Dynamic Power Management for Portable Systems

Tajana Simunic Luca Benini* Peter Glynn† Giovanni De Micheli

Computer Systems Laboratory Stanford University

*DEIS University of Bologna

†Management Science and Engineering Department Stanford University

ABSTRACT
Portable systems require long battery lifetime while still delivering high performance. Dynamic power management (DPM) policies trade off the performance for the power consumption at the system level in portable devices. In this work we present the time-indexed SMDP model (TISMDP) that we use to derive optimal policy for DPM in portable systems. TISMDP model is needed to handle the non-exponential user request interarrival times we observed in practice. We use our policy to control power consumption on three different devices: the SmartBadge portable device [18], the Sony Vaio laptop hard disk and WLAN card. Simulation results show large savings for all three devices when using our algorithm. In addition, we measured the power consumption and performance of our algorithm and compared it with other DPM algorithms for laptop hard disk and WLAN card. The algorithm based on our TISMDP model has 1.7 times less power consumption as compared to the default Windows timeout policy for the hard disk and three times less power consumption as compared to the default algorithm for the WLAN card.

1. INTRODUCTION
Battery-operated portable systems demand tight constraints on energy consumption. System-level *dynamic power management* [1] algorithms increase the battery lifetime by selectively placing idle components into lower power states. System resources can be modeled using state-based abstraction where each state trades off performance for power [2]. State transitions are controlled by commands issued by a *power manager* (PM) that observes the workload of the system and decides when and how to force power state transitions. The power manager makes state transition decisions according to the *power management policy*. The choice of the policy that minimizes power under performance constraints (or maximizes performance under power constraint) is a constrained policy optimization problem which is very important for all portable systems.

Several heuristic power management policies have been investigated in the past for controlling hard disks and interactive terminals. The most common policy for hard disks is a *timeout policy* implemented in most operating systems. The drawback of this policy is that it wastes power while waiting for the timeout to expire.

Timeout policy is also at the heart of the power management implementation in new IEEE 802.11 standard for wireless LANs. Power management is implemented at *medium access control* (MAC) and physical layers [23]. The standard requires that a *central access point* (AP) send out a beacon every 100ms followed by a *traffic indication map* (TIM). Each card that desires to communicate has to actively listen for the beacon in order to synchronize the clock with AP, and for the TIM to find out if any data is arriving for it. If it does not need to transmit or receive, the card can then go to low-power state until the next beacon. The IEEE standard does not address the power management implementation at the system level. If the card is turned off when it is not being used, much larger power savings can be observed.

Another group of policies studied in the past are predictive policies for hard disks [3, 14, 15, 9] and for interactive terminals [4, 5, 16]. Predictive policies force the transition to a low power state as soon as a component becomes idle if the predictor estimates that the idle period will last long enough. A wrong estimate can cause both performance and energy penalties.

All heuristic policies discussed above have to be tested with simulations or measurements to assess their effectiveness and as such do not offer any globally optimal results. The policy optimization technique proposed by Paleologo et al. [6, 7] solves the policy optimization problem with globally optimal results using an approach based on *discrete-time Markov decision processes* (DTMDP). The DTMDP approach requires that all state transitions follow stationary geometric distribution, which is not true in many practical cases. Non-stationary user request rates can be treated using an adaptive policy interpolation procedure presented in [17]. A limitation of both stationary and adaptive DTMDP policies is that decision evaluation is repeated periodically, even when the system is in the sleep state, thus wasting power. For example, for a 10 W processor, the DTMDP policy with period of 1s would waste as much as 1800 J of energy out of the battery during a 30min break. The advantage of the discrete time approach is that decisions are re-evaluated periodically so the decision can be reconsidered thus adapting better to arrivals that are not truly geometrically distributed.

An extension to the DTMDP model is a *continuous-time Markov decision process* (CTMDP) model [10, 11]. In CTMDP power manager (PM) issues commands upon event occurrences instead of at discrete time settings. System transitions are assumed to follow exponential distribution. We show that models that are exclusively based on the exponential arrival times in the idle state can have high energy costs and large performance penalty because the power manager makes one decision as soon as the system goes idle. If the decision is to stay awake, the system will wait until another arrival before revising the decision, possibly missing large idle times, such as a lunch break. CTMDP model was further generalized with *semi-Markov decision process model* (SMDP) [12]. SMDP model can treat a general distribution occuring at the same time with an exponential distribution. In [12] the hard disk transition to and from the sleep state was modeled using uniform distribution, while the exponential distribution still represented the user request arrivals. Some measurements constrasting CTMDP and SMDP are shown in [13].

Although exponential distribution model can be used to model arrivals in the active states, we show that the arrivals in the idle states are better modeled with Pareto distribution after filtering out small interarrival times. Non-exponential arrival distribution coupled with uniform transition distribution requires the *time-indexed Markov chain SMDP model* (TISMDP) which we present in this work. The policy optimization problem based on the time-indexed SMDP model can be solved *exactly* and in polynomial time by solving a linear optimization problem. This model combines the advantages of event-driven SMDP model with discrete-time MDP model. Policy decisions are made in event-driven manner in all states but idle state thus saving power by not forcing policy re-evaluations. In the idle state, policy decision are re-evaluated until the transition is made into the sleep state, thus saving power during longer breaks.

We not only obtain globally optimal results for policy optimization using the TISMDP model, but we also present simulation and, more importantly, real measurements of a laptop hard disk and WLAN card. Our measurement results show that reduction in power can be as large as 1.7 times for the hard disk and 3 times for the WLAN card with a small performance penalty. As the SmartBadge operating system does not have API for controlling power states of its components, we simulated the effect out algorithm has on power savings based on previously collected user trace.

The rest of the paper is organized as follows. Section 2 describes the stochastic models of the system components. The models are based on the experimental data collected. Next we present the time-indexed semi-Markov decision process model for the dynamic power management policy optimization problem in Section 3. Our model guarantees optimal power management policy given the stochastic models we derived from the measurements. We show simulation results for the SmartBadge, followed by simulated and measured results of the hard disk and the WLAN card in Section 4. Finally, we summarize our findings in Section 5.

2. SYSTEM MODEL

The optimization of energy consumption under performance constraint (or vice versa) is performed for three different devices: a hard disk in a laptop computer, a personal communication interactive device, SmartBadge [18], and a WLAN card [24].

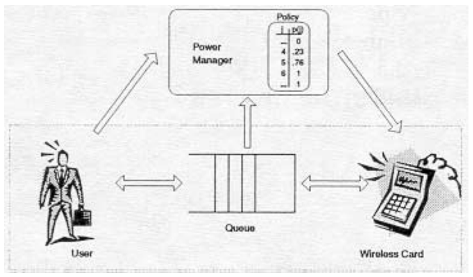

Figure 1: System Model

The system modeled consists of the user, device (hard disk, SmartBadge or WLAN card) with queue (the buffer associated with the device) as shown in Figure 1. The power manager observes the all event occurrences of interest and makes decisions on what state the system should transition to next, in order to minimize energy consumption for a given performance constraint. Each system component is described in the next sections.

2.1 User Model

We collected 11hr user request trace for the laptop hard disk running a Windows operating system with standard software. In the case of the SmartBadge, we monitored the accesses to the server. For WLAN card we used *tcpdump* utility [25] to get the user request arrival times for two different applications (telnet and web browser).

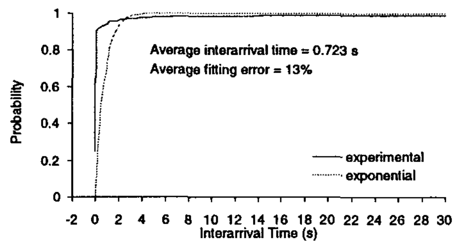

Figure 2: User request arrivals in active state for hard disk

The request interarrival times in the active state (the state where one or more requests are in the queue) for all three devices are exponential in nature. Figure 2 shows exponential cumulative distribution fitted to measured results of the hard disk. Similar results have been observed for the other two devices in the active state. Thus we can model the user (service requestor or SR) in active state with rate λ_{SR} and the request interarrival time $\frac{1}{\lambda_{SR}}$ where the probability of the hard disk or the SmartBadge receiving a user request within time interval t follows the cumulative probability distribution shown below.

$$E_{SR}(t) = 1 - e^{-\lambda_{SR}t} \qquad (1)$$

The exponential distribution does not model well arrivals in the idle state. The model we use needs to accurately describe the behavior of long idle times as the largest power savings are possible over the long sleep periods. We first filter out short user request interarrival times in the idle state in order to focus on longer idle times. Thus, all arrivals within time period smaller than the filter interval are just considered to be one long busy period. We found that filter intervals from 0.5s to about 2s are most appropriate to use for the hard disk, while for the SmartBadge and the WLAN card filter intervals are considerably shorter (50-200ms) since these devices respond much faster than the hard disk.

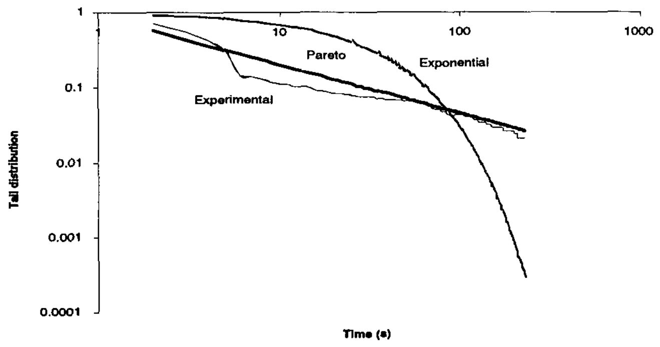

Figure 3: Hard disk idle state arrival tail distribution

Figure 3 shows measured tail distribution of idle periods fitted with Pareto and exponential distributions for the hard disk and Figure 4 shows the same measurements for the WLAN card. The Pareto distribution shows a much better fit for the long idle times as compared to the exponential distribution. The Pareto cumulative distribution is defined in Equation 2. Pareto parameters are $a = 0.9$ and $b = 0.65$ for the hard disk, $a = 0.7$ and $b = 0.02$ for WLAN web requests and $a = 0.7$ and $b = 0.06$ for WLAN telnet requests. SmartBadge arrivals behave the same way as the WLAN arrivals.

$$E_{SR}(t) = 1 - at^{-b} \tag{2}$$

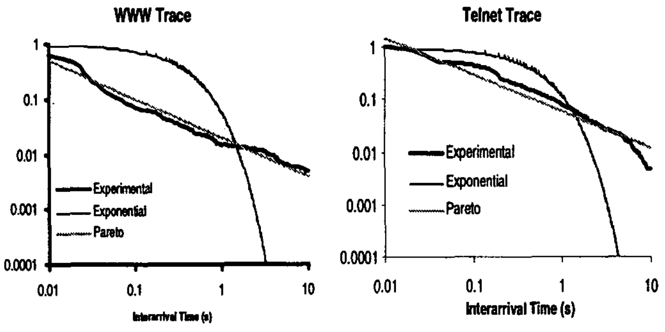

Figure 4: WLAN idle state arrival tail distribution

Since Pareto distribution of arrivals in the idle state is not memoryless as the exponential distribution is, the optimization model needs to include the time-indexing of the states. The new model is presented in Section 3.

2.2 Portable Devices

Portable devices typically have multiple power states. The hard disk interface supports the ACPI standard [2]. The hard disk we used has three states that can be observed from the filter driver we have implemented: *active*, *idle* and *sleep* state. The SmartBadge has three power states: *active*, *idle* and *standby*. The wireless card has multiple power states: two active states, *transmitting*, *receiving*, and two inactive states, *doze* and *off*. We will treat WLAN's doze state as a low-power idle state. The goal of power management policy is to maximize the time each device spends in its low power state (standby, sleep and off respectively), with good performance.

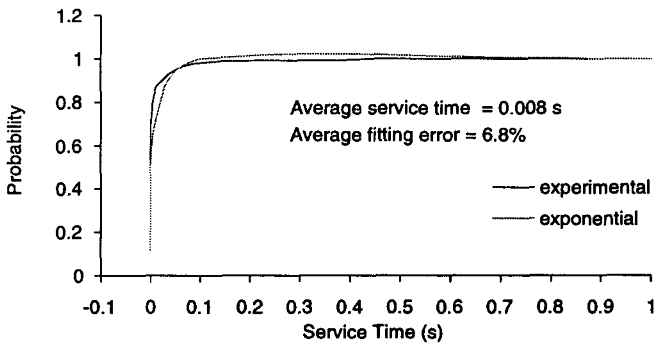

Figure 5: Hard disk service time distribution

In the active state, the device services requests coming from the user. Service times on the hard disk most closely follow an exponential distribution as shown in Figure 5. We found similar results for the SmartBadge and the WLAN card. The average service time is defined by $\frac{1}{\lambda_{SP}}$ where λ_{SP} is the average service rate. Equation 3 defines the cumulative probability of the device (service provider or SP) servicing a user request within time interval t.

$$E_{SP}(t) = 1 - e^{-\lambda_{SP} t} \tag{3}$$

2.2.1 Hard Disk

The Fujitsu MHF 2043AT hard disk we used in our experiments supports two states in which the disk is spinning – idle and active with average power consumption of $0.95W$. When the data is being read or written on the disk, power consumption is $2.5W$, but since the service rate is very high, the average power is $0.95W$. The hard disk automatically transitions into idle state as soon as it is done reading or writing to the disk. The transition from sleep to active state requires spin-up of the hard disk, which is very power intensive; in this case $2.1W$. While in the sleep state, the disk consumes only $0.13W$.

The power manager can control the transitions between the idle and the sleep state. Once in the sleep state, the hard disk waits for the first service request arrival before returning back to the active state. The transition between active and sleep states are best described using uniform distribution, where t_0 and t_1 can be defined as $t_{ave} - \Delta t$ and $t_{ave} + \Delta t$ respectively. The cumulative probability function for the uniform distribution is shown below.

$$E_{SP}(t) = \begin{cases} 0 & t \leq t_0 \\ \frac{t-t_0}{t_1-t_0} & t_0 \leq t \leq t_1 \\ 1 & t \geq t_1 \end{cases} \quad (4)$$

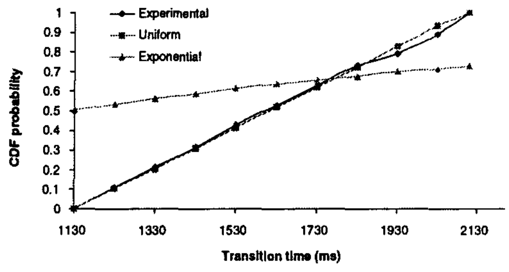

Figure 6: Hard disk transition from sleep to active state

Figure 6 shows the large error that would be made if transition to sleep state were approximated using an exponential distribution. The transition from the active state into the sleep state takes on average $0.67s$ with variance of $0.1s$. The transition back into the active state is much longer, requiring $1.6s$ on average with $0.5s$ variance.

2.2.2 *SmartBadge*

The SmartBadge, shown in Figure 7, is an embedded system consisting of the Sharp's display, WLAN RF link, StrongARM-1100 processor, Micron's SDRAM, FLASH, sensors, and modem/audio analog front-end on a PCB board powered by the batteries through a DC-DC converter. The initial goal in designing the SmartBadge was to allow a computer or a human user to provide location and environmental information to a location server through a heterogeneous network. The SmartBadge could be used as a corporate ID card, attached (or built in) to devices such as PDAs and mobile telephones, or incorporated in computing systems.

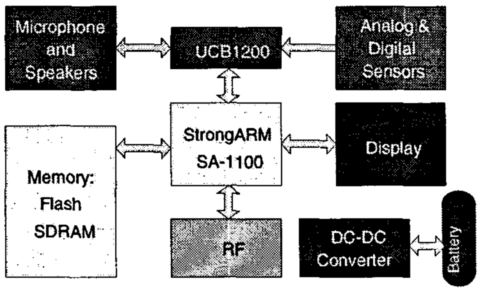

Figure 7: SmartBadge

The SmartBadge supports two lower power states: idle and standby. The idle state is entered immediately by each component in the system as soon as that particular component is not accessed. The standby state transitions can be controlled by the power manager. Similar to the hard disk, the transition from standby into the active state can be best described using the uniform probability distribution. Components in the SmartBadge, the power states and the transition times of each component from standby into active state are shown in the Table 1. Note that the SmartBadge has two types of data memory – slower SRAM (1MB, 80ns) from Toshiba and faster DRAM (4MB, 20ns) from Micron that is used only during MPEG decode.

Table 1: SmartBadge components

Component	Active Pwr (mW)	Idle Pwr (mW)	Standby Pwr (mW)	t_{ave} (ms)
Display	1000	1000	100	100
RF Link	1500	1000	100	80
SA-1100	400	170	0.1	10
FLASH	75	5	0.023	0.6
SRAM	115	17	0.13	5.0
DRAM	400	10	0.4	4.0
Total	3.5 W	2.2 W	200 mW	150 ms

2.2.3 *WLAN card*

Lucent's wireless card has multiple power states: two active states, *transmitting*, *receiving*, and two inactive states, *doze* and *off*. Transmission power is 1.65W, receiving 1.4W, the power consumption in the doze state is 0.045W [24] and in off state it is 0W. When the card is awake (not in the off state), every 100ms it synchronizes its clock to the access point (AP) by listening to AP beacon. After that it listens to TIM map to see if it can receive or transmit during that interval. Once both receiving and transmission are done, it goes to doze state until the next beacon. This portion of the system is fully controlled from the hardware and thus is not accessible to the power manager that has been implemented at the OS level.

The power manager can control the transitions between the doze and the off states. Once in the off state, the card waits for the first user request arrival before returning back to the doze state. We measured the transitions between doze and off states using *cardmgr* utility. The transition from the doze state into the off state takes on average $t_{ave} = 62ms$ with variance of $t_{var} = 31ms$. The transition back takes $t_{ave} = 34ms$ with $t_{var} = 21ms$ variance. The transition between doze and off states are best described using uniform distribution.

2.3 Queue

Portable devices normally have a buffer for storing requests that are have not been serviced yet. Since we did not have access to the detailed information about the real-time size of each queue, we measured the queue size of maximum 10 jobs with an experiment on a hard disk using a typical user trace. Because the service rate in the SmartBadge and WLAN card is higher, and the request arrival rate is comparable, we assume that the same maximum queue size can be used. As the requests arriving at the hard disk do not have priority associated with them, and the SmartBadge requests by definition do not have priority, our queue model contains only the number of jobs waiting service. Active and sleep states can be differentiated then by the number of jobs pending for service in the queue.

3. THEORETICAL BACKGROUND

Power manager (PM) observes user request arrivals, the number of jobs in the queue at run time and the time elapsed since last entry into idle state. In our system, the only decisions PM can make is whether to leave the device in idle state or turn it off completely. If it is left on, the device will wait in the idle state until the first request arrives from the user. Upon arrival of request, it transitions into active state. If the PM turns the device off or places it into sleep state, the system starts a transition between idle and off state. If during the transition time a request arrives from the user, the device starts the transition to active state as soon as transition to off state is completed. If no request arrives during the transition to off state, the device stays off (or asleep) until the next request arrives. Upon request arrival, the transition back into active state starts. Once transition into active state is completed, the device services requests, and then again returns to idle state.

In this section we present the power management optimization problem formulation. Our goal is to minimize performance penalty under energy consumption constraint (or vice versa). We first present the average-cost semi-Markov decision process optimization problem [8] and then extend it into time-indexed SMDP for modeling general interarrival times. We use upper-case bold letters (*e.g.*, **M**) to denote matrices, lower-case bold letters (*e.g.*, **v**) to denote vectors, calligraphic letters (*e.g.*, $\mathcal{S}$) to denote sets, upper-case letters (*e.g.*, S) to denote scalar constants and lower-case letters (*e.g.*, s) to denote scalar variables.

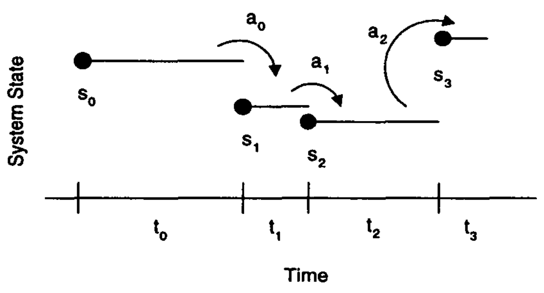

Figure 8: SMDP Progression

Figure 8 shows a progression of the SMDP through event occurrences, called decision epochs. The power manager makes decisions at each event occurrence. *Inter-event time set* is defined as $\mathcal{T} = \{t_i, \text{ s.t. } i = 0, 1, 2, \ldots, T\}$ where each t_i is the time between the two successive event arrivals We denote by $s_i \in \mathcal{S}$ the system state at decision epoch i. Commands are issued whenever the system state changes. We denote by $a_i \in \mathcal{A}$ an action (or command) that is issued at decision epoch i. When action a_i is chosen in system state s_i, the probability that the next event will occur by time t_i is defined by the cumulative probability distribution $E(t_i|s_i, a_i)$. Also, the probability that the system transition to state s_{i+1} at or before the next decision epoch t_i is given by $p(s_{i+1}|t_i, s_i, a_i)$.

The SMDP model also defines cost metrics. The average cost incurred between two successive decision epochs (events) is defined in Equation 5 as a sum of the lump sum cost $k(s_i, a_i)$ incurred when action a_i is chosen in state s_i, followed by the cost incured at rate $c(s_{i+1}, s_i, a_i)$ as long as the system is in state s_{i+1} after choosing action a_i in state s_i.

$$cost(s_i, a_i) = k(s_i, a_i) + \int_0^\infty [E(du|s_i, a_i) \sum_{s_{i+1}} \int_0^u c(s_{i+1}, s_i, a_i) p(s_{i+1}|t_i, s_i, a_i)]dt \quad (5)$$

With this formalizam the policy optimization is to find a policy such that Equation 5 is minimized. The policy optimization problem can be solved in polynomial time (in $S \cdot A$) with linear programming [8]. The linear programming (LP) formulation of the average-cost SMDP problem can be naturally extended with performance constranits. The LP formulation used in this work to obtain the optimal policy is shown below.

$$\begin{aligned} \textbf{LP: } \min \quad & \sum_{s \in S} \sum_{a \in A} cost_{energy}(s, a) f(s, a) & (6) \\ \text{s.t.} \quad & \sum_{a \in A} f(s, a) - \sum_{s' \in S} \sum_{a \in A} m(s'|s, a) f(s', a) = 0 \\ & \sum_{s \in S} \sum_{a \in A} y(s, a) f(s, a) = 1 \\ & \sum_{s \in S} \sum_{a \in A} cost_{perf}(s, a) f(s, a) < Constraint \end{aligned}$$

where the probability of arriving to state s_{i+1} given that the action a_i was taken in state s_i are defined by:

$$m(s_{i+1}|s_i, a_i) = \int_0^\infty p(s_{i+1}|t_i, s_i, a_i) E(dt|s_i, a_i) \quad (7)$$

and the expected time spent in each state is given by:

$$y(s_i, a_i) = \int_0^\infty t \sum_{s_{i+1} \in S} p(s_{i+1}|t_i, s_i, a_i) E(dt|s_i, a_i) \quad (8)$$

The $A \cdot S$ unknowns in the LP, $f(s, a)$, called *state-action frequencies*, are the expected number of times that the system is in state s and command a is issued. It has been shown that the exact and the optimal solution to the SMDP policy optimization problem belongs to the set of Markovian randomized stationary policies [8]. A *Markovian randomized stationary policy* can be compactly represented by associating a value $x(s, a) \leq 1$ with each state and action pair in the SMDP. The probability of issuing command a when the system is in state s, $x(s, a)$, is defined in Equation 9.

$$x(s_i, a_i) = \frac{f(s_i, a_i)}{\sum_{a_i \in A} f(s_i, a_i)} \quad (9)$$

The average-cost SMDP formulation presented above is based on the assumption that at most one of the underlying processes in each state transition is not exponential in nature. On transitions where the processes are not exponential, time-indexed Markov chain formulation needs to be used to keep

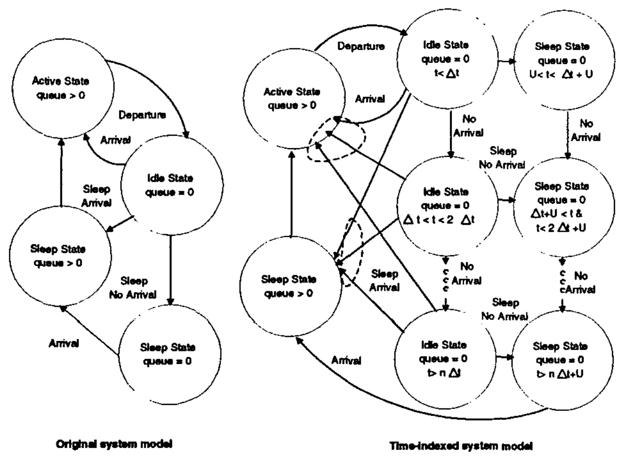

Figure 9: Time-indexed SMDP states

the history information [19]. Time-indexing is done by dividing the time line into a set of intervals of equal length Δt. The original state space is expanded by replacing one idle and one sleep with queue empty state with a series of time-indexed idle and sleep empty states as showing in Figure 9. Note that time-indexed SMDP can contain non-indexed states. If an arrival occurs while in the idle state, the system transitions automatically to the active state. When no arrival occurs during the time spent in a given idle state the power manager can choose to either stay awake, in which case the system enters the next idle state or to transition into sleep state. When the transition to sleep occurs from an idle state, the system can arrive to the sleep state with queue empty or with jobs waiting in the queue. The sleep state with queue empty is indexed by the time from first entry into idle state from active state, much in the same way idle states are indexed, thus allowing accurate modeling of the first arrival. The LP formulation for average-cost SMDP still holds, but the cost, probability and expected time functions have to be redefined for time-indexed states in SMDP. Namely, for time-indexed states Equation 5 is replaced by:

$$cost(s_i, a_i) = k(s_i, a_i) + \sum_{s_{i+1} \epsilon S} c(s_{i+1}, s_i, a_i) y(s_i, a_i) \quad (10)$$

and Equation 8 by:

$$y(s_i, a_i) = \int_{t_i}^{t_i+\Delta t} \frac{(1 - E(t))dt}{1 - E(t_i)} \quad (11)$$

Similarly, the probability of getting an arrival is defined using the time indices for the system state where $t_i \le t \le t_i + \Delta t$:

$$p_\pi(s_{i+1}|t_i, s_i, a_i) = \frac{E(t_i + \Delta t) - E(t_i)}{1 - E(t_i)} \quad (12)$$

Finally, the probability of transitioning to the next idle state defined in Equation 7 is set to $m(s_{i+1}|s_i, a_i) = 1 - p(s_{i+1}|t_i, s_i, a_i)$ and of transitioning back into the active state is $m(s_{i+1}|s_i, a_i) = p(s_{i+1}|t_i, s_i, a_i)$. The general cumulative distribution of event occurrences is given by $E(t_i)$. For example, $E(t_i)$ defines the probability of getting a Pareto distributed arrival in the idle state by time t_i.

In a specific case of Pareto distribution, it can be shown that the probabilities of going to sleep are monothonically increasing and can thus be interpreted as randomized timeout. Such policies filter out short idle times, and then with an increasing probability give command to go to sleep until the timeout value, at which point the system goes to sleep (or is turned off). Thus we can simplify the implementation of power manager (PM). If the PM's decision upon entry to idle state is to leave the device active, the PM waits until the time where the probability of transition to sleep or off state is greater than zero. From that point on, the decision is reevaluated every Δt seconds until either the device is turned off or a user request arrives and the device transitions into the active state. At each evaluation the PM generates a random number using a pseudo-random number generator. If the number generated is less than probability of going to sleep for the current time interval, the PM shuts down the card. Otherwise the device stays in the idle state until either the next evaluation time or until request arrives that forces the transition into active state. Once the device is turned off, it stays off until the first request arrives, at which point it transitions into active state.

4. RESULTS

We perform the policy optimization using lp_solve, the solver for linear programs [20]. The optimization runs in just under 1 minute on a 300MHz Pentium processor. We first verified the optimization results using simulation. Inputs to the simulator are the system description, the expected time horizon, the number of simulations to be performed and the policy. The system description is characterized by the power consumption in each state, the performance penalty, and the function that defines the transition time probability density function and the probability of transition to other states given a command from the power manager. Note that our simulation used both probability density functions (pdfs) we derived from data and the original traces. When using pdfs, we just verified the correctness of our problem formulation. With real traces we were able to verify that indeed pdfs we derived do in fact match well the data from the real system, and thus give optimal policies for the real systems. The results of the optimization are in close agreement with the simulation results.

4.1 SmartBadge

The simulated power and performance tradeoffs for the SmartBadge with our TISMDP policy are shown in Figure 10. Performance penalty is defined as the percent of time system spends in a low-power state with non-empty queue. In general, the goal is to have as few requests as possible waiting for service. For systems with hard real-time constraint, this penalty can be set to large values to force less agressive power management, thus resulting is less requests queued up for service. In systems where it is not as critical to meet time deadlines, the system can stay in a low-power state longer, thus accumulating more requests that can be serviced upon return to the active state.

Because of the particular design characteristics of the SmartBadge, the tradeoff curves of performance penalty and power savings are very close to linear. When probability of going to sleep is zero, no power can be saved, but performance penalty can be reduced by 85% as compared to the case where probability is one. On the other hand, about 50% of the power can be saved when system goes to sleep upon entery to idle state.

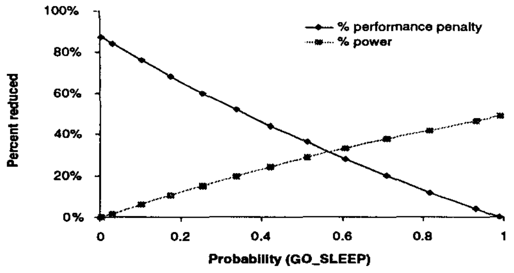

Figure 10: SmartBadge DPM results

4.2 Hard Disk

We implemented the power manager as part of the filter driver template discussed in [21]. A filter driver is attached to the vendor-specific device driver. Both drivers reside in the operating system, on the kernel level, above the ACPI driver implementations. Application programs such as word processors or spreadsheets send requests to the OS. When any event occurs that concerns the hard disk, power manager is notified. When the PM issues a command, the filter driver creates a power transition call and sends it to the device which implements the power transition using ACPI standard.

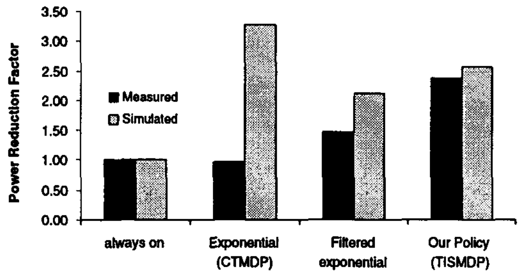

Figure 11: Measured and simulated hard disk power consumption

We measured and simulated three different policies based on stochastic models and compared them with the always-on policy. All stochastic policies minimized power consumption under a 10% performance constraint. The results are shown in Figures 11 and 12. These figures best illustrate the problem we observed when user request arrivals are modeled only with exponential distribution as in CTMDP model [10]. The simulation results for the exponential model (CTMDP) show large power savings, but measurement results show no power savings and a very high performance penalty. As the exponential model is memoryless, the resulting policy makes a decision as soon as the device becomes idle. If the idle time is very short, the exponential model gets a large performance penalty due to the wakeup time of the device and a considerable cost in shut-down and wakeup energies. In addition, if the decision upon entry to idle state is to stay awake, large idle times, such as lunch breaks, will be missed. When we filter out short arrival times (filtered exponential), the results improve but not significantly. The best results are obtained with our policy based on TISMDP model. In fact, our policy uses 2.4 times less power than the always-on policy. These results show that it is critically important to not only simulate, but also measure the results of each policy and thus verify the assumptions made in modeling. In fact, we found that modeling based on simple Markov chains is not accurate, and that we do require more complex time-indexed semi-Markov decision process model presented in this paper.

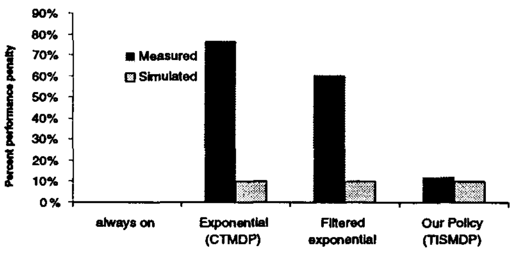

Figure 12: Measured and simulated hard disk performance

The effect of filtering arrivals into the idle state is best shown in Figure 13 for the policy with the performance penalty of the laptop hard disk limited to 10%. This plot shows that the best filtering intervals are on the order of seconds. Similar plots have been obtained for the SmartBadge and WLAN card. The best filtering time depends on device design. For example, hard disk spin-up time is on the order of seconds, thus filtering time of the same order makes most sense.

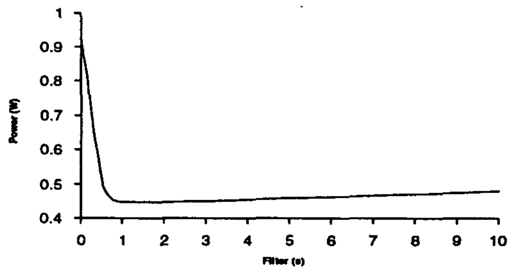

Figure 13: Hard disk power consumption vs. filtering time

Finally, we measured on a real laptop four different policies including ours. Comparison is shown in Table 2. The Karlin's algorithm analysis [22] with timeout set to the break even time of the hard disk (5.43 seconds) is guaranteed to yield a policy that consumes at worst twice the minimum amount of power consumed by the policy computed with perfect knowledge of the user behavior. The reason for this result is that the worst ase happens for workloads with repeated idle periods of length equal to the break even time, separated by pointwise activity. The Karlin's policy consumes 10% more power and has worse performance than the policy based on our time-indexed semi-Markov decision process model. In addition, our TISMDP policy consumes 1.7 times less power than the default Windows timeout policy of 120s and 1.4 times less power than the 30s timeout policy. The policy based on continuous-time model (CTMDP) performs worse then always-on policy, primarily due to the exponential interarrival request assumption. This policy both misses some long idle periods, and tends to shut-down too aggressively, as can be seen from its very short average sleep time.

Performance of the algorithms can be compared using three different measures. The number of times the policy issued sleep command (N_{sd}) should be larger than the number of times sleep command was issued and the hard disk was asleep for shorter than the time it takes to recover the cost of spinning down and spinning up the disk (N_{ud}). Finally, the average length of time spent in the sleep state (T_{ss}) should be as large as possible while still keeping the power consumption down. From our experience with the user interaction with the hard disk, the TISMDP algorithm performs well, thus giving us low-power consumption with still good performance.

Table 2: Hard Disk Measurement Comparison

Algorithm	Pwr (W)	N_{sd}	N_{ud}	$T_{ss}(s)$
oracle	0.33	250		118
TISMDP	0.40	326	76	81
Karlin's	0.44	323	64	79
30s timeout	0.51	147	18	142
120s timeout	0.67	55	3	238
always on	0.95	0	0	0
CTMDP	0.97	391	359	4

The event-driven nature of our algorithm, as compared to algorithms based on discrete time intervals, saves considerable amount of power while in sleep state as it does not require policy evaluation until an event occurs. For example, if we assume that the processor consumes 10 W in the active state and that the discrete time based policy is set to 1s re-evaluation timeout and takes 100ms to execute, during a normal 30 minute break, the policy would use 1800 J of energy out of the battery. With event-driven policy, the processor could be placed in low-power mode during the break time thus saving a large portion of battery capacity.

4.3 WLAN card

For WLAN measurements we used Lucent's WLAN 2Mb/s card [24] running on the laptop. As mobile environment is continually changing, it is not possible to reliably repeat the same experiment. As a result, we needed to use trace-based methodology discussed in [26]. The methodology consists of three phases: collection, distillation and modulation. We used *tcpdump* [25] utility to get the user's trace for two different applications: web browsing and telnet. During distillation we prepared the trace for the modeling step. We had a LAN-attached host read the distilled trace and delay or drop packets according to the parameters we obtained from the measurements. In this way we were able to recreate the experimental environment, so that different algorithms can be reliably compared.

We implemented three different versions of our policy for each application, each with a different combination of power and performance penalty values (TISMDP a,b,c for web browserand TISMDP 1,2,3 for telnet). Since web and telnet arrivals behave differently (see Figure 4), we observe through OS what application is currently actively sending and use the appropriate power management policy.

Table 3: DPM for WLAN Web Browser

Policy	N_{sd}	N_{wd}	$T_{penalty}$ (sec)	P_{ave} (W)
Oracle	395	0	0	0.467
TISMDP(a)	363	96	6.90	0.474
TISMDP(b)	267	14	1.43	0.477
Karlin's	623	296	23.8	0.479
TISMDP(c)	219	9	0.80	0.485
CTMDP	3424	2866	253.7	0.539
Default	0	0	0	1.410

In addition to measuring the energy consumption (and then calculating average power), we also quantified performance penalty using three different measures. Delay penalty, T_p, is the time system had to wait to service a request since the card was in the sleep state when it should not have been. In addition, we measure the number of shutdowns, N_{sd} and the number of wrong shutdowns, N_{wd}. A shutdown is viewed as wrong when the sleep time is not long enough to make up for the energy lost during transition between the idle and off state. The number of shutdowns is a measure of how eager the policy is, while a number of wrong shutdowns tells us how accurate the policy is in predicting a good time to shut down the card.

Table 4: DPM for WLAN Telnet Application

Policy	N_{sd}	N_{wd}	$T_{penalty}$ (sec)	P_{ave} (W)
Oracle	766	0	0	0.220
TISMDP(1)	798	21	2.75	0.269
TISMDP(2)	782	33	2.91	0.296
Karlin's	780	40	3.81	0.302
TISMDP(3)	778	38	3.80	0.310
CTMDP	943	233	20.53	0.361
Default	0	0	0	1.410

The measurement results for a 2.5hr web browsing trace are shown in Table 3. Our policies (TISMDP a,b,c) show on average a factor of three in power savings with low performance penalty. The Karlin's algorithm [22] guarantees to be within a factor of two of oracle policy. Although its power consumption is low, it has a performance penalty that is an order of magnitude larger than for our policy. A policy that assumes exponential arrivals only, CTMDP [10], has a very large performance penalty because it makes the decision as soon as the system enters idle state.

Table 4 shows the measurement results for a 2hr telnet trace. Again our policy performs best, with a factor of five in power savings and a small performance penalty. Telnet application allows larger power savings because on average it transmits and receives much less data then the web browser, thus giving us more chances to shut down the card.

5. CONCLUSIONS

We presented a new optimal approach based on the time-indexed semi-Markov decision processes (TISMDP) for optimizing power management policies of portable systems. We simulated and measured large power savings using our approach on three different portable devices: the SmartBadge, the Sony Vaio laptop hard disk and Lucent's WLAN card. The measurements for the hard disk show that our policy gives 1.7 times lower power consumption as compared to the default Windows timeout policy. In addition, our policy obtains on average 3 times lower power consumption for the wireless card relative to the default policy. As our approach is event-driven, we obtain further system power savings in low-power modes as compared to the discrete time based approaches.

6. ACKNOWLEDGMENTS

This work was supported in part by NSF under grant number CCR-9901190. In addition, we would like to thank Hewlett-Packard Laboratories for their support.

7. REFERENCES

[1] L. Benini and G. De Micheli, *Dynamic Power Management: design techniques and CAD tools*, Kluwer, 1997.

[2] Intel, Microsoft and Toshiba, "Advanced Configuration and Power Interface specification", available at *http://www.intel.com/ial/powermgm/specs.html*, 1996.

[3] R Golding, P. Bosch and J. Wilkes, "Idleness is not sloth" *HP Laboratories Technical Report* HPL-96-140, 1996.

[4] M. Srivastava, A. Chandrakasan. R. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Transactions on VLSI Systems*, vol. 4, no. 1, pp. 42–55, March 1996.

[5] C.-H. Hwang and A. Wu, "A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation", in *Proceedings of the Int.l Conference on Computer Aided Design*, pp. 28-32, 1997.

[6] G. Paleologo, L. Benini, A. Bogliolo and G. De Micheli, "Policy Optimization for Dynamic Power Management", in *Proceedings of Design Automation Conference*, pp. 173–178, 1998.

[7] L. Benini, A. Bogliolo, G. Paleologo and G. De Micheli, "Policy Optimization for Dynamic Power Management", in *Transactions on CAD*, 1999.

[8] M. Puterman, *Finite Markov decision processes*, John Wiley and Sons, 1994.

[9] Y. Lu and G. De Micheli, "Adaptive Hard Disk Power Management on Personal Computers", *IEEE Great Lakes Symposium on VLSI*, pp. 50–53, 1999.

[10] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time markov decision processes", *Design Automation Conference*, pp. 55–561, 1999.

[11] Q. Qiu, Q. Wu and M. Pedram, "Stochastic Modeling of a Power-Managed System: Construction and Optimization", *Proceedings of ISLPED*, pp. 194-199, 1999.

[12] T. Simunic, L. Benini and G. De Micheli, "Event-driven power management", *Proceedings of ISSS*, pp. 18-23, 1999.

[13] T. Simunic, L. Benini and G. De Micheli, "Power Management of Laptop Hard Disk", *Proceedings of DATE*, p.736, 2000.

[14] F. Douglis, P. Krishnan and B. Bershad, "Adaptive Disk Spin-down Policies for Mobile Computers", *Second USENIX Symposium on Mobile and Location-Independent Computing*, pp. 121–137, 1995.

[15] D. Helmbold, D. Long and E. Sherrod, "Dynamic Disk Spin-down Technique for Mobile Computing", *IEEE Conference on Mobile Computing*, pp. 130–142, 1996.

[16] L. Benini, R. Hodgson and P. Siegel, "System-Level Power Estimation and Optimization", *International Symposium on Low Power Electronics and Design*, pp. 173–178, 1998.

[17] E. Chung, L. Benini and G. De Micheli, "Dynamic Power Management for non-stationary service requests", *Proceedings of DATE*, pp. 77–81, 1999.

[18] G. Q. Maguire, M. Smith and H. W. Peter Beadle "SmartBadges: a wearable computer and communication system", *6th International Workshop on Hardware/Software Codesign*, 1998.

[19] H. Taylor and S. Karlin, *An Introduction to Stochastic Modeling*, Academic Press, 1998.

[20] M. Berkelaar, *www.cs.sunysb.edu /algorith /implement /lpsolve /implement.shtml*

[21] Y. Lu, T. Šimunić and G. De Micheli, "Software Controlled Power Management", *CODES*, pp. 157–161, 1999.

[22] A. Karlin, M. Manesse, L. McGeoch and S. Owicki, "Competitive Randomized Algorithms for Nonuniform Problems", *Algorithmica*, pp. 542–571, 1994.

[23] The Editors of IEEE 802.11, *IEEE P802.11D5.0 Draft Standard for Wireless LAN*, July, 1996.

[24] Lucent, *IEEE 802.11 WaveLAN PC Card - User's Guide*, p.A-1.

[25] V. Jacobson, C. Leres, S. McCanne, *The Tcpdump Mannual Page*, Lawrence Berkeley Laboratory.

[26] B. Noble, M. Satyanarayanan, G. Nguyen, R. Katz, "Trace-based Mobile Network Emulation *Proceedings of ACM SIGCOMM*, 1997.