

# The Case for Higher-Level Power Management\*

Carla Schlatter Ellis  
Duke University  
carla@cs.duke.edu

## Abstract

*Reducing the energy consumed in the use of computing devices is becoming a major design challenge. While the problem obviously must be addressed with improved low-level technology, we claim there is potential value in a higher-level perspective, as well. In our approach, the needs of applications serve as the driving force for the development of power-management functions in the operating system and of a power-based API that allows a partnership between applications and the system in setting energy policy. The development of a PalmPilot application is used as an illustration. We advocate that reducing energy consumption should be raised to first-class status among performance goals when software is being designed. In support of this objective, new programming models, measurement tools, and system support mechanisms must be developed. These needs motivate our Milly Watt Project.*

## 1. Introduction

Mobile computing is dramatically changing our day-to-day lives, especially with the popularity of small devices such as personal digital assistants (PDAs) and with the embedding of substantial processing capabilities in devices such as telephones and cameras. Reducing the energy consumed in using these devices, thereby extending the lifetime of the batteries that power them (and reducing the weight of carrying spares), is one of the major challenges in designing such systems. Power consumption is an issue that extends well beyond the realm of battery-powered mobile devices to any computing platform in which heat or fan noise production may be factors (e.g. medical applications). Finally, energy efficiency of computers is desirable in its own right from both the economic and environmental points of view.

This problem can be addressed at various levels: by improving battery technology, by engineering more efficient electronics and components, and by designing computer architectures and software with power as a primary measure of performance. Power is a critical, limited, and shared system resource. Traditionally, the operating system plays a major role in managing such resources. We also believe that the applications hold an important key, in the application-specific constraints and opportunities for saving energy that can be known only at that level. Therefore, our approach is to have the needs of the applications be the driving force for the development of power management functions in the operating system and of a power-based API that allows a partnership between applications and the system in setting energy use policy.

Our higher-level perspective on the energy use problem grew out of firsthand experience with the design of a palmtop application, which we discuss in the next section. In that exercise, we were frustrated in our attempts to tailor the energy use to the specific needs of the application by several factors: a lack of feedback on what effect various design decisions might have on energy use, an inadequate programming model of power consumption in the target platform, a gap between our vision of how the application should control its energy use and the ability to express that to the system, and limitations on the flexibility of interactions between the operating system and the given architecture. Identifying and alleviating these drawbacks in the state-of-the-art of energy conservation has motivated our Milly Watt Project<sup>1</sup>.

To summarize our position statement, we advocate the following points:

- Software (applications and OS) can have a positive impact on and should be involved in power management.

---

\*This work is supported, in part, by NSF grant CDA-9720545.

---

<sup>1</sup>The name was inspired by Reddy Kilowatt, a mascot for the electric power industry, whose heyday was in the 1950's.

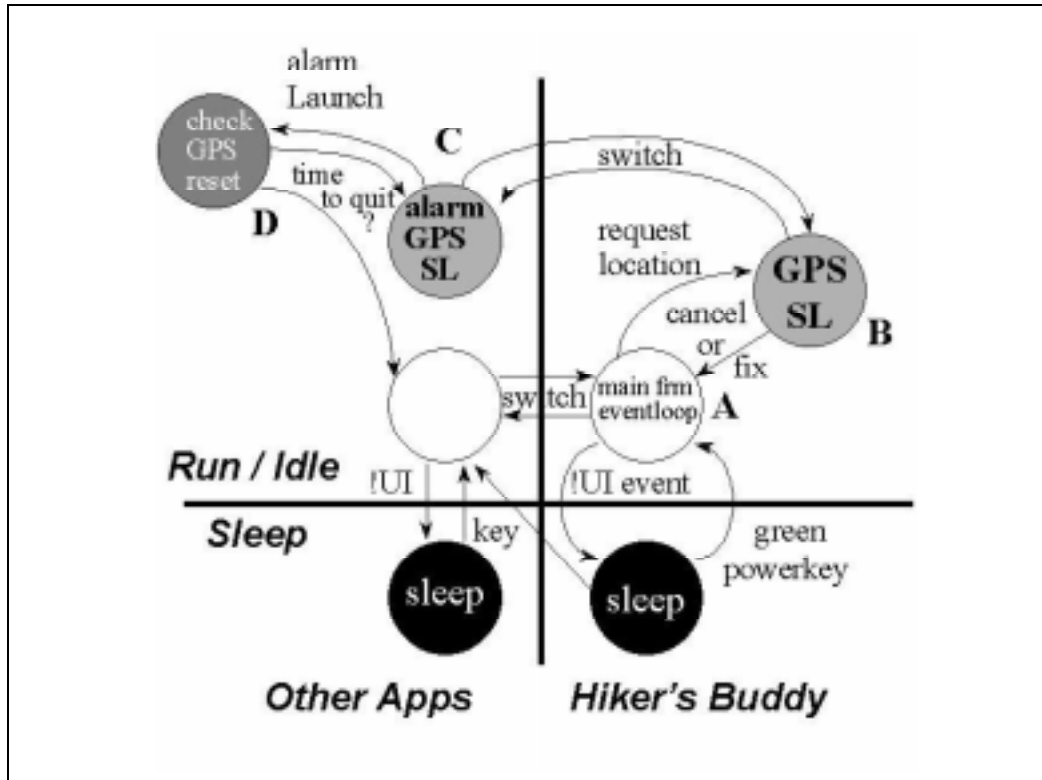


Figure 1. Power State Model

- Energy consumption will be a major concern in emerging applications of computing technology. Therefore, performance measurement studies should regularly include appropriate energy metrics in addition to the more traditional ones (e.g. time, bandwidth).
- Tools and mechanisms for power management are currently inadequate and require more research and development efforts.

## 2. A Concrete Example

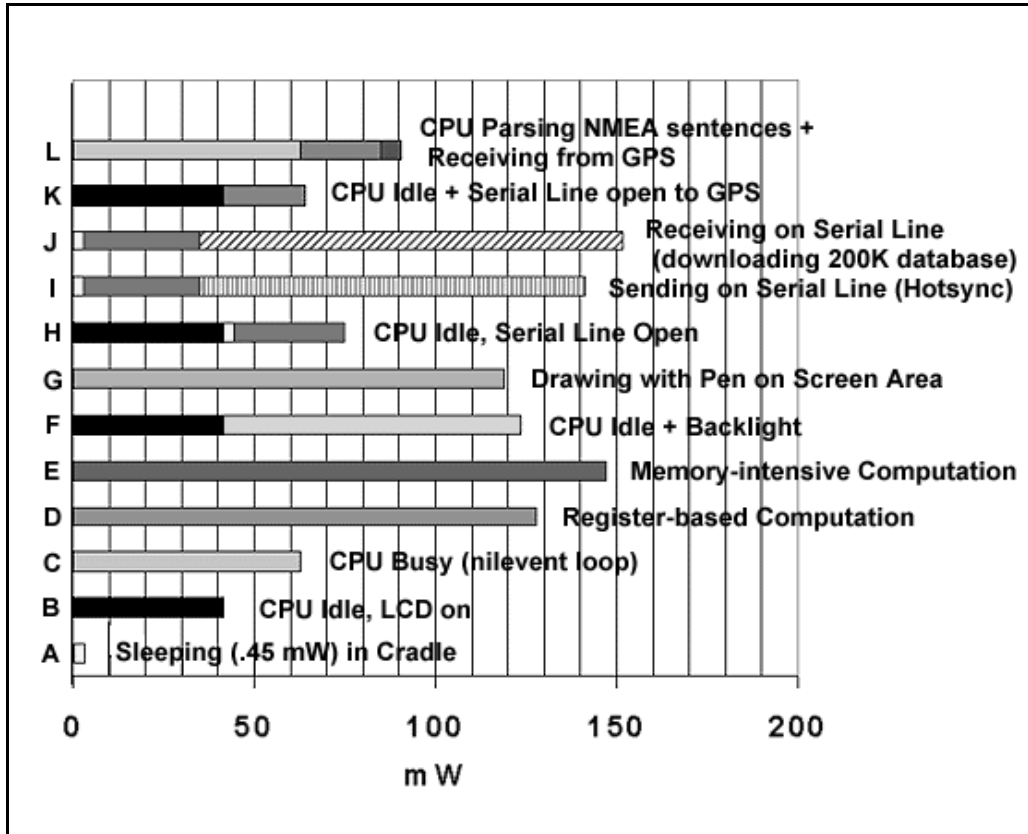
We describe a preliminary case study that encompasses most of the aspects of our research agenda. We chose to build a compelling mobile application for the PalmPilot and then to study application-directed power management in the context of that specific application and device. Our current work involves generalizing from this experience.

The application we chose to build is called Hiker's Buddy. It takes NMEA protocol strings sent by a Global Positioning System (GPS) receiver (a Delorme Tripmate<sup>™</sup>) and plots the hiker's current location on a topographical map found by searching a database of map

segments that have been transcoded for the PalmPilot's limited display and then downloaded from a server prior to starting out on the hike. This application is a particularly appealing vehicle: it is necessarily mobile and battery-powered; the small form factor and light weight of the device are assets to this task and not simply constraints of today's technology; it involves an external, also battery-powered peripheral; and it is a natural for disconnected operation (it is assumed that the user might be hiking out of coverage range for wireless communications) although plausible roles for intermittent wireless connectivity present themselves.

Many of the first problems that arose in programming Hiker's Buddy had power-management implications. In particular, interactions between the external GPS device and the standard PalmOS power management were poorly integrated. While trying to characterize the *correctness* issues that were raised by these interactions, we developed a model of power states for Hiker's Buddy.

The high level representation (shown in a simplified form in Figure 1) was originally intended to capture only the qualitative aspects of the power-related behavior, including interactions with other applications. For example, the interpretation of the state labeled A would be



**Figure 2. Measured Power Consumption (PalmPilot Pro and Hiker’s Buddy)**

“executing the main event-handling loop, which implies display:on, serial line:closed, GPS:off, processor: predominately idle, and infrequent pen inputs”; whereas state *B* means “GPS polling loop, with serial line:open, GPS:on, auto off: disabled (to prevent entering sleep mode which would drop the serial line and turn the GPS unit off), CPU alternating between idle and receiving/parsing the NMEA string, and display:on”. The states to the right of the heavy vertical line represent Hiker’s Buddy as the foreground application; whereas, those states to the left of this line that pertain to Hiker’s Buddy (*C* and *D*) capture the device requirements remaining when another application is in the foreground. Thus, state *C* means “GPS left running, with serial line open and an alarm set to return control to Hiker’s Buddy within an appropriate time interval”.

We later recognized that the model would also be helpful in designing power consumption measurement experiments. The modeling methodology is undergoing further development, especially related to “composing” the power demands of multiple activities.

### 3. Experimentation

It is impossible to address power management without an understanding of what different operations cost in terms of power consumption. Thus we designed a subproject to quantify the power consumption of the PalmPilot, including both generic operations and functions specific to Hiker’s Buddy.

The experiments were done by developing a driver application that could put the PalmPilot into a known power state for a specified test period during which measurements of current were performed with a multimeter. Voltage values were obtained from the sysBatteryInfo system call before and after each of the individual tests. A PalmPilot Professional (1 MB) running PalmOS version 2.0.4 was used. The results are shown in Figure 2 (the details of each test and interpretation of the results are available at [18]). The measured power values can be associated with particular power states in our model. For example, state *B* in Figure 1 corresponds with the bar in Figure 2 labeled *L*. Some of the bars show a breakdown of estimated contributions of various factors to the overall result. Bar *L* shows estimates of the power

consumed by the CPU activity (light gray), by having the serial line open to the GPS receiver (medium gray) and receiving data over it from the GPS (dark gray), for the measured power level. The driver was running the GPS polling and NMEA parsing loop during the test interval to produce this value.

From Figure 2 we can identify which of the activities offered by Hiker's Buddy incur the greatest power costs and appear to be good candidates for optimization. Of particular significance are bars *L*, the GPS polling loop, *J*, corresponding to the map database download, and *E*, a memory-intensive computation. Bar *E* was intended to represent a worst case power cost of searching the map database. Subsequent measurements of the actual map search routine has yielded a more moderate cost of just under 100 mW for this operation (not shown).

Since energy expenditure depends on both the power consumed and the elapsed time, we instrumented Hiker's Buddy to capture the time spent in various power states during execution. Then, assuming the correspondence between power states and measured power levels, we did coarse estimates of the energy expended for a number of alternatives. Since downloading of new maps occurs only occasionally (in addition to happening at home near sources of new batteries), this operation was considered unlikely to cause a serious problem even though the power cost is high. Thus, we focused on the acquisition and use of the GPS data in the field.

One set of experiments has studied varying the polling frequency for the GPS data. This is an interesting case because it involves nontrivial tradeoffs between processor idleness and the length of time required to recognize a GPS fix. Intuitively, the faster the polling and NMEA parsing occurs, the quicker Hiker's Buddy can discover that there is a good GPS fix available thus reducing the time spent in a costly power state. Failure to immediately recognize that the GPS has determined its location means unnecessary time with the serial line open and the display kept on. It routinely can take as long as three minutes for the GPS unit to acquire the requisite number of satellites and return a valid position; so further delay is undesirable. On the other hand, polling less frequently allows the PalmPilot processor to spend more time in the idle mode, thus lowering the overall power cost of this loop. If this does not significantly affect the total time for the loop, energy use is reduced. It is not immediately obvious how to choose a timeout value for this polling loop to balance these factors. Experiments have indicated that an adaptive polling frequency that begins slowly, with significant processor idle time, then

shortens the polling interval when the fix is getting close (evident in the NMEA sentences as more satellites are acquired) is usually a better policy for energy use than any fixed value. The best case used an adaptive strategy with a base timeout of 90 seconds and a subsequent timeout of either 1 or 10 seconds and resulted in energy consumption of from 6 to 9 Joules to get one position. This compared to polling with a fixed timeout of 90 seconds which consumed from 12 to 14 Joules. Smaller base timeouts (for both fixed and adaptive algorithms) yielded marginally higher energy costs than our best case because of increases in processor busy time. Any delay imposed on recognizing a fix was the more significant factor, however, affecting the larger timeouts.

As a feasibility study of our higher-level approach, our experiments have addressed the question of whether design decisions made at the application-level can have an impact on the energy consumption of using Hiker's Buddy. Our preliminary work demonstrates that the application designer can, in fact, make a difference if given the tools to tune the energy behavior of the application program.

Promising results were achieved with minimal OS support. However, the system interface stifled further efforts to fully exploit the power savings opportunities offered by our application. Requirements that were straightforward to articulate could not be simply implemented because of the assumed coupling between components in the available power management system calls. For example, Hiker's Buddy had no need for the display to be on while the GPS was acquiring satellites, but this came with the requirement to keep the serial line open to the GPS.

From this experience, we have begun to compile a wish-list of capabilities for a power-wise OS and what should be exposed to the applications layer. An obvious first step would be to decouple the states of various devices so they can be independently specified. A notification mechanism for imminent power-related events (e.g. the device is about to enter into sleep mode) with an opportunity to respond would be useful. It should be possible to express both hints and requirements to the system. A more difficult issue involves "background" activity (e.g. maintaining the serial line state during GPS polling, even when switching to another foreground application) and how the power requirements of multiple applications are to be combined. In the PalmOS model, setting an alarm that will result in a later alarm launch seems a likely place to attach lingering power constraints.

## 4. Conclusions

The potential benefits of application-level involvement in power management have been largely untapped in work to date. There is a substantial body of work in hardware design for low power, both at the circuit design level and at the architectural level. Since we are primarily interested in software-based power management, the architectural features, of interest, are those that offer explicit power management “hooks” that software can directly exploit. Input to architects about OS/application needs might enhance the low power architecture work significantly, but this requires more attention to power by the OS research community. Previous OS-level studies focusing on power management include work on scheduling for low power processor modes [12, 13, 16], spindown policies for disks and alternatives [1, 2, 3, 4, 6, 10, 11, 17], and managing wireless communication [7, 9, 15]. A consortium of companies has developed a specification [8] that addresses the lower-level OS / device interface, providing one model for gross system-wide power states and per-component power states as a basis for the development of OS-directed power management. The OS / application power management interface has been largely ignored. Odyssey [14] is probably the best example of work aimed at the interface for application-aware resource management (although not specifically addressing power issues) and, certainly, notification mechanisms (key in Odyssey) show up on our wish-list. Finally, in the area of tools for power measurement, PowerScope [5] combines hardware instrumentation with CPU profiling techniques to map power consumption to program structure.

The key point of this position paper is that reducing energy consumption should be raised to first-class status among the performance goals of OS design and application-involvement should be encouraged from the outset. Even for OS developments that are not explicitly aimed at energy conservation, it is legitimate and valuable to raise the question of their impact on energy consumption.

## 5. References

[1] Mary Baker, Satoshi Asami, Etienne Deprit, John Ousterhout, and Margo Seltzer, “Non-volatile memory for fast, reliable file systems” in *Proceedings of the 5<sup>th</sup> International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 1992, pp. 10-22.

[2] Fred Douglass, P. Krishnan, and Brian Bershad, “Adaptive Disk Spin-down Policies for Mobile Computers”, *2<sup>nd</sup> USENIX Symposium on Mobile and Location-Independent Computing*, April 1995.

[3] Fred Douglass, Ramon Caceres, Brian Marsh, Frans Kaashoek, Kai Li, and Joshua Tauber, “Storage alternatives for mobile computers”, in *Proceedings of the First Symposium on Operating Systems Design and Implementation (OSDI)*, Monterey CA, Nov. 94, pp. 25-37.

[4] Fred Douglass, P. Krishnan, and Brian Marsh, “Thwarting the Power Hungry Disk”, in *Proceedings of the 1994 Winter USENIX Conference*, Jan. 1994, pp. 293-306.

[5] Jason Flinn and M. Satyanarayanan, “PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications”, *Second Workshop on Mobile Computing Systems & Applications (WMCSA’99)*, New Orleans, February, 1999.

[6] D. Helmbold, D. Long, and B. Sherrod, “A Dynamic Disk Spin-Down Technique for Mobile Computing”, *Proc. of the 2nd ACM International Conf. on Mobile Computing (MOBICOM96)*, 130-142, Nov. 1996.

[7] Tomasz Imielinski, Monish Gupta, and Sarma Peeyeti, “Energy Efficient Data Filtering and Communications in Mobile Wireless Computing”, *Proceedings of Usenix Symposium on Location Dependent Computing*, April, 1995.

[8] Intel Corporation, Microsoft Corporation, and Toshiba Corporation, “Advanced Configuration and Power Interface Specification”, Dec. 1996, <http://www.teleport.com/~acpi>

[9] R. Kravets, and P. Krishnan, “Power Management Techniques for Mobile Communication”, *Proc. of the 4th International Conf. on Mobile Computing and Networking (MOBICOM98)*, 157-168, Oct. 1998.

[10] P. Krishnan, P. Long, and J. Vitter, “Adaptive disk spin-down via optimal rent-to-buy in probabilistic environments”, *Proceedings of the 12<sup>th</sup> International Conference on Machine Learning*, July 1995, pp. 322-330.

[11] Kester Li, Roger Kumpf, Paul Horton, and Thomas Anderson, “A Quantitative Analysis of Disk Drive Power Management in Portable Computers”, *USENIX*

*Association Winter Technical Conference Proceedings*, 1994, pp. 279-291.

[12] Jacob Lorch and Alan J. Smith, "Reducing processor power consumption by improving processor time management in a single-user operating system", *Proc. of the 2nd ACM International Conf. on Mobile Computing (MOBICOM96)*, 143-154, Nov. 1996.

[13] J. Lorch and A.J. Smith, "Scheduling Techniques for Reducing Processor Energy Use in MacOS", *Wireless Networks*, 3(5), 311-324, Oct. 1997.

[14] B. Noble, M. Price, and M. Satyanarayanan, "A programming interface for application-aware adaptation in mobile computing", *Computing Systems*, 8(4):345-363, 1995.

[15] Mark Stemm and Randy Katz, "Measuring and Reducing energy consumption of network interfaces in hand-held devices", *Proceedings of 3<sup>rd</sup> International Workshop on Mobile Multimedia Communications (MoMuC-3)*, Sept. 1996.

[16] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker, "Scheduling for Reduced CPU Energy", USENIX Association, *Proceedings of First Symposium on Operating Systems Design and Implementation (OSDI)*, Monterey CA, Nov. 94.

[17] John Wilkes, "Predictive Power Conservation", Technical report HPL-CSP-92-5, Hewlett-Packard Labs, Feb. 1992.

[18] [//www.cs.duke.edu/~carla/HB\\_power.html](http://www.cs.duke.edu/~carla/HB_power.html)