Characterizing System Level Energy Consumption in Mobile Computing Platforms

Cintia B. Margi , Katia Obraczka, Roberto Manduchi
Computer Engineering Department
University of California Santa Cruz
1156 High Street
Santa Cruz, CA 95064

***Abstract*— This paper approaches energy consumption characterization in mobile computing platforms by assessing energy consumption of ”basic” application-level tasks, such as as processing, input/output (disk, display, etc.), communication (transmission and reception over the network), and combinations thereof. Besides providing information on the energy consumption behavior of typical tasks performed by mobile computers, task-level energy characterization enables power management decisions, such as whether, in a distributed computation, the task at hand can be executed locally or should be assigned to a different machine (given the machine's current energy budget, the energy cost of executing the task locally, and the cost of sending the required information over the network to a peer). We employ a task-level energy consumption characterization benchmark that accounts for basic tasks such as processing, disk access (including reads and writes), terminal usage, and communication (transmission and reception). Using the benchmark, we perform an energy characterization case study using the Dell Latitude C600 running two versions of the Linux operating system.**

I. Introduction

Due to a combination of technology advances in fields such as wireless communications and circuit integration, the last ten years have witnessed a proliferation of mobile computing platforms. Examples of such platforms, which vary widely in terms of capability and functionality, include laptops (e.g., notebooks, tablets, etc.), pocket PCs, personal digital assistants (PDAs), cell phones, wireless single-board computers, sensor nodes, etc. Following the general trend in the consumer electronics market, the cost of these devices has been steadily decreasing while their capacity (i.e., processing, storage, communication) has been steadily increasing. However, the fact that they are typically powered by non-continuous energy sources imposes serious limitations to these devices' utility from the end user's point-of-view.

As a result, energy consumption in mobile computing platforms has been an area of intense research spanning many fields such as computer architecture, operating systems, computer networks, and application design [1]. In the computer architecture community, energy characterization is usually performed at the instruction level. Proposed power savings mechanisms include shutting off parts of the processor not currently being used, designing machine-level instructions that trade generality for power efficiency, etc. At the operating systems level, example of power management strategies include disk spin-down, periodic system hibernation, etc. Power-efficient network protocols have also been attracting considerable attention from the networking research community and include a variety of techniques such as hibernation of idle nodes, controlling transmission power and/or direction, routing based on remaining energy in nodes, etc.

This paper takes a different approach to energy consumption characterization and focuses on characterizing energy consumed by ”basic” application-level tasks. Our focus is on mobile computing platforms, e.g. laptops. Applications executed by multi-purpose mobile devices like laptops can be as general as “fixed computing” applications and typically consist of basic tasks such as processing, input/output (disk, display, etc.), communication (transmission and reception over the network), or a combination thereof. Consider reading e-mail: it includes communication with the email server and thus network transmission and reception, processing information received, and storing it on disk.

Characterizing energy consumption at the task level allows us to (1) predict whether the energy currently available is sufficient to execute a given application, and (2) perform application-level power management. For example, in a distributed computation, given the task at hand and how much energy there is left, a machine's task manager decides whether the task can be executed locally or needs to be shipped elsewhere. In order to make that decision, the task manager, given the machine's current energy budget, considers the amount of energy the execution of the task will consume versus the amount of energy consumed by sending the necessary information over the network to another machine.

In this paper, we identify a set of basic tasks representative of mobile computing workloads. Based on these tasks, we define a set of benchmarks that consider each task in isolation or task combination. To observe the battery discharge behavior as a benchmark executes, we employ the Advanced Configuration and Power Interface (ACPI)[13] to monitor the battery's discharge rate. A brief description of the ACPI is presented in Section II.

As case studies, we use the benchmark set to characterize the energy consumption of a mobile computing platform (the Dell Latitude C600) under different operating systems, namely Debian [21] and Mandrake [16] Linux. We used the Dell Latitude C600 because it was easily available to us (we have

Cintia Margi is supported by CNPq/Brazil.

[1]Section VII reviews related research in areas that are more closely related to the focus of this paper.

several of them in our lab). As future work, we intend to run the benchmark on other mobile computing platforms.

In summary, the main contribution of this work is a task-level energy consumption characterization benchmark that accounts for basic tasks such as processing, disk access (including read and write access), terminal usage, and communication (transmission and reception). Such characterization is critical to power management decisions.

The remainder of the paper is organized as follows. Section II provides an overview of the ACPI standard. Sections III and IV describes the proposed energy consumption benchmark and our experimental methodology. Section V presents results from our case studies, while Section VI discusses benchmark set results application. We summarize related work in Section VII. Section VIII concludes the paper and discusses future work.

II. Background

The Advanced Configuration and Power Interface (ACPI) is an open standard developed by a consortium involving Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba [13]. It is a standard that defines power and configuration management interfaces between an operating system and the BIOS [6].

The ACPI provides both static information about the battery such as model number, serial number, design voltage, etc., as well as current battery status, e.g., whether the battery is charging or discharging, current voltage, discharge rate, estimate of the remaining battery capacity, etc. In Linux, the ACPI name space is mapped to the */proc* filesystem. For example, in Debian, the ACPI name space is mapped to */proc/acpi/*; and battery related information is in */proc/acpi/battery*.

The ACPI updates battery information (both static and dynamic) every time the corresponding file (i.e., */proc/acpi/battery/info /proc/acpi/battery/state* is read [2]. Therefore the frequency these information is updated depends on the application doing the battery monitoring.

III. Energy Consumption Benchmark

As previously discussed, our goal is to characterize energy consumption macroscopically at the task level (instead of, for example, at the machine instruction level). To this end, we define a set of ”basic” application-level tasks that are representative of typical mobile computing workloads. In the case of laptops, basic tasks include: processing, input/output (disk, display, etc.), and network communication (transmission and reception). Since the focus of this work is mobile nodes, we consider wireless network interfaces. Often, applications consist of a combination of such tasks.

For our benchmarks, we define four main task categories, namely *baseline*, *processing intensive*, *storage intensive*, and *communication intensive*. We also consider the effect of the display by turning it off and on when executing some of the benchmark tasks.

a) Baseline: The baseline benchmark captures the energy consumption behavior of the mobile when no user activity is taking place, i.e., only basic operating system tasks are running. This benchmark characterizes energy consumption when the system is idle and also serves as a reference for all other benchmarks. We disable the wireless network interface to isolate the effects of this device on energy consumption.

b) Processing-intensive: To characterize processing-intensive tasks, we use the FFT benchmark [1], which is part of SPEC's CPU2000 [20], an industry-standardized CPU-intensive benchmark suite. FFT, short for fast Fourier transform, is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse.

c) Storage-intensive: We chose the IOzone [18] filesystem benchmark to characterize energy consumed by tasks that are disk intensive. IOzone can be configured to perform a variety of disk access operations.

We run IOzone in two different modes, one which performs only read accesses and another that only writes to disk. In both cases, it accesses a 3GB file. We use an option which purges the disk cache before each file operation. The write tests includes writes of new files and re-writes of existing files, and the read tests reads and re-reads a file.

d) Communication-intensive: We characterize communication-intensive tasks by using Iperf [23], a tool designed to measure TCP available bandwidth.

Network transmission is implemented using Iperf in client mode, while the reception task uses Iperf in server mode. In both cases, Iperf is configured to generate UDP traffic at 10Mbs for all the experiment duration. A 10Mbs rate was chosen because this is maximum nominal capacity of the wireless card.

e) Display: When we needed to turn the display off, we used the $xset$ [19] utility. $xset$ can be run from the command line and allows the setting of several user preference options for the display, including an option that turns the video card and LCD display off.

IV. Measurements

While the tasks run, we observe the battery discharge behavior through measurements provided by the ACPI. In particular, we monitor the battery discharge rate. As mentioned in Section II, we can obtain this information by sampling */proc/acpi/battery/state*. We run the sampling script which periodic reads the current values of the battery discharge rate, while the benchmark is executing.

When deciding which sampling rate to use, accuracy is an important consideration. However, making sure that the measurements do not interfere with the observations is also critical. In other words, we want to sample as frequently as possible without being intrusive. Preliminary experiments showed that a sampling rate of deci-seconds is not intrusive.

V. Case Study: Dell Latitude C600

We use our benchmark set to characterize energy consumption in the Dell Latitude C600 which has a 750 Mhz Pentium III (Coppermine) processor with 256K cache, 256 MB RAM, and 20Gb hard-disk. Its is powered by a Li-ion battery, with eight cells, design voltage of 14.8 VDC and nominal capacity of 59.0 Wh. As the network interface, we use the Cisco Aironet 350. As the operating system, we use two of the most widely used versions of the Linux operating system,

[2]Simon Fowler, maintainer of $wmacpi$ [12], provided this information.

namely Debian [21] (Debian kernel 2.6.1) and Mandrake [16] (Mandrake 10.1 kernel 2.6.8).

In the remainder of this section, we present energy consumption results for this platform when executing the basic benchmarks described in Section III as well as some combinations thereof.

A. Basic Tasks

We execute all six tasks described in Section III: baseline, processing (FFT), disk writes (IOzone write), disk reads (IOzone reads), network transmission (Iperf client) and reception (Iperf server). We run both operating systems using their default configuration. The script that monitors the battery discharge rate runs at a one second sampling rate.

The discharge rate time series are shown in Figure 1. We summarize these results in Table I which tabulates average discharge rate and the corresponding standard deviation for all tasks under both Debian and Mandrake [3].

From Table I, we note that, for both operating systems, the most energy hungry task is FFT (i.e., intensive CPU activity), followed by disk writes. This is somewhat surprising as we expected disk write intensive tasks to be more expensive than processing intensive tasks in terms of energy consumption. For Debian, network transmission is the next task in energy consumption descending order, followed by disk reads and network reception, both of which have about the same energy cost. For Mandrake, the order is slightly different and has disk reads as the third most expensive task followed by network transmission and then network reception.

Operating System	Debian	Mandrake
Task	Mean and St. Deviation	Mean and St. Deviation
Baseline	10.586 W, 4.285	10.525 W, 4.904
Processing	25.111 W, 1.155	24.836 W, 1.189
Disk Write	19.588 W, 5.218	22.429 W, 4.838
Disk Read	16.233 W, 5.124	21.849 W, 4.747
TX	18.315 W, 4.295	18.301 W, 4.549
RX	16.041 W, 4.236	16.350 W, 4.201

TABLE I
MEAN DISCHARGE RATE FOR EACH TASK ON DELL C600

We should note that we used two laptops, one with Debian and one with Mandrake. They have exactly the same configuration except for their hard disks. Even though both hard disks are manufactured by Hitachi, they have different specifications (e.g., number of heads) and firmware revisions. The configuration parameters are the same for both disks, and they use the same file system (EXT3). However, there is one considerable difference between the disks: the disk on the Mandrake laptop achieves much higher throughput than the one in the Debian laptop. We executed a simple test using IOzone in both platforms, and it took about seven minutes (real time) in the Mandrake laptop, while it took about 10 minutes (real time) in the Debian laptop to write 3GB. This indicates that the difference in the mean discharge rate for both experiments could be due the fact that the Mandrake laptop can transfer more data, i.e. do more writes and reads per experiment.

[3]Since the discharge rate curves for all six tasks on Mandrake exhibit very similar behavior to the corresponding curves obtained for Debian, we omit the Mandrake curves.

Next, we do some back-of-the-envelope calculations to validate the discharge rate results.

According to Intel's 750 Mhz Pentium III Coppermine data-sheet, the voltage and current for the processor core is $Icc_{core} = 15A$ and $Vcc_{core} = 1.65V$ and for the processor in sleep state is $Icc_{sleep} = 2.5A$ and $Vcc_{sleep} = 5V$. If we calculate the power from these two values, we obtain $Pcc_{core} = 24.75W$ and $Pcc_{sleep} = 12.5W$. If we compare Pcc_{sleep} and average discharge rate for baseline experiment, we notice that Pcc_{sleep} is about 20% higher. As for Pcc_{core} and average discharge rate for the FFT experiment, Pcc_{core} is about 4% lower.

During the network transmission experiment, 6.7GB were transmitted at an average bandwidth of 5.7Mb/s. In the network reception experiment, 12.1GB were received at an average bandwidth of 6.6Mb/s. Note that the network reception experiment runs for a longer time than network transmission one because it has lower discharge rate (and thus take longer to reach the cutoff voltage). Since we are using UDP as the transport layer protocol, packet size is 1472 bytes. Since the two laptops are the only two wireless nodes in the lab and they are close to each other, we can assume that in the transmission experiment the time spent in the transmission state (e.g., when compared to the time spent sensing the medium, backing off, performing the 802.11 [8] handshake) dominates. Similarly, for the reception experiment, the time spent in receiving dominates. Thus, we can do a simple calculation considering the bandwidth and the power consumed by a wireless card to verify that the results measured are reasonable. According to the measurements done by Feeney et al [10], a WaveLAN wireless card at 1 Mbps consumes 1400 mW while transmitting and 1300 mW while receiving. If we consider that the mean discharge rate for baseline is about 10.5 W, and we add to this $5.7 * 1.4 = 7.98W$, we obtain a total of 18.3W for the power consumed, which is similar to the 18.48 W mean discharge rate observed in our experiments. Similarly for the reception, if we add 10.5 W to $6.6 * 0.9 = 5.94W$, we obtain 16.44 W, comparable to the 16.04 W mean discharge rate observed in our experiments.

A similar validation for the disk experiments is not as straightforward (and is one of our future work items) since other factors such as disk seek times, default power management techniques as well as the effect of disk caches need to be accounted for.

B. Effect of the Display

In all previous experiments, we had the display off. However, it is a well-known fact that the display is a major source of energy consumption. In this set of experiments, we monitor the energy consumption for some of the basic tasks while the display is on. In particular, we show results for baseline and processing. We ran these experiments on the Debian laptop.

The discharge rate for the baseline experiment is shown in Figure 2 (a). We notice that during part of the experiment the bottom of the discharge rate decreases to about 9 W, and this is because the display was turned off and then the discharge

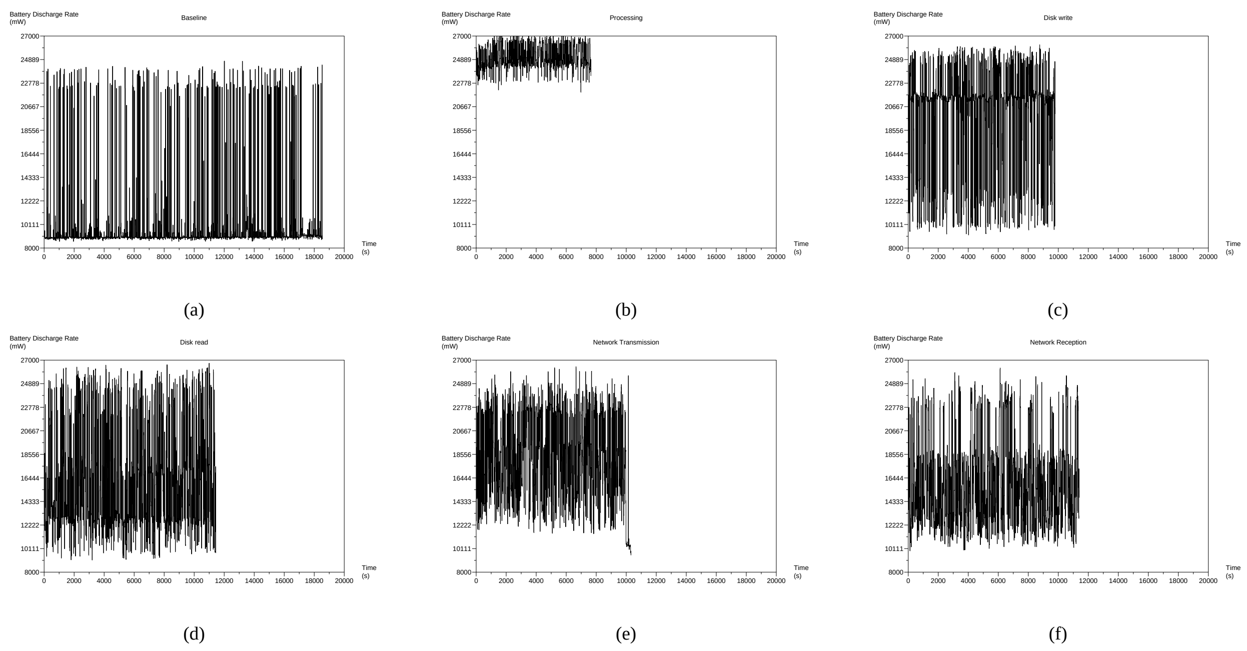

Fig. 1. Discharge Rate for Dell C600 with Linux Debian: (a) baseline, (b) processing task (c) disk write, (d) disk read, (e) network transmission, (f) network reception

rate is about the same we obtained for the baseline task. Thus we observe that the display consumes about as much energy as a disk read or a network reception task.

Figure 2 (b) shows the discharge rate for the processing experiment. Again we observe that the discharge rate decreases to about 24W, which is the mean discharge rate for the basic processing task. This happens because at this point the display was turned off.

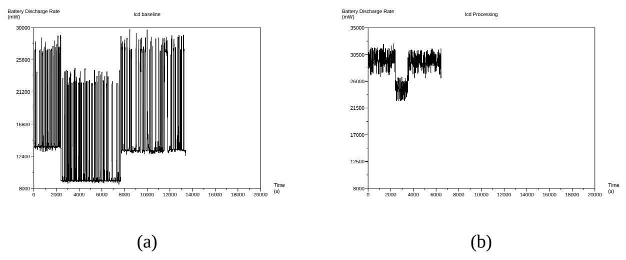

Fig. 2. Battery discharge rate for Dell C600 with Linux Debian and display on: (a) baseline, (b) processing

Table II shows the mean discharge rate and standard deviation for the basic tasks with display on. When comparing this table with Table I, we notice, as expected, an increase in the mean discharge rate, since the display increases the energy consumption.

Task	Mean	St. Deviation
Baseline	14.516 W	6.129
Processing	28.648 W	2.147

TABLE II
MEAN DISCHARGE RATE FOR BASIC TASKS WITH DISPLAY ON ON DELL C600 WITH LINUX DEBIAN

C. Combining Tasks

When we were in the process of choosing the tasks for our energy consumption benchmark, we conjectured that a generic user task consists of a combination of idle, processing, disk access and network communication. In this section, we aim at evaluating the energy consumption and the battery discharge behavior for different combinations of basic tasks. For instance, we choose the following combinations:

- Combo 1: processing (1.2s real time) and disk access: write (10MB of data, 2.8s real time) and read (10MB, 0.4s real time);
- Combo 2: processing (1.2s real time) and network transmission (10M o data, 2.3s real time);
- Combo 3: processing (1.2s real time), network transmission (10M o data, 2.3s real time) and disk reads (10MB, 0.4s real time).

We ran these benchmarks on the same platform, i.e. Dell C600 running Linux Debian.

Figure 3 presents the battery discharge rate curves for each of the composite tasks, i.e. Combo 1, Combo 2 and Combo 3.

The mean discharge rate and the standard deviation for each task are presented in Table III. The mean discharge rate for Combo 1 is less than the basic processing task (Table I, second row), but it is more that the disk access (Table I, third and fourth rows), which is the dominating task, i.e. the task that takes longer to execute within the combo. For Combo 2, the mean discharge rate is higher than the discharge rate for network transmission (Table I, fifth row), while for Combo 3 it is about the same as the discharge rate for disk writes (Table I, third row). From this results we note that the dominating task influences the most the discharge rate.

VI. DISCUSSION

Our premise in this paper is that task-level energy consumption information is key to achieving adequate power-aware task distribution in wireless distributed computing environments. Consider, for instance, the case of emergency rescue

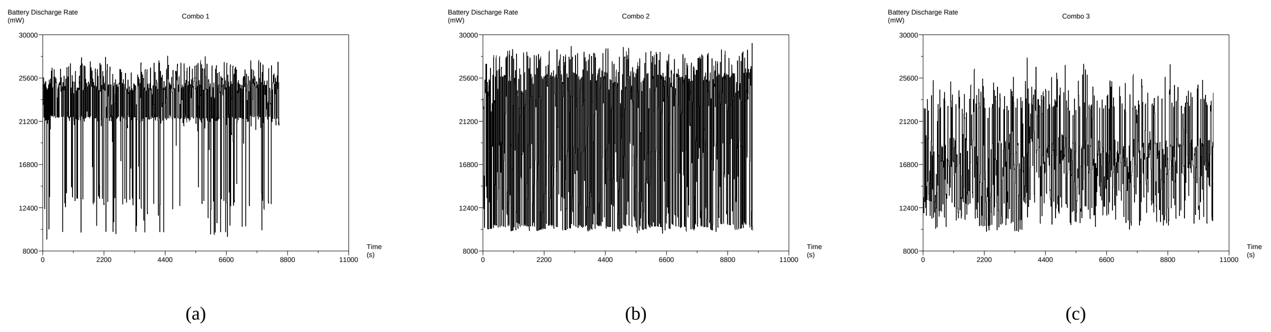

Fig. 3. Battery Discharge Rate for (a) Combo 1, (b) Combo 2 and (c) Combo 3, all on Dell C600.

Task	Mean	St. Deviation
Combo 1	22.455 W	4.343
Combo 2	19.511 W	6.754
Combo 3	17.486 W	4.392

TABLE III
MEAN DISCHARGE RATE FOR COMBINATION OF TASKS ON DELL C600 RUNNING LINUX DEBIAN

operations needed after a major (e.g., natural) disaster which destroyed (completely or partially) basic infrastructure such as the power grid and data communication network. Emergency rescue crews would then use their mobile wireless devices to perform all needed computation to assess damage, find survivors, etc. For example, in collapsed buildings, the rescue crew can use information from seismic sensors embedded in the building to perform structural assessment in order to find portions of the building that have (or not) been affected, what is the probability they will collapse (if they haven't yet), and when that will occur. This computation should be distributed among all (or some) of the participating networked nodes to load balance the computational load and the energy spent. To do so effectively, information on the current energy budget of the nodes as well as the amount of energy consumed by ”basic” tasks must be employed. By looking at the typical mix of ”basic” tasks to be executed, the ”task distribution manager” will be able to assess whether some node can take part on the computation and what is the operational lifetime of the network.

VII. RELATED WORK

In this section, we summarize related work in areas that are more relevant to our work, including energy consumption measurements for network interfaces and mobile devices, system's power management, storage energy consumption issues, etc.

Stem et al [22] measures the power consumption of some network interface cards (NICs) when used by different end-user devices. Authors also report on transport- and application-level strategies to reduce energy consumption by NICs. Later, Feeney et al [10] reported detailed energy consumption measurements of some commercially-available IEEE 802.11 NICs operating in ad hoc mode. Along the same lines, Erbert et al [9] assessed the impact of transmission rate, transmit power, and packet size on energy consumption in a typical wireless network interface.

In [15], energy consumption in ad hoc mobile terminals is modeled using the Advanced Configuration Power Interface [13], or ACPI. ACPI was used to measure energy consumption due to transmission/reception. The resulting energy consumption model includes two states: **high consumption state**, where the host receives and transmits, and **low consumption state**, where the node receives or is in idle. While this approach to model battery discharge empirically is based on values that laptop power management would see in real systems, it is platform-dependent.

In order to understand the issues on energy consumption of storage on mobile devices, Zheng et al [27] evaluated three different storage alternatives: a compact flash, a micro-drive and a wireless LAN card (which would be used to access a remote storage). By considering these different devices, their different power management schemes were studied, as well as the energy cost of their states were measured. Also the read/write latency and bandwidth was measured. Authors considered two types of files systems: update-in-place and log-structured. Results show that the energy consumption behavior depends on the device power management scheme, on the distribution of idleness in the workload, and on the file system strategies.

Dempsey [26] extends the Disk-Sim simulator to provide a simulation environment that also includes an energy consumption. Dempsey considers the power consumed for several different disk tasks, such as seek, rotation, read, write and stand-by.

Displays are a major consumer of energy. Iyer et al [14] discuss the use of energy-adaptive displays sub-systems. Authors use OLED displays and propose a software optimization called dark windows. Dark windows allows the windowing environment to change colors and brightness of areas that are not of interest to the user. According to a characterization of display usage done by authors, users use about 60% of the screen available. Thus, by changing the colors and brightness of areas not in the window of focus, energy can be saved.

Energy efficiency in mobile devices ranging from phones, laptops or hand-held devices is critical. Monticelli [17] presents an scheme using adaptive voltage scaling to control power management on 3G phones, and suggests that this same approach could be used in RF circuits. Yin et al [25] describe an power-aware prefetch scheme, which dynamically adjusts the number of prefetches based on the current energy level. Another approach to save energy on mobile devices is the remote power control of wireless interface cards [2], where a remote server uses its knowledge of the workload to perform traffic reshaping. Another approach is μSleep [7], which puts the processor in sleep mode for short periods (less than a second), and reduces energy consumption by up to 60%.

Balakrishnan et al.[4] propose a power management software architecture developed at user level using a standard power interface (ACPI - Advanced Configuration and Power Interface [13]) that provides information about current hardware state (e.g. estimated battery lifetime, temperature, etc.). This architecture is implemented and tested for disk spin down and thermal management.

Anand et al [3] developed a middle-ware for Linux on iPAQs in order to allow better power management. The power management implemented consider what is the state of all devices to be used (in this platform, wireless card and micro-drive), and hints given by the application when it requested data access and the device was not available. It also provides a user interface that allows one to control the priorities in terms of performance and power conservation. By taking these factors into account, authors show that it is possible to increase performance and decrease energy consumption.

PowerScope [11] is a tool that combines hardware instrumentation to measure current level and software support to perform statistical sampling of system activity, allowing energy profiling of process and procedures, which can then be optimized to reduce energy consumption.

Barr and Asanovic [5] present an interesting scheme considering the trade-offs between data transmission and compression. They used the Skiff platform[4] to perform measurements of energy consumption in wireless Ethernet card, StrongARM CPU, DRAM and Flash memory, under different compression algorithms. Since the energy cost of compressing data in each part of the hardware is know, as well as the cost to transmit data, authors propose a scheme where they compress data to be transmitted in case this procedure will allow energy savings.

VIII. Conclusions

Application-level power management, which is critical when wireless computing platforms are employed, can only be performed based on energy consumption information. In this paper, we presented a task-level energy consumption characterization benchmark that accounts for basic tasks such as processing, disk access (including read and write access), terminal usage, and communication (transmission and reception). Using this benchmark set, we characterized the power consumption of such tasks as presented in Section V on the Dell Latitude C600 running Linux.

Future work includes: (1) run more experiments considering other systems variables (hard-disk power management, control brightness on the display, test different processor speeds, etc.); (2) run other task combinations, so that we can cover a wider range of user-level applications; (3) run the benchmark in other mobile platforms, including more modern laptops as well as sensor network devices (e.g., Crossbow Stargates and motes [24]).

[4]The Skiff platform is based on the iPAQ hardware, but it has a differentiated circuitry to allow easy power consumption measurement.

References

[1] A. Aburto. FFT double precision benchmarks. ftp://ftp.nosc.mil/pub/aburto/, 2001.

[2] A. Acquaviva, T. Simunic, V. Deolalikar, and S. Roy. Remote power control of wireless network interfaces. In *Proc. of PATMOS in Lecture Notes in Computer Science*, Turin, September 2003. Springer-Verlag.

[3] M. Anand, E. B. Nightingale, and J. Flinn. Ghosts in the machine: Interfaces for better power management. In *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys 2004)*, Boston, USA, June 2004.

[4] S. Balakrishnan and J. Ramanan. Power-aware operating system using acpi. CS736 Project - Fall 2001, 2001.

[5] K. Barr and K. Asanovic. Energy aware lossless data compression. In *The First International Conference on Mobile Systems, Applications, and Services*, San Francisco, CA, May 2003.

[6] BIOS central. http://www.bioscentral.com, 2004.

[7] L. S. Brakmo, D. A. Wallach, and M. A. Viredaz. usleep: A technique for reducing energy consumption in handheld devices. In *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys 2004)*, Boston, USA, June 2004.

[8] B. Crow, I. Widjaja, J. Kim, and P. Sakai. Ieee 802.11 wireless local area networks. *IEEE Communications Magazine*, 35(9):116–26, September 1997.

[9] J. Ebert, S. Aier, G. Kofahl, A. Becker, B. Burns, and A. Wolisz. Measurement and simulation of the energy consumption of an WLAN interface. Technical Report TKN-02-010, Technical University Berlin, Telecommunication Networks Group, Germany, June 2002.

[10] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies.INFOCOM 2001*, volume 3, pages 1548–1557. IEEE, April 2001.

[11] J. Flinn and M. Satyanarayanan. Powerscope: A tool for profiling the energy usage of mobile applications. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, Louisiana, February 1999.

[12] S. Fowler. wmacpi: A battery monitor dockapp for ACPI based systems. http://himi.org/wmacpi-ng/, 2004.

[13] Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba. ACPI: Advanced configuration and power interface. http://www.acpi.info/, 2004.

[14] S. Iyer, L. Luo, R. Mayo, and P. Ranganathan. Energy-adaptive display system designs for future mobile environments. In *The First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, San Francisco, USA, June 2003.

[15] E. Lochin, A. Fladenmuller, J.-Y. Moulin, and S. Fdida. Energy consumption models for ad-hoc mobile terminals. In *Med-Hoc Net*, 2003.

[16] MandrakeSoft. Mandrakelinux. http://www.mandrakelinux.com, 2004.

[17] D. Monticelli. System approaches to power management. In *Applied Power Electronics Conference and Exposition, 2002. APEC 2002. Seventeenth Annual IEEE*, Dallas, TX, USA, March 2002.

[18] W. D. Norcott and D. Capps. IOzone filesystem benchmar. http://www.iozone.org/, 2004.

[19] B. Scheifler and D. Krikorian. Unix man pages: xset (1). http://www.mcsr.olemiss.edu/cgi-bin/man-cgi?xset+1, 2004.

[20] Standard performance evaluation corporation. http://www.spec.org, 2004.

[21] SPI. Debian – the universal operating system. http://www.debian.org/, 2004.

[22] M. Stemm and R. H. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Trans. on Communications*, 8(E80-B):1125–1131, 1997.

[23] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. Iperf. http://dast.nlanr.net/Projects/Iperf/, 2003.

[24] A. Woo. Mote documentation and development information. http://www.eecs.berkeley.edu/ awoo/smartdust/, 2000.

[25] L. Yin, G. Cao, C. Das, and A. Ashraf. Power-aware prefetch in mobile environments. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 571–578, 2002.

[26] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang. Modeling hard-disk power consumption. In *Second Conference on File and Storage Technologies*, San Francisco, USA, March 2003.

[27] F. Zheng, N. Garg, S. Sobti, C. Zhang, R. Joseph, A. Krishnamurthy, and R. Wang. Considering the energy consumption of mobile storage alternatives. In *MASCOTS'2003*, Orlando, USA, October 2003.