

MODELING, ESTIMATION AND CONTROL FOR POWER MANAGEMENT IN EMBEDDED SYSTEMS

Saulo O. D. Luiz, Antonio M. N. Lima, Angelo Perkusich

Departamento de Engenharia Elétrica
Centro de Engenharia Elétrica e Informática
Universidade Federal de Campina Grande
58.429-970 Campina Grande, PB, Brazil
{saulo, amnlima, perkusic}@dee.ufcg.edu.br

Abstract: Energy efficiency is one of the key challenges for a large class of electronic systems. Power savings are possible because electronic systems generally have an idle state, when, for example, the processor can run in a low power state. Thus, the correct estimation of the transition probabilities and the workload model are essential in the decision of which and when a power state transition should be performed by the electronic system. This paper introduces a feedback control policy for dynamic power management in non-stationary environments. The design of this feedback policy is discussed and its comparison with the policy that is obtained by solving a linear programming problem for a given performance and consumption constraints is also presented. Exploiting feedback control concepts in power management systems leads to a relatively simple solution that allows real-time adjustment of performance and consumption constraints. The preliminary results were obtained with a relatively simple proportional and integral controller which was designed such that the closed-loop should track a given average request loss. These results are relatively better than the ones obtained with the policy derived from the linear programming formulation.

Keywords: Power management, Markov chains, Parameter estimation, Feedback control.

1. INTRODUCTION

In the past years, mobile embedded systems such as mobile phones, internet tablets and personal digital assistants (PDAs) have increased power consumption due to new capabilities such as faster processors, cameras, LCDs, network interfaces, etc. This has become a critical problem on battery powered mobile devices due to the fact that the capacity of batteries has not increased as fast as power consumption requirements [1, 2]. Therefore, improving the lifetime of the battery through efficient energy use has become one of the key challenges to the design of embedded systems.

Power consumption reduction techniques are classified as static and dynamic [3]. The static techniques are those ap-

plied at design time, such as synthesis and compilation for low power consumption. The dynamic techniques, named *Dynamic Power Management* (DPM), allow power reduction at runtime by shutting down or reducing operation frequency/tension of idle system components. For a given device, a *power manager* (PM) monitors and controls the power states of the components and of the system through a procedure named power management policy [2].

During operation, the system experiences idle and busy periods of time due to its workload. To optimize power consumption reduction, the power management policy must consider a workload model, what can be achieved by modeling it as a stochastic process. However, in real-life systems the workload cannot be pre-characterized due to its nonstationarity. To better understand this problem, let us consider a general purpose processor. The combination of applications running on such system may strongly vary the workload, depending on what is being executed. Moreover, the workload may drastically change during the day, or over the days of the week, or even between different users [2].

Benini et al. [4] have proposed a stochastic solution for power management using discrete time Markov decision processes for stationary workloads. An exact solution from a linear programming problem has been derived directly from the optimization problem of the power policy. Chung et al. [2] introduced an adaptive energy optimization technique to handle workloads initially unknown or nonstationary by means of fixed size sliding windows to estimate the parameters of the workload [2]. The optimization problem consists of the choice of a power policy which minimizes the expected consumption under a performance penalty constraint. An optimum power policy is able to reduce power consumption while still satisfying the performance penalty constraint, that is, the policy which degrades the system performance up to an acceptable level (the performance penalty constraint), is the one which minimizes power consumption. However, the solution of the optimization problem in [4] is not guaranteed to provide the performance penalty constraint for every workload, and therefore, may not offer optimal energy sav-

ings. Thus, power policies using stochastic models may be considered an open-loop control system, because they do not adjust the actual performance penalty to the reference performance penalty constraint. The optimal solution strongly depends on the quality of the model used to derive the solution, i.e. the parameters of the stochastic model and the knowledge of the workload. If the performance penalty constraint changes during operation of the power-managed system, then there are two possible solutions: (i) the optimization problem must be solved on-line, what is not practical, because this demands a considerable energy and time; (ii) or the optimal policies must be pre-computed at design time and stored in a memory, what demands considerable memory space. These limitations of stochastic solution [4] justify the search for other solutions for power management.

Wang et al. [5] developed feedback algorithms for the power management of a group of servers by using frequency scaling. The algorithms are targeted both for efficiency and power capping, to deliver a good trade-off between power capping, efficiency, and application performance. The efficiency algorithm is designed using an integral controller, where the input is the observed error in utilization of a server and the output is the clock frequency. The power capping algorithm is also designed using an integral controller, where the input is the error between the power budget of the server and the actual consumption, and the output is the clock frequency.

Sridharan, Gupta and Mahapatra [6] introduced algorithms to perform power management of real-time embedded systems and to preserve the overall system reliability. A proportional feedback controller is used to assess the additional number of copies of a job to ensure the reliability constraints at run-time.

Ogras, Marculescu and Marculescu [7] have divided a network-on-chip (NoC) into multiple voltage-frequency islands, which are interconnected by queues. A state-space model of the queues occupancy is derived from the utilization of the inter-domain queues by the voltage-frequency islands. Feedback controllers are designed to control the speed of each voltage-frequency island by means of dynamic voltage-frequency scaling to maintain the queues occupancy at reference levels.

Tian et al. [8] developed a control design methodology for power management by means of dynamic voltage scaling (DVS). Their methodology is tailored for embedded microprocessors that run a known set of real-time control tasks. An analytical model is derived for the system considering the worst-case execution time of the real-time control tasks, and the controller is designed using feedback control theory. The power manager is composed of a proportional-integral controller, where the input is the error in processor utilization and the output is the processor frequency. Unfortunately, the output of the controller is a continuous value, whereas real-life processors only have a limited number of voltage/frequency levels.

Lu, Lach and Stan [9] addressed the problem of power management of a system running video playback. A buffer was placed between the decoder and the display, and a proportional integral controller adjusts the decoder's speed by

means of dynamic voltage scaling (DVS). The input to the controller is the buffer occupancy information and the output is the frequency/voltage decision. If the number of frames in the buffer is less than a lower threshold or more than a higher threshold, then the feedback controller is used to pull back the number of frames to the desired region.

Minerick, Freeh and Kogge [10] have implemented for the Linux operating system a feedback mechanism named *energy conservation*, which controls the frequency and voltage of the processor of a battery-powered system to insure that the energy in the battery meets or exceeds a reference value of energy at a specified time in the future. The main drawback of this technique is that the performance may be reduced below acceptable levels because energy conservation is more critical.

In this work, we combine the *Markov chain model*, which accurately models the stochastic behavior of the power-managed system, and *feedback control*, which offers simpler control laws and has the ability of adapt itself at run-time. This approach is based on the assumption that it is possible to approximate the stochastic behavior of the power managed system by a discrete-time dynamic model.

This paper is organized as follows: In Section 2, the system model using discrete-time Markov chains is reviewed. In Section 3, the problem of workload identification is emphasized. The discrete-time Markov chains model is extended to a battery-aware technique in Section 4. In Section 5, the proposed feedback control power management policy is introduced. Then, in Section 6, a comparison study of power management policies is performed. Finally, in Section 7 the features of the proposed policy are summarized and the directions for future research are outlined.

2. STOCHASTIC APPROACH FOR POWER MANAGEMENT

In this section we review the dynamic power management (DPM) approach described in [2] and [4]. Both approaches are based on stochastic models. The techniques which handle unknown nonstationary workloads introduced in [2] are extensions of the system model presented in [4].

2.1. Policies for stationary workloads

The power managed system is modeled as a *service provider (SP)* with a *power manager (PM)*. The service provider processes the requisitions from a unique *service requester (SR)*, and has several power states, each one with a service rate and a specific power consumption level. Thus, the service requester represents the workload of the system and there is a *service queue (SQ)* that stores requests which have not been serviced by the service provider yet because the latter is busy or has zero service rate. This system model is shown in Fig. 1. The components are modeled as stationary discrete-time Markov chains. Two basic assumptions are: (i) the requests are indistinguishable, with no service priority, and (ii) the service queue has finite size. The power manager takes decisions with respect to the service provider based on the history of the service provider, the service queue and the service requester, and issues commands to change the power

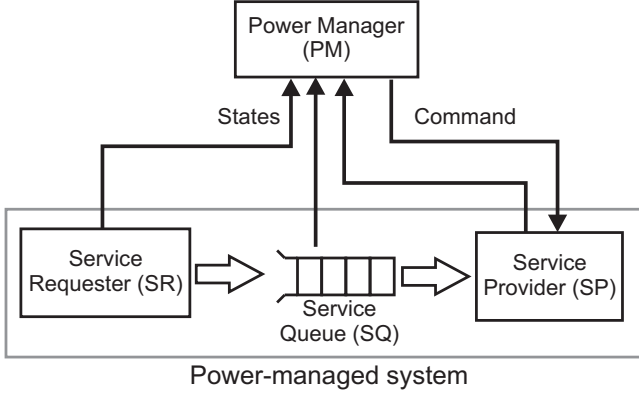


Figure 1 – Components of the system model of DPM in known stationary environment.

state of the service provider. The time is considered discrete: $t = 0, 1, 2, \dots$. The interval between two consecutive values of time is named *time slice*.

Definition 1 A service requester (SR) is a pair $(M_{SR}, z(r))$ where:

- M_{SR} is a Markov chain with state set $\mathcal{R} = \{r_i, |i = 0, 1, \dots, (R-1)\}$ and transition matrix \mathbf{P}^{SR} , where R is a positive integer;
- $z(r)$ is a function $z : \mathcal{R} \rightarrow \mathbb{N}$ which represents the number of requests issued per time slice when the SR is in state r .

The probability that the service requester is in state r_j is $\rho_j \triangleq \lim_{t \rightarrow \infty} P[s_r(t) = r_j]$. For the case of a stationary service requester with two states, the expressions for ρ_0 and ρ_1 are shown in (1) and (2) respectively. As shown in Fig 2, the probability ρ_1 that the service requester is in state r_1 is a function of both the transition probabilities p_{r_0, r_0}^{SR} and p_{r_1, r_1}^{SR} . The greater the probability ρ_1 , the more intensive the workload is, because the probability that a request is issued at a given time slice increases.

$$\rho_0 = \frac{1 - p_{r_1, r_1}^{SR}}{2 - (p_{r_0, r_0}^{SR} + p_{r_1, r_1}^{SR})} \quad (1)$$

$$\rho_1 = \frac{1 - p_{r_0, r_0}^{SR}}{2 - (p_{r_0, r_0}^{SR} + p_{r_1, r_1}^{SR})} \quad (2)$$

Definition 2 A service provider (SP) is a triple $(M_{SP}(a), b(s, a), c(s, a))$ at which:

- $M_{SP}(a)$ is a stationary controllable Markov chain with state set $\mathcal{S} = \{s_i, |i = 0, 1, \dots, (S-1)\}$, control set $\mathcal{A} = \{a_i, |i = 0, 1, \dots, A-1\}$ and transition matrix $\mathbf{P}^{SP}(a)$, at which S is a positive integer, and A is the number of commands of the power manager (PM);
- $b(s, a)$ is a function $b : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, which represents the probability of the SP completing the service of a request in the present time slice, given that the SP is in state $s \in \mathcal{S}$ and the command $a \in \mathcal{A}$ has been issued;

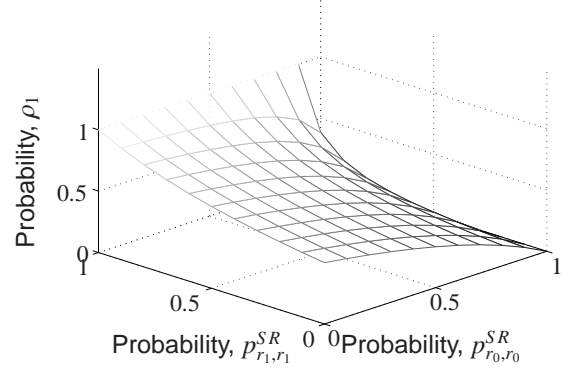


Figure 2 – Probability ρ_1 that a service requester with state set $\{r_0, r_1\}$ be in state r_1 .

- $c(s, a)$ is a function $c : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, which represents the power consumption associated to the state $s \in \mathcal{S}$ and the command $a \in \mathcal{A}$, i.e. the consumption (in arbitrary units: W, A or J) of the SP in the present time slice.

Definition 3 A service queue (SQ) of length $Q - 1$ is a stationary controllable Markov chain $M_{SQ}(a, s, r)$ with state set $\mathcal{Q} = \{q_i, |i = 0, 1, \dots, (Q-1)\}$, control set $\mathcal{A} \times \mathcal{S} \times \mathcal{R}$ and transition matrix $\mathbf{P}^{SQ}(a, s, r)$. The transition probabilities are defined in (3).

$$p_{q_i, q_j}^{SQ}(a, s, r) = \begin{cases} 1 - b(s, a), & \text{if } j = i + z(r) \text{ and } 0 < i + z(r) < Q \\ b(s, a), & \text{if } j = i + z(r) - 1 \text{ and } 0 \leq i + z(r) - 1 < Q \\ 1, & \text{if } j = i = 0 \text{ and } z(r) = 0 \\ 1, & \text{if } j = Q - 1 \text{ and } i + z(r) \geq Q \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The system state $x = (s, r, q)$ is the composition of the states of the service provider, service requester and service queue respectively. Thus, the system has state set $\mathcal{X} = \mathcal{S} \times \mathcal{R} \times \mathcal{Q}$ with $X = S \cdot R \cdot Q$ states. The command C to be issued by the power manager (PM) is a random variable with discrete probability distribution $\delta^{(t)} = (p_{a_0}, p_{a_1}, \dots, p_{a_{A-1}})$, named the decision of the power manager, which assigns a probability value p_a to each command $a \in \mathcal{A}$ at time t . The policy π of the power manager is a set of decisions $[\delta^{(0)}, \delta^{(1)}, \dots]$ over time $t = 0, 1, \dots$.

The transition matrix $P(a)$ of the system when command $a \in \mathcal{X}$ is taken is a $X \times X$ matrix with each element $p_{x_i, x_j}(a)$ as defined in (4). The transition matrix $P_{\delta^{(t)}}$ of the system when the decision $\delta^{(t)}$ is taken at time t is defined in (5). The transition matrix $P_{\pi}(t)$ of the system from time 0 to t is defined in (6).

$$\begin{aligned} p_{x_i, x_j}(a) &= \text{Prob}[x_j = (s', r', q') | x_i = (s, r, q), a] = \\ &= p_{s, s'}^{SP}(a) p_{r, r'}^{SR} p_{q, q'}^{SQ}(a, s, r) \end{aligned} \quad (4)$$

$$P_{\delta^{(t)}} = \sum_{p_a \in \delta^{(t)}} p_a P(a) \quad (5)$$

$$P_\pi(t) = \prod_{\tau=0}^{t-1} P_{\delta(\tau)} \quad (6)$$

The system is supposed to be used until a certain *time horizon* T_f that is finite and random, for example, the time when the battery discharges. Thus, every transition probability of the system must be multiplied by a *discount factor* $0 < \xi < 1$, and the time horizon T_f is modeled as a geometrically distributed random variable with mean $E[T_f] = (1 - \xi)^{-1}$. Once $E[T_f]$ is estimated, the discount factor $\xi = 1 - 1/E[T_f]$. Using the discount factor ξ , the transition matrix $P_\pi(t)$ of the system from time 0 to t is $P_\pi(t) = \prod_{\tau=0}^{t-1} (\xi P_{\delta(\tau)}) = \xi^{t+1} \prod_{\tau=0}^{t-1} P_{\delta(\tau)}$. Let $p(0)$ be the initial state probability distribution of the system at time 0. Then the state probability distribution of the system at time t is $p(t) = p(0)P_\pi(t)$.

A *stationary policy* $\pi = [\delta, \delta, \dots]$ uses the same decision δ every time t . The decision δ of a stationary policy is a function of the system state $x \in \mathcal{X}$, thus it is denoted by δ_x . The expected consumption of the system in state $x \in \mathcal{X}$ and when decision δ_x is taken is $c(x, \delta_x) = \sum_{p_a \in \delta_x} p_a c(s, a)$. The performance penalty of the system in state $x \in \mathcal{X}$ is $d(x)$ and is user defined, e.g., $d(x) = q$ is the number of pending requests in the queue. The consumption and performance penalty vectors of decision δ are shown in (7). The expected consumption and expected performance penalty of the service provider (SP) at time slice t under a policy π are $E_\pi[\mathbf{c}_\delta(t)] = p^{(t)}\mathbf{c}_\delta$ and $E_\pi[\mathbf{d}_\delta(t)] = p^{(t)}\mathbf{d}_\delta$ respectively.

$$\mathbf{c}_\delta = \begin{pmatrix} c(x_0, \delta_{x_0}) \\ \vdots \\ c(x_{X-1}, \delta_{x_{X-1}}) \end{pmatrix} \quad \mathbf{d}_\delta = \begin{pmatrix} d(x_0) \\ \vdots \\ d(x_{X-1}) \end{pmatrix} \quad (7)$$

The *policy optimization problem* (PO), expressed in (8), consists of the choice of the stationary policy π which minimizes the expected consumption for a given performance penalty constraint, where D is the maximum acceptable performance penalty.

$$\text{PO :} \quad \min_{\pi} \sum_{t=0}^{\infty} E_\pi[\mathbf{c}_\delta(t)] \\ \text{such that } \sum_{t=0}^{\infty} E_\pi[\mathbf{d}_\delta(t)] \leq D \quad (8)$$

The optimization problem in (8) is converted into the linear programming problem, in (9) and the result is a stationary policy, expressed as a matrix, named *decision table*, which has X lines, i.e. the number of states of the system, and A columns, i.e. the number of commands issued by the power manager to the service provider. Each line of the decision table is a decision δ_x associated to a system state $x \in \mathcal{X}$. Thus, each element of the decision table is the probability of the power manager issuing a command $a \in A$ when the system is in state $x = (s, r, q)$. In (9) the variables $f_{x,a}$ are named *state-action frequencies* and represent the expected number of times the system state is x and the command a is issued by

the power manager.

$$\text{PL1 :} \quad \min \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} f_{x,a} c(x, a) \quad (9) \\ \text{tal que } \sum_{a \in \mathcal{A}} f_{x,a} - \xi \sum_{y \in \mathcal{X}} \sum_{a \in \mathcal{A}} p_{y,x}(a) f_{y,a} = p_x^{(1)}, \forall x \in \mathcal{X} \\ \sum_{x \in \mathcal{X}} \sum_{a \in \mathcal{A}} f_{x,a} d(x) \leq D \\ f_{x,a} \geq 0, \forall x \in \mathcal{X}, a \in \mathcal{A}$$

To solve the linear programming problem, in (9), it is possible to use: the tool *PCx* [11] or the tool *linprog* of the optimization toolbox of Matlab [12]; in both tools the algorithm of interior points is used to solve the optimization problem. The probability $m_{x,a}$ that the power manager issues the command a when the system state is x is estimate in (10). The probabilities $m_{x,a}$ are organized in a decision table M_π as in (11) where the lines represent the system states $x_0, x_1 \dots x_{X-1}$ and the columns represent the commands of the power manager $a_0, a_1, \dots a_{A-1}$. At run time, the energy policy takes the line M_π of the corresponding to the current system state x : $(m_{x,a_0} \ m_{x,a_1} \ \dots \ m_{x,a_{A-1}})$, named *decision*. This line represents the probability distribution that the power manager issues the command a with probability $m_{x,a}$ when the system state is x .

$$m_{x,a} = \frac{f_{x,a}}{\sum_{a' \in \mathcal{A}} f_{x,a'}} \quad (10)$$

$$M_\pi = \begin{bmatrix} m_{x_0,a_0} & m_{x_0,a_1} & \dots & m_{x_0,a_{A-1}} \\ m_{x_1,a_0} & m_{x_1,a_1} & \dots & m_{x_1,a_{A-1}} \\ \vdots & \vdots & \vdots & \vdots \\ m_{x_{X-1},a_0} & m_{x_{X-1},a_1} & \dots & m_{x_{X-1},a_{A-1}} \end{bmatrix} \quad (11)$$

Example 1 Consider a service provider (SP) $(M_{SP}(a), b(s, a), c(s, a))$ where: $M_{SP}(a)$ is a stationary controllable Markov chain with state set $\mathcal{S} = \{s_0, s_1\}$; control set $\mathcal{A} = \{a_0, a_1\}$, meaning switch to s_0 and switch to s_1 , respectively; transition matrix $\mathbf{P}^{SP}(a)$ in (12) and (13); the matrix $c(s, a)$ in (14) specifies the current consumption (in mA) of the service provider when in state s and when command a is issued by the power manager; and the matrix $b(s, a)$ in (15) specifies the service rate of the service provider when in state s and when command a is issued by the power manager, i.e., the probability of completing a request in a time slice. The state s_0 has greater consumption and better performance than the state s_1 . From (12) and (13), the transition probabilities of the service provider are modeled as deterministic, because, in the system modeled, the transition times between the states of the service provider are negligible with respect to the discretization time step. For this same reason, the current consumption $c(s, a)$ and the service rate $b(s, a)$ in (14) only depend on the current state s of the service provider. The Markov chain model of the service provider is shown in Fig. 3.

$$\mathbf{P}^{SP}(a_0) = \begin{matrix} & s_0 & s_1 \\ s_0 & \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \\ s_1 & \end{matrix} \quad (12)$$

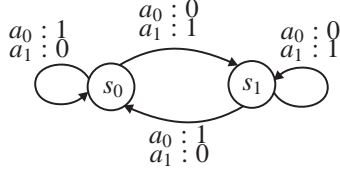


Figure 3 – Markov chain representing a service provider - SP.

$$P^{SP}(a_1) = \begin{matrix} & s_0 & s_1 \\ \begin{matrix} s_0 \\ s_1 \end{matrix} & \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \end{matrix} \quad (13)$$

$$c(s, a) = \begin{matrix} & a_0 & a_1 \\ \begin{matrix} s_0 \\ s_1 \end{matrix} & \begin{bmatrix} 318.7 & 318.7 \\ 294.9 & 294.9 \end{bmatrix} \end{matrix} \quad (14)$$

$$b(s, a) = \begin{matrix} & a_0 & a_1 \\ \begin{matrix} s_0 \\ s_1 \end{matrix} & \begin{bmatrix} 1 & 1 \\ 0.3125 & 0.3125 \end{bmatrix} \end{matrix} \quad (15)$$

Now consider a service requester (SR) ($M_{SR}, z(r)$) with state set $\mathcal{R} = \{r_0, r_1\}$ with function $z(r)$ identifying the number of requests that the service requester issues per time slice when in state r . For the service requester of this example, we use a one-to-one correspondence between images of z and service requester states: $z(r_0) = 0$, $z(r_1) = 1$. The transition matrix \mathbf{P}^{SR} is shown in (16) and the Markov chain model of the service requester in Fig. 4.

$$P^{SR} = \begin{matrix} & r_0 & r_1 \\ \begin{matrix} r_0 \\ r_1 \end{matrix} & \begin{bmatrix} 0.6 & 0.4 \\ 0.5 & 0.5 \end{bmatrix} \end{matrix} \quad (16)$$

The service queue (SQ) $M_{SQ}(a, s, r)$ of length $Q = 1$ has

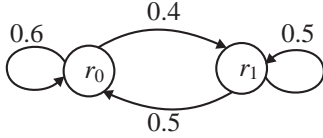


Figure 4 – Markov chain representing a service requester - SR.

state set $\mathcal{Q} = \{q_0, q_1\}$, i.e., in q_0 the service queue is empty, and in q_1 the service queue has one pending service request. The transition matrix \mathbf{P}^{SQ} is shown in (17) and (19). The Markov chain model of the service queue is shown in Fig. 5.

$$P^{SQ}(a, s, r = 0) = \begin{matrix} & q_0 & q_1 \\ \begin{matrix} q_0 \\ q_1 \end{matrix} & \begin{bmatrix} 1 & 0 \\ b(s, a) & 1 - b(s, a) \end{bmatrix} \end{matrix} \quad (17)$$

$$P^{SQ}(a, s, r = 1) = \begin{matrix} & q_0 & q_1 \\ \begin{matrix} q_0 \\ q_1 \end{matrix} & \begin{bmatrix} b(s, a) & 1 - b(s, a) \\ 0 & 1 \end{bmatrix} \end{matrix} \quad (19)$$

$$(20)$$

This system model is to be simulated during a time horizon $T_f = 10000$ time slices. Thus, the discount factor $\xi = 1 - 1/10000 = 0.9999$. The performance penalty constraint defined for this system is the average request loss L .

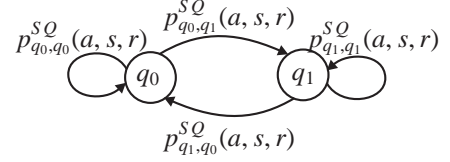


Figure 5 – Markov chain representing a service queue - SQ.

A request loss happens if a new request is generated by the service requester (SR), but the service queue (SQ) is already full. The maximum average request loss is 10%, whose performance penalty constraint is $L = 0.10T_f = 1000$. Let E be a random variable with sample space $\{0, 1\}$ and probability distribution: $p(E = 0) = 1 - b(s, a)$ i.e. $E = 0$ when the service provider does not service a request; and $p(E = 1) = b(s, a)$ i.e. $E = 1$ when the service provider services a request. Let e be a realization of E . For each system state $x = (s, r, q)$, the number of pending requests in the queue $d(x) = q$ i.e. the queue state. Considering the request loss $l(x = (s, r, q))$ in (21), there is a request loss if the number of incoming requests plus the queue occupation minus “one possible request serviced” exceeds the queue capacity $Q - 1$. In practice, a request loss does not represent a real lack of service, but the undesirable condition of many requests waiting to be serviced [4].

$$l(x = (s, r, q)) = \begin{cases} 1, & \text{if } z(r) + q - e > Q - 1 \\ 0, & \text{otherwise} \end{cases} \quad (21)$$

The optimization problem in (8) is converted into a linear programming problem in [4] and the solution is the decision table shown in (22).

$$M_\pi = \begin{matrix} & a_0 & a_1 \\ \begin{matrix} (s_0, r_0, q_0) \\ (s_0, r_0, q_1) \\ (s_0, r_1, q_0) \\ (s_0, r_1, q_1) \\ (s_1, r_0, q_0) \\ (s_1, r_0, q_1) \\ (s_1, r_1, q_0) \\ (s_1, r_1, q_1) \end{matrix} & \begin{pmatrix} 0.5433 & 0.4567 \\ 0.3811 & 0.6189 \\ 1.0000 & 0.0000 \\ 0.3253 & 0.6747 \\ 0.2748 & 0.7252 \\ 1.0000 & 0.0000 \\ 1.0000 & 0.0000 \\ 0.3988 & 0.6012 \end{pmatrix} \end{matrix} \quad (22)$$

The average current consumption for this system is 310.0939mA, i.e. 2.7% less than if the system is always in state s_0 . The average request loss is 2.22% and is smaller than the performance penalty constraint. \triangle

Example 2 Consider the same service provider, service queue and service requester as in Example 1. The current consumption and average request loss will be analyzed for different combinations of the transitions probabilities of the service requester, that is, for different workloads. As shown in Fig 2, the greater the probability π_1 that the service requester is in state r_1 , the more intensive the workload is. Two examples of traces of the service requester (SR) states are shown in Fig. 6: (i) the service requester transition probabilities are $p_{r_0,r_0}^{SR} = 0.1$ and $p_{r_1,r_1}^{SR} = 0.9$, thus $\pi_1 = 0.9$; (ii) the service requester transition probabilities are $p_{r_0,r_0}^{SR} = 0.9$ and

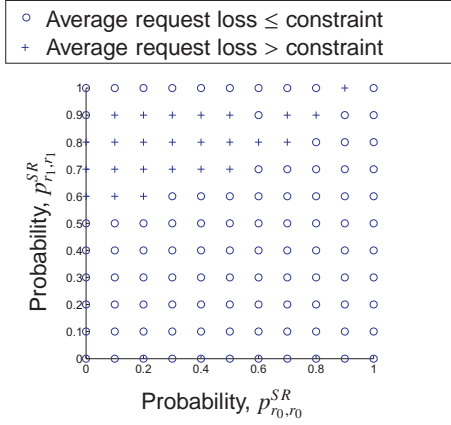


Figure 8 – Policies which (do not) satisfy the performance constraint.

$p_{r_1, r_1}^{SR} = 0.1$, thus $\pi_1 = 0.1$. In order to keep the performance penalty under its constraint, i.e., maintain the performance of the system at acceptable limits, the power manager issues commands to change the power state of the service provider, either increasing the service rate for intensive workloads or decreasing the service rate for less intensive workloads.

The system model of this example is simulated for different stationary service requesters, with $p_{r_0, r_0}^{SR} = 0, 0.1, 0.2, \dots, 1.0$ and $p_{r_1, r_1}^{SR} = 0, 0.1, 0.2, \dots, 1.0$, and the results of power consumption (average Current in mA) and performance penalty (average request loss) are shown in Fig. 7. For the most intensive workload, when $p_{r_0, r_0}^{SR} = 0$ and $p_{r_1, r_1}^{SR} = 1$, the average current is 318.7mA because the service provider is always in state s_0 which has the greatest power consumption (318.7mA), but has the best service rate $b(s_0, a) = 1$ for $a = a_0, a_1$, that is, when the service provider is in state s_0 , it completes a request in the present time slice with probability 1. Thus, the average request loss is 0. For the less intensive workload, when $p_{r_0, r_0}^{SR} = 1$ and $p_{r_1, r_1}^{SR} = 0$, the average current is 294.9mA because the service provider is always in state s_1 which has the smallest power consumption (294.9mA), but has the worst service rate $b(s_1, a) = 0.3125$ for $a = a_0, a_1$, that is, when the service provider is in state s_1 , it completes a request in the present time slice with probability 0.3125. Nevertheless, the average request loss is 0 because no request is issued at all, since $p_{r_0, r_0}^{SR} = 1$ and $p_{r_1, r_1}^{SR} = 0$. For some workloads, the request loss is greater than the performance constraint, and the policies optimized for these workloads are not acceptable, as shown in Fig. 8. Besides, for some workloads, the request loss is less than the performance constraint, i.e. the policies satisfy the performance penalty constraint, but is not guaranteed to fully exploit it and save more power.

In general, the $p_{r_0, r_0}^{SR} p_{r_1, r_1}^{SR}$ plane may be divided into two regions: (i) one for which $\pi_1 > 0.5 \Rightarrow (1 - p_{r_0, r_0}^{SR}) / [2 - (p_{r_0, r_0}^{SR} + p_{r_1, r_1}^{SR})] > 0.5 \Rightarrow p_{r_1, r_1}^{SR} > p_{r_0, r_0}^{SR}$ and related to intensive workloads; (ii) and one for which $\pi_1 < 0.5 \Rightarrow (1 - p_{r_0, r_0}^{SR}) / [2 - (p_{r_0, r_0}^{SR} + p_{r_1, r_1}^{SR})] > 0.5 \Rightarrow p_{r_1, r_1}^{SR} < p_{r_0, r_0}^{SR}$ and related to soft workloads. \triangle

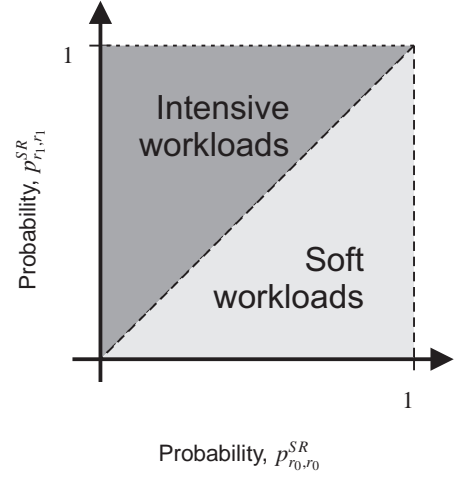


Figure 9 – Classification of system workloads.

2.2. Policies for nonstationary system workloads

The dynamic power management (DPM) system model in known stationary environments [4] assumes that the model of the service requester (SR) is constant over time and is known before operation time, because the transition probabilities of the service requester are estimated from request records of the real system. These two assumptions impose limitations to dynamic power management, since real workloads may present nonstationarity, i.e. the transition probabilities of the service requester may change at run time. In this case, the transition probabilities of the service requester Markov chain model depend on time as shown in Fig. 10(a) for the case of two states. The approach of Chung et al. [2] is based on the paradigm of the *principle of estimation and control* (PEC). The transition probabilities of the service requester Markov chain model are estimate on-line, and optimal policies computed off-line are interpolated to offer a close-to-optimal solution for the estimated parameters.

A nonstationary workload U^l is represented as a series of N_u stationary workloads u_s , $s = 0, 1, \dots, N_u - 1$ with different transition probabilities $R_i = p_{r_i, r_i}^{SR}, \forall r_i \in \mathcal{R}$. Thus, R_i is a function of u_s , as shown in Fig. 10(b). The *best-adaptive policy* is the technique that applies the decision table optimized for the R_i of each workload u_s [2]. This technique is optimal for the nonstationary case, but it is impractical because *a priori* knowledge of the nonstationary workload is needed.

The dynamic power management (DPM) technique using fixed size sliding windows is shown in Fig. 11. The

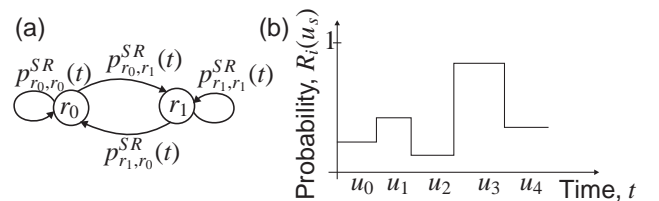


Figure 10 – (a) Nonstationary service requester; (b) Nonstationary workload and the relationship between the probability R_i and the present stationary workload u_s .

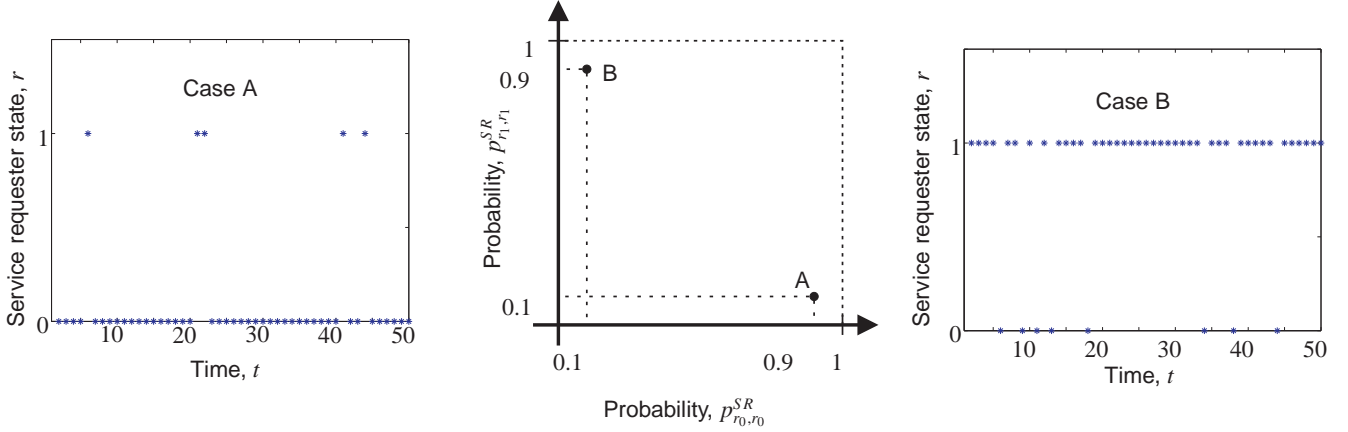


Figure 6 – Examples of system workloads.

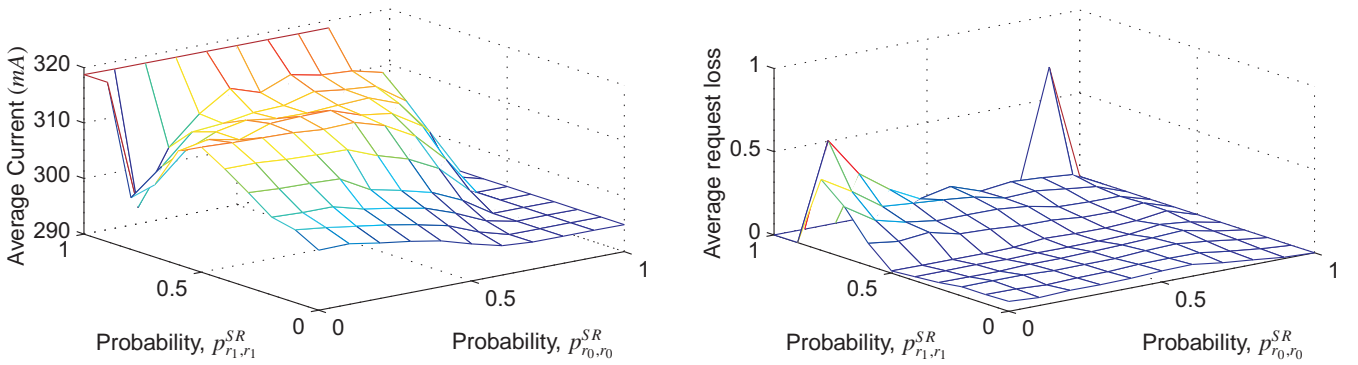


Figure 7 – Power consumption and request losses of a power-managed system for different transition probabilities of the service requester (SR).

power manager (*PM*) estimates the workload of the system and chooses a command to issue to the service provider. To do this, the power manager uses R fixed size *request windows*, 1, of the service requester (*SR*) to evaluate *estimates* of the transition probabilities of the service requester. A *policy table*, 4, contains decision tables generated at design time by the policy optimization problem (*OP*). Thus, the *policy interpolator*, 5, uses the states of the service requester, 7, service queue, 8, service provider, 9, and the *estimates* of the transition probabilities of the service requester for the interpolation of the decision tables. The result is a *decision* which is used for the *choice of the command*, 6, to be issued to the service provider.

The block *request windows* as shown in Fig. 11 contains a multiwindow (first introduced in [2]) that stores service requester (*SR*) requests and evaluates the estimates of the transition probabilities of the service requester. An example is shown in Fig. 12 for a service requester with two states. Each window is denoted by W_i , $i = 0, 1, \dots, (R - 1)$, and corresponds to a state r_i of the service requester. The notation $W_i^k(t)$ denotes the content of position k in window W_i at time t . Every window has size l_w . If there is a transition of the service requester (*SR*) from a state $s_r(t - 1) = r_i$ to a state $s_r(t) = r_j$, as shown in Fig. 13, then the *Previous request buffer (PRB)* stores the state of the service requester $s_r(t - 1) = r_i$ and controls the *window selector*, which chooses the window W_i such that $s_r(t - 1) = r_i$. Only the

selected window W_i shifts right its content by one position, i.e., $W_i^{k+1}(t) = W_i^k(t)$, $\forall k = 0, \dots, l_w - 2$, discards the rightmost element $W_i^{l_w-1}(t)$, and stores $s_r(t) = r_j$ in position $W_i^0(t)$. The block *Compute estimate* can evaluate the estimate $\hat{p}_{r, r'}^{SR}(t, l_w)$ of the transition probability of the service requester from a state r to a state r' at time t for the

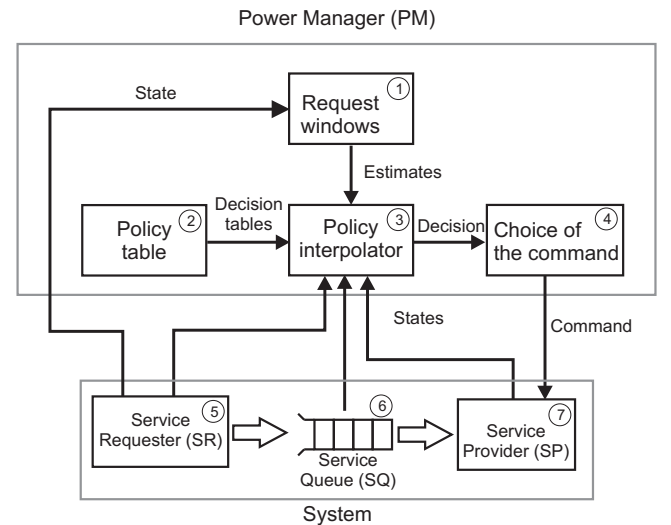


Figure 11 – Power management for nonstationary system workloads.

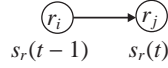


Figure 13 – Transition of the service requester from state $s_r(t-1) = r_i$ to state $s_r(t) = r_j$.

window size l_w . In the case of Fig. 12, only $\hat{p}_{r_0, r_0}^{SR}(t, l_w)$ and $\hat{p}_{r_1, r_1}^{SR}(t, l_w)$ are evaluated because the others can be calculated whenever they are necessary: $\hat{p}_{r_0, r_1}^{SR}(t, l_w) = 1 - \hat{p}_{r_0, r_0}^{SR}(t, l_w)$ and $\hat{p}_{r_1, r_0}^{SR}(t, l_w) = 1 - \hat{p}_{r_1, r_1}^{SR}(t, l_w)$.

The estimates of the transition probabilities of the service requester (SR) Markov chain model are evaluated by the block *Compute estimate* in Fig. 12 by means of maximum likelihood estimation. Let “==” be the equivalence operation, i.e., the result is “1” if the arguments are equal, and “0” otherwise. Let the expression $(W_i^k(t) == j)$ identify whether the state stored in $W_i^k(t)$ is j , meaning the occurrence of a transition from state i to state j . Thus, the estimate $\hat{p}_{r_i, r_j}^{SR}(t, l_w)$ is shown in (23).

$$\hat{p}_{r_i, r_j}^{SR}(t, l_w) = \frac{\sum_{k=0}^{l_w-1} (W_i^k(t) == j)}{l_w}, \quad \forall i, j \quad (23)$$

Example 3 Consider a nonstationary service requester (SR) ($M_{SR}, z(r)$) with state set $\mathcal{R} = \{r_0, r_1\}$ and transition probabilities specified in (24) and (25). This service requester can be represented by the series of stationary workloads: u_0 during $0 \leq t \leq 2000$ time slices, and u_1 during $2000 < t \leq 4000$ time slices. To identify the service requester (SR) model, it was used the multiwindow technique (MW) with fixed window sizes 100 and 400, and the estimates $\hat{p}_{r_0, r_0}^{SR}(t, 100)$ and $\hat{p}_{r_0, r_0}^{SR}(t, 400)$ are shown in Fig. 14. The estimates of the transition probabilities $\hat{p}_{r_1, r_1}^{SR}(t, 100)$ and $\hat{p}_{r_1, r_1}^{SR}(t, 400)$ are not shown here. As shown in Fig. 14, it is possible to observe that, after the transition from u_0 to u_1 at $t = 2000$ time slices, $\hat{p}_{r_0, r_0}^{SR}(t, 400) \approx p_{r_0, r_0}^{SR}(t)$ at $t \geq 2500$ time slices. In this case,

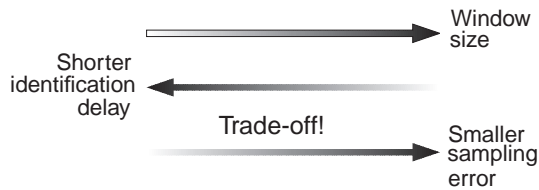
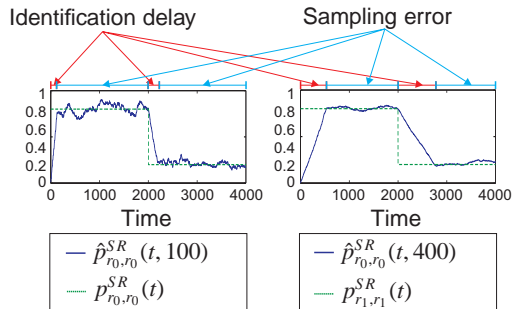


Figure 14 – Trade-off between sampling error and identification delay.

500 time slices was the period of time necessary to completely fill the window with size 400 only with transitions of the new workload u_1 , and it is called the identification delay (named adaptation time in [2]). After that period of time, the estimates oscillate around the true transition probabilities value $p_{r_0, r_0}^{SR}(t)$ what is called the sampling error (named resolution error in [2]). \triangle

$$p_{r_0, r_0}^{SR}(t) = \begin{cases} 0.8, & \text{if } 0 \leq t \leq 2000 \\ 0.2, & \text{if } 2000 < t \leq 4000 \end{cases} \quad (24)$$

$$p_{r_1, r_1}^{SR}(t) = \begin{cases} 0.4, & \text{if } 0 \leq t \leq 2000 \\ 0.1, & \text{if } 2000 < t \leq 4000 \end{cases} \quad (25)$$

As shown of Fig. 11, the block *policy interpolator* evaluates an optimum policy for the service requester transition probability estimates. Since it is not efficient both in time and energy consumption to solve the optimization problem in (8) online for each transition probability estimate, a set of optimal policies is evaluated off-line and then interpolated on-line. The transition matrix \mathbf{P}^{SR} of a service requester with two states can be represented only by the pair of probabilities (R_0, R_1) where $R_i = p_{r_i, r_i}^{SR}$, $i = 0, 1$. For each service requester state i , the transition matrix can be sampled into a finite set of values, each one represented as $R_{ij} = j/(NS_i - 1)$, $j = 0, 1, \dots, NS_i - 1$, where NS_i is the number of samples for the state i . And the optimization problem in (8) is solved for each pair of probabilities (R_{0j}, R_{1k}) , $j = 0, 1, \dots, NS_0 - 1$ e $k = 0, 1, \dots, NS_1 - 1$, resulting in a set of decision tables, which are organized on a *policy table*, as shown in Fig. 15. Each element of the policy table is a decision table optimized for a service requester with transition probabilities (R_{0j}, R_{1k}) , $j = 0, 1, \dots, NS_0 - 1$ and $k = 0, 1, \dots, NS_1 - 1$.

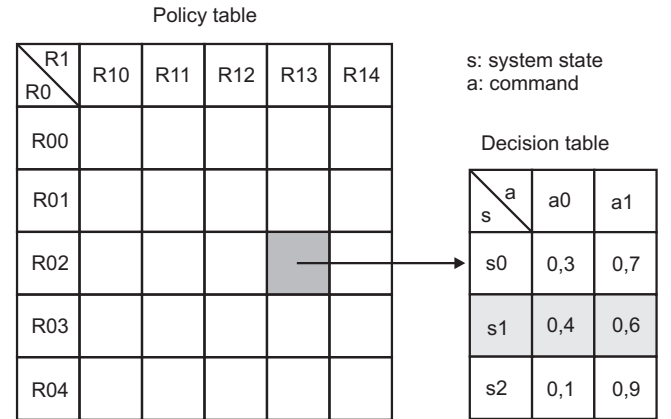


Figure 15 – Example of policy table with $NS_0 = NS_1 = 5$.

As shown in Fig. 11, the block *request windows*, evaluates estimates ($\hat{R}_0 = \hat{p}_{r_0, r_0}^{SR}(t, l_w)$, $\hat{R}_1 = \hat{p}_{r_1, r_1}^{SR}(t, l_w)$) of the transition probabilities of the service requester, and two consecutive indexes are chosen for each service requester state: $R_{0j} \leq \hat{R}_0 \leq R_{0(j+1)}$ and $R_{1k} \leq \hat{R}_1 \leq R_{1(k+1)}$. The decision table for the pair (\hat{R}_0, \hat{R}_1) is evaluated using the decision tables of (R_{0j}, R_{1k}) , $(R_{0j}, R_{1(k+1)})$, $(R_{0(j+1)}, R_{1k})$, $(R_{0(j+1)}, R_{1(k+1)})$. Each one of these four decision table has one line associated to the current system state, named CS. One technique of bi-dimensional interpolation is applied to these four lines to

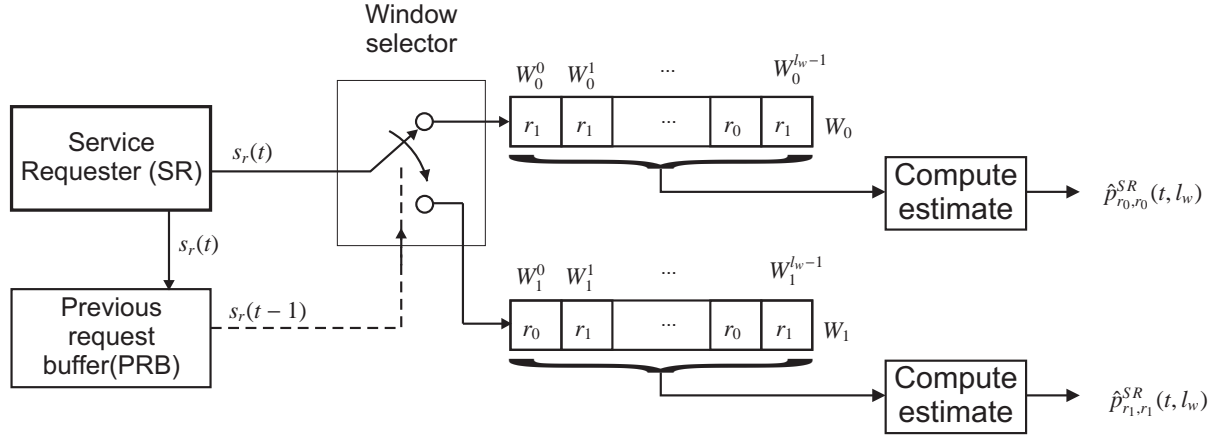


Figure 12 – Multiwindow for a service requester with two states.

evaluate the *final decision*. Let $f : \mathbb{R} \rightarrow \mathbb{R}$ in (26) be a one-dimensional interpolation function. To perform bi-dimensional interpolation, $f : \mathbb{R} \rightarrow \mathbb{R}$ in (26) is applied recursively. Extrapolation is applied whenever $\hat{R}_i > R_{i(N_S, i-1)}$ or $\hat{R}_i < R_{i0}$.

$$f(x) = \frac{(f(x_2) - f(x_1))x + x_2f(x_1) - x_1f(x_2)}{x_2 - x_1} \quad (26)$$

Each element of the *final decision* ($p_{a_0}, \dots, p_{a_{A-1}}$) is the probability of the power manager issuing a command $a \in A$ when the system is in state $x = (s, r, q)$. As shown in Fig. 11, the block *choice of the command* outputs the command a to be issued to the service provider. In this block, a D random number uniformly distributed in the interval $[0, 1]$ is chosen, and the command a_i is such that $\sum_{k=0}^{i-1} p_{a_k} < D \leq \sum_{k=0}^i p_{a_k}$ if $0 < D \leq 1$, or a_0 if $D = 0$, as shown in Fig. 16.

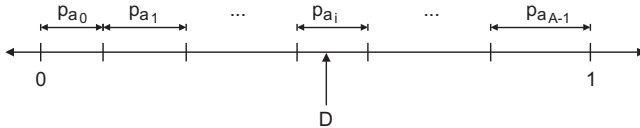


Figure 16 – Choosing power management commands.

3. IDENTIFICATION OF NONSTATIONARY WORKLOADS

The effects of sampling error and identification delay of a non-stationary workload of a battery powered device were analyzed by Luiz, Perkusich and Lima [13]. Simulations using a Markov chain model of an embedded system and an analytical model of the battery suggested that the optimal window size l_w of the block *request windows* as shown of Fig. 11 is the one which results in a minimum mean square error of the estimation of the workload parameters, thus increasing the battery lifetime with an acceptable performance penalty. In [14], Luiz, Perkusich and Lima applied stochastic learning-based weak estimation to identify non-stationary workloads. The identification delay and sampling

error effects found in stochastic learning-based weak estimation were formally stated and analytical expressions were derived. This technique outperformed previous approaches in estimation accuracy, performance and power consumption. Furthermore, the stochastic learning-based weak estimation requires less computation effort than fixed size sliding windows.

The dynamic power management (DPM) technique using stochastic learning-based weak estimation can be described with the help of the block diagram shown in Fig. 17. The proposed technique has been designed for nonstationary environments as described in the following. The power manager (PM) estimates the workload of the system and chooses a command to issue to the service provider. To do this, the power manager uses stochastic learning-based weak estimation, 1, to evaluate *estimates* of the transition probabilities of the service requester. A *policy table*, 2, contains decision tables generated at design time by the policy optimization problem (OP). Thus, the *policy interpolator*, 3, introduced in [2], uses the states of the service requester, 5, service queue, 6, service provider, 7, and the *estimates* for the interpolation of the decision tables. The result is a *decision* which is used for the *choice of the command*, 4, to be issued to the service provider. In what follows, the stochastic learning-based weak estimation for workload identification is analyzed.

3.1. Stochastic learning-based weak estimation

In the block *stochastic learning-based weak estimation* shown in Fig. 17, the estimates of the transition probabilities of the service requester are evaluated. Let $\hat{P}_{r_i, r_j}^{SR}(t, \lambda)$ be the estimate of the transition probability from state r_i to state r_j with parameter $0 < \lambda < 1$. An example is shown in Fig. 18 for a service requester with two states, where only $\hat{P}_{r_0, r_0}^{SR}(t, \lambda)$ and $\hat{P}_{r_1, r_1}^{SR}(t, \lambda)$ are evaluated because the others can be calculated whenever they are necessary: $\hat{P}_{r_0, r_1}^{SR}(t, \lambda) = 1 - \hat{P}_{r_0, r_0}^{SR}(t, \lambda)$ and $\hat{P}_{r_1, r_0}^{SR}(t, \lambda) = 1 - \hat{P}_{r_1, r_1}^{SR}(t, \lambda)$. The value of the estimate $\hat{P}_{r_i, r_j}^{SR}(t, \lambda)$ is only updated when the input $s_r(t-1)$ to the block *Compute estimate* equals r_i , that is, when the state of the ser-

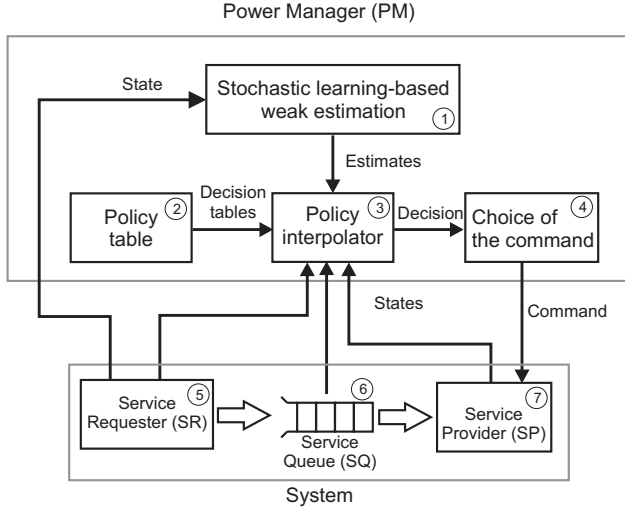


Figure 17 – Stochastic learning system workload estimation for nonstationary environments.

vice requester $s_r(t-1) = r_i$, as shown in Fig. 13. Otherwise, the block *Compute estimate* as shown in Fig. 18 outputs its previous evaluated estimate $\hat{p}_{r_i, r_j}^{SR}(t-1, \lambda)$.

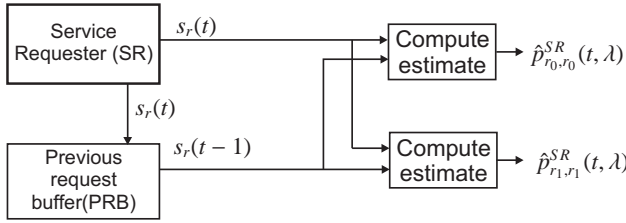


Figure 18 – Block *stochastic learning-based weak estimation* for a service requester with two states.

3.1.1 Statistical properties of the estimates

The estimates of the transition probabilities of the service requester (SR) Markov chain model are evaluated by the block *Compute estimate* in Fig. 18 by means of stochastic learning-based weak estimation [15]. The statistical properties of this estimation approach are explained and mathematically analyzed here. Let “=” be the equivalence operation, i.e., the result is “1” if the arguments are equal, and “0” otherwise. And let \wedge be the logical AND. Let $X(t) = (s_r(t-1) == r_i) \wedge (s_r(t) == r_j)$ be a random variable identifying whether the state of the service requester (SR) is r_i at time $t-1$, and r_j at time t . When $X(t) = 1$, there was a transition from state r_i to state r_j . Thus, X is a discrete random variable with sample space $\{0, 1\}$ with probabilities $P(X = 1) = p_{r_i, r_j}^{SR}(t)$ and $P(X = 0) = 1 - p_{r_i, r_j}^{SR}(t)$. The expected value $E(X)$ and the variance $V(X)$ are presented in (27) and (3.1.1) respectively.

$$E(X) = \sum_{i=1}^2 x_i P(X = x_i) = p_{r_i, r_j}^{SR}(t) \quad (27)$$

$$V(X) = E(X - E(X))^2 = p_{r_i, r_j}^{SR}(t) - [p_{r_i, r_j}^{SR}(t)]^2$$

The operation performed in the block *Compute estimate* (Fig. 18) is shown in the Algorithm (1) (whose version for binomial distributions was first introduced in [15]). In this algorithm, $\hat{P}(X = 1)$ and $\hat{P}(X = 0)$ are internal variables. At $t = 0$, they are initialized to user defined values, e.g. $\hat{P}(X = 1) = 0.5$ and $\hat{P}(X = 0) = 0.5$, and are updated whenever $(s_r(t-1) == r_i)$.

Algorithm 1: Compute estimate.

Data: $\hat{P}(X = 1)$, $\hat{P}(X = 0)$ and λ

input : $s_r(t)$ and $s_r(t-1)$

output: $\hat{p}_{r_i, r_j}^{SR}(t, \lambda)$

if $(s_r(t-1) == r_i)$ **then**

if $(s_r(t) == r_j)$ **then**

$\hat{P}(X = 1) = 1 - \lambda \hat{P}(X = 0)$;

else

$\hat{P}(X = 1) = \lambda \hat{P}(X = 1)$;

$\hat{P}(X = 0) = 1 - \hat{P}(X = 1)$;

$\hat{p}_{r_i, r_j}^{SR}(t, \lambda) = \hat{P}(X = 1)$;

The estimate quality is assessed by the mean squared error (MSE). Let $e_{ij}(t) = \hat{p}_{r_i, r_j}^{SR}(t) - p_{r_i, r_j}^{SR}(t)$ be the estimated error of the probability $p_{r_i, r_j}^{SR}(t)$ of a service requester during time slices $t = 1 \dots m$. The mean squared error $MS E_{ij}$ of the estimate $\hat{p}_{r_i, r_j}^{SR}(t)$ with respect to $p_{r_i, r_j}^{SR}(t)$ is computed in (28).

$$MS E_{ij} = \frac{1}{m} \sum_{k=1}^m (e_{ij}(t_k))^2 = \frac{1}{m} \sum_{k=1}^m [\hat{p}_{r_i, r_j}^{SR}(t_k) - p_{r_i, r_j}^{SR}(t_k)]^2 \quad (28)$$

3.1.2 Identification delay

As shown in Fig. 19, when a transition from one stationary workload u_r to another u_s occurs at $t = t_s$, there is a period of time T_{p_i} necessary so that the estimates $\hat{p}_{r_i, r_j}^{SR}(t, \lambda)$ converge to the new workload u_s . This period of time is denoted the *identification delay* (named *adaptation time* in [2]). For the service requester state r_i , the identification delay is denoted T_{p_i} . Let T_p be the identification delay for all of the service requester states r_i , $i = 0, \dots, R-1$.

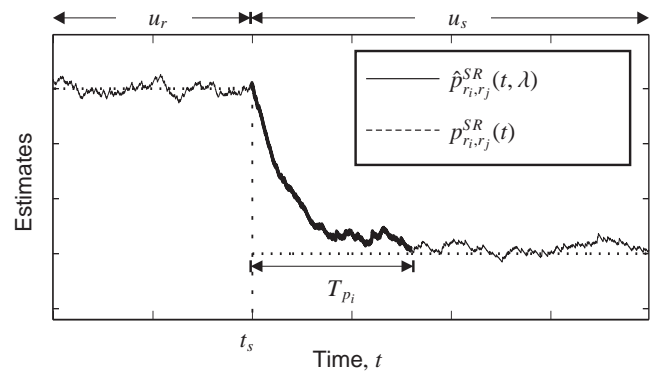


Figure 19 – The identification delay (bold line) during the transition from one stationary workload u_r to another u_s .

Theorem 1 If a transition from one stationary workload u_r to another u_s occurs at $t = t_s$, and the workload u_s remains for at least T_{p_i} time slices as shown in Fig. 19, then, for a given $0 < \epsilon < 1$, there exists a T_{p_i} such that the inequality in (29) holds, where p_{r_i, r_j}^{SR} is the constant value of $p_{r_i, r_j}^{SR}(t)$ during the workload u_s . The period of time T_{p_i} is named the identification delay for the service requester state r_i and satisfies (30), where $\pi_i \triangleq \lim_{t \rightarrow \infty} P[s_r(t) = r_i | u_s]$ is the probability that the service requester (SR) is in state r_i . And the identification delay T_p for all of the service requester states r_i , $i = 0, \dots, R-1$ is shown in (31).

$$|E[\hat{p}_{r_i, r_j}^{SR}(t_s + T_{p_i}, \lambda)] - p_{r_i, r_j}^{SR}| \leq \epsilon \quad (29)$$

$$T_{p_i} \geq \frac{\ln(\epsilon)}{\pi_i \ln(\lambda)} \quad (30)$$

$$T_p = \max_i \{T_{p_i}\} \geq \frac{\ln(\epsilon)}{\ln(\lambda)} \max_i \left\{ \frac{1}{\pi_i} \right\} \quad (31)$$

Proof 1 Since the value of the estimate $\hat{p}_{r_i, r_j}^{SR}(t, \lambda)$ is only updated when the state of the service requester $s_r(t-1) = r_i$, let us consider that the service requester state $s_r(t-1)$ is observed for T_{p_i} time slices and there are n occurrences of the state r_i as shown in Fig 20. Then, for sufficiently large T_{p_i} , we have $n/T_{p_i} \cong \pi_i \Rightarrow n \cong T_{p_i} \pi_i$.

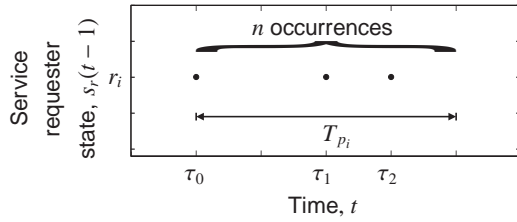


Figure 20 – Time slices when the value of the estimate $\hat{p}_{r_i, r_j}^{SR}(t, \lambda)$ is updated.

Now, let us consider the sequence $(\tau_0, \tau_1, \tau_2, \dots)$ of time slices $\tau_n = t_s + t$ such that $s_r(t_s + t - 1) = r_i$ for $n = 0, 1, 2, \dots$ as shown in Fig. 20, that is, the sequence of time slices when the value of the estimate $\hat{p}_{r_i, r_j}^{SR}(t_s + t, \lambda)$ is updated. We will compute now the necessary number n of time steps τ_n so that the estimates $\hat{p}_{r_i, r_j}^{SR}(\tau_n, \lambda)$ converge to the new workload u_s . From the Algorithm (1), the expected value of estimate $\hat{p}_{r_i, r_j}^{SR}(\tau_n, \lambda)$ given $\hat{p}_{r_i, r_j}^{SR}(\tau_{n-1}, \lambda) = \hat{P}(X = 1)$ is shown in (32). Taking expectations a second time, the expression in (32) is rewritten in (33) and extended to $E[\hat{p}_{r_i, r_j}^{SR}(\tau_0, \lambda)]$ in (34).

$$\begin{aligned} E[\hat{p}_{r_i, r_j}^{SR}(\tau_n, \lambda) | \hat{p}_{r_i, r_j}^{SR}(\tau_{n-1}, \lambda)] &= \\ &= p_{r_i, r_j}^{SR} (1 - \lambda (1 - \hat{p}_{r_i, r_j}^{SR}(\tau_{n-1}, \lambda))) + \\ &+ (1 - p_{r_i, r_j}^{SR}) \lambda \hat{p}_{r_i, r_j}^{SR}(\tau_{n-1}, \lambda) = \\ &= (1 - \lambda) p_{r_i, r_j}^{SR} + \lambda \hat{p}_{r_i, r_j}^{SR}(\tau_{n-1}, \lambda) \end{aligned} \quad (32)$$

$$E[\hat{p}_{r_i, r_j}^{SR}(\tau_n, \lambda)] = (1 - \lambda) p_{r_i, r_j}^{SR} + \lambda E[\hat{p}_{r_i, r_j}^{SR}(\tau_{n-1}, \lambda)] \quad (33)$$

$$= (1 - \lambda^n) p_{r_i, r_j}^{SR} + \lambda^n E[\hat{p}_{r_i, r_j}^{SR}(\tau_0, \lambda)] \quad (34)$$

The necessary number n of time steps τ_n so that the estimates $\hat{p}_{r_i, r_j}^{SR}(t, \lambda)$ converge to the new workload u_s must satisfy the inequality in (29). Using the expression for $E[\hat{p}_{r_i, r_j}^{SR}(\tau_n, \lambda)]$ in (34), this inequality is rewritten in (35). By considering that $|E[\hat{p}_{r_i, r_j}^{SR}(\tau_0, \lambda)] - p_{r_i, r_j}^{SR}(u_s)| \leq 1$, $0 < \epsilon < 1$ and $0 < \lambda < 1$ the inequality in (36) holds.

$$\lambda^n |E[\hat{p}_{r_i, r_j}^{SR}(\tau_0, \lambda)] - p_{r_i, r_j}^{SR}| \leq \epsilon \Rightarrow \quad (35)$$

$$\Rightarrow \lambda^n \leq \frac{\epsilon}{|E[\hat{p}_{r_i, r_j}^{SR}(\tau_0, \lambda)] - p_{r_i, r_j}^{SR}|} \Rightarrow$$

$$\Rightarrow \lambda^n \leq \epsilon \Rightarrow n \geq \frac{\ln(\epsilon)}{\ln(\lambda)} \quad (36)$$

Replacing $n \cong T_{p_i} \pi_i$ in (36), then the inequality (30) holds. And the identification delay for all of the service requester states T_p may be defined as the maximum of all of the identification delays T_{p_i} , $i = 0, \dots, R-1$, as expressed in (31).

When the parameter $0 < \lambda < 1$ increases, the identification delay T_p also increases. Therefore, to combat the identification delay, the parameter λ must be kept as small as possible.

3.1.3 Sampling error

After the identification delay from one stationary workload u_r to another u_s , as shown in Fig. 21, the estimates oscillate around the true transition probabilities values $p_{r_i, r_j}^{SR}(t)$. This effect is denoted the *sampling error* (named *resolution error* in [2]). The decrease in the parameter λ is beneficial to reduce the identification delay, but it increases the sampling error.

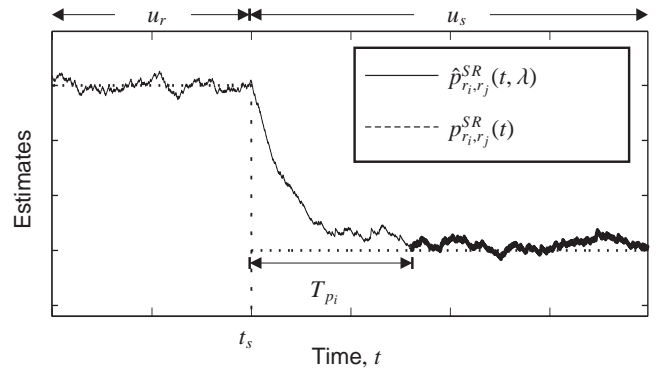


Figure 21 – The sampling error (bold line) after the transition from one stationary workload u_r to another u_s .

The variance of the estimate $\hat{p}_{r_i, r_j}^{SR}(t, \lambda)$, shown in (37), is a metric of the sampling error (the proof for this expression of the variance is shown in [15]). The variance $V(\hat{p}_{r_i, r_j}^{SR}(t, \lambda))$ is minimum for $\lambda = 1$ and maximum for $\lambda = 0$. Therefore, when λ increases, the sampling error decreases. But, as shown in Section 3.1.2, when λ increases, the identification delay T_p also increases. So, there is a trade-off between

sampling error and identification delay.

$$V(\hat{p}_{r_i, r_j}^{SR}(t, \lambda)) = \frac{(1 - \lambda) p_{r_i, r_j}^{SR}(t) (1 - p_{r_i, r_j}^{SR}(t))}{(1 + \lambda)} \quad (37)$$

In [16], the problem of power management under non-stationary workload conditions was rigorously analyzed and solved using a multi-size sliding window (MSW) technique which results in close to optimal power management policies. The estimates were computed taking into account either multiple windows with different sizes or multiple parameters. The introduced technique is able to adapt itself whether the workload of a real-life system is stationary or highly nonstationary. Furthermore, the introduced multi-size sliding window technique can be easily combined with any technique used to compute the estimates. In [16], it was combined either with maximum likelihood estimation or stochastic learning-based weak estimation.

4. BATTERY-AWARE POLICIES FOR NONSTATIONARY WORKLOADS

Optimization of the battery lifetime using power management strategies has become one of the key challenges in the design of mobile embedded systems. Battery-aware DPM strategies must take into account the workload, the electric parameters (e.g. values of currents) of the electronic system and the electrochemical features of the battery [1]. The approaches of [4] and [2] are both energy optimization techniques, what applies perfectly to a linear battery model, but may not optimize the battery lifetime at a real battery powered device, at which the rate-capacity characteristic and the relaxation effect are not negligible. Luiz et al. [17] proposed a battery-aware dynamic power management technique that exploits an accurate analytical battery model in a non-stationary environment. This technique is briefly described in this section.

In [18] an arbitrary *load profile* is defined as a series of N load “steps” $(I_1, \Delta_1), (I_2, \Delta_2), \dots, (I_N, \Delta_N)$, where I_k, Δ_k e t_k are the magnitude, duration and start time, respectively, of the k^{th} load step. The total duration of the load profile is $\Delta = \sum_{k=1}^N \Delta_k$. At [19], Rakhmatov and Vrudhula [19] provide a battery-aware cost function $\sigma(n)$ (see equation 38) to be minimized and thus maximize battery lifetime.

$$\sigma(n) = \sum_{k=0}^{n-1} I_k F(T, t_k, t_k + \Delta_k, \beta) \quad (38)$$

where

$$F(t, t_i, t_f, \beta) = t_f - t_i + 2 \sum_{m=1}^{\infty} \frac{e^{-\beta^2 m^2 (t-t_f)} - e^{-\beta^2 m^2 (t-t_i)}}{\beta^2 m^2} \quad (39)$$

Replacing function F in (38) with its expression at 39 we get (40).

$$\sigma(n) = \underbrace{\sum_{k=0}^{n-1} I_k \Delta_k}_{l(n)} + \underbrace{\sum_{k=0}^{n-1} I_k \left(2 \sum_{m=1}^{\infty} \frac{e^{-\beta^2 m^2 (t_n - t_k - \Delta_k)} - e^{-\beta^2 m^2 (t_n - t_k)}}{\beta^2 m^2} \right)}_{u(n)} \quad (40)$$

As pointed out in [18], the first term of (40) is the *actual charge lost*, and the second term is interpreted as the *unavailable charge*. Let α be the battery’s full charge capacity. Then $a(n) = \alpha - \sigma(n)$ is the *available charge* when using the Rakhmatov-Vrudhula battery model. And $a_L(n) = \alpha - l(n)$ is the *available charge* when using the linear battery model. Since $u(n) \geq 0, \forall t_n \geq 0, a(n) \leq a_L(n)$, i.e. the available charge of the Rakhmatov-Vrudhula model is less than or equal to the available charge of the linear model, as shown in (41).

$$\sigma(n) \geq l(n) \Leftrightarrow -\sigma(n) \leq -l(n) \Leftrightarrow \alpha - \sigma(n) \leq \alpha - l(n) \quad (41)$$

By including the Rakhmatov-Vrudhula battery model in the *policy optimization* problem (PO) in (8), it is possible to consider the battery non-linear effects. For this task, the cost function must be rewritten as (42) where the function F is defined in (39). The expected consumption is shown in (43), consumption vector is shown in (44). The system consumption is $\sum_{n=1}^{\infty} E_{\pi} [\bar{\mathbf{b}}_{\delta}(n)]$ and the battery lifetime optimization problem *PObat* is shown in (45).

$$b(s, a, t_k) = I(a, s, t_k) F(T, t_k, t_k + \Delta_k, \beta) \quad (42)$$

$$\bar{b}(x, \delta_x, t_k) = \sum_{p_a \in \delta_x} p_a b(s, a, t_k) \quad (43)$$

$$\bar{\mathbf{b}}_{\delta(k)} := \begin{pmatrix} \bar{b}(x_1, \delta_{x_1}, t_k) \\ \vdots \\ \bar{b}(x_X, \delta_{x_X}, t_k) \end{pmatrix} \quad (44)$$

$$\mathbf{PObat} : \min_{\pi} \sum_{t=0}^{\infty} E_{\pi} [\bar{\mathbf{b}}_{\delta}(t)] \quad \text{such that} \quad \sum_{t=0}^{\infty} E_{\pi} [\mathbf{d}_{\delta}(t)] \leq D \quad (45)$$

It is important to notice that the cost function $b(s, a, t_k)$ in *PObat* is time dependent because of the function $F(T, t_k, t_k + \Delta_k, \beta)$, which reproduces the battery non-linear effects. The cost function $c(s, a)$ in (8) is not time dependent and therefore PO2 is simplified to the linear program PL1 in (9), where there is no temporal dependency. Thus, the approach of Benini et al. [4] does not support the solution of the policy optimization problem with a time dependent cost function, such as the problem *PObat* in (45). In this section, it is presented an alternative para to increase the battery lifetime using the approach of Chung et al. [2].

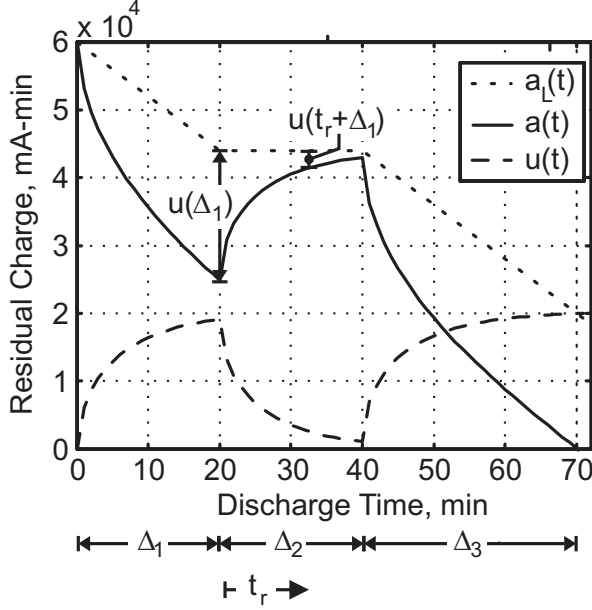


Figure 22 – Battery discharging process.

An example of a discharge process is shown in figure 22 for a battery of parameters $\alpha = 60000$ e $\beta = 0.35$. For the first 20 minutes, the discharge current was 800 mA. Then the current became zero during 20 minutes, and raised again to 800 mA until the battery was fully discharged. We notice at time interval Δ_2 (see figure 22) that $a(t)$ approaches $a_L(t)$ because of the recovery effect, since the current is zero at that time interval. One may see that the difference between the available charge of the linear model and the one of the Rakhmatov-Vrudhula model is equal to unavailable charge, as shown in equation 46.

$$a_L(n) - a(n) = \alpha - l(n) - \alpha + \sigma(n) = \sigma(n) - l(n) = u(n) \quad (46)$$

Thus, the recovery effect decreases $u(t)$. So we desire that when the current is zero, the idle time be as long as possible so that $u(t)$ approaches zero as much as possible. The *recovery time* $t_r(\varepsilon)$ of a load profile is defined at [18] as the additional time after the end of the load profile so that the unavailable charge reaches a fraction ε of its value at the end of the load profile. For example, considering the load profile (I_1, Δ_1) at figure 22, $t_r(\varepsilon = 0.25)$ is the additional time after Δ_1 so that $u(t_r + \Delta_1) = \varepsilon u(\Delta_1)$. For an arbitrary load profile, the recovery time is defined in (47).

$$\frac{u(t_r + \Delta)}{u(\Delta)} = \frac{\sum_{k=1}^N I_k F(t_r + \Delta, t_k, t_k + \Delta_k, \beta)}{\sum_{k=1}^N I_k F(\Delta, t_k, t_k + \Delta_k, \beta)} = \varepsilon \quad (47)$$

In spite of the difficulty to find a closed form expression for $t_r(\varepsilon)$ from (47), the upper bound reproduced in (48) may be obtained [18].

$$\bar{t}_r(\varepsilon) = -\frac{1}{\beta^2} \log(1.5\varepsilon), \forall \varepsilon < 0.4 \quad (48)$$

The power management approach in [2] may benefit from the battery model in [19] by using power policies optimized

with lower performance constraints at idle intervals, thus extending the intervals of time at which the battery “recovers”, which considerably increases the battery lifetime.

When a power managed system enters a low power state which demands zero current from the battery, the relaxation effect makes the battery “recover” available charge and thus the lifetime may be considerably extended. Luiz et al. [17] extended the DPM technique proposed at [2], lowering the performance of the system when it is at low power states, what increases the period of time the battery “recovers” and consequently its lifetime. The proposed technique is described for a service provider with two states, *on* and *off*, but it can be extended to handle a SP with more states by combining low power states which demand zero or negligible current from the battery, so that the proposed technique is applied whenever the SP enters one of these low power states.

When low performance is acceptable, and the SP is off, it may take more time to switch on, on the expense of more unserved requests waiting at the SQ. Let T be the time resolution. In the case of the discrete time Markov chain model of the SP, the expected transition time from state *off* to state *on* ($\bar{t}_{off,on}(s_{on})$) when the power manager issues a command to switch on (s_{on}) is inversely proportional to the probability of the SP transitioning from state *off* to state *on* ($p_{off,on}^{SP}(s_{on})$), as shown in equation 49.

$$\bar{t}_{off,on}(s_{on}) = \frac{T}{p_{off,on}^{SP}(s_{on})} \quad (49)$$

Thus, in general, for states of the system at which the service provider is off, the probability that the power manager issues a command to switch the SP on is smaller when the performance penalty constraint D in the optimization problem presented by [4] is greater.

If a battery powered system spends more time at low power states, it benefits from the relaxation effect and extends battery lifetime. The approach of Luiz et al. [17] uses: power policies optimized for low performance penalty constraints at high power states; and power policies optimized for high performance penalty constraints at low power states.

Consider a service provider such that, for a *on* state, a minimum performance penalty constraint $D_{min} = 0.2$ is desired. And for an *off* state, a maximum performance penalty constraint $D_{max} = 0.4$ is acceptable. Immediately after the service provider turns off, it is not desired that the power manager uses a power policy optimized for the high performance penalty constraint $D_{max} = 0.4$, because, if so, the system user will experience a high delay to turn on the system again. Thus, the transition between $D_{min} = 0.2$ to $D_{max} = 0.4$ must be performed smoothly.

An exponential function was used, and intervals of values which this function may assume were associated to values of D between D_{min} and D_{max} as shown in figure 23. The exponential function $f(t_{off})$ is shown in (50), where t_{off} is the time the SP remains at the off state, $\bar{t}_r(\varepsilon)$ is the upper bound for the recovery time [see (48)] of the battery used.

$$f(t_{off}) = 1 - e^{-t_{off}/\bar{t}_r(\varepsilon)} \quad (50)$$

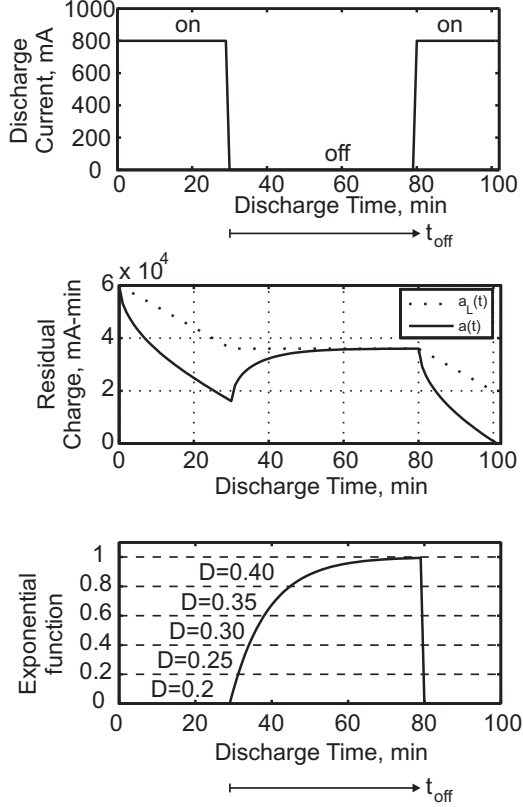


Figure 23 – Exponential function to vary performance penalty constraints when the SP turns off

Let N_D be the number of equally spaced performance penalty constraint values $D(i_D)$ [see (51)], where $i_D = 0, 1, 2, \dots, N_D$. The performance penalty constraint $D(i_D)$ value is chosen such that $(i_D - 1)/N_D \leq f(t_{off}) \leq i_D/N_D$.

$$D(i_D) = \frac{(i_D - 1)(D_{max} - D_{min})}{N_D - 1} + D_{min} \quad (51)$$

The system is represented by a discrete-time Markov chains coupled to the battery model. Such model allows a rigorous mathematical formulation of the problem and a trade-off between performance and battery lifetime. Experimental results have shown that the technique introduced results in longer battery lifetime compared to previous techniques.

Nevertheless, every dynamic power management technique based on stochastic context [2, 4, 13, 14, 16, 17] applies the optimization problem in (8). As shown in Example 2, this solution may satisfy the performance penalty constraints, but is not guaranteed to fully exploit them. Thus, there are still opportunities for power saving. In the next section, it is shown how to guarantee that the actual performance penalty of the system follows the performance penalty constraint by means of feedback control.

5. FEEDBACK CONTROL FOR MARKOV MODELED POWER-MANAGED SYSTEMS

Feedback control techniques for power management in general apply a Wscheme where: the reference value is a

reference performance value, the performance of the system is measured and compared to the reference performance value, resulting in the error of performance, which is the input to a controller, whose output is a command to change the power state of the power-managed system, as shown in Fig 24. Thus, the system performance is guaranteed to achieve the desired performance penalty constraint. Feedback control has been applied on power management in many different scenarios, such as: servers [5], real-time embedded systems [6], networks-on-chip [7], embedded microprocessors [8] and video playback [9].

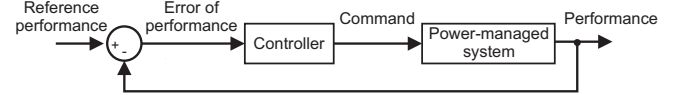


Figure 24 – Power management based on feedback control principle.

The dynamic power management (DPM) technique proposed in this work can be described with the help of the block diagram shown in Fig. 25. The 'controlled variable' is chosen to be the average request loss $l(t)$ of the system. The 'manipulated variable' is the state of the service provider s , which is changed by the means of the command a issues by the power manager. Since the system user accepts that the average request loss $l(t)$ of the system be degraded up to a certain user defined performance penalty constraint L , than the power manager can issue a command a to change the state s of the service provider and thus reduce the service rate $b(s, a)$. The 'setpoint' $l_C(t)$ is the performance penalty constraint L (such as D used for the *policy optimization* problem in (8)). According to the states of the service requester r , service queue q , and service provider s , there may be a request loss $l(x = (s, r, q))$, as shown in (21), which is used to estimate the performance of the system by means of the block *Estimate request loss*, whose output $\hat{l}(t)$ is subtracted from the setpoint $l_C(t)$ to create the error signal $e(t)$. A PI controller takes the error $e(t)$ to produce the control signal $u(t)$. But, according to 2, the control set \mathcal{A} of the service provider is discrete and finite. Thus, the *Quantizer* block is used to map the continuous control signal $u(t)$ into a discrete command $a(t)$. In what follows, a stochastic model for the feedback control for Markov modeled power-managed systems is analyzed.

5.1. The power-managed system

For systems like processors with dynamic voltage and frequency scaling, the switching time between the dynamic voltage and frequency scaling power states may be considered negligibly less than the period T of the power manager, then the switching costs (energy and time overheads) can be neglected. Thus, the design of the service provider Markov chain model of such systems is based on two assumptions: (i) the consumption $c(s, a)$ and service rate $b(s, a)$ only depend on the current service provider state s because the energy and time overheads for transitions are negligible; (ii) the transitions between power states are considered deterministic, that is, each element $p_{s_i, s_j}^{SP}(a_i)$ of the transition matrix $\mathbf{P}^{SP}(a)$ is

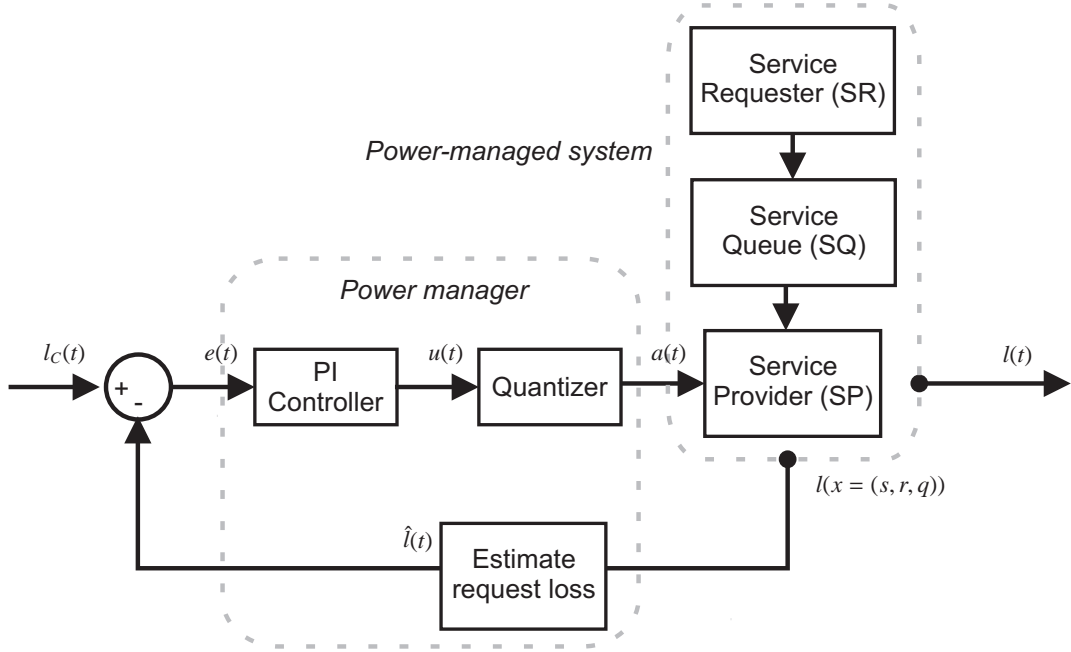


Figure 25 – Block diagram of the feedback policy for power management.

evaluated as in (52), where the command a_l means to “switch to s_l ”.

Let us reorder the service provider states in decreasing order of power consumption, that is, $c(s_0) > c(s_1) > \dots > c(s_{S-1})$. This decreasing order of power consumption corresponds to an decreasing order of service rate $b(s_0) > b(s_1) > \dots > b(s_{S-1})$ and an increasing order of performance penalty $1 - b(s_0) < 1 - b(s_1) < \dots < 1 - b(s_{S-1})$. The reasoning for this is the fact, as shown in Fig. 25, the estimate of request loss $\hat{l}(t)$ is subtracted from the setpoint $l_C(t)$ to create the error signal $e(t) = l_C(t) - \hat{l}(t)$. If $\hat{l}(t) > l_C(t) \Rightarrow e(t) < 0$, i.e. the current estimated request loss is greater than the performance penalty constraint, then the power manager must issue a command a so that the service provider enters a state with less performance penalty $1 - b(s)$. Accordingly, if $\hat{l}(t) < l_C(t) \Rightarrow e(t) > 0$, i.e. the current estimated request loss is less than the performance penalty constraint, then the power manager must issue a command a so that the service provider enters a state with more performance penalty $1 - b(s)$.

$$p_{s_i, s_j}^{SP}(a_l) = \begin{cases} 1, & \text{if } s_j = s_l \\ 0, & \text{otherwise} \end{cases} \quad (52)$$

The expressions derived in the previous section for the expected consumption and expected performance penalty of the service provider (SP) at time slice t were based on the fact that, using a stochastic policy π , the command C to be issued by the power manager (PM) is a *random variable* with discrete probability distribution $\delta_x = (p_{a_0}, p_{a_1}, \dots, p_{a_{A-1}})$ which depends on the system state x . In the case of the feedback control technique shown in Fig. 25, the command $a(t)$ is *known*. Let $p(t)$ be the probability distribution of the system states $\mathcal{X} = \{x_0, \dots, x_{X-1}\}$ at time t , let $P(a)$ be the system transition matrix when command $a \in \mathcal{X}$ is issued by the power manager, and let $l(t)$ be the request loss at time

t , which is either 0 or 1, as defined in (21). Let l be the performance penalty vector. Then the evolution of the probability distribution of the system states and the performance penalty is as shown in (53). Since the performance penalty of interest is *request loss*, l in (54) contains the expected request loss associated to each system state x . Considering the request loss $l(x = (s, r, q))$ in (21), the expected request loss associated to each system state x is $E[l(x = (s, r, q))] = 1 \times (1 - b(s)) + 0 \times b(s) = 1 - b(s)$.

$$\begin{aligned} p(t+1) &= p(t)P(a) \\ E[l(t)] &= p(t)l \end{aligned} \quad (53)$$

$$l = \begin{bmatrix} E[l(x_0)] \\ E[l(x_1)] \\ \vdots \\ E[l(x_{X-1})] \end{bmatrix} \quad (54)$$

5.2. The Quantizer block

The *Quantizer* block as shown in Fig. 25 associates values of the control signal $u(t)$ to commands $a(t)$ according to the function $f_Q(u(t))$ in (55) and as shown in Fig. 26. Then, $a(t) = f_Q(1 - b_{s_i}) = a_i$, for $i = 0, \dots, X - 1$. If $u(t) = 1 - b_{s_i}$ is kept constant for sufficient time, then $\lim_{t \rightarrow \infty} s(t) = s_i$ and $\lim_{t \rightarrow \infty} l(t) = 1 - b_{s_i}$ as shown in Fig. 27. Thus, the quantizer is designed so that the static gain is unity. Also, if $1 - b(s_i, a_i) < u(t) < 1 - b(s_{i+1}, a_{i+1})$ the least expected request loss is $1 - b(s_i, a_i)$.

$$a(t) = f_Q(u(t)) = \begin{cases} a_0, & \text{if } u(t) < 1 - b(s_1) \\ a_1, & \text{if } 1 - b(s_1) \leq u(t) < 1 - b(s_2) \\ \vdots & \\ a_{X-1}, & \text{if } u(t) \geq 1 - b(s_{X-1}, a_{X-1}) \end{cases} \quad (55)$$

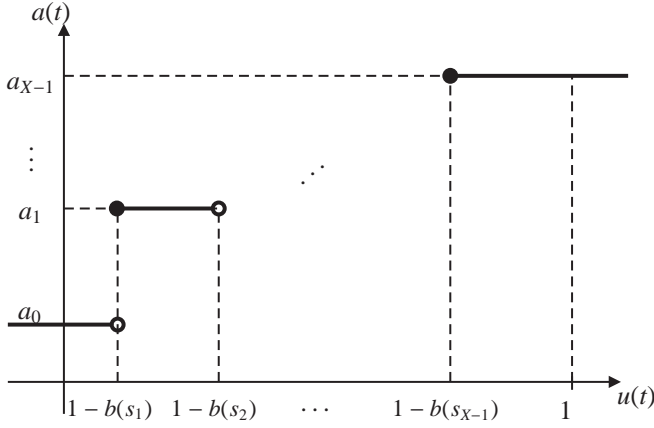


Figure 26 – The command quantizer.

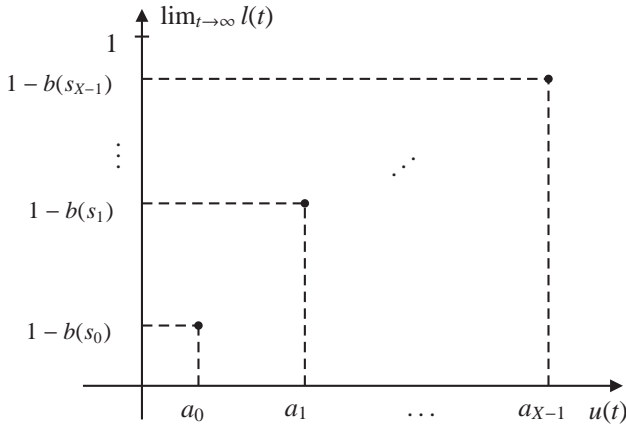


Figure 27 – The expected request loss.

5.3. The Estimate request loss block

As shown in Fig. 25, the output of the power-managed system is the request loss $l(t)$, but the goal of power management is that the expected request loss $E[l(t)]$ must satisfy the performance penalty constraint, which, in the formulation of the control problem is represented by the control signal $l_C(t)$. Therefore, the *Estimate request loss* block observes the current request loss $l(t)$ and outputs an estimate $\hat{l}(t)$ to be compared to the performance penalty constraint $l_C(t)$. Since we are interested in the expected value of $l(t)$, one possible approach is Stochastic learning-based weak estimation [15].

The operation performed in the *Estimate request loss* block (Fig. 25) is depicted in the Algorithm (2) (whose version for binomial distributions was first introduced in [15]). In this algorithm, the parameter $0 < \lambda < 1$, $\hat{P}(X = 1)$ and

$\hat{P}(X = 0)$ are internal variables. At $t = 0$, they are initialized to user defined values, e.g. $\hat{P}(X = 1) = l_C(0)$ and $\hat{P}(X = 0) = 1 - l_C(0)$, and are updated every time t .

Algorithm 2: Estimate request loss.

Data: $\hat{P}(X = 1)$, $\hat{P}(X = 0)$ and λ

input : $l(t)$

output: $\hat{l}(t)$

if $l(t) \neq 0$ **then**

$\hat{P}(X = 0) = \lambda \hat{P}(X = 0)$;

else

$\hat{P}(X = 0) = 1 - \lambda \hat{P}(X = 1)$;

$\hat{P}(X = 1) = 1 - \hat{P}(X = 0)$;

$\hat{l}(t) = \hat{P}(X = 1)$;

The decrease in the parameter λ is beneficial to reduce the identification delay, but it increases the sampling error. According to [15], one interesting interval for this parameter is $0.9 \leq \lambda < 1$. In this work the value used is $\lambda = 0.999$.

5.4. The discrete-time system model

As shown in Fig. 25, the interfaces of the controller with the process it controls are the error signal $e(t)$ and control signal $u(t)$. The controller knows neither the internal structure of the power managed system nor its Markov chain model. Therefore, *service requester*, *service queue*, *service provider*, *Quantizer* and *Estimate request loss* blocks may be represented as a unique entity, named the *process*. Thus, to design the controller transfer function $G_C(z)$, it is necessary to identify the process transfer function $G_P(z)$, as shown in Fig. 28. Unfortunately, the stochastic model in (53) is not useful for this task because of several reasons: (i) it is neither in the form of difference equation nor in state-space form; (ii) the input a is a parameter of the system transition matrix, thus it is not additive to the system states probability distribution $p(t)$; (iii) the instantaneous value of the expected request loss $E[l(t)]$ cannot be directly measured because it is an expected value. But the process transfer function $G_P(z)$ can be estimated using some model structure [20] such as ARX in (56) or ARMAX in (57), where $A(q)$, $B(q)$ and $C(q)$ are the polynomials defined in (58), (59) and (60) respectively, and $e_C(t)$ is the equation error model.

$$A(q)\hat{l}(t) = B(q)u(t) + e_C(t) \quad (56)$$

$$A(q)\hat{l}(t) = B(q)u(t) + C(q)e_C(t) \quad (57)$$

$$A(q) = 1 + a_1q^{-1} + \dots + a_{n_a}q^{-n_a} \quad (58)$$

$$B(q) = 1 + b_1q^{-1} + \dots + b_{n_b}q^{-n_b} \quad (59)$$

$$C(q) = 1 + c_1q^{-1} + \dots + c_{n_c}q^{-n_c} \quad (60)$$

The process transfer function $G_P(z)$ identification procedure must follow the steps [20]: (i) design an experiment to collect input $u(t)$ and output $\hat{l}(t)$ data; (ii) chose a model set (ARX, ARMAX, ...); (iii) chose the best model in the set according to a given criterion.

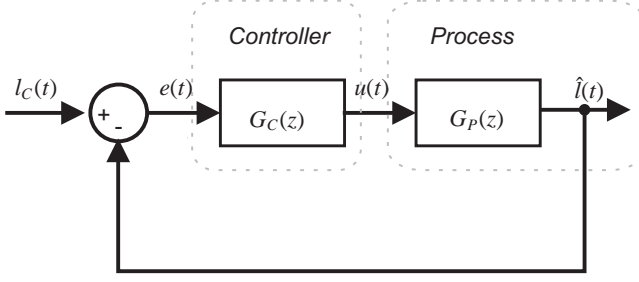


Figure 28 – Block diagram of the proposed power management technique.

5.5. The controller design

Let us suppose that the process transfer function $G_P(z)$ as obtained by the identification procedure described in the previous section has the form in (61). Let us use a PI controller for this process with transfer function in (62), where k_P and k_I are the proportional and integral gains respectively. Then, the closed-loop transfer function is as in (63). Let $\alpha \pm \beta i$ be the desired closed-loop poles, then the characteristic polynomial of the closed-loop system is of the form $(z - \alpha - \beta i)(z - \alpha + \beta i) = z^2 + 2\alpha z + (\alpha^2 + \beta^2)$. Based on the principle of pole placement, the parameters of the PI controller are evaluated in (64) and (64), resulting in the solutions in 66 and 66.

$$G_P(z) = \frac{b_1}{z + a_1} \quad (61)$$

$$G_C(z) = \frac{(k_P + k_I)z - k_P}{z - 1} \quad (62)$$

$$G(z) = \frac{G_C(z)G_P(z)}{1 + G_C(z)G_P(z)} = \quad (63)$$

$$= \frac{b_1(k_P + k_I)z - b_1k_P}{z^2 + [a_1 - 1 + b_1(k_P + k_I)]z - (a_1 + b_1k_P)}$$

$$a_1 - 1 + b_1(k_P + k_I) = 2\alpha \quad (64)$$

$$-(a_1 + b_1k_P) = \alpha^2 + \beta^2 \quad (65)$$

$$k_I = \frac{2\alpha - a_1 + 1}{b_1} \quad (66)$$

$$k_P = -\frac{a_1 + \alpha^2 + \beta^2}{b_1} \quad (67)$$

Example 4 To choose α and β , it is possible to use e.g. dead-beat design [21], where the desired poles $\alpha \pm \beta i$ are all allocated at the origin, resulting in the characteristic polynomial of the closed-loop system as $P(z) = z^n$, where n is the order of the characteristic polynomial of the closed-loop system, and, in the present case, is 2. Then, using $\alpha = 0$ and $\beta = 0$ the parameters of the PI controller are evaluated in (68) and (69).

△

$$k_I = -\frac{a_1 - 1}{b_1} \quad (68)$$

$$k_P = -\frac{a_1}{b_1} \quad (69)$$

6. RESULTS AND DISCUSSION

In this section we compare two different power management techniques:

1. the stochastic approach [4] using Markov decision processes
2. the proposed technique of feedback control for Markov modeled power-managed systems

Consider a processor with dynamic voltage and frequency scaling whose power states are shown in Table 1. In what follows, frequencies f and voltages v will be normalized. For example, the normalized frequency f_n of state s_1 is the rate of the actual speed 168 Mhz to the full frequency 192 Mhz. According to [22], the dynamic (switching) power P of CMOS circuits is proportional to fv^2 . Thus the dynamic power of the power states of Table 1 can also be normalized, as shown in Table 2. The power manager is triggered every $T = 1$ s to control the power state of the processor. Since the switching time between the dynamic voltage and frequency scaling power states is negligibly less than the period T of the power manager, then the switching costs (energy and time overheads) are neglected.

Table 1 – Power states

Name	Frequency, f (MHz)	Voltage, v (V)
s_0	192	1.5
s_1	168	1.5
s_2	84	1.5
s_3	84	1.1
s_4	60	1.5
s_5	60	1.1

Table 2 – Normalized power states

Name	Normalized frequency, f_n	Normalized voltage, v_n	Normalized power, $f_n v_n^2$
s_0	1	1	1
s_1	0.875	1	0.875
s_2	0.4375	1	0.4375
s_3	0.4375	0.733	0.235
s_4	0.3125	1	0.3125
s_5	0.3125	0.733	0.168

The model for the processor is a service provider (SP) ($\mathcal{M}_{SP}(a), b(s, a), c(s, a)$) where: $\mathcal{M}_{SP}(a)$ is a stationary controllable Markov chain with state set $\mathcal{S} = \{s_0, s_1, \dots, s_5\}$; control set $\mathcal{A} = \{a_0, a_1, \dots, a_5\}$, where a_i means switch to s_i , $i = 0, 1, \dots, 5$. Since the switching costs (energy and time overheads) are neglected, the transitions between power states are considered deterministic. Each element $p_{s_i, s_j}^{SP}(a_i)$ of the transition matrix $\mathbf{P}^{SP}(a)$ is evaluated as in (70); the power consumption $c(s, a)$ is the normalized power $f_n v_n^2$ of the state s , and the service rate $b(s, a)$ is the normalized frequency of the state s , as shown in Table 2.

$$p_{s_i, s_j}^{SP}(a_i) = \begin{cases} 1, & \text{if } s_j = s_i \\ 0, & \text{otherwise} \end{cases} \quad (70)$$

The service queue (SQ) $M_{SQ}(a, s, r)$ of length $Q = 1$ has state set $\mathcal{Q} = \{q_0, q_1\}$, i.e., in q_0 the service queue is empty, and in q_1 the service queue has one pending service request.

Consider a service requester (SR) ($M_{SR}, z(r)$) with state set $\mathcal{R} = \{r_0, r_1\}$ with function $z(r)$ identifying the number of requests that the service requester issues per time slice when in state r : $z(r_0) = 0$, $z(r_1) = 1$. The state r_0 represents an idle state of the service requester, e.g. when neither the operating system kernel is running nor an active process needs the processor. In the Linux scheduler, e.g., this situation takes place when the *idle process* is scheduled [23]

It is important to notice that the power states s_2 and s_4 have the same frequencies as s_3 and s_5 respectively. But, in spite of having the same frequencies, they have different voltages. Thus, when using either s_2 and s_4 , it is interesting to lower voltage as soon as possible. The block *Quantizer* as shown in Fig. 25 has the function $f_Q(u(t))$ in (71), where $1 - b(s_2)$ and $1 - b(s_4)$ are made slightly less than $1 - b(s_3)$ and $1 - b(s_5)$ respectively, so that the states with lower voltage are entering just after their higher voltages counterparts.

$$a(t) = f_Q(u(t)) = \begin{cases} a_0, & \text{if } u(t) < 0.1250 \\ a_1, & \text{if } 0.1250 \leq u(t) < 0.5620 \\ a_2, & \text{if } 0.5620 \leq u(t) < 0.5625 \\ a_3, & \text{if } 0.5625 \leq u(t) < 0.6870 \\ a_4, & \text{if } 0.6870 \leq u(t) < 0.6875 \\ a_5, & \text{if } u(t) \geq 0.6875 \end{cases} \quad (71)$$

The *Estimate request loss* block (Fig. 25) applies the Algorithm (2) with parameter $\lambda = 0.999$.

Since all the elements of the *process* are specified, it is necessary to estimate the process transfer function $G_P(z)$. The experiment executed to collect the data was designed for the worst case workload as in (72), i.e., there is an incoming request every time t . The input $u(t)$ was designed to be representative of every command a in (73) as shown in Fig. 29.

$$P^{SR} = \begin{matrix} & r_0 & r_1 \\ r_0 & \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \end{matrix} \quad (72)$$

$$u(t) = \begin{cases} 0.6875, & \text{if } t \leq 10000 \\ 0.6870, & \text{if } 10000 < t \leq 20000 \\ 0.5625, & \text{if } 20000 < t \leq 30000 \\ 0.5620, & \text{if } 30000 < t \leq 40000 \\ 0.1250, & \text{if } 40000 < t \leq 50000 \\ 0, & \text{if } 50000 < t \leq 60000 \\ 0.6875, & \text{if } 60000 < t \leq 70000 \\ 0.6870, & \text{if } 70000 < t \leq 80000 \\ 0.5625, & \text{if } 80000 < t \leq 90000 \\ 0.5620, & \text{if } 90000 < t \leq 100000 \\ 0.1250, & \text{if } 100000 < t \leq 110000 \\ 0, & \text{if } 110000 < t \leq 120000 \end{cases} \quad (73)$$

The ARX and ARMAX model structures were used for the estimation procedure with orders of the polynomials in (58), (59) and (60) ranging from 1 to 11. The best fit was obtained for the ARX model structure $A(q)y(t) = B(q)u(t) + e(t)$ with $A(q)$ in (74) and $B(q)$ in (75). The process

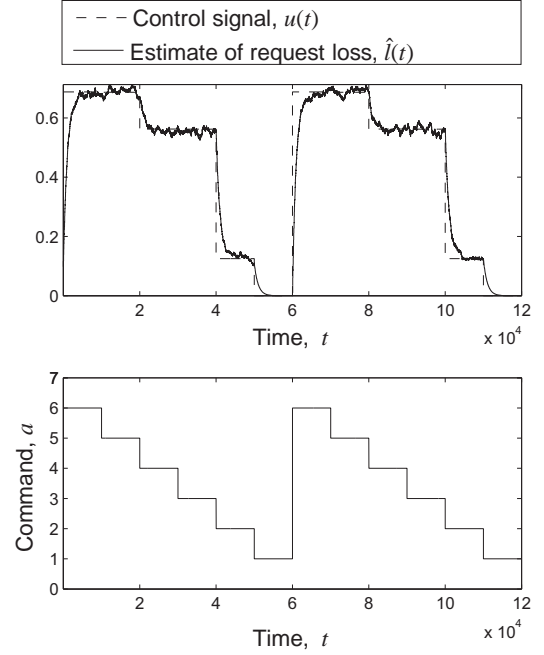


Figure 29 – Test for collecting input $u(t)$ and output $\hat{l}(t)$ data.

transfer function $G_P(z)$ is shown in (76). Using the deadbeat controller design in Example 4, the parameters of the PI controller are $k_P = 996.7289$ and $k_I = 997.7299$.

$$A(q) = 1 - 0.999q^{-1} \quad (74)$$

$$B(q) = 0.001002q^{-1} \quad (75)$$

$$G_P(z) = \frac{B(z)}{A(z)} = \frac{0.001002}{z - 0.999} \quad (76)$$

The performance penalty constraint is: maximum average request loss of $L = 10\%$. Thus the 'setpoint' $l_C(t) = 0.1$. The system model of this example is simulated for different stationary service requesters, with $p_{r_0, r_0}^{SR} = 0, 0.1, 0.2, \dots, 1.0$ and $p_{r_1, r_1}^{SR} = 0, 0.1, 0.2, \dots, 1.0$. Two metrics were evaluated to assess the performance of the power management approaches:

- P : average power consumption
- \hat{L} : average request loss

Using the stochastic approach [4], the power consumption and average request loss are shown in Fig. 30. For some workloads, the request loss is greater than the performance constraint, and the policies optimized for these workloads are not acceptable, as shown in Fig. 31. Among all of the transition probabilities of the service requester, the maximum average request loss is 0.5039.

Using the proposed technique of feedback control for Markov modeled power-managed systems, the power consumption and average request loss are shown in Fig. 32. For some workloads, the request loss is slightly greater than the performance constraint, as shown in Fig. 32, but, among all of the transition probabilities of the service requester, the

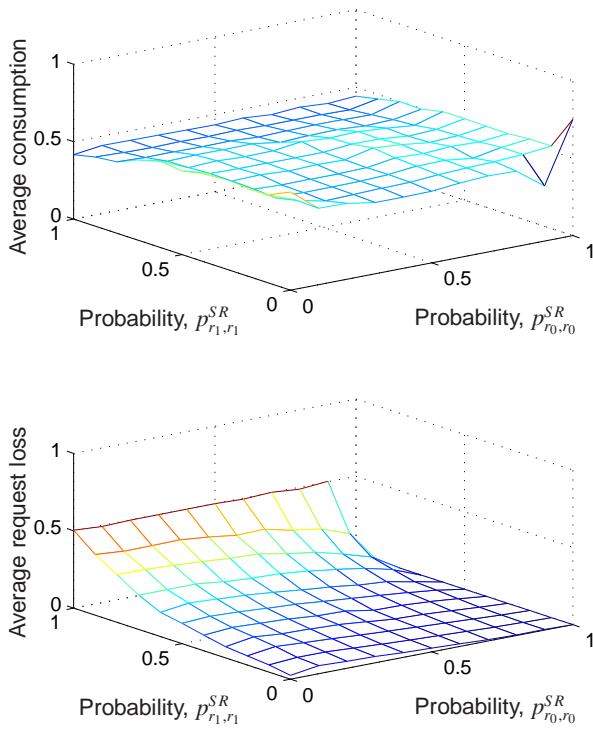


Figure 30 – Stochastic approach: power consumption and request losses of a power-managed system for different transition probabilities of the service requester (SR).

maximum average request loss is 0.1023. Depending on the power-managed system, this difference of the maximum average request loss from the performance constraint (10%) may be neglected or the 'setpoint' $l_C(t)$ can be changed to a value slightly below 0.1. For some soft workloads, the request loss is far less than the performance constraint, and it does not represent a limitation of the controller. Actually, it happens because these are soft workloads and the rate of requests per time slice is not enough to produce an average request loss equal to the performance penalty, even with the service provider in the state with lowest service rate.

The difference between the average power consumption of the stochastic control [4] and the feedback control is shown in Fig. 34. If we consider only the stochastic policies which satisfy the performance constraint, then some point are removed from the difference plot in Fig. 34 to produce the plot in Fig. 35. For soft workloads, feedback control offers less consumption than stochastic control. Besides, as shown in Fig. 31, stochastic control is not able to satisfy the performance constraint for a intensive workloads and does not fully exploit the opportunities for power saving for soft workloads. Thus feedback control guarantees that the actual performance penalty of the system follows the performance penalty constraint and fully exploits power saving.

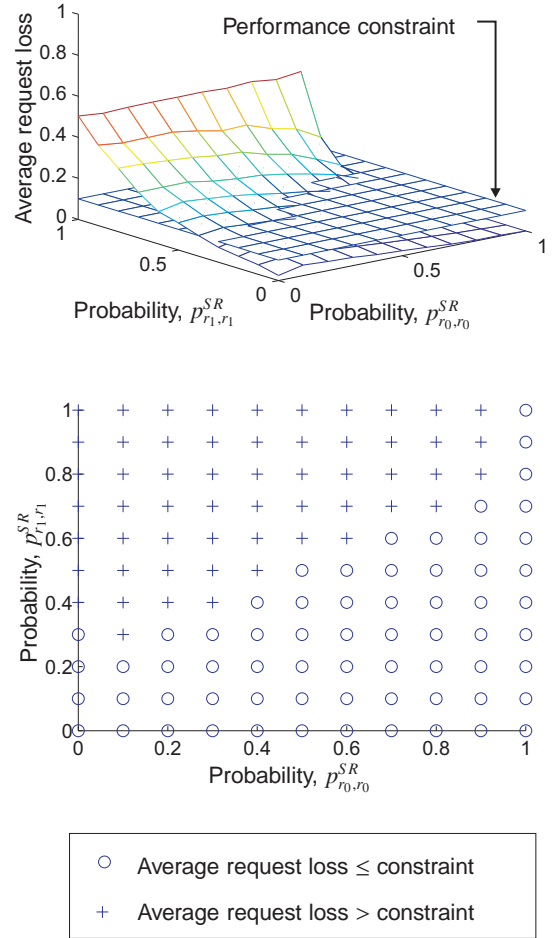


Figure 31 – Stochastic approach: policies which (do not) satisfy the performance constraint.

7. CONCLUSIONS AND FUTURE WORK

The stochastic behavior of power-managed systems is accurately modeled by means of controllable Markov chains. Using this model, policies can be derived from the solution of an optimization problem. But the effectiveness of stochastic approach for power management depends on the accurate identification of the system workload. In this work, an workload estimation technique using a stochastic learning-based weak estimation was described. The identification delay and sampling error effects found in stochastic learning-based weak estimation were formally stated and analytical expressions were derived.

The extension of battery lifetime through efficient energy utilization has become one of the key challenges to the design of embedded systems. We presented a battery-aware power management technique that benefits from the relaxation effect of batteries, trading off performance for battery lifetime at run time in a non-stationary environment.

In this work it was derived a discrete-time dynamic model from the stochastic model of the power-managed system. The combination of feedback control and the Markov chain model of the power-managed system, was effective, resulting in power policies which outperformed the previous stochas-

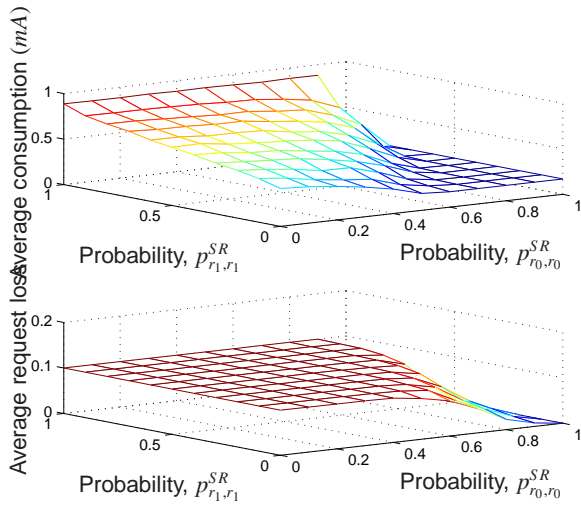


Figure 32 – Feedback control: power consumption and request losses of a power-managed system for different transition probabilities of the service requester (SR).

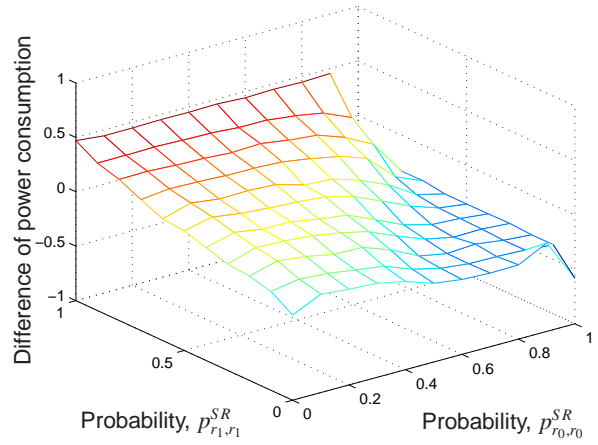
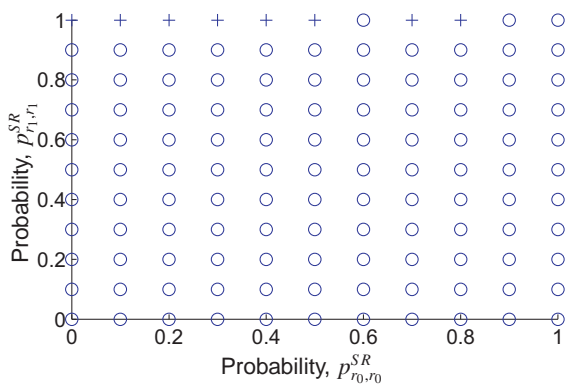
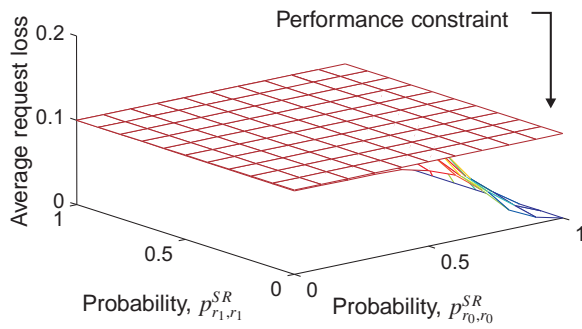


Figure 34 – Difference of power consumption. Negative values represent workloads for which feedback control offers less power consumption than stochastic control.



- Average request loss \leq constraint
- + Average request loss $>$ constraint

Figure 33 – Feedback control: policies which (do not) satisfy the performance constraint.

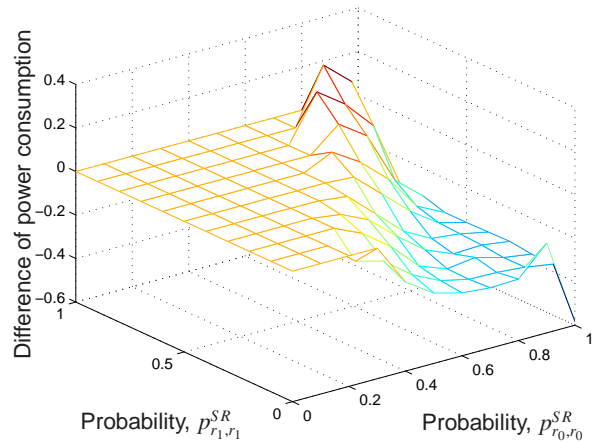


Figure 35 – Difference of power consumption for the acceptable stochastic policies. Negative values represent workloads for which feedback control offers less power consumption than stochastic control.

tic approaches. The integration of the discrete-time dynamic model of the controller with the power managed system was described, and a methodology to design the controller was presented.

By means of feedback control for power-managed systems, the design of the power manager is simpler than the previous stochastic approaches, and, at run time, power policy obtained takes considerably less time to execute. Besides, if a new performance penalty constraint is desired during run-time, the power manager can be adjusted automatically. In the case of the previous stochastic approaches, a new set of optimum policies must be computed. Thus, the proposed feedback control for power management is an adaptive power policy.

As future work, we are planning to use workload estimation techniques to update online the stochastic model of the power-managed system and apply adaptive control to adjust the controller parameters.

ACKNOWLEDGMENTS

The authors would like to thank all the members of the Embedded Systems and Pervasive Computing Laboratory of the Universidade Federal de Campina Grande. This work was partially supported by Universidade Federal de Campina Grande (UFCG) Electrical Engineering Graduate Program (Programa de Pós-Graduação em Engenharia Elétrica - PPgEE), CNPq and CAPES.

REFERENCES

- [1] P. Rong and M. Pedram, "Battery-aware power management based on markovian decision processes," in *Conf. Rec. ICCAD '02*. New York, NY, USA: ACM Press, 2002, pp. 707–713.
- [2] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. D. Micheli, "Dynamic power management for non-stationary service requests," *IEEE Trans. Comput.*, vol. 51, no. 11, pp. 1345–1361, 2002.
- [3] Y.-H. Lu and G. D. Micheli, "Comparing system-level power management policies," *IEEE Design & Test of Computers*, vol. 18, no. 2, pp. 10–19, 2001.
- [4] L. Benini, A. Bogliolo, G. Paleologo, and G. De Micheli, "Policy optimization for dynamic power management," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 6, pp. 813–833, June 1999.
- [5] Z. Wang, X. Zhu, C. McCarthy, P. Ranganathan, and V. Talwar, "Feedback control algorithms for power management of servers," in *Conf. Rec. FeBid'08*, Jun. 2008.
- [6] R. Sridharan, N. Gupta, and R. Mahapatra, "Feedback-controlled reliability-aware power management for real-time embedded systems," in *Conf. Rec. DAC'08*. New York, NY, USA: ACM, 2008, pp. 185–190.
- [7] U. Y. Ogras, R. Marculescu, and D. Marculescu, "Variation-adaptive feedback control for networks-on-chip with multiple clock domains," in *Conf. Rec. DAC'08*. New York, NY, USA: ACM, Jun 2008, pp. 614–619.
- [8] Y.-C. Tian, F. Xia, Y. Sun, and J. Dong, "Control-theoretic dynamic voltage scaling for embedded controllers," *Computers & Digital Techniques, IET*, vol. 2, no. 5, pp. 377–385, Sep. 2008.
- [9] Z. Lu, J. Lach, M. Stan, and K. Skadron, "Reducing multimedia decode power using feedback control," *Conf. Rec. ICCD'03*, pp. 489–496, Oct. 2003.
- [10] R. J. Minerick, V. W. Freeh, and P. M. Kogge, "Dynamic power management using feedback," in *Conf. Rec. COLP'02*, Sep. 2002, pp. 6–1–6–10.
- [11] J. Czyzyk, S. Mehrotra, M. Wagner, and S. J. Wright, "Pcx user guide (version 1.1)," Nov. 1997. [Online]. Available: <http://www-unix.mcs.anl.gov>
- [12] I. The MathWorks, "linprog," 2008. [Online]. Available: <http://www.mathworks.com>
- [13] S. O. D. Luiz, A. Perkusich, and A. M. N. Lima, "Workload estimation for power management," in *Conf. Rec. ICCE'09*. Las Vegas: IEEE Consumer Electronics Society, Jan 2009, pp. 7.2–4.
- [14] —, "Stochastic learning-based weak estimation for dynamic power management," in *Conf. Rec. IECON'09*, Porto, Portugal, 2009, not published.
- [15] B. J. Oommen and L. Rueda, "Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments," *Pattern Recognition*, vol. 39, no. 3, pp. 328 – 341, 2006. [Online]. Available: <http://www.sciencedirect.com>
- [16] S. O. D. Luiz, A. Perkusich, and A. M. N. Lima, "Multi-size sliding window in workload estimation for dynamic power management," *IEEE Trans. Comput.*, not published.
- [17] S. O. D. Luiz, A. Perkusich, A. M. N. Lima, and K. Gorgônio, "Técnica de gerenciamento dinâmico de energia orientada à autonomia da bateria para sistemas embarcados," in *Conf. Rec. CBA'08*. Juiz de Fora, MG, Brasil: Anais do CBA 2008, 2008.
- [18] R. Rao and S. Vrudhula, "Battery optimization vs energy optimization: which to choose and when?" in *Conf. Rec. ICCAD '05*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 439–445.
- [19] D. Rakhmatov and S. Vrudhula, "Energy management for battery-powered embedded systems," *ACM Transactions on Embedded Computing Systems*, vol. 2, no. 3, pp. 277–324, 2003.
- [20] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice Hall PTR, December 1998, pp. 14–15, 81–88.
- [21] K. J. Åström and B. Wittenmark, *Computer-Controlled*

Systems: Theory and Design, 3rd ed. New Jersey: Prentice Hall, Jan. 1997, pp. 131–132.

- [22] F. Shearer, *Power Management in Mobile Devices*, 1st ed. Burlington, USA: Newnes, 2008, pp. 44, 89–93, 132–135.
- [23] L. Benini, A. Bogliolo, S. Cavallucci, and B. Ricco;, “Monitoring system activity for os-directed dynamic power management,” in *Conf. Rec. ISLPED’98*. New York, NY, USA: ACM Press, 1998, pp. 185–190.