HPCs reflect the activity levels of each hardware component in a processor and can be monitored during runtime.

D. Brooks et al [9] implemented an HPC-based power model into an offline power estimator on top of a performance simulator. C. Isic et al. [4] showed an HPC-based runtime energy profiler, with a median error rate up to 7.2%. They used several HPCs for each hardware component for estimating the power consumption of that component. However, previous models [2, 3, 4, 5] could not be directly used in our experimental device, because the number of HPCs that can be monitored simultaneously on our experimental device is less than that required for the previous models.

Derived from [2, 3, 4, 5], we also apply the regression method in our system-level power modeling, but our work differs from previous work in the way that we apply the nonnegative coefficient feature for regression variable selection. In addition, we extend the usage of regression-based modeling from component-level power modeling to system-level power modeling.

Some power monitoring software, such as *PowerTutor*[3] for Android-based mobile platform, claim to be able to monitor the power consumption at the system level. Because the models they use for calculating the power breakdown of the hardware components and applications are not publicly available, it is difficult to compare their models with ours. According to the published information, there are at least two differences between our model and *PowerTutor*. First, *PowerTutor* only considers the CPU frequency level when estimating the power consumption of CPU, whereas we consider the I/O operations in addition. Second, in addition to uplink data rate, we consider downlink data rate and the 802.11 power saving mode, which have not been taken into account in *PowerTutor*.

III. Regression-based Power Modeling

We use linear regression as a baseline method to build a system-level power model that takes three hardware components into account, namely, the processor, the WNI and the display. Our method requires no extra effort for parameter tuning. The linearity is consistent with our previous findings on the power characteristics of hardware resources, for example, the linear relationship between the increments in HPCs and the processor power consumption [2, 3, 4, 5], as well as the linear increase in the network transmission cost with the network data rate [6]. In this paper we describe our modeling methodology, which follows the five steps below.

- Defining the regression variables, such as CPU cycle rate and network data rates, which reflect the activity levels of the corresponding hardware resources.
- Designing the energy benchmark, which stresses each regression variable and explores their cross

[2] http://www.openmoko.com/freerunner.html

[3] http://www.powertutor.org

product. In practice, the benchmark runs a batch of real-life mobile applications (workloads) in ways that correspond to different activity levels of the hardware resources.

- Running the energy benchmark to generate two data sets. One for model fitting, and the other for model evaluation.
- Forming a linear regression model based on the least square method [10]. The model can be used for runtime power estimation.
- Validating the power model with the testing data set. The prediction accuracy is evaluated by the prediction percentage error.

A. Linear Regression Model

Suppose that in a large universe of interest, a subset of *n* observations is known including the values of *p* predictor variables and the values of the corresponding responses. The linear regression approach builds a linear relationship between the *p* predictor variables and the responses based on the *n* observations. Denote a response by $y_i(i \in [1..n])$, the corresponding predictor variables by $x_{i,j}(i \in [1..n], j \in [1..p])$, and the variable coefficient by $\beta_j(j \in [1..p], \beta_j \geq 0)$.

Differently from previous work [2, 3, 4, 5], we introduce the nonnegative coefficient feature [11] into our power model for the following reasons. First, the power consumption of the processor, the WNI and the display is non-subtractive. Second, it is well-known in machine learning theory that linear regression with nonnegative coefficients can show strong sparse effects, which means most of the coefficients will be close to zero [11]. Because the value of the coefficient for each independent variable reflects the amount of the effect that the variable has on the response, the sparse model can be utilized to reduce the size of the regression variable set. This feature is desired for cases where a small proportion of the variables have to be selected from a set of variables. On our experimental device, at most 3 out of 17 HPCs can be used for runtime power estimation.

The power consumption is estimated based on the linear regression model [11] below.

$$f(y_i) = \beta_0 + \sum\nolimits_{j=1}^{p} \beta_j g_j(x_{i,j}), \qquad (1)$$

where $g_j(x_{i,j})$ is a preprocessing function of the original values of the predictors, β_0 is the intercept, which implies the response when all the variables are set to zero. The preprocessing functions used for the predictors in our model are listed in Table I. The observed power consumption can then be expressed as $y_i = f(y_i) + e_i$, where e_i is an additive noise with zero mean and constant variance.

The values of the intercept and each coefficient are automatically adjusted during the model fitting towards a model in which the response can be the best predicted from the predictor variables. To evaluate the goodness of the prediction, we define the sum of the squared deviations of the predicted response, $S(\beta_0,...,\beta_p)$, as below.

$$S(\beta_0,...,\beta_p) = \sum\nolimits_{i=1}^{n} (y_i - f(y_i))^2 \qquad (2)$$

During model fitting, we apply the least square method [10] that minimizes the value of $S(\beta_0,...,\beta_p)$. The results of the model fitting are presented in Section 4.2.

TABLE I. DESCRIPTION OF REGRESSION VARIABLES AND THEIR PREPROCESSING FUNCTIONS.

Hardware Resource	**Regression Variable $x_{i,j}$**	**Preprocessing Function $g_j(x_{i,j})$**	**Description**
Processor	17 HPC-based event rates: $x_{i,j}(j \in [0..16]) = \begin{cases} \frac{c_{i,0}}{d_i}, for\ \ CPU\_CYCLES, \\ \frac{c_{i,j}}{c_{i,0}}, for\ \ other\ \ HPCs, \end{cases} \quad (3)$ where $c_{i,0}$ is the increment in CPU_CYCLES, and $c_{i,j}(j \in [1..16], i \in [1..n])$ is the increment in any other HPC during the same monitoring period d_i.	Normalization function: $g_j(x_{i,j}) = \frac{x_{i,j} - mean(x_{i,j})}{standard\ \ deviation(x_{i,j})} \quad (4)$ The mean and the standard deviation are calculated from the data set used for model fitting.	HPCs available on ARM 1136: CPU_CYCLES, DCACHE_MISS, TLB_MISS, ITLB_MISS, CYCLES_DATA_STALL, INSN_EXECUTED, DTLB_MISS, DCACHE_ACCESS, DCACHE_MISS, EXP_EXTERNAL, DCACHE_ACCESS_ALL, IFU_IFETCH_MISS, BR_INST_MISS_PRED, CYCLES_IFU_MEM_STALL, LSU_STALL, PC_CHANGE, BR_INST_EXECUTED.
WLAN Interface	Download data rate (KB/s) $x_{i,17}$ Upload data rate (KB/s) $x_{i,18}$ CAM switch $x_{i,19}(x_{i,19} \in [0..1])$	$g_j\ (x_{i,j}) = x_{i,j}$	$x_{i,19} = 1$: CAM enabled AND network data rate is lower than a threshold [6]; $x_{i,19} = 0$: Otherwise.
Display	Brightness level $x_{i,20}(x_{i,20} \in [0..5])$	$g_j\ (x_{i,j}) = x_{i,j}$	6 Brightness levels on a Nokia N810: 0: off; 1..5: brightness from low to high.

B. Regression Variables

We initially select 21 regression variables, which reflect the power characteristics of the processor, the WNI and the display respectively. The variables and their preprocessing functions are described in Table I.

First, we use HPCs to estimate the power consumed by the CPU processing and the memory access. As shown in [2, 3, 4, 5], HPCs reflect the activity levels of the hardware components in a processor such as data cache and instruction cache. In addition, monitoring some HPCs such as the L3 cache miss counter allows us to track the use of the off-chip memory.

There are 17 HPCs available on our experimental processor ARM 1136, as listed in Table I, whereas only the CPU cycle counter CPU_CYCLES and any other two HPCs can be monitored simultaneously during runtime. We define an event rate for each HPC as shown in (3). HPCs are aggregate counters. For CPU_CYCLES, we define its event rate as the consumption rate of CPU cycles, which can be calculated as the number of CPU cycles elapsed in a unit of time. For the other HPCs, similarly with [2], we define the event rate of each HPC as the increment in the HPC during a CPU cycle. It can be calculated as a ratio of the increment in the HPC to the corresponding increment in CPU_CYCLES during the same monitoring period. The event rates are normalized using (4). The statistics used for normalization are calculated based on the data set used for model fitting.

Second, we select 3 network parameters based on the knowledge gained from a previous study about power modeling of data transmission [6]. The power consumption of the data transmission through an 802.11 WLAN linearly increases with the upload/download data rate. Hence, we select the upload and download data rates as regression variables because they reflect the workload of the WNI. Moreover, the 802.11 power saving mode has impact on the power consumption. For instance, when the data rate is lower than a threshold such as 32KB/s [6], the power consumption is higher if the Continuously Active Mode (CAM) is enabled. Concerning the requirement of nonnegative coefficients, we choose the CAM switch as a regression variable. When the power saving mode is disabled, the value of the CAM switch is set to 1. Otherwise, it is set to 0.

A mobile internet application can invoke the protocol processing in a processor, the read/write operations in a memory, and the receive/transmit operations on a network interface. The first two types of operations can be reflected by HPCs, and the last one can be measured by the network parameters. Hence, with HPCs and the network parameters, our model can be used for modeling the power consumption of network applications, which has not been widely discussed in previous work.

TABLE II. DESCRIPTIONS OF THE WORKLOADS USED IN OUR ENERGY BENCHMARK.

Category	Description	Test Case
Idle with Different Brightness Levels	CPU and memory workload: Low Wireless connection: No Brightness level: 0~5	Keep the system idle without running any applications and set the brightness level of the display to different values.
Audio/ Video Players	CPU and memory workload: Low ~ High Wireless connection: No Brightness level: 0 for audio player; 5 for video player.	Media player on N810: mplayer[4] Media file storage: Phone memory Audio format: MP3, OGG, RM Number of audio players in parallel: 1, 2, 3 Video format: AVI, MPEG Number of video players in parallel: 1, 2
Audio/ Video Recorders	CPU and memory workload: Medium Wireless connection: No Brightness level : 5	Run an embedded audio recorder to record an audio file played on a machine close to the experimental device.
		Use the embedded camera to record a video.
File Download/ Upload at Different Data Rates	CPU and memory workload: Low ~High WLAN connection: On Network data rate: 16KB/s ~ 400KB/s. Brightness level: 0	N810: netcat[5] Linux Server: netcat, Trickle[6](bandwidth limiting utility) Data rate limit: 16, 32, 128, 256, and 400KB/s. CAM: On/off (data rate < 32KB/s); Off (data rate ≥32KB/s) Download Storage: phone memory, /dev/null Upload storage: phone memory
Streaming	CPU and memory workload: High WLAN connection: On WLAN Power saving mode: Enabled Brightness level: 5	Watch online TV programs transferred from www.itv.com. Encoding rate: 16 ~ 72KB/s
		Listen to radio programs from three different radio websites. Download date rate: around 24KB/s.
		Use web browser to watch YouTube videos online. Download data rate: 46~136KB/s depending on the network conditions.

[4] http://mplayer.garage.maemo.org/

[5] http://netcat.sourceforge.net

[6] http://monkey.org/~marius/pages/?page=trickle

Third, similarly with *PowerTutor*, we use brightness level to model the power consumption of the display. Assuming that the resolutions of the displays are fixed during runtime, we divide the workload of the display into 6 brightness levels, compatible with the screen configuration on our experimental device.

C. Energy Benchmark

We develop a new benchmark to generate data for fitting the regression-based power model, because the existing system-level energy benchmarks such as *JouleSort* [12] do not take the power consumption of the display and the WNI into account [13]. Our benchmark captures the values of the regression variables defined in Section 3.2 at a certain sampling frequency when running the workloads. Each run of the workload lasts for a certain monitoring period such as 60 seconds.

The workloads we use are categorized into five types, namely, idle with different brightness levels, audio/video players, audio/video recorders, file download/upload at different network data rates, and streaming. In each category there are multiple test cases as described in Table II. They are chosen based on the following three principles.

First, similarly with the micro-benchmarks used in [4], our benchmark stresses the selected variables and explores the space of their cross product. The resource consumption of a real-life mobile application can be described in terms of four elements, CPU processing intensity, memory access rate, network data rates, and the brightness level of the display. Each element can be represented by one or multiple regression variables. We define the values of the first three elements to be low, medium and high, and the value of the fourth element to be equal to the brightness level of the display set in the screen configuration. We choose the test cases in which the resource consumption corresponds to as many different combinations of the values of the four elements as possible, which makes it possible for our model to be independent of usage scenarios.

Second, we choose the workloads with a fixed demand on hardware resources over a sampling interval of the HPC event rates. Accordingly, the power consumption is considered to be stable during the sampling interval. A monitoring period includes at least one sampling interval. For the workloads with a fixed demand on the hardware resources over a given monitoring period, the values of the corresponding HPCs are increasing at a constant rate. For example, in a case where a video playback with a fixed frame rate is a workload with a fixed demand on the hardware resources over the monitoring period, the increment in the HPCs such as CPU_CYCLES is stable in each sampling interval. In this case, we calculate the average value of all the samples and consider it as one observation.

For the workloads with varying demand on the hardware resources over the monitoring period, we try to divide the monitoring period into several smaller periods in each of which the demand can be considered to be stable. For example, an internet radio test case can be divided into two periods: downloading only, downloading together with playback. To simplify the synchronization of the HPCs, the network parameters and the power consumption during data collection, we define the test cases in the way that the demand of the hardware resources can be stable for a relatively long time.

Third, we choose the applications that are typical on the mobile devices in question. All the applications we use are either embedded in our experimental device, or easy to download and install from the support website of the device. The detailed information about the applications we use is listed in Table II.

IV. EXPERIMENTATION

A. Data Collection

Our experimental device, a Nokia Internet Tablet N810, is equipped with an ARM 1136 processor at 330 MHz, an 802.11 WNI, and a high-resolution WVGA display (800×480 pixels). It is running a Linux-based OS, Maemo.

Our benchmark ran *oprofile*[7] to access the HPCs from user space, and logged the readings of HPCs every 1 second. The sampling interval of the HPCs was set to 100000 CPU cycles during the initialization of *oprofile*. During runtime, 3 HPCs at most can be accessed simultaneously on ARM 1136, and one counter is reserved to count the CPU cycles. Similarly with the multiplexing technique presented in [14], to get a full observation including all the 17 event rates, our benchmark repeated the same test case for 8 times with two different HPCs monitored each time. We used the CPU cycles as timers to multiplex the event rates. In other words, the 8 samples from different runs can be merged into one full observation only when the event rate of CPU_CYCLES in each sample is equal to each other.

In practice, we collected 60 samples continuously from the beginning of a test case in each run except for the streaming cases in which we started the monitoring when the playback started. After 8 runs, we got 60 full observations. According to our observation, in most of test cases, the difference among the 60 samples in a line is close to 0. In these cases, instead of importing 60 similar observations into our data sets, we only imported one observation, which included the average value of each HPC in the 60 observations. In total, our benchmark ran each test case for 48 times to get 6 full observations for each.

Our test cases can be divided into 5 categories as described in Table II. We use different test cases to generate two different data sets for model fitting and evaluation, respectively. For example, we used the data collected from three types of workload for model fitting, including idle with different brightness levels, audio/video players, and file download/upload at different network data rates. For model evaluation, we used the data collected from streaming, audio/video recorders, and file download/upload at different network data rates. Even for the same type of workload, the test cases used in model evaluation are different from those in model fitting.

We connected the N810 to a sample multimeter to measure the power consumption. The sampling frequency of

[7] http://maemo.org/development/tools/doc/diablo/oprofile/

the multimeter was synchronized with that of the event rates and was set to 1Hz. In the test cases with the N810 connected to an 802.11g network, the timeout of the 802.11 power saving mode was set to 100ms. The network data rates in the file transmission cases were limited by a traffic shaper running on a Linux server, and the data rates in the streaming cases depended on the streaming protocols, the media formats and the network conditions. We captured the network traffic by running *Wireshark*[8], a network analyzer, on a laptop that was connected to the same WLAN. We adjusted the CAM settings and the display brightness levels through the user interfaces.

B. Model Fitting

We used a function called *lsqnonneg* in Matlab[9], which is meant for optimizing the least-square objectives with non-negativity constraint. During model fitting, we ran this function twice. In the first round, each observation included 17 HPC-based variables, 3 network parameters and 1 display parameter. After executing *lsqnonneg*, we chose 3 HPC-based variables with the biggest coefficient values. They were the event rates of DCACHE_WB, TLB_MISS and CPU_CYCLES. In the second round, we fitted the observations including the three selected event rates, the three network parameters and the one display parameter to a regression model by running *lsqnonneg* again. Our final power model is presented in (5).

$$\begin{aligned} Power(W) &= 0.7655 + 0.2474 \times g_0(x_0) + 0.0815 \times g_1(x_1) \\ &+ 0.0606 \times g_2(x_2) + 0.0011 \times g_{17}(x_{17}) \\ &+ 0.0015 \times g_{18}(x_{18}) + 0.3822 \times g_{19}(x_{19}) \\ &+ 0.125 \times g_{20}(x_{20}). \end{aligned} \tag{5}$$

where $g_j(x_j)(j \in [0..2, 17..20])$ is the preprocessing function as described in Table I. Let c_0, c_1 and c_2 be the increment in CPU_CYCLES, DCACHE_WB and TLB_MISS in the monitoring period d, respectively.

$$\begin{aligned} & g_0(x_0) = \frac{x_0 - 1316.84}{1349.423}, x_0 = \frac{c_0}{d}, \\ & g_1(x_1) = \frac{x_1 - 0.000901}{0.00045}, x_1 = \frac{c_1}{c_0}, \\ & g_2(x_2) = \frac{x_2 - 0.000513}{0.000365}, x_2 = \frac{c_2}{c_0}, \\ & g_{17}(x_{17}) = x_{17}, x_{17} : download \quad data \quad rate(KB/s), \\ & g_{18}(x_{18}) = x_{18}, x_{18} : upload \quad data \quad rate(KB/s), \\ & g_{19}(x_{19}) = x_{19}, x_{19} : CAM \quad switch, \\ & g_{20}(x_{20}) = x_{20}, x_{20} : brightness \quad level. \end{aligned} \tag{6}$$

We define the IDLE mode of a mobile device as the status when there is no application running. The value of the intercept, 0.7655, is close to the power consumption in the IDLE mode with the display turned off. Among the three HPC-based variables, the event rate of CPU_CYCLES describes the general workload of the processor, which takes a big part of the total power consumption. The event rates of TLB_MISS and DCACHE_WB reflect the memory access efficiency in a CPU cycle. The coefficient values of the non-HPC variable show an increase in the power consumption of the WNI or the display when the corresponding variable increases by one unit. For example, an increase of 1 KB/s in the upload data rate costs on average 1.5mW more power, and the network transmission with CAM enabled costs on average 0.3822W more when the network data rate is less than 32KB/s on the experimental device.

C. Model Evaluation

We used our testing data set to evaluate the regression model obtained in Section 4.2. Although we computed the prediction values offline, the computation followed the same steps and gained the same results as the online computation for runtime power estimation. First, we calculated the event rates based on the collected HPC values. Second, we normalized the event rates based on the statistics of the data set used for model fitting. Third, we obtained the upload/download data rates from traffic traces, and the display settings from the user interface. The data rates can also be monitored during runtime through networking APIs. Fourth, we estimated the power consumption following (5) and compared the estimates to the physical measurement.

For the data set used for model evaluation, the median percentage error in power estimation is 2.62%, and the standard deviation of the error rate is 0.0376. Compared to [4] whose median percentage error is 3.9% ~7.2% depending on the testing data sets, our model still provides adequate accuracy while using fewer HPCs and extending the power estimation to network applications.

To analyze the prediction accuracy for different workloads, we calculated the median error for each category of workload. As shown in Fig. 1, our power model tracks the power consumption in each category fairly well. Only in the cases of video recorder is the error a little bigger. We attribute the error in the power estimates for the video

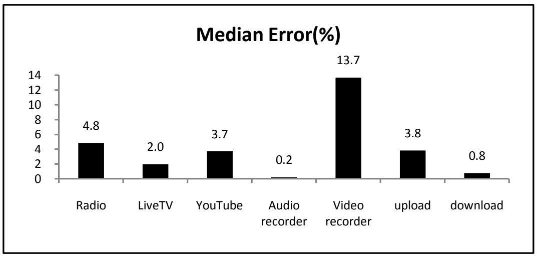

Figure 1. Median percentage error in the power estimates of different test cases.

[8] http://www.wireshark.org

[9] http://www.mathworks.com/access/helpdesk/help/toolbox/optim/ug/lsqnonneg.html

recorder to the usage of the camera, which could not be monitored through our variables. In future we will consider the power consumption of the camera in different operating modes, such as standby and capturing, in our model.

V. Discussion

Our model can be used for power simulation at the system level when the regression variables are available. The HPCs are available on the performance simulators such as *Wattch* [9], and the network transmission parameters can be found from network simulators such as NS-2. However, to the best of our knowledge, no prior simulator can provide both the HPCs and the network transmission parameters. In large-scale distributed computing networks, both computational and communication workload consumes high power consumption on a mobile device. Hence, for future work it is worth developing a mobile device simulator, which combines performance and network simulators, and supports power simulation at the system level.

Dynamic power management on a mobile device requires knowledge about the power consumption at different granularities depending on the policies it uses. For example, total power consumption is required for estimating the remaining battery lifetime, while the power breakdown of processes is needed for task scheduling in OS. Although we only showed the estimation of total power consumption in this paper, our model can also be used for more detailed power analysis.

Because the HPCs can be monitored for each process, we propose to estimate the computational power consumption of each process based on the per-process HPC values. Assume that there are N processes contributing to the HPCs during a monitoring period d. For process $i (i \in [0..N-1])$, the increments in DCACHE_WB, TLB_MISS, and CPU_CYCLES are defined as w_i, m_i and u_i, respectively. The total number of CPU cycles elapsed in d is defined as $c_0 = \sum_{i=0}^{N-1} u_i$. We reform the preprocessing functions of the HPC-based regression variables defined in (6) as below.

$$\begin{aligned} g_0(x_0) &= \sum\nolimits_{i=0}^{N-1} \frac{u_i/d - 1316.84}{1349.423}, \\ g_1(x_1) &= \sum\nolimits_{i=0}^{N-1} \frac{w_i/c_0 - 0.000901}{0.00045}, \\ g_2(x_2) &= \sum\nolimits_{i=0}^{N-1} \frac{m_i/c_0 - 0.000513}{0.000365}. \end{aligned} \tag{7}$$

It is possible to estimate the computational power consumption of process i as shown in (8). However, because there is no power meter available for measuring the power consumption of each process, we have not been able to validate our power breakdown of the processes.

$$\begin{aligned} Power(W) = 0.7655 &+ 0.2474 \times \frac{u_i/d - 1316.84}{1349.423} \\ &+ 0.0815 \times \frac{w_i/c_0 - 0.000901}{0.00045} \\ &+ 0.0606 \times \frac{m_i/c_0 - 0.000513}{0.000365}. \end{aligned} \tag{8}$$

The power consumption of a WNI is estimated based on the aggregate data rates in our model. It is derived from the model basing on the operating mode of the WNI and the 802.11 power saving mode [6]. For example, the WNI is in TRANSMIT or RECEIVE mode when it is transmitting or receiving data, and it is in either IDLE or SLEEP mode during traffic intervals depending on the settings of the 802.11 power saving mode. The power consumption of a WNI includes the power consumed in each operating mode. When there are multiple flows sharing a WNI, the traffic intervals can be inside a flow or between flows, and there is no common rule in assigning traffic intervals to each flow. If the power consumption during the traffic intervals is ignored, the power consumed by each flow depends on how much data rate it contributes to the aggregate data rate. For example, if a flow contributes 50% of the data rate, this flow consumes 50% of the power consumption in TRANSMIT and RECEIVE modes.

In this paper we present a regression-based method of building a system-level power modeling. When a new hardware component is installed into the mobile device, there are two ways to update the system-level power model. One way is to add regression variables, which describe the activity levels of the new hardware component, define new test cases to stress the new variables, and fit the new data sets to a regression model. The other way is to directly add an analytical power model of the new hardware component to the existing system-level model. For the latter method, an analytical power modeling of the hardware component must be built and validated beforehand. In future we will compare these two methods by adding more hardware resources such as 3G interface and GPS receiver in the system.

VI. Conclusion

We have proposed a power model for mobile internet devices taking the major hardware resources such as the processor, the WLAN interface and the display into account. We showed how to build such a system-level power model using linear regression with nonnegative coefficients. The model we built can be used for runtime power estimation and offline power simulation, with a median error of 2.62%, which can help improve the efficiency of power management during runtime.

Acknowledgment

This work was supported by TEKES as part of the Future Internet program of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT). We thank Khaled Chowdhury for his help in data collection.

REFERENCES

[1] A. Carroll and G. Heiser, “An analysis of power consumption in a smartphone,” in *USENIXATC'10: Proceedings of the 2010 USENIX conference on USENIX annual technical conference*. Berkeley, CA, USA: USENIX Association, Jun. 2010, pp. 21–34.

[2] K. Singh, M. Bhadauria, and S.A. McKee, “Real time power estimation and thread scheduling via performance counters,” *SIGARCH Comput. Archit. News*, vol. 37, no. 2, Jul. 2009, pp.46–55.

[3] B. C. Lee and D. M. Brooks, “Accurate and efficient regression modeling for microarchitectural performance and power prediction,” in *ASPLOSXII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems* , New York, NY, USA: ACM, Oct. 2006, pp. 185–194.

[4] C. Isci and M. Martonosi, “Runtime power monitoring in high-end processors: methodology and empirical data,” in *MICRO-36: Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, IEEE Press, Dec. 2003, pp. 93–104.

[5] B. Goel, S. McKee, R. Gioiosa, K. Singh, M. Bhadauria, and M. Cesati, “Portable, scalable, per-core power estimation for intelligent resource management,” in *IGCC'10: Proceedings of the 2010 International conference on Green Computing*, IEEE Press, Aug. 2010, pp. 135 –146.

[6] 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 *e-Energy'10: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*. New York, NY, USA: ACM, Apr. 2010, pp. 75–84.

[7] M. Onouchi, T. Yamada, K. Morikawa, I. Mochizuki, and H. Sekine, “A system-level power-estimation methodology based on ip-level modeling, power-level adjustment, and power accumulation,” in *ASP-DAC'06: Proceedings of the 2006 Asia and South Pacific Conference on Design Automation*, IEEE Press, Jan. 2006, pp. 547–550.

[8] V. Tiwari, S. Malik, A. Wolfe, and M.-C. Lee, “Instruction level power analysis and optimization of software,” *VLSI Signal Processing*, vol. 13, 1996, pp. 1–18.

[9] D. Brooks, V. Tiwari, and M. Martonosi, “Wattch: A framework for architectural-level power analysis and optimizations,” in *ISCA'10: Proceedings of the 27th Annual International Symposium on Computer Architecture*, New York,NY, USA: ACM, 2000, pp. 83–94.

[10] C.Lawson and R. Hanson. Solving Least Squares Problems. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[11] F.I. Kushnirskii and M.E. Primak, “Regression with nonnegative coefficients,” *Cybernetics and Systems Analysis*, vol. 9, no. 1, Springer New York, Jan. 1973.

[12] S.Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis, “Joulesort: a balanced energy-efficiency benchmark,” in *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, Jun. 2007, pp. 365–376.

[13] S. Rivoire, M. A. Shah, P. Ranganathan, C. Kozyrakis, and J. Meza, “Models and metrics to enable energy-efficiency optimizations,” *Computer*, vol. 40, no. 12, 2007, pp. 39–48.

[14] R. Azimi, M. Stumm, and R. W. Wisniewski, “Online performance analysis by statistical sampling of microprocessor performance counters,” in *ICS'05: Proceedings of the 19th annual international conference on Supercomputing*. New York, NY, USA: ACM, Jun. 2005, pp. 101–110.