Every Joule is Precious: The Case for Revisiting Operating System Design for Energy Efficiency

Amin Vahdat, Alvin Lebeck, and Carla Schlatter Ellis
Department of Computer Science
Duke University
Durham, NC 27708–0129
{vahdat,alvy,carla}@cs.duke.edu

1. Introduction

By some estimates, there will be close to one billion wireless devices capable of Internet connectivity within five years, surpassing the installed base of traditional wired compute devices. These devices will take the form of cellular phones, personal digital assistants (PDA's), embedded processors, and "Internet appliances". This proliferation of networked computing devices will enable a number of compelling applications, centering around ubiquitous access to global information services, just in time delivery of personalized content, and tight synchronization among compute devices/appliances in our everyday environment. However, one of the principal challenges of realizing this vision in the post-PC environment is the need to reduce the energy consumed in using these next-generation mobile and wireless devices, thereby extending the lifetime of the batteries that power them. While the processing power, memory, and network bandwidth of post-PC devices are increasing exponentially, their battery capacity is improving at a more modest pace.

Thus, to ensure the utility of post-PC applications, it is important to develop low-level mechanisms and higher-level policies to maximize energy efficiency. In this paper, we propose the systematic re-examination of all aspects of operating system design and implementation from the point of view of energy efficiency rather than the more traditional OS metric of maximizing performance. In [7], we made the case for energy as a first-class OS-managed resource. We emphasized the benefits of higher-level control over energy usage policy and the application/OS interactions required to achieve them. This paper explores the implications that this major shift in focus can have upon the services, policies, mechanisms, and internal structure of the OS itself based on our initial experiences with rethinking system design for energy efficiency.

Our ultimate goal is to design an operating system where major components cooperate to explicitly optimize for energy efficiency. A number of research efforts have recently investigated aspects of energy-efficient operating systems (a good overview is available at [16, 20]) and we intend to leverage existing "best practice" in our own work where such results exist. However, we are not aware of any systems that systematically revisit system structure with energy in mind. Further, our examination of operating system functionality reveals a number of opportunities that have received little attention in the literature. To illustrate this point, Table 1 presents major operating system functionality, along with possible techniques for improving power consumption characteristics. Several of the techniques are well studied, such as disk spindown policies or adaptively trading content fidelity for power [8]. For example, to reduce power consumption for MPEG playback, the system could adapt to a smaller frame rate and window size, consuming less bandwidth and computation.

One of the primary objectives of operating systems is allocating resources among competing tasks, typically for fairness and performance. Adding energy efficiency to the equation raises a number of interesting issues. For example, competing processes/users may be scheduled to receive a fair share of *battery* resources rather than CPU resources (e.g., an application that makes heavy use of DISK I/O may be given lower priority relative to a compute-bound application when energy resources are low). Similarly, for tasks such as ad hoc routing, local battery resources are often consumed on behalf of remote processes. Fair allocation dictates that one battery is not drained in preference to others. Finally, for the communication subsystem, a number of efforts already investigate adaptively setting the polling rate for wireless networks (trading latency for energy).

Our efforts to date have focused on the last four areas highlighted in Table 1. For memory allocation, our work explores how to exploit the ability of memory chips to transition among multiple power states. We also investigate met-

Operating System Functionality	Energy Efficient Techniques
Disk scheduling	Spindown policies [18, 6, 5, 14, 11]
Security	Adaptive cryptographic policy based on computation/ communication overhead
CPU scheduling	Voltage scaling, idle power modes [32, 19, 22]
Application/OS Interaction	Agile content negotiation trading fidelity for power, APIs [8]
Memory allocation	Adaptive placement of memory blocks, switching of hardware energy conservation modes
Resource Protection/Allocation	Fair distribution of battery life among both local and distributed tasks, "locking" battery for expensive operations
Communication	Adaptive network polling, energy-aware routing, placement of distributed computation, and server binding [27, 13, 28, 25, 26]

Table 1. Operating system functionality along with potential techniques for optimizing energy utilization.

rics for picking energy-efficient routes in ad hoc networks, energy-efficient placement of distributed computation, and flexible RPC/name binding that accounts for power consumption.

These last two points of resource allocation and remote communication highlight an interesting property for energy-aware OS design in the post-PC environment. Many tasks are distributed across multiple machines, potentially running on machines with widely varying CPU, memory, and power source characteristics. Thus, energy-aware OS design must closely cooperate with and track the characteristics of remote computers to balance the often conflicting goals of optimizing for energy and speed.

The rest of this paper illustrates our approach with selected examples extracted from our recent efforts toward building an integrated hardware/software infrastructure that incorporates cooperative power management to support mobile and wireless applications. The instances we present in subsequent sections cover the resource management policies and mechanisms necessary to exploit low power modes of various (existing or proposed) hardware components, as well as power-aware communications and the essential role of the wide-area environment. We begin our discussion with the resources of a single machine and then extend it to the distributed context.

2. Resource Management

A fundamental OS task is efficient management of host resources. With energy as the focus, the question becomes how to make the basic interactions of hardware and software as energy efficient as possible for local computation (e.g., disconnected operation). One trend observed in traditional, performance-centric resource management involves latency hiding techniques. A significant difference and challenge in energy-centric resource management is that power consumption is not easy to hide.

As one instance of power-aware resource management, we consider memory management. Memory instructions are among the more power-hungry operations on embedded processors [29], making the hardware/software of memory management a good candidate for optimization. Intel's guidelines for mobile power [12] indicate that the target for main memory should be approximately 4% of the power budget (e.g. an average 1.3W for 96MB)for year 2000 laptops. This percentage can dramatically increase in systems with low power processors (e.g., Transmeta Crusoe [10]), displays [21], or without hard disks. Since many small devices have no secondary storage and rely on memory to retain data, there are power costs for memory even in otherwise idle systems. The amount of memory available in mobile devices is expanding with each new model to support more demanding applications (e.g., multimedia) while the demand for longer battery life also continues to grow significantly.

Our work on energy in memory systems [15] is motivated by the emergence of main memory technologies for portable systems composed of one or more DRAM devices with the ability to control the power state of individual memory chips. Direct Rambus DRAM (RDRAM) [23] is one concrete example. RDRAM's novel communication protocol allows individual devices to be in any of the following power states in decreasing order of power consumption and increasing order of access time: Active, Standby, Nap, and Powerdown. Figure 1 shows the power states and their *relative* power costs as well as the possible transitions and *relative* transition times into active mode, as required for access.

Our work investigates intelligent page allocation for energy efficiency that can be used by an informed OS to com-

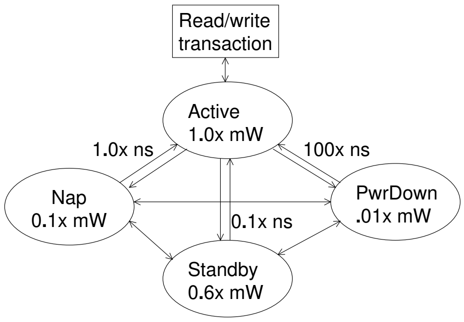

Figure 1. RDRAM Power States

plement hardware power management strategies. We consider both static and dynamic hardware policies for determining the power state of a memory chip. A static hardware policy is one in which every power-aware DRAM chip in the system resides in the same base power mode when there are no outstanding memory requests for that device. A dynamic hardware policy tries to determine the appropriate power state for a chip, based on access patterns, using thresholds of time between accesses to trigger transitions to the next lower power state. OS support appears to be crucial for fully exploiting such hardware capabilities. We consider the effect of code and data placement policies that choose the mapping of virtual pages to physical locations within power-aware memory chips. Our "sequential first-touch" allocation policy aims to group active pages according to their temporal locality. It allocates pages in the order they are accessed, filling an entire chip before moving on to the next. In this way, we place related pages within the smallest number of DRAM devices, in effect trading power for available parallelism. This technique enables more devices to be in a low-power state while providing performance close to that achieved by placing all DRAM devices in the fastest but highest power state. Further, this technique can reduce the energy overhead of reading data from memory.

We have explored the effectiveness of these policies at improving energy efficiency in a set of simulation experiments using two simulators: a trace-driven simulator and a detailed out-of-order execution-driven processor simulator. Our trace-driven simulation uses a simplified processor and memory system model to process instruction and data address traces from a set of productivity applications [17] as a workload representative of mobile laptop and handheld devices. We also use an execution-driven simulator with a more detailed processor and memory model to evaluate a set of programs from the integer SPEC2000 suite that place higher demands on the memory system than the available traces. We measure the energy savings within the RDRAM-based memory system and any additional delay in execution time resulting from these power-aware page placement strategies, expressed in terms of an Energy•Delay metric.

Our simulation results show that

- Power-aware page allocation by an informed operating system coupled with dynamic hardware policies can dramatically improve energy efficiency of memory. Power-aware allocation allows a 6% to 50% improvement in Energy•Delay over the best static hardware policy which uses nap as its base power state. This translates to an improvement of 99% to 80% over a traditional full-power memory system without cooperative hardware/software policies.

- Power-aware page allocation when used with just static hardware policies can improve Energy•Delay by 12% to 30%.

- Dynamic hardware policies without informed OS support (i.e., using random page allocation) do not improve energy efficiency as measured by Energy•Delay.

3. Communications

Another fundamental OS function is to support communication among computing devices. As new devices (e.g., PDAs, sensors, etc.) continue to permeate our daily lives, wireless communication becomes an ever more important aspect of system design. However, the combination of new scenarios and restrictions on energy consumption create new challenges for designing wireless communication substrates.

Recent advances in ad hoc networks (for an overview see [1]) allow mobile devices to communicate with one another, even in the absence of pre-existing base-stations or routers. All mobile devices are able to act as routers, forwarding packets among devices that may otherwise be out of communication range of one another. Important challenges include discovering and evaluating available routes among mobile devices and maintaining these routes as devices move, continuously changing the "topology" of the underlying wireless network. In applications with limited battery power, it is important to minimize energy consumption in supporting this ad-hoc communication.

There are numerous opportunities for power optimizations in such environments, including: i) reducing transmission power adaptively based on the distance between sender and receiver, ii) adaptively setting transmission power in route discovery protocols, iii) balancing hop count and latency against power consumption in choosing the "best" route between two hosts, and iv) choosing routes to fairly distribute the routing duties (and the associated power consumption) among nodes in an ad-hoc network [27].

Our initial investigations exploit the following property of wireless communication: received signal strength decreases exponentially with increased distance from the transmitter. In the simplest model, the signal decreases with the square of the distance from the transmitter [24], and decreases even faster when obstacles or ground reflection are considered. Although wireless network interfaces can be a large component of overall power consumption [13, 28], in many cases transmitter power consumption may be larger than required. Many devices transmit a signal strong enough to reach the receiver from their maximum range independent of their actual distance from the receiver.

We are currently investigating the design, implementation, and evaluation of a power-aware version of the DSR (Dynamic Source Routing) ad-hoc routing protocol that we call DSRp. We modify the RTS/CTS exchange in the IEEE 802.11 specification to determine the amount of power required to transmit to a given destination (such hardware support will be available, for example, in Bluetooth devices). The fundamental observation is that five ten-meter hops to a given destination will likely consume more power than ten five-meter hops (at the cost of increased store and forward latency). We use this information to augment route discovery to include end-to-end power requirements in addition to hop count, allowing the sending host to optimize its route selection along multiple axes. For example, time sensitive communication may minimize latency, while normal communication may minimize power. Our preliminary results, using the monarch/ns simulation infrastructure, show that adaptively reducing transmit power can reduce overall energy consumption by 32% to 56% for minimum hop-count routes.

One interesting implication from our work is that the total power consumed on behalf of wireless communication can be dramatically reduced by limiting transmission power. Our work in DSRp attempts to find power-optimal connected paths through ad hoc networks. A related effort into Epidemic Routing [31] demonstrates that it is possible to deliver application data among mobile hosts even when there is never a connected path from source to destination. In this work, we leverage pairwise connectivity among hosts to selectively flood application data (as oppossed to the control packets typically flooded for route discovery) through connected subsets of an ad hoc network. Unseen messages are exchanged among hosts when they come into contact with one another. We employ per-message hop limits and a FIFO queue to limit the resources consumed in delivering any single packet. Our simulation results in monarch/ns demonstrate high message delivery rates (100% if no resource limits are enforced given random movement), with message delivery latencies directly predicted by the relative connectivity of the underlying network.

With respect to energy consumption, Epidemic Routing demonstrates that it is possible to maintain high delivery rates in ad hoc networks even when unilaterally reducing the maximum transmission range of network interface cards to the point where not all hosts are directly connected to one another. For example, reducing tranmission range to 70% of maximum could reduce power consumption in transmitting packets by 50%. Such mechanisms will be increasingly important with the emergence of short-range wireless technologies (e.g., Bluetooth [9]) where power is a primary consideration.

4. Leveraging Remote Computation

Many post-PC applications depend upon location-independent access to Internet resources. System support for energy efficient access to data located on the web is thus an important priority for an energy-aware OS. Beyond that, the question becomes whether remote computational resources can also be exploited to conserve local energy consumption. Rather than statically assigning program execution to the local host (e.g., handheld device, PDA), the system may dynamically determine that it is more efficient to migrate a program to a remote site (e.g., with line power or with more computational resources) and to retrieve its results when execution completes. If the time or energy saved on the mobile device is more than the time or energy consumed to: i) send the process, ii) idle until results can be retrieved, and iii) receive the results, then this method may be effective for conserving either time or energy. Thus, energy can be viewed as an available Internet resource similar to other remote resources.

Our approach is based upon achieving such efficiency through mobile code—programs able to dynamically migrate and replicate in response to the local and wide-area environment. Mobile code is emerging as the enabling technology for transparent access to Internet resources.

This work extends our previous work in mobile code within the WebOS project [30] to embrace mobile and wireless clients. There are a number of similarities between wide-area and mobile/embedded systems—for instance, highly unpredictable network and computational characteristics and the associated need to address such variability. Our primary goal is to support intelligent program placement to achieve an appropriate balance between performance and energy consumption. We are developing energy efficient mechanisms for placement and migration of programs originating from diverse sources (e.g., handheld computers to servers) as well as policies to determine the conditions under which a program should migrate.

The considerations and tradeoffs are inherently different when optimizing for time versus power. The client-side must evaluate the cost (in time and energy) of transferring the computation over a slow and/or expensive wireless link versus continuing to run it locally, with variable and uncertain information upon which to base its decision. Our initial investigations have shown that the key to effectively using remote energy for execution is to be *very selective* in choosing the processes which are likely to benefit, where the computation is significant relative to the data shipped and the processor/implementation on the server receiving the task is significantly faster than on the mobile device. We believe that application-specific knowledge will be required to determine the proper tradeoffs.

An interesting problem in this domain is the need to maintain information about network characteristics and remote computational resources. For example, to determine whether a remote server is an appropriate target for offloading local computation, a mobile host must estimate the characteristics of the network between it and the remote server (to determine the amount of time/energy required to transmit the request and to retrieve the results) as well as remotely available CPU, memory, and disk resources (to determine the amount of time necessary to carry out the request at the remote site). The naive approach to maintaining such information might involve periodic communication among the mobile host and remote sites to track remote characteristics. We are investigating the use of anti-entropy techniques and efficient bounding of numerical error [33] (e.g., remote CPU load or available bandwidth) in replicated data to minimize the communication required to maintain an accurate view of remote resources.

5. Content Specialization

One special case of exploiting remote execution is the use of a transcoding proxy for transforming multimedia web content to save bandwidth (and thus energy consumed by the local device by receiving less data) or to specialize image data for display, given the destination device characteristics (thus saving energy for rendering). Unfortunately, one danger with aggressively transcoding is that the result delivered may become degraded in quality so much as to be unusable. In our preliminary work [2], we have developed an informed *quality-aware transcoding* technology that addresses this deficit, preserving quality by allowing transcoding to be selectively applied. We have demonstrated [3] that this technique can be applied at a proxy to reduce the wireless bandwidth required to deliver good quality images.

Mobile devices can also achieve improved average image quality and reduced power consumption by performing transcoding operations locally. In [4], we show that our techniques allow a battery-limited mobile/wireless device that generates images (e.g., a digital camera) to maximize the number of quality pictures that can be stored before memory or battery resources are exhausted.

6. Implementation and Structure

Finally, we need to reconsider the internal structure and activity of the OS itself from this new perspective as existing structures are likely to be wasteful of energy. For example, daemon processes often perform periodic maintenance functions (e.g. compaction, garbage collection); however, any nonessential activity is an energy drain and possibly should be deferred until the device is once again plugged in and fully charged unless the energy costs of operating in a degraded state (e.g. highly fragmented disk) outweigh the overhead. Tradeoffs have to be reevaluated for techniques that exploit "idle" cycles to perform optional management functions since they can no longer be considered free when energy consumption replaces time as the important metric.

7. Summary and Conclusions

This paper advocates revisiting all aspects of Operating System design and implementation with energy-efficiency as the primary objective rather than the traditional OS metrics of maximizing performance and fairness. Energy is an increasingly important resource for post-PC battery-powered devices. Our initial investigations have indicated that this change in viewpoint has significant implications for the services, policies, mechanisms, and structure of the OS. We illustrate with examples of power-aware memory management, power-aware ad hoc routing protocols, and energy exploitation of remote resources in the context of a cooperative wide-area distribution system that encompasses mobile and wireless devices.

References

[1] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, October 1998.

[2] S. Chandra and C. S. Ellis. JPEG Compression Metric as a Quality Aware Image Transcoding. In *2nd Symposium on Internet Technologies and Systems*, Boulder, CO, October 1999. USENIX.

[3] S. Chandra, C. S. Ellis, and A. Vahdat. Multimedia Web Services for Mob ile Clients Using Quality Aware Transcoding. In *Proceedings of the Second ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM'99)*, Seattle, WA, August 1999. ACM SIGMOBILE.

[4] S. Chandra, C. S. Ellis, and A. Vahdat. Managing the storage and battery resources in an image capture device (digital camera) using dynamic transcoding. In *Proceedings of*

the Second ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM'00), August 2000.

[5] F. Douglis, P. Krishnan, and B. Bershad. Adaptive Disk Spin-down Policies for Mobile Computers. In *2nd USENIX Symposium on Mobile and Location-Independent Computing*, April 1995. Monterey CA.

[6] F. Douglis, P. Krishnan, and B. Marsh. Thwarting the Power Hungry Disk. In *Proceedings of the 1994 Winter USENIX Conference*, pages 293–306, January 1994.

[7] C. S. Ellis. The Case for Higher-Level Power Management. In *Proceedings of the 7th Workshop on Hot Topics in Operating Systems*, Rio Rico, AZ, March 1999.

[8] J. Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *Symposium on Operating Systems Principles (SOSP)*, pages 48–63, December 1999.

[9] J. Haartsen, M. Naghshineh, J. Inouye, O. J. Joeresson, and W. Allen. Bluetooth: Vision, Goals, and Architecture. *ACM Mobile Computing and Communications Review*, 2(4):38–45, October 1998.

[10] T. Halfhill. Transmeta breaks x86 low-power barrier. *Microprocessor Report*, February 2000.

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

[12] Intel Corporation. Mobile Power Guidelines 2000. `ftp://download.intel.com/design/mobile/intelpower/mpg99r1.pdf`, December 1998.

[13] R. Kravets and P. Krishnan. Power Management Techniques for Mobile Communication. In *Proc. of the 4th International Conf. on Mobile Computing and Networking (MOBICOM98)*, pages 157–168, October 1998.

[14] P. Krishnan, P. Long, and J. Vitter. Adaptive Disk Spin-Down via Optimal Rent-to-Buy in Probabilistic Environments. In *Proceedings of the 12th International Conference on Machine Learning*, pages 322–330, July 1995.

[15] A. R. Lebeck, X. Fan, H. Zeng, and C. S. Ellis. Power aware page allocation. Technical Report CS-2000-08, Department of Computer Science, Duke University, June 2000.

[16] D. Lee. Energy Management Issues for Computer Systems. `//www.cs.washington.edu/homes/dlee/frontpage/mypapers/generals.ps.gz`.

[17] D. C. Lee, P. J. Crowley, J.-L. Baer, T. E. Anderson, and B. N. Bershad. Execution characteristics of desktop applications on Windows NT. In *Proceedings of the 25th Annual International Symposium on Comput er Architecture*, pages 27–38, June 1998.

[18] K. Li, R. Kumpf, P. Horton, and T. Anderson. A Quantitative Analysis of Disk Drive Power Management in Portable Computers. In *USENIX Association Winter Technical Conference Proceedings*, pages 279–291, 1994.

[19] J. Lorch and A. J. Smith. Scheduling Techniques for Reducing Processor Energy Use in MacOS. *Wireless Networks*, 3(5):311–324, October 1997.

[20] J. Lorch and A. J. Smith. Software Strategies for Portable Computer Energy Management. *IEEE Personal Communications Magazine*, 5(3):60–73, June 1998.

[21] MicroOptical Corp. *Eyeglass Display*, 1999.

[22] T. Pering, T. Burd, and R. Brodersen. Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System. In *Power Driven Microarchitecture Workshop, attached to ISCA98*, June 1998.

[23] Rambus. *RDRAM*, 1999. `http://www.rambus.com/`.

[24] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 1996.

[25] A. Rudendo, P. Reiher, G. Popek, and G. Kuenning. Saving portable computer battery power through remote process execution. *Mobile Computing and Communications Review, SIGMOBILE*, 2(1):19–26, January 1998.

[26] A. Rudendo, P. Reiher, G. Popek, and G. Kuenning. The remote processing framework for portable computer power saving. In *Proceedings of ACM Symposium on Applied Computing*, Feb 1999.

[27] S. Singh, M. Woo, and C. S. Raghavendra. Power-Aware Routing in Mobile Ad Hoc Networks. In *The Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 181–190, 1998.

[28] M. Stemm and R. Katz. Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices. In *Proceedings of 3rd International Workshop on Mobile Multimedia Communications (MoMuC-3)*, September 1996.

[29] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: A first step towards software power minimization. *IEEE Transactions on Very Large Scale Integration*, 2(4):437–445, December 1994.

[30] A. Vahdat, T. Anderson, M. Dahlin, E. Belani, D. Culler, P. Eastham, and C. Yoshikawa. WebOS: Operating System Services for Wide-Area Applications. In *Proceedings of the Seventh IEEE Symposium on High Performance Distributed Systems*, Chicago, Illinois, July 1998.

[31] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Technical Report CS-2000-06, Duke University, April 2000.

[32] M. Weiser, B. Welch, A. Demers, and S. Shenker. Scheduling for Reduced CPU Energy. In *USENIX Association, Proceedings of First Symposium on Operating Systems Design and Implementation (OSDI)*, November 1994. Monterey CA.

[33] H. Yu and A. Vahdat. Design and Evaluation of a Continuous Consistency Model for Replicated Services. In *Proceedings of Operating Systems Design and Implementation*, October 2000.