# Software Strategies for Portable Computer Energy Management*

Jacob R. Lorch† Alan Jay Smith†

February 24, 1998

## Abstract

Limiting the energy consumption of computers, especially portables, is becoming increasingly important. Thus, new energy-saving computer components and architectures have been and continue to be developed. Many architectural features have both high performance and low power modes, with the mode selection under software control. The problem is to minimize energy consumption while not significantly impacting the effective performance. We group the software control issues as follows: *transition*, *load-change*, and *adaptation*. The transition problem is deciding when to switch to low-power, reduced-functionality modes. The load-change problem is determining how to modify the load on a component so that it can make further use of its low-power modes. The adaptation problem is how to create software that allows components to be used in novel, power-saving ways. We survey implemented and proposed solutions to software energy management issues created by existing and suggested hardware innovations.

## 1 Introduction

Limiting energy consumption has become an important aspect of modern computing. The most important reason for this is the growing use of portable computers, which have limited battery capacities. Another reason is that high energy consumption by desktop computers translates into heat, fan noise, and expense. One way to reduce energy consumption is to simply use components that consume less power. Another way is to use components that can enter low-power modes by temporarily reducing their speed or functionality. This paper will discuss the software-implemented optimizations that arise from such hardware features, and what solutions have been proposed to deal with these issues. The aim of this paper is not to discuss hardware techniques for reducing power, but to discuss software techniques for taking advantage of low-power hardware that has already been designed.

We classify the software issues created by power-saving hardware features into three categories: transition, load-change, and adaptation. The transition problem involves answering the question, "When should a component switch from one mode to another?" The load-change problem involves answering the question, "How can the functionality needed from a component be modified so that it can more often be put into low-power modes?" The adaptation problem involves answering the question, "How can software be modified to permit novel, power-saving uses of components?" Each of the software strategies we will consider addresses one or more of these problems.

Different components have different energy consumption and performance characteristics, so it is generally appropriate to have a separate energy management strategy for each such component. Thus in this paper we will generally consider each component separately. For each component, first we will discuss its particular hardware characteristics, then we will discuss what transition, load-change, and adaptation solutions have been proposed for that component. The components whose software power management issues are most significant are the secondary storage unit, the processing unit, the wireless communication unit, and the display unit, but we will also briefly discuss other components.

| Category | Description |
|---|---|
| Transition | When should a component switch between modes? |
| Load-change | How can a component's functionality needs be modified so it can be put in low-power modes more often? |
| Adaptation | How can software permit novel, power-saving uses of components? |

Table 1: Categories of energy-related software problems

This paper is organized as follows. Section 2 discusses general issues in developing and evaluating solutions to the problems we have discussed. Sections 3, 4, 5, and 6 talk about specific problems and solutions involving the secondary storage unit, the processing unit, the wireless communication unit, and the display unit, respectively. Section 7 considers other, miscellaneous, components. Section 8 talks about strategies that deal with the system itself as a component to be power-managed. Finally, in section 9, we conclude.

## 2 General Issues

### 2.1 Strategy types

We call a strategy for determining when to switch from one component mode to another a *transition strategy*. Transition strategies require two sorts of information about a component: knowledge about its mode characteristics and information about its future functionality requirements. By mode characteristics we mean the advantages and disadvantages of each mode the component can be in, including how much power is saved by being in it, how much functionality is sacrificed by entering it, and how long it will take to return from it.

Mode characteristics are generally more easily obtained than future functionality requirements, so the most difficult part of a transition strategy is predicting future functionality requirements. Thus, transition strategies are sometimes called *prediction strategies*. The most common, but not the only, prediction tactic is to assume that the longer a component has been inactive, the longer it will continue to be inactive. Combining this prediction method with knowledge about mode characteristics then leads to a period $t$ such that whenever the component is inactive in a certain mode for longer than $t$, it should be placed in a lower-power mode. Such a period is called an *inactivity threshold*, and a strategy using one is called an inactivity threshold strategy.

We call a strategy for modifying the load on a component in order to increase its use of low-power modes a *load-change strategy*. Disk caching is an example, since it can reduce the load on a hard disk and thereby reduce its power consumption. Note that modifying component load does not always mean reducing it; sometimes merely reordering service requests can reduce power consumption. For instance, the hard disk will consume less power if one makes a disk request immediately before spinning the disk down than if one makes the request immediately after spinning it down.

*This material is based upon work supported in part by the National Science Foundation under grants MIP-9116578 and CCR-9117028, by NASA under Grant NCC 2-550, by the State of California under the MICRO program, and by Apple Computer Corporation, Intel Corporation, Sun Microsystems, Fujitsu Microelectronics, Toshiba Corporation, and Sony Research Laboratories.

†Computer Science Division, EECS Department, University of California, Berkeley, California, 94720-1776

We call a strategy for allowing components to be used in novel, power-saving ways an *adaptation strategy*. An example is modifying file layout on secondary storage so that magnetic disk can be replaced with lower-power flash memory.

### 2.2 Levels of energy management

Energy management can be done at several levels in the computer system hierarchy: the component level, the operating system level, the application level, and the user level. The end-to-end argument [68] suggests that this management should be performed at the highest level possible, because lower levels have less information about the overall workload. However, certain types of strategy are inappropriate for the highest levels. Most strategies are inappropriate for the user, since the user lacks knowledge about power consumption of each component, is unable to make decisions within milliseconds or faster, and is generally unwilling to make frequent energy management decisions. Problems with the application level are that applications operate independently and that applications lack certain useful information about the state of the machine because of operating system abstraction. For these reasons, most energy management is performed at the operating system level. The user typically just makes a few high-level decisions and applications typically just reduce their use of components.

One way to get the advantages of application-level management without most associated disadvantages is to use application-aware adaptation [37, 60]. In such a system, each application explicitly tells the operating system what its future needs are, and the operating system notifies each application whenever is a change in the state of the system relevant to energy management decisions. Thus, if an energy management strategy has to be implemented at the operating system level, it can still get information about the needs of an application from the definitive source: the application itself. Furthermore, if an energy management strategy is best implemented at the application level, it can be performed using machine state information normally confined to the operating system. Unfortunately, it is seldom the case that applications have the necessary knowledge or sophistication to take advantage of the ability to obtain or supply power-relevant information.

### 2.3 Strategy evaluation

When evaluating power management strategies, there are several points to remember. First, the effect of a strategy on the overall system power consumption is more important than its effect on the particular component it concerns. For example, a 50% reduction in modem power sounds impressive, but if the modem only accounts for 4% of total power consumption, this savings will only result in a 2% decrease in total power.

Second, it is important to use as the baseline the current strategy, rather than the worst possible strategy. For example, it would not be sufficient to simply know that a new strategy causes the disk motor to consume 85% of its maximum possible power. If the current strategy already caused it to be off 80% of the time, this would represent a small power reduction, but if the current strategy only turned it off 20% of the time, this would represent a significant power reduction.

Third, maximum battery lifetime is not necessarily what users want—they want to maximize the amount of work they can accomplish before the battery runs out, not simply the amount of time the computer can remain running before the battery runs out. For example, consider a strategy that halves the CPU speed and increases battery lifetime by 50%. If the sluggish response time makes papers take 10% longer to write, it is not reasonable to call the new strategy a 50% improvement just because the machine stays on 50% longer. The user can only write 36% more papers with one battery, so the strategy is really only a 36% improvement. Thus, to completely evaluate a new strategy, one must take into account not only how much power it saves, but also how much it extends or diminishes the time tasks take.

| Component | Hyp. 386 | Duo 230 | Duo 270c | Duo 280c | Avg. |
|---|---|---|---|---|---|
| Processor | 4% | 17% | 9% | 25% | 14% |
| Hard disk | 12% | 9% | 4% | 8% | 8% |
| Backlight | 17% | 25% | 26% | 25% | 23% |
| Display | 4% | 4% | 17% | 10% | 9% |
| Modem | n/a | 1% | 0% | 5% | 2% |
| FPU | 1% | n/a | 3% | n/a | 2% |
| Video | 26% | 8% | 10% | 6% | 13% |
| Memory | 3% | 1% | 1% | 1% | 2% |
| Other | 33% | 35% | 28% | 22% | 30% |
| Total | 6 W | 5 W | 4 W | 8 W | 6 W |

Table 2: For various portable computers, percentage of total power used by each component when power-saving techniques are used [48, 53]

Fourth, when evaluating a strategy, it is important to consider and quantify its effect on components it does not directly manipulate. For example, a strategy that slows down the CPU may cause a task to take longer, thus causing the disk and backlight to be on longer and consume more energy.

Fifth, to be completely accurate, one also has to consider that battery capacity is not constant. Battery capacity can vary depending on the rate of power consumption [63] and on the way that rate changes with time [83]. Thus, it may be important to understand not only how much a strategy reduces power consumption, but also how it changes the function of power consumption versus time. Also, it means that computing battery lifetime is more difficult than just dividing a rated energy capacity by total power consumption.

In conclusion, there are four things one must determine about a component power management strategy in order to evaluate it: how much it reduces the power consumption of that component; what percentage of total system power, on average, is due to that component; how much it changes the power consumption of other components; and how it affects battery capacity through its changes in power consumption. The first, third, and fourth require simulation of the strategy; the second requires a power budget describing the average power consumption of each system component. In the next subsection, we will give some such budgets.

### 2.4 Power budget

Table 2 shows examples of average power consumption for the components of some portable computers when power-saving techniques are used. This table shows measurements taken only when the computers were running off battery power, since we are most concerned with power management at such times; power management when a machine is plugged in is less critical, may have different tradeoffs, and may experience different user behavior. Note that power supply inefficiency is not treated as a separate category, but rather as a "tax" on all power consumed by each component. So, for instance, if the power supply system is 80% efficient, then instead of attributing 20% of power consumption to the power supply we increase the effective power consumption of each component by 25%. The first machine is a hypothetical 386DXL-based computer [53]. The next three examples describe measurements of Macintosh PowerBook Duo machines [48]. The Duo 230 has a supertwist monochrome display while the other Duos have active-matrix color displays.

The power budget of Table 2 indicates the magnitude of possible power savings. For instance, since the hard disk consumes only 8% of total power on the Duo 280c given its current power-saving methods, better techniques for managing hard disk power could save at most 8% of total system power, increasing battery lifetime by at most 9%. With power management active, the main consumers of power include the backlight, processor, video system, and hard disk. Thus, these are the components for which further power-saving methods will be most important.

Note that these breakdowns are likely to change as time progresses [32]. For instance, wireless communication hardware will probably eventually appear in most portable computers, at first adding about 1 W to total power consumption. Hardware improvements will decrease the power consumption of various other components, but this rate of decrease will be different for different components. Evidence of changing breakdown percentages can be seen from trends in modern portable computers [21]. Later models of portable computers seem to spend a greater percentage of their power consumption on the hard disk than earlier models. Presumably, this is because later models have substantial relative savings in other components' power but not as much savings in hard disk power. These forecasts suggest that as time progresses, power-saving techniques might become more important for the display and hard disk, and less important for the processor. Intuitively, this seems reasonable—mechanical motion and light generation seem inherently more power intensive than digital circuits. Note also that most existing components have been designed almost exclusively for performance and capacity, with little thought for power consumption. Components designed with minimal power use as a primary goal might behave very differently.

### 2.5 Battery technology

The importance of energy management arises as much from limited battery capacity as from high power use by portable computers. And, unfortunately, battery technology has been improving at only a modest pace in terms of increased capacity per unit weight and volume. The highest capacity battery technology currently used in portables is lithium ion, providing as much as 380 W-h/L and 135 W-h/kg [28]. This is an improvement over the roughly 260–330 W-h/L and 120 W-h/kg achievable from them in 1995 and the roughly 180 W-h/L achievable from them in 1991. Most impressive, though, is their improvement over earlier battery technologies, such as nickel-metal hydride with its 150 W-h/L and 50 W-h/kg in 1995 and nickel-cadmium with its 125 W-h/L and 50 W-h/kg in 1995 [63]. Technologies in development, such as lithium polymer, lithium anode, zinc-manganese dioxide, and zinc-air, may lead to even higher battery capacities in the future [63]. As an example of the battery capacity one can get today, a modern Apple laptop comes with a 29–32 W-h lithium ion battery [3].

## 3 Secondary Storage

### 3.1 Hardware features

Secondary storage in modern computers generally consists of a magnetic disk supplemented by a small amount of DRAM used as a disk cache; this cache may be in the CPU main memory, the disk controller, or both. Such a cache improves the overall performance of secondary storage. It also reduces its power consumption by reducing the load on the hard disk, which consumes more power than the DRAM.

Most hard disks have five power modes; in order of decreasing power consumption, these are active, idle, standby, sleep, and off [32]. In active mode, the disk is seeking, reading, or writing. In idle mode, the disk is not seeking, reading, or writing, but the motor is still spinning the platter. In standby mode, the motor is not spinning and the heads are parked, but the controller electronics are active. In sleep mode, the host interface is off except for some logic to sense a reset signal; thus, if there is a cache in the disk controller, its contents are lost. Transitions to active mode occur automatically when uncached data is accessed. Transitions to standby and sleep modes occur when explicit external directives are received; this is how software power-saving strategies influence hard disk power consumption.

Having the motor off, as in the standby mode, saves power. However, when it needs to be turned on again, it will take considerable time and energy to return to full speed. If this energy is greater than the savings from having the motor off, turning the motor off may actually increase energy consumption. Turning off the motor also has a performance impact, since the next disk request will be delayed until the motor returns to full speed. In addition, while the disk is returning to full speed, other components will typically continue consuming power, also increasing energy use. Going to sleep mode is an analogous operation, although one in which the savings in power, as well as the overhead required to return to the original state, are greater. Table 3 quantifies some time and energy considerations for various hard disks.

| **Hard disk** | **Maxtor Mobile-Max 251350** | **Road Warrior 815 MB Slimline** | **Toshiba MK2720** | **WD Portfolio** |
|---|---|---|---|---|
| Capacity | 1.35 GB | 815 MB | 1.35 GB | 1.0 GB |
| Idle power | 0.9 W | 0.9 W | 1.4 W | 0.95 W |
| Standby power | 0.23 W | 0.5 W | 0.35 W | 0.20 W |
| Sleep power | 0.025 W | 0.15 W | 0.15 W | 0.095 W |
| Spin-up time | 1 sec | 5 sec | 5 sec | 6 sec |
| Spin-up energy | 4.4 J | 17.5 J | 19.5 J | 30 J |

Table 3: Characteristics of various hard disks [57, 65, 76, 79]

A possible technology for secondary storage is an integrated circuit called flash memory [9, 19, 40, 56, 81]. Like a hard disk, such memory is nonvolatile and can hold data without consuming energy. Furthermore, when reading or writing, it consumes only 0.15 to 0.47 W, far less than a hard disk. It has a read speed of about 85 ns per byte, similar to DRAM, but a write speed of about 4–10 $\mu s$ per byte, about 10–100 times slower than hard disk. However, since flash memory has no seek time, its overall write performance is not that much worse than that of magnetic disk; in fact, for sufficiently small random writes, it can actually be faster. Flash memory is technically read-only, so before a region can be overwritten it must be electrically erased. Such erasure is done one full segment at a time, with each segment 0.5–128 KB in size and taking about 15 $\mu s$ per byte to erase [81]. A segment can only be erased 100,000 to 1,000,000 times in its lifetime before its performance degrades significantly, so the operating system must ensure that the pattern of erasures is reasonably uniform, with no single segment getting repeatedly erased. The current cost per megabyte of flash is \$2–4, making it about 17–40 times more expensive than hard disk but about 2–5 times less expensive than DRAM. Cáceres et al. [9] point out that the costs of flash and hard disk may become comparable in the future since the per-year increases in megabytes per dollar are about 25% for magnetic disk and 40% for flash; given these assumptions, the two prices could converge in 6–8 years. Flash memory offers great opportunities for secondary storage power savings if it can be substituted for the hard disk or used for caching. Before that, however, software must be designed to overcome the many limitations of flash memory, especially its poor write performance.

### 3.2 Transition strategies

Transition strategies for magnetic disks can be of three kinds: deciding when to go to sleep mode, deciding when to go to standby mode, and deciding when to turn off the disk completely. Most are of the first kind. We know of no studies of the second kind, for reasons we will discuss in the next paragraph. Strategies of the third kind also exist, but are generally simple inactivity threshold strategies that have not been experimentally scrutinized.

One reason for the lack of study of transition strategies for deciding when to enter standby mode is that this mode is a relatively new feature on disks. Another reason is that it may often be better to enter sleep mode than standby mode. Sleep mode consumes less power, and since the time it takes to go from sleep to idle mode is dominated by the spin-up time of the motor, this transition takes no longer than that from standby to idle mode. The main advantage to standby mode is that on-disk cache contents are

preserved; this may or may not be significant, depending on the caching algorithm in the disk controller, and whether or not the main memory disk cache is a superset of the contents of the controller disk cache.

#### 3.2.1 Fixed inactivity threshold

The most common transition strategy for going into sleep mode is to enter that mode after a fixed inactivity threshold. When hard disks allowing external control over the motor were first developed, their manufacturers suggested an inactivity threshold of 3–5 minutes. However, researchers soon discovered that power consumption could be minimized by using inactivity thresholds as low as 1–10 seconds; such low thresholds save roughly twice as much power as a 3–5 minute threshold [21, 45].

The greater power savings from using a smaller inactivity threshold comes at a cost, however: perceived increased user delay. Spinning down the disk more often makes the user wait more often for the disk to spin up. The inactivity threshold yielding minimum disk power results in user delay of about 8–30 seconds per hour; some researchers believe this to be an unacceptable amount of delay [21], although in absolute terms, this amount is trivial. Another problem with short inactivity thresholds is that disks tend to last for only a limited number of start-stop cycles, and excessively frequent spin up-spin down cycles could cause premature disk failure. Thus, the best disk spin-down policy is not necessarily the one that minimizes power consumption, but the one that minimizes power consumption while keeping user delay and start-stop frequency at an acceptable level.

It is worth pointing out, although it should be obvious, that the time between disk accesses is not exponentially distributed; the expected time to the next disk access is generally an increasing function of the time since the last access. If interaccess times for disk reference were exponentially distributed, the correct strategy would use an inactivity threshold of either zero or infinity [30].

#### 3.2.2 Changing inactivity threshold

There are several arguments for dynamically changing the inactivity threshold, not necessarily consistent with each other. The first argument is that disk request interarrival times are drawn independently from some unknown stationary distribution. Thus, as time passes one can build up a better idea of this distribution, and from that deduce a good threshold. The second argument is that the interarrival time distribution is nonstationary, i.e. changing with time, so a strategy should always be adapting its threshold to the currently prevailing distribution. This distribution can be inferred from samples of the recent distribution and/or from factors on which this distribution depends. The third argument is that worst-case performance can be bounded by choosing thresholds randomly—any deterministic threshold can fall prey to a particularly nasty series of disk access patterns, but changing the threshold randomly eliminates this danger.

If disk interarrival times are independently drawn from some unknown stationary distribution, as the first argument states, then no matter what this distribution, there exists an inactivity threshold that incurs a cost no more than $e/(e-1)$ times that of the optimal off-line transition strategy [39]. One could find this threshold by keeping track of all interarrival times so that the distribution, and thus the ideal threshold, could be deduced.

One algorithm of that type, using constant space, builds up a picture of the past interarrival time distribution in the following indirect way [42]. It maintains a set of possible thresholds, each with a value indicating how effective it would have been. At any point, the algorithm chooses as its threshold the one that would have performed the best. Incidentally, "best" does not simply mean having the least power consumption; the valuation might take into account the relative importance of power consumption and frequency of disk spin-downs specified by the user. This algorithm has been shown to perform well on real traces, beating many other practical algorithms.

Another strategy using a list of candidate thresholds is based on the second argument, that disk access patterns change with time [33]. In this strategy, each candidate is initially assigned equal weight. After each disk access, candidates' weights are increased or decreased according to how well they would have performed relative to the optimal off-line strategy over the last interaccess period. At any point, the threshold chosen for actual use is the weighted average of all the candidates. Simulations show that this strategy works well on actual disk traces. The developers of this strategy only considered using it to minimize power consumption; however, it could easily be adapted to take frequency of spin-ups into account.

Another dynamic strategy based on the second argument tries to keep the frequency of annoying spin-ups relatively constant even though the interaccess time distribution is always changing [20]. This strategy raises the threshold when it is causing too many spin-ups and lowers it when more spin-ups can be tolerated. Several variants of this strategy, which raise and lower the threshold in different ways, are possible. Simulation of these variants suggests that using an adaptive threshold instead of a fixed threshold can significantly decrease the number of annoying spin-ups experienced by a user while increasing energy consumption by only a small amount.

Note that all the dynamic strategies we have described that are based on the second argument make inferences about the current distribution of disk access interarrival times based on recent samples of this distribution. However, there are likely other factors on which this distribution depends and on which such inferences could be based, such as the current degree of multiprogramming or which applications are running. Additional research is needed to determine which of these factors can be used effectively in this way.

By the third argument, a strategy should make no assumptions about what the disk access pattern looks like, so that it can do well no matter when disk accesses occur. One such strategy chooses a new random threshold after every disk access according to the cumulative distribution function

$$\pi(t) = \frac{e^{t/c} - 1}{e - 1},$$

where $c$ is the number of seconds it takes the running motor to consume the same amount of energy it takes to spin up the disk [39]. This strategy has been proven ideal among strategies having no knowledge of the arrival process. Note, however, that almost all transition strategies described in this paper do purport to know something about the arrival process, and thus are capable of beating this strategy. In other words, although this strategy does have the best worst-case expected performance, it does not necessarily have the best typical-case performance.

#### 3.2.3 Alternatives to an inactivity threshold

Some transition strategies have been developed that do not use an inactivity threshold explicitly [21]. One such strategy is to predict the actual time of the next disk access to determine when to spin down the disk. However, simulations of variants of this strategy show that they provide less savings than the best inactivity threshold strategy, except when disk caching is turned off. This may be because filtering a pattern of disk accesses through a disk cache makes it too patternless to predict. Another strategy is to predict the time of the next disk request so the disk can be spun up in time to satisfy that request. However, no techniques proposed for this have worked well in simulation, apparently because the penalty for wrong prediction by such strategies is high. Despite the shortcomings of the non-threshold-based transition strategies studied so far, some researchers remain hopeful about the feasibility of such strategies. Simulation of the optimal off-line strategy indicates that such strategies could save as much as 7–30% more energy than the best inactivity threshold method.

### 3.3 Load-change strategies

Another way to reduce the energy consumption of a hard disk is to modify its workload. Such modification is usually effected by changing the configuration or usage of the cache above it.

One study found that increasing cache size yields a large reduction in energy consumption when the cache is small, but much lower energy savings when the cache is large [45]. In that study, a 1 MB cache reduced energy consumption by 50% compared to no cache, but further increases in cache size had a small impact on energy consumption, presumably because cache hit ratio increases slowly with increased cache size [82]. The study found a similar effect from changing the dirty block timeout period, the maximum time that cache contents are permitted to be inconsistent with disk contents. Increasing this timeout from zero to 30 seconds reduced disk energy consumption by about 50%, but further increases in the timeout delay had only small effects on energy consumption [45]. Another possible cache modification is to add file name and attribute caching. Simulation showed a moderate disk energy reduction of 17% resulting from an additional 50 KB of cache devoted to this purpose [45].

Prefetching, a strategy commonly used for performance improvement, should also be effective as an energy-saving load-change strategy. If the disk cache is filled with data that will likely be needed in the future before it is spun down, then more time should pass before it must again be spun up. This idea is similar to that of the Coda file system [69], in which a mobile computer caches files from a file system while it is connected so that when disconnected it can operate independently of the file system.

Another approach to reducing disk activity is to design software that reduces paging activity. This can be accomplished by reducing working set sizes and by improving memory access locality. There are many things operating system and application designers can do to achieve these goals.

### 3.4 Adaptation strategies for flash memory as disk cache

Flash memory has two advantages and one disadvantage over DRAM as a disk cache. The advantages are nonvolatility and lower power consumption; the disadvantage is poorer write performance. Thus, flash memory might be effective as a second-level cache below the standard DRAM disk cache [56]. At that level, most writes would be flushes from the first-level cache, and thus asynchronous. However, using memory with such different characteristics necessitates novel cache management strategies.

The main problem with using flash memory as a second-level cache is that data cannot be overwritten without erasing the entire segment containing it. One solution is to ensure there is always a segment with free space for writing; this is accomplished by periodically choosing a segment, flushing all its dirty blocks to disk, and erasing it [56]. One segment choosing strategy is to choose the one least recently written; another is to choose the one least recently accessed. The former is simpler to implement and ensures no segment is cleaned more often than another, but the latter is likely to yield a lower read miss ratio. Unfortunately, neither may be very good in packing together blocks that are referenced together. One approach is to supplement the strategy with a copying garbage collector, such as that found in LFS [66], to choose recently used blocks of a segment about to be erased and write them into a segment that also contains recently used blocks and is not going to be erased.

Simulations have shown that using a second-level flash memory cache of size 1–40 MB can decrease secondary storage energy consumption by 20–40% and improve I/O response time by 30–70% [56]. Thus, using appropriate cache management policies seem to allow a flash memory second-level cache to reduce energy consumption and still provide equal or better performance than a system using a traditional cache. The simulations would have been more persuasive, however, if they had compared the system with flash to one with a DRAM second-level cache rather than to one with no second-level cache.

### 3.5 Adaptation strategies for flash memory as disk

Flash memory is a low-power alternative to magnetic disk. However, the large differences between flash memory and magnetic disk suggest several changes to file system management. Since flash has no seek latency, there is no need to cluster related data on flash memory for the purpose of minimizing seek time [9]. Since flash is practically as fast as DRAM at reads, a disk cache is no longer important except to be used as a write buffer [9]. Such a write buffer would make writes to flash asynchronous, thus solving another problem of flash memory: poor write performance. In fact, if SRAM were used for this write buffer, its permanence would allow some writes to flash to be indefinitely delayed [19, 81]. Finally, unlike magnetic disk, flash memory requires explicit erasure before a segment can be overwritten, a slow operation that can wear out the medium and must operate on a segment at a time. One solution to these problems is to use a log-structured file system like LFS [40, 66], in which new data does not overwrite old data but is instead appended to a log. This allows erasures to be decoupled from writes and done asynchronously, thus minimizing their impact on performance. A flash file system also needs some way to ensure that no physical segment is cleaned especially often. One way to do this is to make sure that physical segments containing infrequently modified data and ones containing frequently modified data switch roles occasionally [81].

Simulations of flash file systems using some of these ideas have found that they can reduce secondary storage power consumption by 60–90% while maintaining aggregate performance comparable to that of magnetic disk file systems [19]. However, at high levels of utilization, the performance of file systems using asynchronous erasure can degrade significantly due to the overhead of that erasure.

### 3.6 Adaptation strategies for wireless network as disk

Another hardware approach to saving secondary storage energy is to use wireless connection to a plugged-in file server instead. Offloading storage has the advantage that the storage device can be big and power-hungry without increasing the weight or power consumption of the portable machine. Disadvantages include increased power consumption by the wireless communication system, increased use of limited wireless bandwidth, and higher latency for file system accesses. Adaptation strategies can help minimize the impact of the disadvantages but retain the advantages.

The general model for using wireless communication as secondary storage is to have a portable computer transmit data access requests to, and receive data from, a server. An improvement on this is to have the server make periodic broadcasts of especially popular data, so the portable computer will have to waste less power transmitting requests [36]. A further improvement is to interleave index information in these broadcasts, so a portable computer can anticipate periods of no needed data and shut its receiver off during them.

Another model for using wireless communication for storage is proposed in extensions to the Coda scheme [69]. In this model, the portable computer storage system functions merely as a large cache for the server file system. When the portable computer is not wired to the file system server, it services cache misses using wireless communication. Because such communication is slow and bandwidth-consuming, the cache manager seeks to minimize its frequency by hoarding files that are anticipated to be needed during disconnection.

A third model for using wireless communication for storage, used by InfoPad [5, 7], is to perform *all* processing on an unmoving server. In this model, the portable "computer" is merely a terminal that transmits and receives low-level I/O information, so the energy consumption for general processing and storage is consumed by plugged-in servers instead of the mobile device. In this way, portable storage and CPU energy consumption are traded for high processing request latency, significant network bandwidth consumption, and additional energy consumption by the portable wireless device.

The limiting factor in all of these cases is network bandwidth; what is practical depends on the bandwidth between the local system and the data source. A packet radio connection at 28.8 Kb/s is very different than the type of multi-megabit per second system that could be implemented within a building.

### 3.7 Future hardware innovations

Researchers working on technological advances in hardware can also do much to aid software techniques in reducing power. In order to minimize the impact of decisions to spin down the hard disk, the energy and time consumed by a disk when spinning up should be reduced. Since disk power use drops roughly quadratically with rotation speed, it would also be useful to enable the disk to be put into low rotation speed modes, so that software could sacrifice some I/O performance to obtain reduced disk power consumption. An additional benefit of reduced rotation speed would be a reduction in spin-up times. To make it easier for software to achieve good performance with flash memory, its design should emphasize fast writing and erasing, as well as the ability to erase at the same time that data is being read or written. Increasing the number of erasures possible in the lifetime of a segment would simplify the management of flash memory.

## 4 Processor

### 4.1 Hardware features

Processors designed for low-power computers have many power-saving features. One power-saving feature is the ability to slow down the clock. Another is the ability to selectively shut off functional units, such as the floating-point unit; this ability is generally not externally controllable. Such a unit is usually turned off by stopping the clock propagated to it. Finally, there is the ability to shut down processor operation altogether so that it consumes little or no energy. When this last ability is used, the processor typically returns to full power when the next interrupt occurs.

In general, slowing down the processor clock without changing the voltage is not useful. Power consumption $P$ is essentially proportional to the clock frequency $f$, the switching capacitance $C$, and the square of the voltage $V$ ($P \propto CV^2f$), but the time $t$ it takes the processor to complete a task is inversely proportional to the clock frequency ($t \propto 1/f$). Since the energy $E$ it takes the processor to complete a task is the product of its power consumption and the time it spends ($E = Pt$), this energy consumption is invariant with clock speed. Thus, reducing the clock speed lengthens processor response time without reducing the amount of energy the processor consumes during that time. In fact, slowing the clock speed will usually increase total energy consumption by extending the time other components need to remain powered. However, if the voltage can be decreased whenever the clock speed is reduced, then energy consumption, which is proportional to the square of the voltage ($E \propto CV^2$), would usually be reduced by slowing the clock.

Turning off a processor has little downside; no excess energy is expended turning the processor back on, the time until it comes back on is barely noticeable, and the state of the processor is unchanged from it turning off and on, unless it has a volatile cache [29]. On the other hand, there is a clear disadvantage to reducing the clock rate: tasks take longer. There may also be a slight delay while the processor changes clock speed.

Reducing the power consumption of the processor saves more than just the energy of the processor itself. When the processor is doing less work, or doing work less quickly, there is less activity for other components of the computer, such as memory and the bus. For example, when the processor on the Macintosh Duo 270c is off, not only is the 1.15 W of the processor saved, but also an additional 1.23 W from other components [48]. Thus, reducing the power consumption of the processor can have a greater effect on overall power savings than it might seem from merely examining the percentage of total power attributable to the processor.

### 4.2 Transition strategies for turning the processor off

When the side effects of turning the processor off and on are insignificant, the optimal off-line transition strategy is to turn it off whenever the processor will not be needed until the next interrupt occurs. With a well-designed operating system, this can be deduced from the current status of all processes. Thus, whenever any process is running or ready to run, the processor should not be turned off; when all processes are blocked, the processor should be turned off [49, 72, 74]. Examples of operating systems using this strategy are Windows [15, 62] and UNIX.

MacOS, however, uses a different strategy, perhaps because its strategy was designed when processors *did* suffer side effects from turning off. It uses an inactivity threshold, as is commonly used for hard disk power management. The processor is shut off when there have been no disk accesses in the last fifteen seconds and no sound chip accesses, changes to the cursor, displaying of the watch cursor, events posted, key presses, or mouse movements in the last two seconds. The savings achievable from this strategy vary greatly with workload [47, 48].

### 4.3 Load-change strategies when the CPU can turn off

Given a transition strategy that turns off the processor when it is not performing any tasks, the goal of a load-change strategy is simply to limit the energy needed to perform tasks. This suggests three approaches: reducing the time tasks take, using lower-power instructions, and reducing the number of unnecessary tasks. Below we present several load-change strategies that use different subsets of these approaches.

One technique, which uses the first two approaches, is to use more efficient operating system code [70]. However, if overall system performance has not been a sufficient reason for system designers and implementors to write good code, energy efficiency considerations are unlikely to make much difference.

Another technique, which uses the same approaches, is to use energy-aware compilers, i.e. compilers that consider the energy efficiency of generated code [75]. Traditional compiler techniques have application as load-change strategies in that they reduce the amount of time a processor takes to complete a task. Another way for the compiler to decrease energy consumption is to carefully choose which instructions to use, since some instructions consume more power than others. However, preliminary studies indicate that the primary gain from code generation is in decreasing the number of instructions executed, not in choosing between different equally long code sequences [75]. There may be some special cases, such as generating entirely integer versus mixed integer and floating point code, where the effects are significant, but we believe that these are the exception.

Another load-change technique uses the third approach, performing fewer unnecessary tasks [49]. In some cases, when an application is idle, it will "busy-wait" for an event instead of blocking. Then, the standard transition strategy will not turn off the processor even though it is not doing any useful work. One way to solve this problem is to force an application to block for a certain period whenever it satisfies certain conditions that indicate it is likely to be busy-waiting and not performing any useful activity. Simulations of such a strategy using traces of machines running on battery power showed that it would allow the processor to be off, on average, 66% of the time, compared to 47% when no measures were taken to forcibly block applications.

### 4.4 Transition strategies for dynamically changing CPU speed

As explained before, slowing the clock is useless if voltage is kept constant. Therefore, when we discuss strategies to take advantage of slowing the processor clock, we are assuming that slowing the clock is accompanied by reducing the voltage. The voltage can be reduced when the clock speed is reduced because under those conditions the longer gate settling times resulting from lower voltage become acceptable.

Previous calculations have shown that the lowest energy consumption comes at the lowest possible speed. However, performance is also reduced by reducing the clock speed, so any strategy to slow the clock must achieve its energy savings at the expense of performance. Furthermore, reducing processor performance may cause an increase in the energy consumption of other components, since they may need to remain on longer. Thus, it is

important to make an appropriate trade-off between energy reduction and performance.

The strategies that have been considered so far for making this trade-off [12, 77] were designed to achieve two general goals. The first goal is to not delay the completion time of any task by more than several milliseconds. This ensures that interactive response times do not lengthen noticeably, and ensures that other components do not get turned off noticeably later. The second goal is to adjust CPU speed gradually. This is desirable because voltage scaling causes the minimum voltage $V$ permissible at a clock speed $f$ to be roughly linear in $f$. Thus, the number of clock cycles executed in an interval $I$ is proportional to $\int_I f(t)\,dt$, while the energy consumed during that interval is proportional to $\int_I [f(t) + C]^3\,dt$. Given these equations, it can be mathematically demonstrated that the most energy-efficient way to execute a certain number of cycles within a certain interval is to keep clock speed constant throughout the interval.

One strategy for adjusting CPU speed seeks to achieve these goals in the following way [77]. Time is divided into 10–50 ms intervals. At the beginning of each interval, processor utilization during the previous interval is determined. If utilization was high, CPU speed is slightly raised; if it was low, CPU speed is slightly lowered. If, however, the processor is falling significantly behind in its work, CPU speed is raised to the maximum allowable. Simulations of this strategy show 50% energy savings when the voltage, normally constrained to be 5 V, can be reduced to 3.3 V, and 70% savings when it can be reduced to 2.2 V. Interestingly, the strategy shows worse results when the voltage can be reduced to 1 V, seemingly because the availability of the low voltage causes the strategy to generate extreme variation in the voltage level over time. Obviously, a strategy should be designed so that it never yields worse results when the range over which parameters can be varied increases.

Another strategy also divides time into 10–50 ms intervals [12]. At the beginning of each interval, it predicts the number of CPU busy cycles that will occur during that interval, and sets the CPU speed just high enough to accomplish all this work. There are actually many variants of this strategy, each using a different prediction technique. In simulations, the most successful variants were one that always predicted the same amount of work would be introduced each interval, one that assumed the graph of CPU work introduced versus interval number would be volatile with narrow peaks, and one that made its prediction based on a weighted average of long-term and short-term CPU utilization but ignored any left-over work from the previous interval. The success of these variants suggests that it is important for such a strategy to balance performance and energy considerations by taking into account both short-term and long-term processor utilization in its predictions. Too much consideration for short-term utilization increases speed variance and thus energy consumption. Too much consideration for long-term utilization makes the processor fall behind during especially busy periods and thus decreases performance.

Using intervals to ensure that no work gets delayed more than a fixed period of time is only one way to limit the performance impact. A better way is to determine appropriate deadlines for all tasks and attempt to delay no task past its deadline [12, 77]. However, no strategies using this approach have been developed yet.

### 4.5 Load-change strategies when functional units can turn off

Typically, if functional units can be turned off, the chip internals turn them off automatically when unused. This makes software transition strategies unnecessary and impossible to implement, but makes load-change strategies tenable. One such load-change strategy is to use a compiler that clusters together several uses of a pipelined functional unit so that the unit is on for less time. Another is, when compiling, to preferentially generate instructions using functional units that do not get power-managed. However, no research has been done yet on such load-change strategies. It is unclear whether the power savings to be obtained from these strategies would be significant.

### 4.6 Future hardware innovations

There are several things that researchers working on technological advances in hardware can do to increase the usefulness of software strategies for processor energy reduction. Perhaps the most important is designing the motherboard so that reduction in the energy consumption of the processor yields a consequent large reduction in the energy consumption of other components. In other words, motherboard components should have states with low power consumption and negligible transition side effects that are automatically entered when the processor is not presenting them with work. Voltage scaling is not widely available, and needs to be made so. Once this happens, the software strategies that anticipate this technology can be put to good use. Finally, if hardware designers can keep the time and energy required to make transitions between clock speeds low, the savings from clock speed transition strategies will be even greater.

## 5 Wireless communication devices

### 5.1 Hardware features

Wireless communication devices are appearing with increasing frequency on portable computers. These devices can be used for participating in a local or wide area wireless network, or for interacting with disconnected peripherals like a printer or mouse. They typically have five operating modes; in order of decreasing power consumption, these are transmit, receive, idle, sleep, and off [32]. In transmit mode, the device is transmitting data; in receive mode, the device is receiving data; in idle mode, it is doing neither, but the transceiver is still powered and ready to receive or transmit; in sleep mode, the transceiver circuitry is powered down, except sometimes for a small amount of circuitry listening for incoming transmissions. Typically, the power consumption of idle mode is not significantly less than that of receive mode [73], so going to idle mode is not very useful. Transitions between idle and sleep mode typically take some time. For example, HP's HSDL-1001 infrared transceiver takes about 10 $\mu s$ to enter sleep mode and about 40 $\mu s$ to wake from it [34], AT&T's WaveLAN PCMCIA card and IBM's infrared wireless LAN card each take 100 ms to wake from sleep mode, and Metricom's Ricochet wireless modem takes 5 seconds to wake from sleep mode [73].

Some devices provide the ability to dynamically modify their transmission power. Reducing transmission power decreases the power consumption of the transmit mode; it also has the advantage of reducing the interference noise level for neighboring devices, leading to a reduction in their bit error rates and enabling higher cell capacity. The disadvantage, however, is that when a device reduces its transmission power, it decreases the signal to noise ratio of its transmissions, thus increasing its bit error rate.

Wireless device power consumption depends strongly on the distance to the receiver. For instance, the wide-area ARDIS system, in which each base station covers a large area, requires transmit power of about 40 W, but the wide-area Metricom system, which uses many base stations each serving a small area, requires mobile unit transmit power of only about 1 W [17]. Local-area networks also tend to provide short transmission distances, allowing low power dissipation. For instance, the WaveLAN PCMCIA card, meant for use in such networks, consumes only about 1.875 W in transmit mode [51]. Even smaller distances, such as within an office or home, yield even smaller power requirements. For instance, the HSDL-1001 infrared transceiver consumes only 55 mW in transmit mode [34], and the CT-2 specification used for cordless telephones requires less than 10 mW for transmission [17].

### 5.2 Transition strategies for entering sleep mode

Transition strategy issues for wireless communication devices entering sleep mode are quite similar to those for hard disks, so the solutions presented for hard disks are generally applicable to them. However, some features of wireless communication suggest two changes to the standard inactivity threshold methods used with hard disks. First, because a wireless

communication device does not have the large mechanical component a hard disk has, the time and energy required to put it in and out of sleep mode are generally much smaller. Further, the user is unlikely to know when the unit is off or on unless some sort of monitor is installed, so users should be unaware of start-up times unless they are unduly long. These factors suggest a more aggressive energy management strategy such as using a much shorter inactivity threshold. Second, it may be necessary to have wireless devices periodically exit sleep mode for a short period of time to make contact with a server, so that the server does not decide the unit is off or out of range and delete state information related to the connection [52].

Experimental simulation has been used to evaluate the effect of some transition strategies [73]. One simulation, which assumed a Ricochet modem was used only for the retrieval of electronic mail, considered a strategy that put the modem to sleep whenever no mail was being retrieved, and woke it up after a certain period of time to check for new mail. It showed that using a period of about four minutes would reduce the energy consumption of the wireless device by about 20%, and only cause mail to be, on average, two minutes old when it was seen by the user. Another simulation assumed a WaveLAN PCMCIA card was used only for Web browsing, and considered the strategy of putting the wireless device to sleep whenever a certain inactivity threshold passed with no outstanding HTTP transactions. It showed that using a very small inactivity threshold reduced power consumption of the device by 67% without noticeably increasing the perceived latency of document retrieval.

### 5.3 Load-change strategies when sleep mode is used

One way to increase the amount of time a wireless device can spend sleeping is simply to reduce network usage altogether. There are many strategies for doing this, some examples of which are as follows. One strategy is to compress TCP/IP headers; this can reduce their size by an order of magnitude, thus reducing the wireless communication activity of a mobile client [18]. Another strategy is to reduce the data transmission rate or stop data transmission altogether when the channel is bad, i.e. when the probability of a dropped packet is high, so that less transmission time is wasted sending packets that will be dropped [83]. Of course, if data transmission is ceased altogether, probing packets must be sent occasionally so that the unit can determine when the channel becomes good again. Yet another strategy is to have servers [60] or proxies [27] use information about client machine characteristics and data semantics to provide mobile clients with versions of that data with reduced fidelity and smaller size; this reduces the amount of energy mobile clients must expend to receive the data. For example, a data server might convert a color picture to a black-and-white version before sending it to a mobile client. A fourth strategy is to design applications that avoid unnecessary communication, especially in the expensive transmit direction.

Another way is to use a medium access protocol that dictates in advance when each wireless device may receive data. This allows each device to sleep when it is certain that no data will arrive for it. For example, the 802.11 LAN standard has access points that buffer data sent to wireless devices and that periodically broadcast a beacon message indicating which mobile units have buffered data. Thus, each wireless device only has to be listening when it expects a beacon message, and if it discovers from that message that no data is available for it, it may sleep until the next beacon message [6]. Of course, strategies such as this either require a global (broadcast) clock or closely synchronized local clocks. Similar protocols, designed to increase battery life in one-way paging systems, include the Post Office Code Standardization Advisory Group (POCSAG) protocol and Motorola's FLEX protocol [54]. A different type of power-conserving protocol, which does not require buffering access points and is thus peer-to-peer, is LPMAC [55]. LPMAC divides time into fixed-length intervals, and elects one terminal in the network the network coordinator (NC). The NC broadcasts a traffic schedule at the beginning of each interval that dictates when each unit may transmit or receive data during that interval. Each interval ends with a short contention period during which any unit may send requests for network time to the NC. Thus, each mobile unit only needs be awake during the broadcast of the traffic schedule, and may sleep until the next such broadcast if the schedule indicates that no one will be sending data to it. This protocol does not require intermediate buffering because data is buffered at the sending unit until the NC gives it permission to send.

### 5.4 Transition strategies for changing transmission power

Many approaches to dynamically changing the transmission power in wireless networks have been proposed. However, few of them were designed with consideration for the battery lifetime of mobile units, being meant solely to achieve goals like guaranteeing limits on signal to noise ratio, balancing received power levels, or maximizing cell capacities [67]. Here we will focus on those strategies that at least address the battery lifetime issue. Of course, a strategy cannot consider battery lifetime alone, but must balance the need for high battery lifetime with the need to provide reasonable cell capacity and quality of service.

A transition strategy to decide when to change power should should take into account the consequences of reducing transmission power: increased battery lifetime, lower bit error rate for neighbors (enabling higher cell capacities), and higher bit error rate for one's own transmissions. Such a strategy can be local, meaning that it accounts only for the needs of the wireless device on which it is running, or global, meaning that it accounts for the needs of other devices on the network. Global strategies seem most appropriate considering that the decisions one wireless unit makes about its transmission power and link bandwidth affect the other wireless units. Local strategies have the advantage of being simpler to implement, especially in a heterogeneous environment where communication between units in an attempt to cooperate is difficult. Global strategies, however, need not require global communication; estimates of the effect of power changes on interference with other units may be sufficient.

One suggested local transition strategy is to choose transmission power at each moment based on the current quality of service required and the current interference level observed, using a function analytically selected to optimize battery lifetime. Simulations show that such an approach can yield significant energy savings and no reduction in quality of service compared to other schemes that do not consider battery lifetime, such as one that attempts to always maintain a certain signal to noise ratio. The amount of energy savings obtained decreases with increasing quality of service required, since more required activity means less opportunity to reduce transmission power and save energy [67].

The developers of that strategy also considered a global variant of it, in which each mobile unit also considers the interference levels it has observed in the past when making its decisions about transmission power [67]. In this scheme, the only communication between mobile units is indirect, via the noise they generate for each other with their transmissions. Using such indirect communication makes implementation simpler, since it requires no protocol for explicit communication of control information. However, it does not allow the units to make intelligent distributed decisions, such as to use time-division multiple access, i.e. to take turns transmitting, so as to minimize interference and maximize use of cell capacity. Nevertheless, simulations indicate that even without explicit communication the strategy seems to achieve reasonable global behavior. One reason for this may be that when devices act independently to reduce their power levels when interference makes transmission unworthwhile, the machines will tend somewhat to take turns in transmitting. This is similar to back-off strategies in Ethernet.

Other global strategies can be imagined that use explicit communication of such things as quality of service needs and transmission power schedules. Such explicit communication would allow mobile units to coordinate their transmission power in an attempt to optimize overall battery lifetime given reasonable overall goals for quality of service and cell capacity.

### 5.5 Load-change strategies when transmission power can change

When a wireless device transmits with reduced power, its bit error rate increases. Load-change strategies can be used to mitigate the effect of this increased bit error rate and thus enable the use of less transmission power. One way to do this is to use more error correction code bits, although this has the side effect of reducing effective data bandwidth by consuming extra bandwidth for the additional code bits. Another way is to request more link bandwidth, to counter the effects of increased error correction and more frequent retransmission of dropped packets.

So, a strategy for modifying transmission power can be made more effective by simultaneously effecting changes in the amount of error correction and link bandwidth used. For instance, suppose the needed quality of service can be attained by transmitting with power $P_1$, using error correction method $E_1$, and consuming bandwidth $B_1$, or by transmitting with power $P_2$, using error correction method $E_2$, and consuming bandwidth $B_2$. Then, the strategy can choose among these two combined options based on how they will influence battery lifetime and cell capacity.

Some researchers have suggested a global strategy that allows mobile units in a network to optimize overall system utility by coordinating which units will be transmitting when, what transmission power each unit will use, and how much error correction each unit will use [50]. They considered only bit error rate and bandwidth seen by each user application in determining system utility, but their model could be adapted to take battery lifetime into account as well. Their system uses explicit communication, but uses a hierarchically distributed algorithm to reduce the complexity, control message bandwidth, and time required to perform the optimization.

## 6 Display and Backlight

### 6.1 Hardware features

The display and backlight have few energy-saving features. This is unfortunate, since they consume a great deal of power in their maximum-power states; for instance, on the Duo 280c, the display consumes a maximum of 0.75 W and the backlight consumes a maximum of 3.40 W [48]. The backlight can have its power reduced by reducing the brightness level or by turning it off, since its power consumption is roughly proportional to the luminance delivered [32]. The display power consumption can be reduced by turning the display off. It can also be reduced slightly by switching from color to monochrome or by reducing the update frequency. Reducing the update frequency reduces the range of colors or shades of gray for each pixel, since such shading is done by electrically selecting each pixel for a particular fraction of its duty cycle. Generally, the only disadvantage of these low-power modes is reduced readability. However, in the case of switches among update frequencies and switches between color and monochrome, the transitions can also cause annoying flashes.

### 6.2 Transition strategies

Essentially the only transition strategy currently used to take advantage of these low-power features is to turn off or down the backlight and display after a certain period of time has passed with no user input. The reasoning behind this strategy is that if the user has not performed any input recently, then it is likely he or she is no longer looking at the screen, and thus the reduced readability of a low-power mode is acceptable for the immediate future. To lessen the effect of a wrong guess about such inactivity on the part of the user, some machines do not shut the backlight off immediately but rather make it progressively dimmer. In this way, if the user is actually still looking at the screen, he or she gets a chance to indicate his or her presence before the entire screen becomes unreadable. One study found that thanks to the use of low-power states, the backlights on some machines consumed only 32–67% of maximum possible energy while running on battery power [48].

A possible modification of this standard strategy is to automatically readjust the inactivity threshold to make it a better predictor of user inactivity. For example, if the user hits a key just as the backlight begins to dim, such a strategy might increase the inactivity threshold on the assumption that its current value is too short.

Another possible transition strategy is to switch the display to monochrome when color is not being displayed, or to switch it to a lower update frequency when the items displayed do not require a high update frequency. The operating system might even switch to a lower-power display mode when those parts of the screen making use of the current display mode are not visually important to the user. For example, if the only color used on the screen were in a window belonging to an application not recently used, the operating system might switch the display to monochrome. The use of such features would be most acceptable if such transitions could be made unobtrusive, e.g. without a flash, and perhaps even with progressive fading.

Other transition strategies become feasible when additional hardware is present on the machine. For example, if a device can detect when the user is looking at the screen, the system can turn off the display and backlight at all other times. Such a device might consist of a light emitter and receiver on the machine and a reflector on the forehead of the (unusually docile) user. Or, it might be similar to those used by some video cameras to watch where the user is looking in order to change focus. If a sensing device can determine the ambient light level, the system can dim the backlight when ambient light is sufficiently bright to see by [71].

### 6.3 Load-change strategies

Currently, there are no formal load-change strategies for reducing the energy consumption of the display or backlight. However, it has been suggested that using a light virtual desktop pattern rather than a dark one can reduce the load on the backlight. This happens because lighter colors make the screen seem brighter and thus encourage users to use lower default backlight levels [44]. Furthermore, since most LCD's are "normally white," i.e. their pixels are white when unselected and dark when selected [78], the display of light colors consumes marginally less power than the display of dark colors. A similar strategy would be for the operating system or an application to decrease the resolution of a screen image by only illuminating a certain fraction of its pixels.

### 6.4 Future hardware innovations

Researchers working on technological advances in display and backlight hardware have many opportunities to make software power management of these components more effective. Switching to low-power modes could be made unobtrusive. If an ambient light sensor were available, the operating system could automatically reduce the backlight level when ambient light brightened. Finally, as for all other components mentioned in this paper, designers of display and backlight hardware should seek to include as many low-power modes as possible that provide reduced but reasonable levels of functionality.

## 7 Other Components

### 7.1 Main memory

Main memory is generally implemented using DRAM with three modes: active, standby, and off. In active mode, the chip is reading or writing; in standby, it is neither reading nor writing but is maintaining data by periodically refreshing it. As an example of the power consumption in these states, 8 MB of EDO memory from Micron consumes about 1.65 mW in standby mode and 580 mW in active mode [58]. The only transition strategy currently used to reduce memory energy makes use of the off state: when it is determined that the entire system will be idle for a significant period of time, all of main memory is saved to disk and the memory system is turned off. The memory contents are restored when the

system is no longer idle. When memory is saved in this manner, the machine state is said to be *suspended*; restoring memory is called *resuming*. Load-change strategies for saving memory power have been discussed before in the context of load-change strategies for saving processor power: they included using energy-aware compilers and using compact and efficient operating system code. Such strategies reduce the load on the memory system by making application and system code more compact and efficient, thus permitting greater use of the standby state. They may also convince the user to purchase a machine with less main memory, thus reducing the energy consumption of the memory system.

In future machines, memory may be divided into banks, with each bank able to turn on and off independently. Such capability broadens the ability of the operating system to manage memory energy. At times when the memory working set could fit in a small amount of memory, unused memory could be turned off. If the contents of a bank of memory were not expected to be used for a long time, they could even be saved to disk. Note that the expected period of idle time would have to be large to make up for the significant energy and time consumed in saving and restoring such memory. A related approach would be to compress, using standard data compression methods, the contents of memory, and turn off the unused banks; memory contents could be uncompressed either as needed or when activity resumed.

### 7.2 Modem

A modem can be transmitting, receiving, idle, or off. Some modems provide another state with power consumption between idle and off, called wake-on-ring; in this state, the modem consumes just enough power to detect an incoming call. MacOS uses no power saving strategies for the modem, relying on the user or an application to turn the modem on and off explicitly [48]. Presumably, this is because the operating system has no idea when data will arrive at the modem, and needs to make sure the modem is enabled whenever such data arrives so that it is not lost. A better strategy would be to have the operating system, or even the modem itself, switch the modem to the off or wake-on-ring state whenever there is no active connection to the modem.

### 7.3 Sound system

The sound system is another miscellaneous consumer of energy that can be active, idle, or off. MacOS uses an inactivity timer to decide when to turn the sound card off [48]. Another possibility is to turn the sound card off whenever a sound request from an application that triggered it turning on is completed; this has the disadvantage of increasing sound emission latency when one sound closely follows another.

## 8 Overall Strategies

It is possible to energy manage the entire computer as if it were a single component. When the computer is unneeded now and probably for some time, the operating system may put the entire system in a low-power state. Just how low-power a state depends on how long the system is expected to be idle since, in general, the lower the power of the state, the greater the time to return the system to full functionality.

Transition strategies for entering low-power system states generally use inactivity thresholds. If the user and all processes have been idle for some set period of time, the next lower system state is entered. APM 1.1 [37], a standard for energy management in computers, allows this decision-making process to be enhanced in at least two ways. First, the user may be allowed to make an explicit request to switch to a lower-power system state. Second, certain applications may be consulted before making a transition to a lower-power system state, so they can reject the request or make internal preparations for entering such a state.

Several low-power system states can be devised, and some examples of these are defined in APM 1.1. In the APM standby state, most devices are in a low-power state but can return to their full-power states quickly. For example, the disk is spun down and the backlight is off. In the APM suspend state, all devices are in very low-power states and take a relatively long time to return to functionality. For instance, the contents of memory are saved to disk and main memory is turned off. In the APM off state, the entire machine is off; in particular, all memory contents are lost and a possibly long boot period is needed to return to functionality.

Note that the sequence with which low power states are entered is significant. For example, if the memory is copied to the disk before the disk is spun down, then the machine, or at least the memory, can be shut down without spinning up the disk to establish a checkpoint.

There are both a disadvantage and an advantage to energy managing the system as a whole instead of separately managing each component. The disadvantage is that it requires all components' energy management be synchronized. Thus, if one component is still active, some other inactive component may not get turned off. Also, if it takes microseconds to determine the idleness of one component but seconds to determine the idleness of another, the first component will not be energy-managed as efficiently as possible. The advantage of treating the system as a single component is simplicity. It is simpler for the operating system to make a single prediction about the viability of entering a system state than to make separate predictions for each component state. It is simpler for an application to give hints to the operating system about when state transitions are reasonable, and to accept and reject requests by the operating system to make such transitions. Also, it is simpler for the user, if he or she is called upon to make energy management decisions, to understand and handle a few system state transitions than to understand and handle an array of individual component transitions. For these reasons, an operating system will typically do both component-level and system-level energy management. For example, APM 1.1 has a system state called enabled, in which individual component energy management is performed. After extended periods of idleness, when most components can be managed uniformly, different low-power system states can be entered.

## 9 Conclusions

Computer hardware components often have low-power modes. These hardware modes raise software issues of three types: transition, load-change, and adaptation. Several solutions to these issues have been implemented in real portable computers, others have been suggested by researchers, and many others have not yet been developed. Generally, each solution targets the energy consumption of one component.

The disk system has been the focus of many software solutions. Currently, the main hardware power-saving feature is the ability to turn the motor off by entering sleep mode. The main existing software solutions consist of entering sleep mode after a fixed period of inactivity and caching disk requests to reduce the frequency with which the disk must be spun up. Other technologies may improve the energy consumption of the storage system further, but present new challenges in file system management. These technologies include flash memory, which can function either as a secondary storage cache or a secondary storage unit, and wireless communication, which can make remote disks appear local to a portable computer.

New low-power modes for the CPU also present software challenges. Currently, the main hardware energy-saving feature is the ability to turn the CPU off, and the main existing software solution is to turn the CPU off when all processes are blocked. Other software strategies include using energy-aware compilers, using compact and efficient operating system code, and forcing processes to block when they appear to be busy-waiting. An energy-saving feature that may soon appear in portable computers is the ability to reduce the processor voltage by simultaneously reducing the clock speed. Proposed solutions that take advantage of this feature have generally been interval-based, attempting to complete all processor work by the end of each interval. However, there may be effective non-interval-based solutions as well.

The wireless communication device is appearing with increasing frequency in portable computers, and is thus an important recent focus of software energy management research. Commonly, the device is put in sleep mode when no data needs to be transmitted or received. To make this happen often, various techniques can be used to reduce the amount of time a device needs to be transmitting or receiving, including the adoption of protocols that let devices know in advance when they can be assured of no incoming data. Some new devices have the ability to dynamically change their transmission power; given this, a device needs a strategy to continually decide what power level is appropriate given quality of service requirements, interference level, and needs of neighboring units.

The display unit, including the backlight, typically consumes more power than any other component, so energy management is especially important for it. Power-saving modes available include dimming the backlight, turning the display and/or backlight off, switching from color to monochrome, and reducing the update frequency. Current system strategies only take advantage of the former two abilities, dimming the backlight and eventually turning off the display unit after a fixed period of inactivity. Other software strategies can be envisioned, especially if future hardware makes transitions to other low-power modes less obtrusive.

Other components for which software power management is possible include main memory, the modem, and the sound system. It is also possible to power manage the entire system as if it were a single component, bringing all components simultaneously to a low-power state when general inactivity is detected. Such system-level power management is simple to implement and allows simple communication with applications and users about power management; however, it should not completely supplant individual component power management because it requires synchronization of all components' power management.

To conclude, there are a few things we believe developers of future solutions to computer energy reduction should keep in mind.

1. A hardware feature is rarely a complete solution to an energy consumption problem, since software modification is generally needed to make best use of it.

2. Energy consumption can be reduced not only by reducing the power consumption of components, but also by introducing lower-power, lower-functionality modes for those components and permitting external control over transitions between those modes.

3. Standard operating system elements may need to be redesigned when dealing with low-power components that have different performance characteristics than the components they replace.

4. On a portable computer, the main goal of a component energy management strategy is to increase the amount of work the entire system can perform on one battery charge; thus, evaluation of such a strategy requires knowledge of how much energy *each* component consumes.

5. Evaluation of a power management strategy should take into account not only how much energy it saves, but also whether the trade-off it makes between energy savings and performance is desirable for users.

6. Seemingly independent energy management strategies can interact.

## References


[1] Adelberg, B., Garcia-Molina, H., and Kao, B. Emulating soft real-time scheduling using traditional operating system schedulers. *Proceedings of the 1994 Real-Time Systems Symposium*, San Juan, Puerto Rico, 292–298, December 1994.

[2] Apple Computer, Inc. *Inside Macintosh, Volume VI*, Addison Wesley Publishing Company, Reading, MA, 1991.

[3] Apple Computer, Inc. *Apple Computer*, on the World Wide Web at http://www.apple.com/, June 1997.

[4] Baker, M., Asami, S., Deprit, E., Ousterhout, J., and Seltzer, M. Non-volatile memory for fast, reliable file systems. *Proceedings of the Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS V)*, Boston, MA, 10–22, October 1992.

[5] Barringer, B., Burd, T., Burghardt, F., Burstein, A., Chandrakasan, A., Doering, R., Narayanaswamy, S., Pering, T., Richards, B., Truman, T., Rabaey, J., and Brodersen, R. InfoPad: a system design for portable multimedia access. *Proceedings of the Calgary Wireless 1994 Conference*, July 1994.

[6] Bellanger, P. and Diepstraten, W. 802.11 MAC Entity: MAC basic access mechanism / privacy and access control. Presented to IEEE P802, March 1996.

[7] Brewer, E., Burd, T., Burghardt, F., Burstein, A., Doering, R., Lutz, K., Narayansaramy, S., Pering, T., Richards, B., Truman, T., Katz, R., Rabaey, J., and Brodersen, R. Design of wireless portable systems. *Proceedings of the IEEE International Computer Society Conference (COMPCON 95)*, San Francisco, CA, 169–176, March 1995.

[8] Bertsch, J., Bernstein, K., Heller, L, Nowak, E., and White, F. Experimental 2.0 V Power/Performance Optimization of a 3.6 V-Design CMOS Microprocessor — PowerPC 601. *1994 Symposium on VLSI Technology Digest of Technical Papers*, Honolulu, HI, 83–84, 1994.

[9] Cáceres, R., Douglis, F., Li, K., and Marsh, B. Operating system implications of solid-state mobile computers. *Proceedings of the Fourth IEEE Workshop on Workstation Operating Systems*, Napa, CA, 21–27, October 1993.

[10] Caruthers, F. Hue and cry for color stirs flat-panel makers. *Computer Design, OEM Integration section*, 33(7):11–16, July 1994.

[11] Caruthers, F. Next-generation flat panels. *Computer Design, OEM Integration section*, 33(7):16–18, July 1994.

[12] Chan, E., Govil, K., and Wasserman, H. Comparing algorithms for dynamic speed-setting of a low-power CPU. *Proceedings of the First ACM International Conference on Mobile Computing and Networking (MOBICOM 95)*, Berkeley, CA, 13–25, November 1995.

[13] Chandrakasan, A. P., Sheng, S., and Brodersen, R. W. Low-power CMOS digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, April 1992.

[14] Chetto, H. and Chetto, M. Some results of the earliest deadline scheduling algorithm. *IEEE Transactions on Software Engineering*, 15(10):1261–1269, October 1989.

[15] Conger, J. *Windows API Bible*, Waite Group Press, Corte Madera, CA, 1992.

[16] Copeland, G., Keller, T., Krishnamurthy, R., and Smith, M. The case for Safe RAM. *Proceedings of the Fifteenth International Conference on Very Large Databases*, Amsterdam, Netherlands, 327–335, August 1989.

[17] Cox, D. C. Wireless personal communications: what is it? *IEEE Personal Communications*, 2(2):20–35, April 1995.

[18] Degermark, M., Engan, M., Nordgren, B., and Pink, S. Low-loss TCP/IP header compression for wireless networks. *Proceedings of the Second ACM International Conference on Mobile Computing and Networking (MOBICOM 96)*, Rye Brook, NY, 1–14, November 1996.

[19] Douglis, F., Kaashoek, F., Marsh, B., Cáceres, R., Li, K., and Tauber, J. Storage alternatives for mobile computers. *Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation*, Monterey, CA, 25–37, November 1994.

[20] Douglis, F., Krishnan, P., and Bershad, B. Adaptive disk spin-down policies for mobile computers. *Proceedings of the Second USENIX Symposium on Mobile and Location-Independent Computing*, Ann Arbor, MI, 121–137, April 1995.

[21] Douglis, F., Krishnan, P., and Marsh, B. Thwarting the power-hungry disk. *Proceedings of the 1994 Winter USENIX Conference*, San Francisco, CA, 293–306, January 1994.

[22] Duchamp, D. Issues in wireless mobile computing. *Proceedings of the Third IEEE Workshop on Workstation Operating Systems*, Key Biscayne, FL, 2–10, April 1992.

[23] Ellis, S. C. Power management in notebook PC's. *Proceedings of the Silicon Valley Personal Computing Conference*, 749–754, 1991.

[24] Fazzio, D. P., Moser, M. A., Polson, C. J., and Scheffel, J. N. Head actuator dynamics of an IBM 5 1/4-inch disk drive. *IBM Journal of Research and Development*, 37(4):479–490, July 1993.

[25] Feibus, M. and McCarron, D. Conquering notebook power. *OEM Magazine*, 2(4):60–69, April 1994.

[26] Forman, G. H. and Zahorjan, J. The challenges of mobile computing. *Computer*, 27(4):38–47, April 1994.

[27] Fox, A., Gribble, S. D., Brewer, E. A., and Amir, E. Adapting to network and client variability via on-demand dynamic distillation. *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS VII)*, Cambridge, MA, 160–170, October 1996.

[28] Fuji Photo Film Co., Ltd. *Fujifilm Develops New Generation Lithium Ion Secondary Battery*, on the World Wide Web at http://www.fujifilm.co.jp/eng/news_e/nr079.html, January 1997.

[29] Gary, S., Dietz, C., Eno, J., Gerosa, G., Park, S., and Sanchez, H. The PowerPC$^{TM}$ 603 microprocessor: a low-power design for portable applications. *Proceedings of the IEEE International Computer Society Conference (COMPCON 94)*, San Francisco, CA, 307–315, February 1994.

[30] Greenawalt, P. M. Modeling power management for hard disks. *Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, Durham, NC, 62–66, February 1994.

[31] Hansche, L. ROM BIOS: the best place for portable PC power-management features. *Proceedings of the Silicon Valley Personal Computing Conference*, 755–760, 1991.

[32] Harris, E. P., Depp, S. W., Pence, W. E., Kirkpatrick, S., Sri-Jayantha, M., and Troutman, R. R. Technology directions for portable computers. *Proceedings of the IEEE*, 83(4):636–657, April 1995.

[33] Helmbold, D. P., Long, D. D. E., and Sherrod, B. Dynamic disk spin-down technique for mobile computing. *Proceedings of the Second ACM International Conference on Mobile Computing and Networking (MOBICOM 96)*, Rye Brook, NY, 130–142, November 1996.

[34] Hewlett Packard. *Infrared IRDA Compliant Transceiver HSDL-1001 Technical Data*, on the World Wide Web at ftp://ftp.hp.com/pub/accesshp/HP-COMP/ir/hsdl1001.pdf, November 1996.

[35] Hospodor, A. D. and Hoagland, A. S. The changing nature of disk controllers. *Proceedings of the IEEE*, 81(4):586–594, April 1993.

[36] Imielinski, T., Viswanathan, S., and Badrinath, B. R. Energy efficient indexing on air. *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, Minneapolis, MN, 25–36, May 1994.

[37] Intel Corporation and Microsoft Corporation. *Advanced Power Management (APM) BIOS interface specification revision 1.1*, September 1993.

[38] Jain, R. and Werth, J. Airdisks and AirRAID: modeling and scheduling periodic wireless data broadcast. *Computer Architecture News*, 23(4):23–28, September 1995.

[39] Karlin, A. R., Manasse, M. S., McGeoch, L. A., and Owicki, S. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, June 1994.

[40] Kawaguchi, A., Nishioka, S., and Motoda, H. A flash-memory based file system. *Proceedings of the 1995 USENIX Technical Conference*, New Orleans, LA, 155–164, January 1995.

[41] Klostermeyer, W. F. and Srinivas, K. Reducing disk power consumption in a portable computer. *Operating Systems Review*, 29(2):27–32, April 1995.

[42] Krishnan, P., Long, P. M., and Vitter, J. S. Adaptive disk spindown via optimal rent-to-buy in probabilistic environments. *Duke University Technical Note CS-1995-08*, Duke University, 1995.

[43] Kuenning, G. H., Popek, G. J., and Reiher, P. L. An analysis of trace data for predictive file caching. *Proceedings of the 1994 Summer USENIX Conference*, Boston, MA, 291–303, June 1994.

[44] Kuty, D. Personal communication.

[45] Li, K., Kumpf, R., Horton, P., and Anderson, T. A quantitative analysis of disk drive power management in portable computers. *Proceedings of the 1994 Winter USENIX Conference*, San Francisco, CA, 279–291, January 1994.

[46] Lin, H.-D., Yan, R.-H., and Yu, D. Improving CMOS speed at low supply voltages. *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, Cambridge, MA, 618–621, October 1994.

[47] Lorch, J. Modeling the effect of different processor cycling techniques on power consumption. *Performance Evaluation Group Technical Note #179*, ATG Integrated Systems, Apple Computer, November 1995.

[48] Lorch, J. A complete picture of the energy consumption of a portable computer. *Masters Thesis*, Computer Science, University of California at Berkeley, December 1995.

[49] Lorch, J. and Smith, A. J. Reducing processor power consumption by improving processor time management in a single-user operating system. *Proceedings of the Second ACM International Conference on Mobile Computing and Networking (MOBICOM 96)*, Rye Brook, NY, 143–154, November 1996.

[50] Lu, Y. and Brodersen, R. Unified power control, error correction coding and scheduling for a CDMA downlink system. *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 96)*, San Francisco, CA, 1125–1132, March 1996.

[51] Lucent Technologies. *WaveLAN/PCMCIA Card User's Guide*, on the World Wide Web at ftp://ftp.wavelan.com/pub/pdf-file/pcmcia/pcman3x.pdf, October 1996.

[52] Lucent Technologies. *WaveLAN/PCMCIA Card*, on the World Wide Web at ftp://ftp.wavelan.com/pub/pdf-file/pcmcia/fs-pcm.pdf, November 1996.

[53] MacDonald, J. Power management for 386DXL-based notebook computers. *Proceedings of the Silicon Valley Personal Computing Conference*, 735–748, 1991.

[54] Mangione-Smith, B. Low power communications protocols: paging and beyond. *1995 IEEE Symposium on Low Power Electronics Digest of Technical Papers*, San Jose, CA, 8–11, October 1995.

[55] Mangione-Smith, W., Ghang, P. S., Nazareth, S., Lettieri, P., Boring, W., and Jain, R. A low power architecture for wireless multimedia systems: lessons learned from building a power hog. *1996 International Symposium on Low Power Electronics and Design Digest of Technical Papers*, Monterey, CA, 23–28, August 1996.

[56] Marsh, B., Douglis, F., and Krishnan, P. Flash memory file caching for mobile computers. *Proceedings of the 27th Hawaii Conference on Systems Sciences*, Wailea, HI, 451–460, January 1993.

[57] Maxtor Corporation. *251350, 251010 and 250840 Specifications*, on the World Wide Web at http://www.maxtor.com/products.html, February 1997.

[58] Micron Technology, Inc. *4 MEG x 16 Preliminary Datasheet MT4LC4M16R6*, on the World Wide Web at http://www.micron.com/mti/marketing/pdfs/datasheets/129.pdf, March 1997.

[59] Motorola. *PowerPC 603$^{TM}$ RISC Microprocessor Technical Summary*, 1993.

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

[61] Parrish, T. and Kelsic, G. Dynamic head loading technology increases drive ruggedness. *Computer Technology Review*, 12(16):51–55, February 1993.

[62] Pietrek, M. *Windows Internals*, Addison Wesley, Reading, MA, 1993.

[63] Powers, R. A. Batteries for low power electronics. *Proceedings of the IEEE*, 83(4):687–693, April 1995.

[64] Ritchie, D. M. and Thompson, K. The UNIX time-sharing system. *Communications of the ACM*, 17(7):365–275, July 1974.

[65] Road Warrior International. *815 MB Slimline Hard Drive Functional Specifications*, on the World Wide Web at http://warrior.com/arihd/815.html, May 1996.

[66] Rosenblum, M. and Ousterhout, J. K. The design and implementation of a log-structured file system. *ACM Transactions on Computer Systems*, 10(1):26–52, February 1992.

[67] Rulnick, J. M. and Bambos, N. Mobile power management for maximum battery life in wireless communication networks. *Proceedings of the IEEE Conference on Computer Communications (INFOCOM 96)*, San Francisco, CA, 443–450, March 1996.

[68] Saltzer, J. H., Reed, D. P., and Clark, D. D. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[69] Satyanarayanan, M., Kistler, J. J., Mummert, L. B., Ebling, M. R., Kumar, P., and Lu, Q. Experience with disconnected operation in a mobile computing environment. *Proceedings of the USENIX Mobile and Location-Independent Computing Symposium*, Cambridge, MA, 11–28, August 1993.

[70] Snyder, J. H., McKie, J. B., and Locanthi, B. N. Low-power software for low-power people. *1994 IEEE Symposium on Low Power Electronics Digest of Technical Papers*, San Diego, CA, 32–35, October 1994.

[71] Sohn, P. Personal communication.

[72] Srivastava, M. B., Chandrakasan, A. P., and Brodersen, R. W. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 4(1):42–55, March 1996.

[73] Stemm, M. and Katz, R. Measuring and reducing energy consumption of network interfaces in hand-held devices. *Institute of Electronics, Information, and Communication Engineers Transactions on Communications*, E80-B(8):1125–1131, August 1997.

[74] Suzuki, N. and Uno, S. Information processing system having power saving control of the processor clock. *United States Patent #5,189,647*, February 1993.

[75] Tiwari, V., Malik, S., Wolfe, A., and Lee, M. T. Instruction level power analysis and optimization of software. *Journal of VLSI Signal Processing*, 13(3):223–238, September 1996.

[76] Toshiba America Information Systems. *MK2720 Functional Specifications*, on the World Wide Web at http://www.toshiba.com/taisdpd/mk2720a.htm, January 1997.

[77] Weiser, M., Welch, B., Demers, A., and Shenker, S. Scheduling for reduced CPU energy. *Proceedings of the First USENIX Symposium on Operating Systems Design and Implementation*, Monterey, CA, 13–23, November 1994.

[78] Werner, K. Flat panels fill the color bill for laptops. *Circuits and Devices*, 10(4):21–29, July 1994.

[79] Western Digital Corporation. *WD Portfolio Specifications*, on the World Wide Web at http://www.wdc.com/acrobat/portfoli.pdf, February 1997.

[80] Wood, C. and Hodges, P. DASD trends: cost, performance, and form factor. *Proceedings of the IEEE*, 81(4):573–585, April 1993.

[81] Wu, M. and Zwaenepoel, W. eNVy: a non-volatile, main memory storage system. *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS VI)*, San Jose, CA, 86–97, October 1994.

[82] Zivkov, B. T. and Smith, A. J. Disk caching in large database and timeshared systems. *Proceedings of the Fifth International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 97)*, Haifa, Israel, 184–195, January 1997.

[83] Zorzi, M. and Rao, R. R. Error control and energy consumption in communications for nomadic computing. *IEEE Transactions on Computers*, 46(3):279–289, March 1997.

**Jacob Lorch** is a Ph.D. student in the computer science division of the electrical engineering and computer science department of the University of California at Berkeley. He is a National Science Foundation Fellowship recipient, as well as a member of ACM and IEEE. He holds a B.S. in computer science and a B.S. in mathematics from Michigan State University, and an M.S. in computer science from the University of California at Berkeley. His current research interests are in operating systems, energy management, and mobile computing.

**Alan Jay Smith** was raised in New Rochelle, New York, USA. He received the B.S. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, Massachusetts, and the M.S. and Ph.D. degrees in computer science from Stanford University, Stanford, California. He was an NSF Graduate Fellow.

He is currently a Professor in the Computer Science Division of the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California, USA, where he has been on the faculty since 1974; he was vice chairman of the EECS department from July, 1982 to June, 1984. His research interests include the analysis and modeling of computer systems and devices, computer architecture, and operating systems. He has published a large number of research papers, including one which won the IEEE Best Paper Award for the best paper in the IEEETC in 1979. He also consults widely with computer and electronics companies.

Dr. Smith is a Fellow of the Institute of Electrical and Electronic Engineers, and is a member of the Association for Computing Machinery, IFIP Working Group 7.3, the Computer Measurement Group, Eta Kappa Nu, Tau Beta Pi and Sigma Xi. He is on the Board of Directors (1993–99), and was Chairman (1991–93) of the ACM Special Interest Group on Computer Architecture (SIGARCH), was Chairman (1983–87) of the ACM Special Interest Group on Operating Systems (SIGOPS), was on the Board of Directors (1985–89) of the ACM Special Interest Group on Measurement and Evaluation (SIGMETRICS), was an ACM National Lecturer (1985–86) and an IEEE Distinguished Visitor (1986–87), was an Associate Editor of the ACM Transactions on Computer Systems (TOCS) (1982–93), is a subject area editor of the Journal of Parallel and Distributed Computing, and is on the editorial board of the Journal of Microprocessors and Microsystems. He was program chairman for the Sigmetrics '89 / Performance '89 Conference, program co-chair for the Second (1990), Sixth (1994), and Ninth (1997) Hot Chips Conferences, and has served on numerous program committees.