An Overview of the Competitive and Adversarial Approaches to Designing Dynamic Power Management Strategies

Sandy Irani, *Member, IEEE,* Gaurav Singh, *Student Member, IEEE,*
Sandeep K. Shukla, *Senior Member, IEEE,* Rajesh K. Gupta, *Fellow, IEEE.*

***Abstract*— Dynamic Power Management or DPM refers to the problem of judicious application of various low power techniques based on runtime conditions in an embedded system to minimize the total energy consumption. To be effective, often such decisions take into account the operating conditions and the system-level design goals. DPM has been a subject of intense research in the past decade driven by the need for low power consumption in modern embedded devices. We present a comprehensive overview of two closely related approaches to designing DPM strategies, namely competitive analysis approach, and model checking approach based on adversarial modeling. Although many other approaches exist for solving the system-level DPM problem, these two approaches are closely related and are based on a common theme. This commonality is in the fact that the underlying model is that of a competition between the system and an adversary. The environment that puts service demands on devices is viewed as an adversary, or to be in competition with the system to make it burn more energy, and the DPM strategy is employed by the system to counter that.**

I. Introduction

A. Dynamic Power Management

MINIMIZATION of power consumption is rapidly becoming the chief optimization criterion in system design for a range of systems from general purpose computing to embedded, mobile computing devices. To be useful, such optimizations must often be done against other competing criteria, such as functionality delivery within performance and timing constraints. Often a balance is sought between the amount of computing (as in local processing) vs the amount of communication that would be needed as computation is reduced [1], [2], [3].

This paper is tutorial in nature, and complements the previous tutorials [4], [5] on DPM, the last of which dates back to 2000, when none of the approaches surveyed here were published. Our focus here is on system-level dynamic power management that can be implemented in the operating system. These power saving measures allow for observation and incorporation of application behavior [6], [7], [4] in a *Power Manager* (PM). The PM can change the power consumption of a device through selection of shutdown/sleep/wakeup states for the device, or by changing its speed through voltage or frequency scaling. For historical reasons, system level DPM generally refers to the techniques that save energy in devices by turning these on and off under operating system control. From an OS point of view, shutdown/wakeup remains a key decision in effective power management because the effectiveness of speed-scaling is sometimes called into question due to the process technology effects such as dominance of leakage [8]. DPM has been studied by several research groups [9], [10], [7], [6], [11], [12], [13], [14], as well as concerted industry efforts such as Microsoft's OnNow [15] and ACPI [16].

We acknowledge support from NSF CAREER award CCR-0237947, NSF NGS award 0204028, NSF grant CCR-0105498, CCF-0514082, CCR-0098335, and SRC Integrated Systems Grant.

B. Previous Survey

A survey of the DPM techniques developed prior to 2000 can be found in [4], [5]. In these extensive reviews, the solution approaches to DPM have been classified into *predictive schemes* and *stochastic optimum control* schemes [17]. Predictive schemes attempt to predict a device's usage behavior in the future, usually based on the past history of usage patterns, and decide to change power states of the device accordingly. The chief parameter of interest here is the idleness threshold, i.e., the time period for a device to transition from an active state to a sleep state. Work on prediction based dynamic power management can be categorized into two groups: *adaptive* and *non-adaptive*. Non-adaptive strategies set the idleness thresholds for the algorithm once and for all and do not alter them based on observed input patterns. Adaptive strategies, on the other hand, use the history of idle periods to guide the decisions of the algorithm for future idle periods. There have been a number of adaptive strategies proposed in the literature [9], [18], [13], [10]. In [9], a system-level power management technique for power savings in event-driven applications is presented. It discusses a predictive system shutdown method which uses an exponential-average approach to predict the upcoming idle period and exploits sleep mode operations for power saving. [18] presents randomized online algorithms for snoopy-caching and spin-block problems, that can be applied to the DPM problem considered here. [19] introduces a finite-state, abstract system model for power-managed systems based on Markov decision processes. Under this model, the problem of finding policies that optimally tradeoff performance for power can be cast as a stochastic optimization problem and solved exactly and efficiently. [13] presents two methods for characterizing non-stationary service requests by means of a prediction scheme based on sliding windows. It also describes how control policies for non-stationary models can be derived.

In [10], authors describe architectural techniques for energy efficient implementation of programmable computation, particularly focussing on the computation needed in portable devices where event-driven user interfaces, communication protocols, and signal processing play a dominant role.

Stochastic approaches make probabilistic assumptions (based on observations) about usage patterns and exploit the nature of the probability distribution to formulate an optimization problem, the solution to which drives the DPM strategy. Examples are in [13], [19], [20], [21], [22], [23], [24], [25], [26]. Until very recently predictive schemes have been mostly based on devices with two power saving states (e.g., standby and sleep). In case of multiple states, the predictive schemes can be extended to use a sequence of idleness thresholds to determine when to transition to the next power state. By comparison, multi-state systems are naturally modeled in most stochastic optimum control approaches [4], [21], [22], [19], [13], [20], [23]. In particular, [21] formulates policy optimization for dynamic power management as a constrained optimization problem on continuous-time Semi-Markov decision processes (SMDP). [22] discusses a modular approach for design and simulation of hardware and software energy consumption at the system level. [20] introduces a continuous-time, controllable Markov process model of a power-managed system. In that paper, the problem of dynamic power management in such a system is formulated as a policy optimization problem and solved using an efficient "policy iteration" algorithm. [23] also formulates the problem of system-level power management as a controlled optimization problem based on the theories of continuous-time Markov decision processes and stochastic networks. Examples of session clustering and prediction strategies are in [27], online strategies are in [12], and adaptive learning based strategies are in [28]. [10] describes architectural techniques for energy efficient implementation of programmable computation. Lu *et. al.* in [29] provide a quantitative comparison of various power management strategies. Most adaptive dynamic power management strategies [9], [10], [12], [18], [13], [27] use a sequence of past idle period lengths to predict the length of the next idle period. These strategies typically describe their prediction for the next idle period with a single value. Given this prediction, they transition to the power state that is optimal for this specific idle period length. In the case the prediction is wrong, they transition to the lowest power state if the idle period extends beyond a fixed threshold value. For the sake of comparison with other approaches, we shall call these predictive DPM schemes *Single-Value Prediction schemes (SVP)*. Among SVPs, of particular interest is [28] that addresses multiple idle state systems using a prediction scheme, based on adaptive learning trees, that improves the hit ratio of the predicted interval significantly.

C. Competitive and Adversarial Modeling

Online algorithms [30] have been designed and analyzed in the theoretical computer science arena primarily using competitive analysis. The idea behind this technique is that an online algorithm is presented input in a continuous stream as it is executing, and hence cannot analyze the input in its entirety before processing it. In this scenario, the usual worst-case analysis can force any algorithm to perform arbitrarily badly and therefore does not provide a meaningful way to distinguish between different algorithms. Therefore, we compare the performance of an online algorithm that processes the input in a continuous stream to an *offline algorithm* that gets the same input in advance and can process the entire input before producing any output. The *competitive ratio* is the worst case over all input sequences of the performance of the online algorithm divided by the performance of the offline algorithm. Another way to view this type of analysis is to think of the online algorithm as playing against an adversary. While the adversary is devising the worst possible input sequence for the algorithm, he must keep in mind that he must also process the input sequence, but in an offline manner. The adversary is devising an input sequence that will maximize the ratio between the cost of the online algorithm and the cost of the optimal offline algorithm.

Now consider another approach. Model checking [31] is a technique of verifying systems which can be looked upon as if an adversary is creating a path into a state transition model of the system to falsify a property that the model checking engine is trying to prove correct. In other words, if a system is trying to save power, and the designer of the system wants to guarantee that the system will never burn more than a certain amount of energy within a certain number of steps, an adversary can create an input sequence which may violate that, and in that case the designer's bound is disproved. If designers bound is proven by the model checker, that establishes a competitive ratio. This view was experimentally validated in [11], and the SMV model checker was used to establish the competitive ratio of a number of DPM strategies which were already calculated using analytical arguments in [32].

The parallel drawn between competitive analysis and model checking in [11] by itself is not as useful other than the analogy it draws between the two. However, it indicated an adversarial framework in which DPM strategies can be designed and analyzed. What became potentially more interesting was the work in [24], [33] where probabilistic model checking tool PRISM was used to carry out this adversarial view further into the domain of stochastic DPM design and analysis using probabilistic models. This tutorial therefore brings together for the first time the main results in competitive analysis based techniques and model checking based techniques under the unified theme of adversarial reasoning/modeling based DPM.

D. The System Model

Consider a single peripheral device whose power state is managed by the operating system. The device can be in one of the n power states denoted by $\{s_1, ..., s_n\}$. The power consumption for state i is denoted by α_i. Without loss of generality, we assume that the states are ordered so that $\alpha_i > \alpha_j$ for $i < j$. Thus, state s_1 is the *ready* state which is the highest power consumption state. (As an example, the ACPI [16] standard specifies the different devices classes and their recommended power states in an Intel PC platform.)

In addition to the states, we are also given (typically either measured or from the device manufacturer's specification) the transition power p_{ij}, and transition times t_{ij}, to move from state s_i to s_j. Often, the power needed and time spent to go from a higher (power consumption) state to a lower state is negligible. In these cases, stronger results can be obtained. However, this condition does not necessarily hold for all systems. Predictive schemes often consider the transition power and transition time numbers as deterministic as we do here [7], [13], [28], [9], [14], [34], [12], whereas in stochastic approaches these numbers are used as parameters to the probability distributions assumed. In some cases, these numbers are experimentally determined [27], [29], [35]. Another characteristic of predictive schemes is that they generally transition to the ready state when powering up and not to an intermediate (higher powered) state. Schemes using predictive wake-up [29], [28] are a notable exception and beyond the scope of this paper. However, stochastic strategies often have probabilistic predictive wakeup built into DPM algorithm. As a result, when discussing deterministic DPM we only need the time and total energy consumed (β_i) in transitioning up from each state i to the ready state.

The input to the PM is a sequence of requests for service that arrive over time. If the device is busy when a new request arrives, it enters a queue and is served on a first-come-first-serve basis. In this case, there is no idle period and the device remains active through the time that the request is finished. Thus, the number of idle periods is less than the number of requests serviced. Whenever a request terminates and there are no outstanding requests waiting in the system, an idle period begins. In these situations, the PM determines the power consumption states the device should transition and at what times.

If the device is not busy when a new request arrives, it will immediately transition to the ready state to serve the new request if it is not already there. In the case where the device is not already in the ready state, the request can not be serviced immediately, but will have to incur some latency in waiting for the transition to complete. This delay will cause future idle periods to be shorter. In fact, if a request is delayed, some idle periods may disappear. Thus, the behavior of the algorithm affects future inputs (idle period lengths) given to the algorithm. Similarly, note that without performance constraints, delaying the servicing of a request will tend to lower the power usage. Consider the extreme case where the power manager remains in the deepest sleep state while it waits for all the requests to arrive and then processes all of them consecutively. This extreme case is not allowed to happen in our model since we require that the strategy transition to the ready state as soon as any request appears. However, it illustrates the natural trade-off which occurs between power consumption and latency. See [12] for a more extensive discussion of this trade-off.

E. Model Checking Approach for DPM

A common method for prediction of the next idle period is to use some form of regression equation over the previous idle periods, and/or use of interpolation or learning-based techniques. In contrast to these *ad hoc* techniques, the stochastic DPM literature tends to be more formal in the sense that assumptions are made as to the characteristics of the probability distribution of idle periods, device response times etc. These are then used to formulate the optimization problems. Much of the stochastic DPM strategy literature uses Markov models, based on assumptions about how and when requests can arrive (whether at certain time points or at any time). For example, discrete-time and continuous-time Markov chains have been used.

Our focus on formal methods is from the point of view of developing DPM strategies that attempt to ensure bounds on the efficiency of achievable power reduction and power/latency tradeoffs without the need for time consuming simulation techniques. We seek methods that can determine these bounds either in the deterministic or probabilistic sense.

The remainder of this tutorial paper is organized as follows. Section 2 focuses mostly on predictive schemes. This presentation complements the survey in [4] by focusing on the more recent work in the area. We introduce the basic concepts of online algorithms and competitive analysis in the context of DPM. Section 3 considers the stochastic approaches to DPM. Section 4 describes the most recent approaches based on probabilistic model checking. Finally, Section 5 summarizes the tutorial.

II. DPM AS AN ONLINE PROBLEM

Dynamic power management is an inherently *online* problem, in that the power manager must make decisions about the expenditure of resources before all the input to the system is available [36]. The input here is the length of an upcoming idle period and the decision to be made is whether to transition to a lower power dissipation state while the system is idle. A short idleness threshold will lead to higher power-up costs, whereas a large threshold would lead to suboptimal power usage. Analytical solutions to such online problems are often best characterized in terms of a *competitive ratio* [30] that compares the cost of an online algorithm to the optimal offline solution which knows the input in advance (and thus chooses the best assignment of power states). Earliest work on competitive analysis of dynamic power management strategies presents bounds on the quality of the DPM solutions [12], [18], [37].

A. Competitve Analysis of Deterministic DPM

An algorithm is $c - competitive$ if, for any input, the cost of the online algorithm is bounded by c times the cost of the optimal offline algorithm for that input. The *competitive ratio* of an algorithm is the infimum over all c such that the algorithm is c-competitive. It has been known for some time that 2 is the optimal competitive ratio that can be achieved for any two-state system by a deterministic algorithm. A succint proof of this fact can be found in [30]. We will sketch the idea behind this result in order to illustrate how competitive analysis works. Since the system has only two states, we call these the active and the sleep state. Let β be the energy cost to

transition from the sleep to the active state. Let α be the power dissipation rate in the active state. Without loss of generality, we assume that the power dissipation in the sleep state is zero.

The optimal offline algorithm is assumed to know the length T of the idle period in advance. Thus, it chooses the best state for this idle period length and stays in that state for the duration of the idle period. Staying in the active state costs αT. Transitioning immediately to the sleep state costs β because the algorithm must transition back to the active state at the end of the idle period. This means that the cost for the optimal offline algorithm for an idle period of length T is $\min\{\alpha T, \beta\}$.

Since the online algorithm does not know the length of the idle period in advance, it selects a threshold, τ (conceptually same as timeout) and stays in the active state for time τ after which it transitions to the sleep state if the system is still idle. If the idle period length T is less than τ, its cost is αT. If the idle period is longer than τ, its cost is $\beta + \alpha\tau$. The online algorithm seeks to minimize the ratio of its cost to the cost of the optimal offline algorithm for all T. It can be shown that if $\tau = \beta/\alpha$, this ratio is never more than two. The worst case for the online algorithm is if the idle period ends immediately after it transitions to the sleep state. This puts a tight bound of 2 for the competitive ratio of any deterministic algorithm.

For multi-state systems, the situation is a bit more complex in that the optimal competitive ratio will, in general, depend on the parameters of the system (e.g. the number of states, power dissipation rates, start-up costs, etc.). In [34], a generalization of the 2-competitive algorithm for two-state systems is given for multi-state systems that also achieves a competitive ratio of 2. In general, this bound is not tight because it may be possible to attain a better competitive ratio for specific systems. The bound holds under the assumption that the cost to transition from a higher power state to a lower power state is negligible. We note that in the special case where for any $i < j < k$, the cost to go from i to j and then from j to k is the same as the cost of going from i directly down to k (i.e. there is no penalty for stopping at a state on the way to another state), the costs to transition downward can be folded into the cost to transition back up to the active state. The algorithm is non-adaptive since it does not use any information about the arrival sequence of jobs to the device.

More recently, [38] have developed competitive algorithms for multi-state systems that work for arbitrary transition costs on the states. The authors give an online algorithm that obtains a competitive ratio of 8. This can be improved to 5.828 under the very reasonable assumption that the transition power to move from state s_i to s_j, p_{ij} is greater than the transition power to move from state s_i to s_l, p_{il} for any $i < l < j$. They also develop a *meta-algorithm* (i.e., a DPM algorithm generator) that takes as input the parameters of a system and produces a DPM strategy (sequence of states and threshold times). The strategy they produce is guaranteed to achieve a competitive ratio that is within an arbitrary ϵ of the best possible competitive ratio for that system. The running time of the meta-algorithm is polynomial in the number of states and $1/\epsilon$. The algorithm uses a decision algorithm which, given a value for ρ and a description of a device (states, power consumption rates and transition costs), determines if there is a ρ-competitive algorithm for that system. Then a binary search for the optimal ρ can be performed since it is known to lie in the interval between one and eight.

Another direction that has recently been undertaken is to combine DPM strategies with Dynamic Speed Scaling (DSS) for devices that have both the ability to run at varying speeds and the ability to shut down when idle [39]. Note that we choose the more general term *Dynamic Speed Scaling* in order to ecompass *Dynamic Frequency Scaling* and *Dynamic Voltage Scaling*. The basic idea of all of these is that the speed of a device can be reduced to save power. In the combined DSS and DPM problem, the power consumption rate of a device is assumed to be a continuous function of the speed at which it runs. In addition, the power consumption rate when the speed is zero (i.e. the device is idle) is greater than zero. The system has the option to transition to a sleep state when idle in order to reduce the power consumption rate. There is a fixed cost then to transition back to the active state. The input to this problem consists of a set of jobs with release times, number of execution cycles and deadlines. Each job must be completed in the interval between its release times and deadline.

Combining the two problems of DSS and DPM, introduces challenges which do not appear in either of the original problems. In DPM, the lengths of the idle intervals are given as part of the input whereas in the combined problem they are created by the scheduler which decides when and how fast to perform the tasks. In DSS, it is always in the best interest of the scheduler to run jobs as slowly as possible within the constraints of the arrival times and deadlines due to the convexity of the power function. By contrast in the combined problem, it may be beneficial to speed up a task in order to create an idle period in which the system can sleep. An offline algorithm is described that is within a factor of two of the optimal algorithm as well as an online algorithm with a constant competitive ratio. Some of the same issues are dealt with in [40] in which process schedulers have some latitude in scheduling the execution of tasks so as to maximize the benefit of dynamic power scheduling.

B. Probabilistic Analysis

As discussed above, competitive analysis often gives overly pessimistic bounds for the behavior of algorithms. This is because competitive analysis is a worst-case analysis. In many applications there is structure in the input sequence that can be utilized to fine tune online strategies and improve their performance. Indeed, important earlier works in this area [19], [20] have relied on modeling the distribution governing inter-arrival times as an exponential distribution. In practice, such stochastic modeling seems to hold well for specific kinds of applications. However, these assumptions have led to complications in other settings due to phenomena such as the non-stationary nature of the arrival process, clustering, and the lack of independence between subsequent events. These problems have been addressed to some extent in [27], [28].

In [34], we introduced an approach that models the upcoming input sequence by a probability distribution that is *learnt* based on historical data. One of the strengths of this

method is that it makes no assumptions about the form of this distribution. Once the distribution is learnt, we can automatically generate a probability-based DPM strategy that minimizes the *expected* power dissipation given that the input is generated according to that distribution based on the notion of a *probabilistic competitive ratio* [14].

The strategies discussed below use a probability distribution governing the length of the idle periods to determine the optimal power-down strategy. The strategy once produced is completely deterministic. It is well known that if the input to an algorithm is generated by a known probability distribution, then using a probabilistic strategy gives no benefit [34]. This is in contrast to the work discussed in the previous section which examines strategies using a worst-case analysis. In analyzing strategies under a worst-case scenario, the use of randomization can improve upon deterministic strategies.

1) Optimizing Power Based on a Probability Distribution: Let us suppose that the length of the idle interval is generated by a fixed, known distribution whose density function is π. Let us consider systems with two states. As before, let β be the start-up energy of the sleep state and α the power dissipation of the active state. Suppose that the online algorithm uses τ as the timeout or threshold at which time it will transition from the active state to the sleep state if the system is still idle. In this case, the *expected* energy cost for the algorithm for a single idle period is given as:

$$\int_0^\tau \pi(t)(\alpha t)dt + \int_\tau^\infty \pi(t)[\alpha\tau + \beta]dt.$$

The best online algorithm will select a value for τ which minimizes this expression. On the other hand, the offline optimal algorithm which knows the actual length of an upcoming idle period will have an expected cost of:

$$\int_0^{\beta/\alpha} \pi(t)(\alpha t)dt + \int_{\beta/\alpha}^\infty \pi(t)\beta dt.$$

It has been shown that for the 2-state case, the online algorithm can pick its threshold τ so that the ratio of its expected cost to the expected cost of the optimal algorithm is at most $e/(e-1) = 1.58$ [18], [34]. Furthermore, the result is tight in that there are distributions for which $e/(e-1)$ is the best ratio that can be achieved. A generalization of the two-state algorithm to the multi-state case is given in [14]. The generalized algorithm is called the Probabilistic Lower Envelope Algorithm (or PLEA) and the authors have shown that for any probability distribution and any system in which power-down costs are negligible, PLEA is no worse than $e/(e-1)$ times the optimal offline algorithm. The results in [38] show that PLEA is optimal in that it minimizes the expected cost over all power-down strategies for any probability distribution for any system in which power-down costs are negligible. (Note that the $e/(e-1)$ bound is only tight for some but not all distributions.) The results in [38] generalize PLEA even further to give a power-down strategy which is provably optimal for any multi-state system, with no restriction on transition costs, when the length of the idle period is generated by a known probability distribution. The analysis of the two-state case plays an important role in all of these extensions to multi-state systems. In particular, it is proven in [38] that if the optimal strategy transitions from some state i directly to some state j, the optimal time for that transition will be the same as the optimal transition time for a two-state system in which state i and state j are the only two states in the system. Thus, the problem of simultaneously optimizing many transition times is reduced to finding pair-wise optimal transitions.

Thus, a knowledge of the input pattern and its use can help bridge the gap between the performance of an online strategy and that of the optimal offline strategy. Results show the the worst case competitive ratio can be improved by 21%, with respect to the deterministic case [34].

2) Learning the Probability Distribution: The algorithm PLEA above assumes perfect knowledge of the probability distribution governing the length of the idle period. Rather than assuming such a distribution, it can be learnt based on recent history. For instance, a learning scheme in conjunction with PLEA is called the Online Probability-Based Algorithm (OPBA). The probability estimator works as follows: a window size w is chosen in advance and is used throughout the execution of the algorithm. The algorithm keeps track of the last w idle period lengths and summarizes this information in a histogram. Periodically, the histogram is used to generate a new power management strategy.

The set of all possible idle period lengths $(0, \infty)$ is partitioned into n intervals, where n is the number of bins in the histogram. Let r_i be the left endpoint of the i^{th} interval. The i^{th} bin has a counter c_i which indicates the number of idle periods among that last w idle periods whose length fell in the range $[r_i, r_{i+1})$. Instead of using the continuous probability distribution π with PLEA as described in the previous section, we use a discrete distribution, where the probability the idle period has length r_i is c_i/w. A similar approach was taken for a two state system in the context of determining virtual circuit holding time policies in IP-over-ATM Networks [37].

Efficient implementation of such an algorithm is important to ensure overall gains in power reduction. In [34], we present an implementation for finding the $m-1$ thresholds in time $O(mn)$, where m is the number of states and n is the number of bins in the histogram. Two important factors which determine the cost (in time expenditure) of implementing our method is the frequency with which the thresholds are updated and the number of bins in the histogram. Selecting the right granularity for the histogram is an important consideration since there is a tradeoff between efficiency and accuracy. The algorithm employs non-uniform bin sizes so as to have a high degree of accuracy in critical regions. The reader is referred to [34] for a description of how system parameters are used to select bin sizes.

While the approach in [34] is provably close to optimal if the probability distribution is known, the techniques used to learn and represent the probability distirbution based on recent history are heuristic. A more formal approach to this problem is taken in [41] which looks at the complexity of determining the optimal power-down treshold in a two-state system in which a long sequence of idle periods is generated by a fixed but unknown probablity distribution. They give a

method which uses $O(1)$ time and space and converges to the optimal online algorithm for that distribution. It would be intersting to extend these ideas to a multi-state case.

III. Stochastic Approaches to DPM

We now discuss the stochastic version of the DPM problem. The problem basically requires one to devise a strategy (policy) which is *probabilistic*, in the sense that the actions to be taken by the strategy have probabilities attached to them. Unlike deterministic strategies, where a particular state of the system will lead the strategy to take a deterministic action, here, the strategy can choose between multiple actions with pre-designated probabilities.

A. Stochastic Learning Feedback Hybrid Automata for DPM

Hybrid automata are composed of discrete states, the states of automaton, and continuous dynamics, the differential equations that govern the continuous variables in each state. In the DPM approach presented in [42], [43], this compositional similarity of hybrid automata with embedded systems (having multiple power-down modes) is exploited to model such systems with a timed hybrid automaton. The discrete states of the hybrid automaton are used to model the power modes of the system, while the continuous dynamics account for the power consumed in each mode. The DPM is formulated as a hybrid automaton control problem where the control strategy is learnt dynamically using Stochastic Learning Hybrid Automata (SLHA) with feedback learning algorithms.

First, a mathematical model of the system is constructed, which includes various states, each representing a power mode of the system. The model uses dynamically updated internal variables for evaluating the total energy spent by the system and the temporary clock. These are governed by the continuous dynamics specific to each state. The total latency incurred by requests in the system and the cumulative request length are updated regularly. Control is added to the model to guide the automaton through power modes while the system is idle. For this, an externally handled control variable is used which manages the sequence of states that the system follows during an idle period.

Next, a stochastic learning feedback hybrid automata (SLHA) model for DPM is developed [42], [43], where the value of the control variable is managed using probabilities of switching between states. For this, a stochastic control is incorporated to the hybrid automaton, and learning feedback is added to the previous mathematical model. In this model, the system attempts to learn the length of the future idle period probabilistically and, accordingly, decides on the behavior to observe during idle time. For this, the external variable of the mathematical model is replaced by the action probabilities, which are frequently recomputed using reinforcement techniques. Every allowed state transition is labeled by an action probability that represents the probability of switching from one state to the other. Several feedback stochastic learning algorithms (General Linear Reward- Penalty Scheme, Symmetric Linear Reward-Penalty Scheme, Linear Reward-Inaction Scheme etc.) are incorporated in this model for educating the system to choose the optimal action. Given a set of permitted actions in every state of a system, the system chooses the optimal action to execute at every stage using these learning techniques.

Two power-up strategies are used in this work, namely, wake-up "On-Demand" and "Preemptive" activation. In both these approaches, the system is ordered to switch to a lower power mode at the start of the idle period. Wake-up "On Demand" orders the system to remain in the lower-power mode until receipt of a request. The system must immediately switch to the active state upon the receipt of a request. With "Preemptive" activation, the system remains in the lower-power state for a lesser period of time, powering-up to the active state in order to be active at the arrival of the request.

To study the behavior of the SLHA systems, a model of a four-state mobile hard-drive from IBM was employed for simulating DPM [42], [43]. First, the simulations were performed to determine the optimal configurations of the SLHA model to reach the correct convergence in stationary environments. Then, these configurations were used to simulate the SLHA model with the real input distributions, obtained using input files that were adapted from trace data obtained from the auspex file server archive. Simulations were performed in two categories: optimization of energy and latency, and optimization of only energy.

From the competitive ratios, it was observed that "On Demand" wake-up tends to better minimize energy and latency expenditure than the "Preemptive" wake-up method. Moreover, configurations corresponding to the first nonlinear reinforcement scheme with high reward and penalty parameters perform the best minimization of energy for the presented traces. Furthermore, configurations corresponding to the second non-linear updating scheme with a high reward parameter and a high degree of nonlinearity perform the best minimization of latency for the presented traces.

In general, the SLHA mathematical model is claimed to prove its superiority compared to the former DPM strategies presented in literature with "Preemptive" wake-up for the examined input patterns. For wake-up "On Demand", results were enhanced either for the conservation of energy or the prevention of latency, but optimality was not reached for both figures simultaneously. The proposed SLHA model is also claimed to offer a high versatility for the DPM problem.

B. Other Stochastic Approaches

In recent years, several other approaches for designing stochastic DPM strategies have been proposed [44], [19], [4], [13], [20], [23], [45], [21], [22]. These methodologies are based on a stochastic model of the DPM problem, which incorporates the probabilistic characteristics of request arrivals to the device, the device response time distribution, the power consumption by the device in various states and the distribution of energy consumption in changing states. From this stochastic model, an exact optimization problem is formulated, the solution to which is the required optimal stochastic DPM policy. The strategy devised must ensure that power savings are not achieved at an undue cost in performance. For example,

a new request should be always served in a reasonable time. The constructed policy optimizes the *average* energy usage while minimizing *average* delay. The policies are usually validated by simulation to check for the soundness of the modeling assumptions, and the effectiveness of the strategies in practice [20], [44].

The stochastic models which have been used in the literature are discrete-time Markov chains [44], [19], continuous-time Markov Chains [20], [23], [45] or their variants [21], [22]. The approaches vary in the model of time. In the continuous-time case, *mode* switching commands can be issued at any time, and events can happen at any time. In the discrete-time case, all events and actions occur at certain discrete time points. The continuous-time assumption makes the formulation of the problem easier. In practice, such stochastic modeling seems to work well for specific kinds of applications. Generally, the stochastic matrices for these models are created manually. In [45], stochastic Petri nets are used, which allows automatic generation of the stochastic matrices and formulation of the optimization problems. [44] describes power-managed systems using a finite-state, stochastic model. In [23], authors formulate the problem of system-level power management as a controlled optimization problem based on the theories of continuous-time Markov decision processes and stochastic networks. [21] formulates policy optimization as a constrained optimization problem on continuous-time Semi-Markov decision processes (SMDP). [22] presents a modular approach for design and simulation of hardware and software energy consumption at the system level.

IV. DPM Analysis using PRISM

A. Short Introduction to Probabilistic Model Checking

Probabilistic Model Checking (PMC) offers a promising way to verify stochastic approaches to DPM as shown in [24], [33]. The idea is to construct a probabilistic model of the system under study. As in the deterministic case, this is usually a labeled transition system which defines the set of all possible states and the transitions between these states. In PMC, the model is augmented with information about the likelihood that each transition will take place. Examples of such models are discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs) and Markov decision processes (MDPs). The properties to be verified, are specified typically in probabilistic extensions of temporal logic. These allow specification of properties such as: "shutdown occurs with probability at most 0.01"; or "the video frame will be delivered within 5ms with probability at least 0.97." The properties can be verified with a probabilistic model checker either as graph-based analysis and solution of linear equation systems or linear optimization problems [46].

Like the conventional, non-probabilistic case, probabilistic model checking usually constitutes verifying whether or not some temporal logic formula is satisfied by a model. The two most common temporal logics for this purpose are PCTL [47], [48] and CSL [49], [50], both extensions of the logic CTL. PCTL is used to specify properties for DTMCs and MDPs and CSL is used for CTMCs. One common feature of the two logics is the probabilistic $\mathcal{P}$ operator, which allows one to reason about the probability that executions of the system satisfy some property. For example, the formula $\mathcal{P}_{\geq 1}[\Diamond\ terminate]$ states that with probability 1, the system will eventually terminate. On the other hand, the formula $\mathcal{P}_{\geq 0.95}[\neg repair\ U^{\leq 200}\ terminate]$ asserts that with probability 0.95 or greater, the system will terminate within 200 time steps and without requiring any repairs. These properties can be seen as analogues of the non-probabilistic case, where a formula would typically state that *all* executions satisfy a particular property, or that *there exists* an execution which satisfies it. CSL also provides the $\mathcal{S}$ operator to reason about steady-state (long-run) behavior. The formula $\mathcal{S}_{<0.01}[queue\_size\ =\ max]$, for example, states that in the long-run, the probability that a queue is full is strictly less than 0.01. Further properties can be analyzed by introducing the notion of *costs* (or, conversely, *rewards*). If each state of the probabilistic model is assigned a real-valued cost, one can compute properties such as the expected cost to reach a certain states, the expected accumulated cost over some time period, or the expected cost at a particular time instant. As in the previous paragraph, such properties can also be expressed concisely and unambiguously in temporal logic [51], [52].

B. PRISM

PRISM [46], [24], [33] is a probabilistic model checker developed at the University of Birmingham in England. In [24] and [33], PRISM was used for deriving stochastic DPM policies for disk-drives, and was shown to be a uniform framework in which DPM policies can be derived and evaluated. The basic approach is to build a probabilistic model of the DPM system from which, for a given constraint, an optimization problem is constructed. The solution to this problem is the optimum randomized power management policy satisfying this constraint.

Once an optimal power management policy has been constructed, it must be validated to ensure it performs as intended. Possible approaches are to use trace-based simulation or to actually implement the schemes in device drivers. The advantage of PMC is that it allows one to validate and analyze the policies statically leading to a wide range of useful information about the policy to be generated.

Modeling DPM in PRISM: While PMC has been applied to both DTMCs [44], [19] as well as CTMCs [20], [23], [45], we focus on the former here. The approach is described through the example of [44], [19], an IBM TravelStar VP disk-drive [53]. The device has 5 power states, labelled *sleep*, *stby*, *idle*, *idlelp* and *active*. It is only in the state *active* that the drive can perform data read and write operations. In state *idle*, the disk is spinning while some of the electronic components of the disk drive have been switched off. The state *idlelp* (idle low power) is similar except that it has a lower power dissipation. The states *stby* and *sleep* correspond to the disk being spun down. Based on the fastest possible transition performed by system, one can choose a time resolution of 1ms for the model, i.e., each discrete-time step of the DTMC will correspond to 1ms.

The system model shown in Figure 1 [19] consists of: a Service Provider (SP), which represents the device under power management control; a Service Requester (SR), which issues requests to the device; a Service Request Queue (SRQ), which stores requests that are not serviced immediately; and the Power Manager (PM), which issues commands to the SP, based on observations of the system and a stochastic DPM policy. Each component is represented by an individual PRISM module, which we now consider in turn. Below, we provide the examples of some of those components.

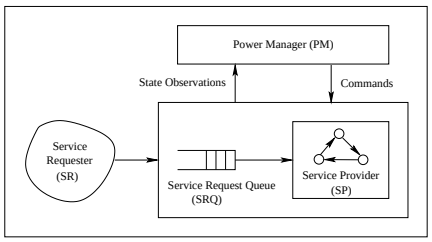

Fig. 1. The System Model

Modeling the Power Manager(PM), Service Requester (SR) and Queue(SRQ) : The PM decides to which state the SP should move at each time step. To model this, each step is split into two parts: in the first, the PM (instantaneously) decides what the SP should do next (based on the current state); and in the second, the system makes a transition (with the SP's move based on the choice made by the PM). These steps are synchronized with other components using two synchronization actions $tick1$ and $tick2$. Figure 3 shows an example PM in PRISM.

Both the SRQ and the SR will synchronize on $\mathrm{tick2}$. The SR has two states: *idle* where no requests are generated and *1req* where one request is generated per time step (1ms). The module of the SR is given by:

```
module SR

    sr : [0..1] init 0;
    // 0 - idle and 1 - 1req

    [tick2] sr=0 → 0.898 : (sr'=0) + 0.102 : (sr'=1);
    [tick2] sr=1 → 0.454 : (sr'=0) + 0.546 : (sr'=1);

endmodule
```

The transitions between the states of the SR module are based on time-stamped traces of disk access measured on real machines [19]. The above module specifies that if SR is in the *idle* state, then it will remain in the same state with probability 0.898, and will transition to the *1req* state with probability 0.102. On the other hand, if SR is in the *1req* state, then it will remain in the same state with probability 0.546, and will transition to the *idle* state with probability 0.454. Both these transitions will take place at $\mathrm{tick2}$.

The SRQ models queue of service requests. It responds to the arrival of requests from the SR and the service of requests by the SP. The queue size will only decrease when the SR and SP are in states *idle* and *active*, respectively. Similarly, it will only increase when the SR is in state *1req* and the SP is not *active*. The PRISM code is as follows:

```
const QMAX = 2; //maximum size of the queue

module SRQ

    q : [0..QMAX] init 0; // size of queue

    // SP is active
    [tick2] sr = 0 ∧ sp = 0 → q' = max(q − 1, 0);
    [tick2] sr = 1 ∧ sp = 0 → q' = q;
    // SP is not active
    [tick2] sr = 0 ∧ sp > 0 → q' = q;
    [tick2] sr = 1 ∧ sp > 0 → q' = min(q + 1, QMAX);

endmodule
```

C. Policy Construction and Analysis

Using the PRISM language description detailed in the previous section, the PRISM model checking tool can be used to construct a generic model of the power management system. From the transition matrix of this system, the linear optimization problem whose solution is the optimal policy can be formulated, as described in [44], [19]. This optimization problem is then passed to the MAPLE symbolic solver. Policies can be constructed such that they satisfy any required constraints. This helps to formulate policies for practical purposes which work under the given constraints. Figure 2 shows policies constructed in this way for a range of constraints on the average size of the service request queue. The first column lists the constraints values; the second column summarizes the corresponding constructed policy.

Average size of SRQ	Constructed Optimum Policy
≤ 2	remain sleeping
≤ 1.5	SP active and queue not full: goto idle SR in state 0, SP sleeping and queue full: remain sleeping with probability 0.99999953 go to active with probability 0.00000047 SR in state 1, SP idle: goto active
≤ 0.5	SP active and queue not full: goto idle SR in state 0, SP sleeping and queue full: remain sleeping with probability 0.99999418 go to active with probability 0.00000582 SR in state 1 and SP idle: goto active
≤ 0.05	SP active, SR in state 0 and queue empty: remain active with probability 0.63683933 go to idle with probability 0.36316067 SP idle: go to active SP sleeping: go to active

Fig. 2. Optimum policies under varying constraints on the average queue size

Once a policy has been constructed, its performance can be investigated using probabilistic model checking. For this, the generic power manager PRISM module is modified to represent a specific policy. Figure 3 shows an example of this for the constraint "queue size is less than 0.05". This can be seen to correspond to the policy in the 4th row of the table in Figure 2. There, the constructed policy states that under this constraint, for cases where SP is active, SR is idle (state = 0), and the queue is empty, SP should remain active with probability 0.63683933 and it should go to the idle state with probability 0.36316067. It also states that if SP is either in the idle state or the sleep state, then it should transition to the active state. PRISM is then used to construct and analyze the DTMC for this policy.

The analysis shows that the average power consumption of a policy decreases as the constraint on queue length used to construct it is relaxed (i.e. the queue size is larger). One can also validate the policy by confirming that the expected size of the queue matches the value in the constraint which was used to construct it. Finally, it is observed that a side-effect of this is that the average number of requests lost is also increased.

```
module PM

    // policy when constraint on queue size equals 0.05
    pm : [0..4];
    // 0 − go to busy, 1 − go to idle, 2 − go to idlelp
    // 3 − go to standby and 4 − go to sleep

    [tick1] sr=0 ∧ sp=0 ∧ q=0 →
                                    0.63683933 : pm′=0 //active
                                    + 0.36316067 : pm′=1; //idle
    [tick1] sp=1 → pm′=0; //active
    [tick1] sp=9 → pm′=0; //active
    [tick1] ¬(sp=9 ∨ sp=1 ∨ (sr=0 ∧ sp=0 ∧ q=0)) →
                                                    pm′=pm;

endmodule
```

Fig. 3. Example input to PRISM for a derived Policy under performance constraint = 0.05

In Figure 4, the graphical results for a range of policies are shown from [33]. Using PRISM, one can associate a cost with each state and then compute the expected accumulated cost of the system until the required time. Using this assignments of model states to costs, for a range of values of T, "expected power consumption by time T", "expected queue size at time T", and "expected number of lost customers by time T", are computed and plotted. The first and third properties are determined by computing expected cost cumulated up until time T; the second by computing the instantaneous cost at time T. Again, we see that policies which consume less power have larger queue sizes and are more likely to lose requests. Here, though, one can get a much clearer view of how these properties change over time. We see, for example, that the expected queue size at time T initially increases and then decreases. This follows from the fact that the strategies wait for the queue to become full before switching the SP on.

In Figure 5, the probability that a request is served by time T, given that it arrived into a certain position in the queue is plotted based on [33]. Figure 6 shows the probability that N requests get lost by time T for $N = 500$ and $N = 1000$. Again this information has been computed for a range of policies and for a range of values of T. These properties are computed by adding additional state variables to the PRISM model. For those in Figure 6, for example, a variable is added which is initially zero and is increased each time a customer is lost (up to a maximum on N). Then, the probability of reaching any state where this variable's value is equal to N, is calculated.

The graphs show that the probability of requests being lost within a certain time bound increases more quickly for those strategies that consume less power. These results are to be expected since, to reduce power, the strategies must force the service provider to spend more time in low power states which cannot service requests, e.g. *sleep* and *standby (stby)*.

Probabilistic model checking has also been applied [24] to the stochastic optimum control approach of [20], [23], [45], which is based on CTMCs rather than DTMCs. Since the model is a CTMC, components change state according to exponentially distributed delays and the PM acts when such a state transition occurs. The construction of optimum policies from the PRISM model follows the approach of [20], [23], [45] but is essentially the same overall process. For analysis of policies, one can consider similar properties to the DTMC case. The main differences are that the logic CSL is used

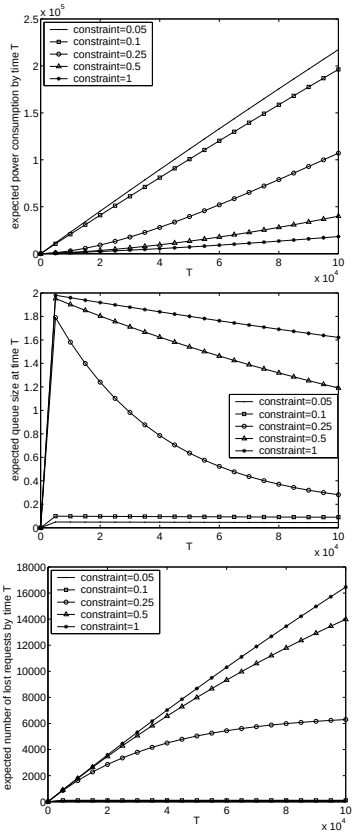

Fig. 4. Power and performance by time T (ms)

as opposed to the logic PCTL, and that the time bound T used in the properties is now a real-value as opposed to a number of discrete steps. In addition, in this case, using the approach of [25] one can also analyze the policies for alternative inter-arrival distributions, to give a more realistic model of the arrival of service requests. For example, Figure 7 shows the performance (average power consumption, average queue size and average number of lost requests) for optimum policies under five different inter-arrival distributions. All the chosen distributions have the same mean and it can be seen that, with the exception of the Pareto distribution, the long-run performance and costs are reasonably close to those of the exponential arrival process. For the Pareto distribution, the average queue size is generally much smaller. This is due to the Pareto distribution's *heavy tail*: in the long run, many requests will not arrive for a very long time, in which case the service provider (SP) will serve all pending requests, leaving the queue empty.

V. Summary

In this tutorial, we focused on techniques for power management that rely on an adversarial modeling approach, namely competitive analysis, and stochastic model checking for the evaluation of the effectiveness of DPM algorithms. For deterministic models of the system, competitive analysis along with learning techniques provide a reaonsable framework for their analysis. Stochastic optimization approaches to DPM can be analyzed using advances in probabilistic model checking techniques.

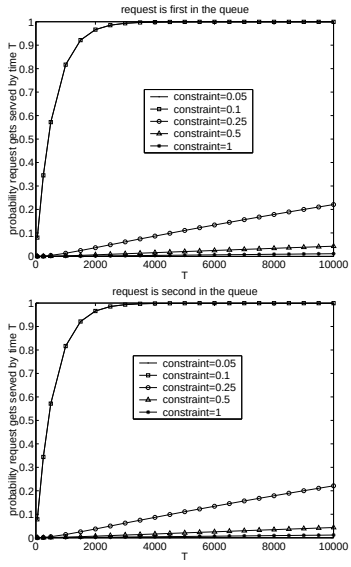

Fig. 5. Probability that a request is served by time T (ms)

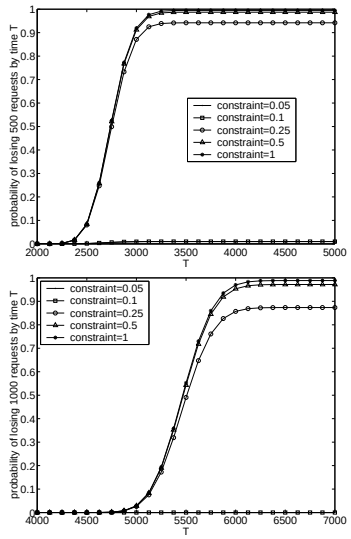

Fig. 6. Probability that N requests gets lost by time T (ms)

We showed (from [24], [33], [26]) how probabilistic model checking allows generation of a wide range of performance measures for the analysis of DPM policies. Statistics such as power consumption, service queue length and the number of requests lost can be computed both in the average case and for particular time instances over a given range. Furthermore, the policies' behavior can be examined under alternative service request inter-arrival distributions such as Erlang and Pareto. In addition to the exhaustive analysis (including corner-case scenarios), probabilistic model checking presents an attractive unified framework for automated construction, validation and analysis of DPM policies. Unfortunately, the details require elaboration which we feel is inappropriate for this article, as it would unnecessarily obscure the main focus. The full details of the handling of non-exponential distribution based arrival processes using PRISM can be found in [26].

In this article, we have not been able to cover some new efforts in power management in the context of power aware ad-hoc network protocols. Notable among these are the use

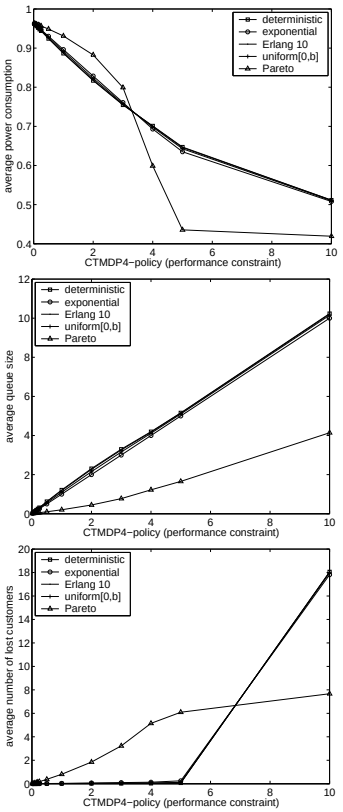

Fig. 7. Analysis of the CTMC case for a variety of inter-arrival distributions

of an economics-based model for networking protocols in [3] and power aware source routing in [54]. We also did not discuss DPM models that consider the battery model which is not considered in any of the approaches discussed above. In [55], Rong and Pedram provide a stochastic model that takes into account the current discharge rates by the batteries in formulating stochastic DPM strategies.

REFERENCES

[1] M. Srivastava, *Chapter 11 of Power Aware Design Methodologies*. Kluwer Academic Press, 2002, ch. Power-Aware Communication Systems.

[2] J. W. Hui, Z. Ren, , and B. Krogh, "Sentry-based Power Management in Wireless Sensor Networks," in *Proceedings of the Information Processing in Sensor Networks (IPSN'03)*, 2003, pp. 458–472.

[3] L. Shang, R. Dick, and N. K. Jha, "An economics-based power-aware protocol for computation distribution in mobile ad-hoc networks," in *Proceedings of the 14th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'02)*, November 2002.

[4] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI'00)*, vol. 8, no. 3, pp. 299–316, 2000.

[5] L. Benini and G. De Micheli, "System level power optimization: Techniques and tools," *ACM Transactions on Design Automation of Electronic Systems (TODAES'00)*, vol. 5, no. 2, pp. 115–192, 2000.

[6] L. Benini, G. De Micheli, and E. Macii, "Designing Low-power Circuits: Practical Recipes," *IEEE Circuits and Systems Magazine*, vol. 1, no. 1, pp. 6–25, Mar. 2001.

[7] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Publications, 1998.

[8] R. Min, M. Bhardwaj, S.-H. Cho, N. Ickes, E. Shih, A. Sinha, A. Wang, and A. Chandrakasan, "Energy-Centric Enabling Technologies for Wireless Sensor Networks," *IEEE Wireless Communications (IEEE WC'02)*, vol. 9, no. 4, pp. 28–39, August 2002.

[9] C.-H. Hwang, C. Allen, and H. Wu, "A Predictive System Shutdown Method For Energy Saving of Event-Driven Computation," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD'96)*, November 1996, pp. 28–32.

[10] M. B. Srivastava, A. P. Chandrakasan, and R. W. Broderson, "Predictive Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI'96)*, vol. 4, no. 1, pp. 42–55, March 1996.

[11] S. Shukla and R. Gupta, "A Model Checking Approach to Evaluating System Level Power Management Policies for Embedded System," in *Proceedings of the IEEE International Workshop on High-Level Design Validation and Test (HLDVT'01)*, November 2001, pp. 53–57.

[12] D. Ramanathan, S. Irani, and R. K. Gupta, "Latency Effects of System Level Power Management Algorithms," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD'00)*, November 2000, pp. 350–356.

[13] E.-Y. Chung, L. Benini, A. Bogliolo, and G. De Micheli, "Dynamic Power Management for Non-Stationary Service Requests," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'99)*, 1999, pp. 77–81.

[14] S. Irani and S. Shukla and R. Gupta, "Competitive analysis of dynamic power management strategies for systems with multiple power saving states," in *Proceedings of the Design, Automation and Test Europe Conference and Exhibition (DATE'02)*, March 2002, pp. 117–123.

[15] Microsoft, "OnNow Power Management Architecture for Applications," Website, August 1997. [Online]. Available: http://www.microsoft.com/hwdev/pcfuture/onnowapp.HTM

[16] Intel, Microsoft and Toshiba, "Advanced Configuration and Power Interface Specification," http://www.acpi.info, December 1996. [Online]. Available: http://www.intel.com/ial/powermgm/specs.html

[17] Y. Lu and G. De Micheli, "Comparing System-Level Power Management Policies," *IEEE Design and Test of Computers (IEEE D&T'01)*, pp. 10–19, 03 2001.

[18] A. Karlin, M. Manasse, L. McGeoch, and S. Owicki, "Randomized competitive algorithms for non-uniform problems," in *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'90)*, 1990, pp. 301–309.

[19] L. Benini, A. Bogliolo, G. Paleologo, and G. De Micheli, "Policy optimization for dynamic power management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD'99)*, vol. 18, no. 6, pp. 813–833, 1999.

[20] Q. Qiu and M. Pedram, "Dynamic Power Management Based on Continuous-Time Markov Decision Processes," in *Proceedings of the Design Automation Conference (DAC'99)*, June 1999, pp. 555–561.

[21] T. Simunic, L. Benini, and G. De Micheli, "Event Driven Power Management of Portable Systems," in *Proceedings of the International Symposium on System Synthesis (ISSS'99)*, November 1999, pp. 18–23.

[22] T. Simunic, "Energy Efficient System Design and Utilization," Ph.D. dissertation, Stanford University, http://si2.epfl.ch/~demichel/graduates/theses/tajana.pdf, 03 2001.

[23] Q. Qiu and Q. Wu and M. Pedram, "Stochastic Modeling of a Power-Managed System: Construction and Optimization," in *Proceedings of the international symposium on Low power electronics and design (ISLPED'99)*, 1999, pp. 194–199.

[24] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, and R. Gupta, "Formal analysis and validation of continuous time Markov chain based system level power management strategies," in *Proceedings of the IEEE International Workshop on High-Level Design Validation and Test (HLDVT'02)*, October 2002, pp. 45–50.

[25] M. Kwiatkowska, G. Norman, and A. Pacheco, "Model checking expected time and expected reward formulae with random time bounds," in *Proceedings of the 2nd Euro-Japanese Workshop on Stochastic Risk Modelling for Finance, Insurance, Production and Reliability*, September 2002, pp. 282–291.

[26] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, and R. Gupta, "Using probabilistic model checking for dynamic power management," *Formal Aspects of Computing (FAC'05)*, vol. 17, no. 2, pp. 160–176, August 2005.

[27] Y. Lu and G. De Micheli, "Adaptive Hard Disk Power Management on Personal Computers," in *Proceedings of the Great Lakes Symposium on VLSI (GLS-VLSI'99)*, March 1999, pp. 50–53.

[28] E.-Y. Chung, L. Benini, and G. De Micheli, "Dynamic Power Management Using Adaptive Learning Trees," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD'99)*, November 1999, pp. 274–279.

[29] Y. Lu, E.-Y. Chung, T. Simunic, L. Benini, and G. De Micheli, "Quantitative Comparison of Power Management Algorithms," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'00)*, March 2000, pp. 20–26.

[30] S. Phillips and J. Westbrook, *Chapter 10 of Algorithms and Theory of Computation Handbook*. CRC Press, Boca Raton, 1999, ch. On-line algorithms: Competitive analysis and beyond.

[31] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, 1999.

[32] D. Ramanathan, S. Irani, and R. Gupta, "An Analysis of System Level Power Management Algorithms and their effects on Latency," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD/ICAS'02)*, vol. 21, no. 3, pp. 291–305, March 2002.

[33] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, and R. Gupta, "Using probabilistic model checking for dynamic power management," in *Proceedings of the 3rd Workshop on Automated Verification of Critical Systems (AVoCS'03)*, April 2003, pp. 202–215.

[34] S. Irani, S. Shukla, and R.Gupta, "Online strategies for dynamic power management in systems with multiple power-saving states," *ACM Transactions on Embedded Computing Systems (TECS'03)*, vol. 2, no. 3, pp. 325–346, August 2003.

[35] Y. Lu, "Power-Aware Operating Systems For Interactive Systems," Ph.D. dissertation, Stanford University, http://si2.epfl.ch/~demichel/graduates/theses/yung.pdf, 12 2001.

[36] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[37] S. Keshav, C. Lund, S. Phillips, N. Reaingold, and H. Saran, "An empirical evaluation of virtual circuit holding time policies in ip-over-atm networks," *IEEE Journal on Selected Areas in Communications (J-SAC'95)*, vol. 13, pp. 1371–1382, 1995.

[38] J. Augustine, S.Irani, and C. Swamy, "Optimal power-down strategies," in *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS'04)*, 2004, pp. 530–539.

[39] S. Irani, S. Shukla, and R.Gupta, "Algorithms for power savings," in *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*, 2003, pp. 37–46.

[40] Y. Lu, L. Benini, and G. De Micheli, "Request-aware power reduction," in *Proceedings of the International Symposium on System Synthesis (ISSS'00)*, 2000, pp. 18–23.

[41] P. Krishnan, P. Long, and J.Vitter, "Adaptive Disk Spindown via Optimal Rent-to-Buy in Probabilistic Environments," *Algorithmica*, vol. 23, no. 1, pp. 31–56, 1999.

[42] T. Erbes, S. K. Shukla, and P. Kachroo, "Stochastic Learning Feedback Hybrid Automata for Dynamic Power Management in Embedded Systems," Master's thesis, http://fermat.ece.vt.edu/teodora/Erbes-Thesis.pdf, 01 2004.

[43] ——, "Stochastic Learning Feedback Hybrid Automata for Dynamic Power Management in Embedded Systems," in *Proceedings of the IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications. (SMCia'05).*, June 2005, pp. 208–213.

[44] G. Paleologo, L. Benini, A. Bogliolo, and G. De Micheli, "Policy Optimization for Dynamic Power Management," *Proceedings of the Design Automation Conference (DAC'98)*, pp. 182–187, 1998.

[45] Q. Qiu and Q. Wu and M. Pedram, "Dynamic power management of complex systems using generalized stochastic petri nets," in *Proceedings of the Design Automation Conference (DAC'00)*, June 2000, pp. 352–356.

[46] D. Parker, "Implementation of symbolic model checking for probabilistic systems," Ph.D. dissertation, University of Birmingham, 2002.

[47] H. Hansson and B. Jonsson, "A logic for reasoning about time and probability," *Formal Aspects of Computing (FAC'94)*, vol. 6, pp. 512–535, 1994.

[48] A. Bianco and L. de Alfaro, "Model checking of probabilistic and nondeterministic systems," in *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95)*, ser. Lecture notes in Computer science (LNCS), vol. 1026. Springer, 1995, pp. 499–513.

[49] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Verifying continuous time Markov chains," in *Proceedings of the Conference on Computer-Aided Verification (CAV'96)*, July 1996, pp. 269–276.

[50] C. Baier, J.-P. Katoen, and H. Hermanns, "Approximative symbolic model checking of continuous-time Markov chains," in *Proceedings of the International Conference on Concurrency Theory (CONCUR'99)*, Eindhoven, August 1999, pp. 146–161.

[51] L. de Alfaro, "Formal verification of probabilistic systems," Ph.D. dissertation, Stanford University, 1997.

[52] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen, "On the logical characterisation of performability properties," in *Proceedings of the*

International Colloquium on Automata, Languages and Programming (ICALP'00), 2000, pp. 780–792.

[53] *Technical specifications of hard drive IBM Travelstar VP 2.5inch.*, http://www.storage.ibm.com/storage/oem/data/travvp.htm, 1996.

[54] M. Maleki, K. Dantu, and M. Pedram, "Power-aware source routing protocol for mobile ad hoc networks," in *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'02)*, 2002, pp. 72–75.

[55] P. Rong and M. Pedram, "Extending the lifetime of a network of battery-powered mobile devices by remote processing: a Markovian decision-based approach," in *Proceedings of the Design Automation Conference (DAC'03)*, June 2003, pp. 906–911.

Sandy Irani graduated magna cum laude with a degree in Electrical Engineering and Computer Science from Princeton University in 1986. She completed her Ph.D. in Computer Science at University of California, Berkeley in 1991 and the following year was a recipient of the University of California President's Postdoctoral Fellowship. In the Fall of 1992, she joined the faculty of University of California at Irvine where she is currently a full professor. Her research has focused on the application of algorithm design and analysis to computing systems. In particular, she has specialized in the area of online algorithms and their applications to scheduling and resource allocation. Dr. Irani co-authored a chapter of "Approximations for NP-Hard Problems" on Online Algorithms (PWS Publishing). She has served as a guest editor of a special edition of the journal, Theoretical Computer Science, and currently serves as an editor of the online journal, Theory of Computing.

Gaurav Singh is a Ph.D. student at the FERMAT Lab of Virginia Tech. He received his B.E. in Electrical Enginnering from IIT Roorkee, India in 1999 and M.S. in Computer Engineering from Virginia Tech in 2004. His research interests include low-power synthesis, dynamic power management, system level design for embedded systems, and software design. His current work involves developing algorithms and strategies for reducing power dissipation during behavioral synthesis of hardware designs. He recently co-authored a chapter on "Parallelizing High-Level Synthesis: A Code Transformational Approach to High-Level Synthesis." in the CRC Handbook of EDA for IC Design.

Sandeep Shukla (M′99-SM′02) is currently an Assistant Professor of computer engineering with the Virginia Polytechnic and State University, Blacksburg. He is also a founder and Deputy Director of the Center for Embedded Systems for Critical Applications (CESCA) and Director of the FERMAT Laboratory. He was also elected as a College of Engineering Faculty Fellow at the Virginia Polytechnic and State University. He has authored or couathored over 100 papers in journals, books, and conference proceedings. He coauthored SystemC Kernel Extensions for Heterogeneous Modeling (Norwell, MA: Kluwer, 2004) and coedited Nano, Quantum and Molecular Computing: Implications to High Level Design and Validation (Norwell, MA: Kluwer, 2004) and Formal Methods and Models for System Design: A System Level Perspective (Norwell, MA: Kluwer, 2004). He has edited a number of special issues for various journals and is on the Editorial Board of IEEE Design and Test. Dr. Shukla has chaired a number of international conferences and workshops. He was the recipient of the NSF PECASE Award for his research in design automation for embedded systems design, which particularly focuses on system-level design languages, formal methods, formal specification languages, probabilistic modeling and model checking, dynamic power management, application of stochastic models and model analysis tools for fault-tolerant system design, and reliability measurement of fault-tolerant systems.

Rajesh Gupta is QUALCOMM endowed chair professor in the Department of Computer Science and Engineering at UC San Diego, California. He received his B. Tech. in Electrical Engineering from IIT Kanpur, India in 1984, M.S. in EECS from UC Berkeley in 1986 and a Ph.D. in Electrical Engineering from Stanford University in 1994. Earlier he worked at Intel Corporation, and on the Computer Science faculty at University of Illinois, Urbana-Champaign and UC Irvine. His current research is focused on energy efficient and mobile computing issues in embedded systems. He is author/co-author of over 150 articles on various aspects of embedded systems and design automation and four patents on PLL design, data-path synthesis and system-on-chip modeling. Gupta is a recipient of the Chancellors Fellow at UC Irvine, UCI Chancellor's Award for excellence in undergraduate research, National Science Foundation CAREER Award, two Departmental Achievement Awards and a Components Research Team Award at Intel. Gupta served as founding Chair of the ACM/IEEE Conference on Models and Methods in Codesign (MEMOCODE) and founding Co-Chair of ACM/IEEE/IFIP Conference on Codesign and System Synthesis (CODES+ISSS). Gupta is editor-in-chief of IEEE Design and Test of Computers and serves on the editorial boards of IEEE Transactions on CAD and IEEE Transactions on Mobile Computing. Gupta is a Fellow of the IEEE and VP of Publications of the IEEE Council on Electronic Design Automation (CEDA).