# Wavelet-Based Dynamic Power Management for Non-stationary Service Requests

A. Abbasian[1], S. Hatami[1], A. Afzali-Kusha[1], and M. Pedram[2]

[1]*Low-Power High-Performance Nanosystems, Nanoelectronics Center of Excellence, University of Tehran*
*North Kargar Ave., P.O. Box 14395/515, Tehran, Iran*
[2]*University of Southern California, Department of Electrical Engineering – Systems*
*Los Angeles, CA, U.S.A.*

a.abbasian@ece.ut.ac.ir, s.hatami@ece.ut.ac.ir, afzali@ut.ac.ir, pedram@usc.edu

## ABSTRACT

***Abstract -*** *In this paper, a wavelet-based dynamic power management policy (WBDPM) is proposed. In this approach, the workload source (service requester) is modeled by a non-stationary time series, which is in turn represented by a non-decimated Haar wavelet as its basis. The proposed approach is robust and has the ability to minimize the energy dissipation under different performance constraints. To assess the accuracy of the model, the algorithm was implemented for the data extracted from the hard disks of computers. Prediction results of this approach for the case of a non-stationary service requester exhibit accuracies of more than 95%.*



**Categories and Subject Descriptors**: D.4.8 Performance, (*Modeling and prediction Stochastic analysis)*

**General Terms**: Design

**Additional Key Words and Phrases**: Dynamic Power Management, Wavelet based prediction, non-stationary service request. Low Power system design


## I. INTRODUCTION

Battery life times in portable systems can be prolonged in two ways: increasing the battery capacity (energy per unit weight) and reducing the power consumption of microelectronic circuits and systems [1]. Between these two alternatives, the latter has been the preferred method because the battery's gravimetric energy density (Watt-hours/lb) has improved only by a factor of two to four over the last 30 years, while the computational power of digital IC's has increased by more than four orders of magnitude [2][3]. Portable electronic systems are far more complex than a single very large scale integrated (VLSI) chip. They contain tens or even hundreds of components, ranging from digital and analog to electromechanical and

optical components. Much of the power dissipation in a portable electronic device comes from non-digital components [5]. For example, the power breakdown for a typical laptop computer shows that, on average, 36% of the total power is consumed by the display, 18% by the hard drive, 18% by the wireless LAN interface, 7% by non-critical components (such as keyboard and mouse) and only 21% by digital VLSI circuitry (mainly memory and CPU) [3]. Reducing the power in the digital logic portion of this laptop by a factor of 15 would reduce the overall power consumption by less than 20% while reducing the power consumption of the non VLSI components (such as the LCD and the HDD) by a factor of 2 leads to more than 25% reduction in the total power dissipation [3].

The power reduction techniques can be classified as static and dynamic. The static techniques, such as low power logic synthesis and physical design, clock gating, and power-aware algorithm selection and software compilation, are applied at the design time while the dynamic techniques, such as dynamic voltage and frequency scaling (DVFS) and dynamic power management (DPM), use runtime behavior of the system to reduce the power when the system is idle or is serving light workloads. DPM for the system components may be considered as the policy and realization of the selective shutdown, slowdown, or the power state transition of the idle or underutilized system components. The DPM policy should often be determined so as to minimize the overall system power consumption subject to a performance constraint.

This paper describes an adaptive and application-independent, wavelet-based predictive method for utilization by the system-level power management solutions under non-stationary workload. The remainder of this paper is organized as follow. A brief review of the background and the related works are presented in Section II while the wavelet based prediction is discussed in Section III. The policy implementation is explained in Section IV

and the experimental results are discussed in Section V. Finally, the summary and the conclusion of the work are presented in Section VI.

## II. REVIEW OF BACKGROUND AND RELATED WORKS

### *A. Modeling the Service Requester and the Service Provider*

One of the key components of the DPM is the modeling of the workload source (e.g., the user, also called service requester, SR.) The workload source, in general, has a non-stationary behavior and, hence, should be modeled as a non-stationary time series whose statistical properties such as auto-covariance would vary in time.

Another important component of the DPM is the device that provides services to the SR. For the sake of simplicity, let us consider a service provider (SP) device with three main states of active (A), idle (I) and sleep (S) as shown in Figure 1(a.) The definition of the active state is that the SP is in its fully functional state and that it is providing service to some SR. In the idle state, the SP is still fully up and operational, but there are no service requests to deal with, and hence, the SP is in its idle state. The transition between the active and idle states is autonomous, i.e., as soon as the SP completes servicing all of the waiting requests, it enters the idle state. Similarly, the SP goes from idle to active as soon as any service request arrives.

In a DPM framework, the SP is transitioned to the sleep state only from an idle state. The duration of the time that the SP is kept in the idle state before it is moved to a sleep state determines the tradeoff between the service latency and power dissipation of the SP. The typical difficulty is that if the power manager adopts an aggressive DPM policy whereby the SP is transitioned into the sleep mode immediately or only after a short period of time, and if the next service request comes early, then the system has to pay for the extra energy and latency of waking up the SP and bringing it to the active state. This is an undesirable degenerate situation where the SP is put to sleep too fast, only to be awakened immediately.

The SP thus uses increasing resources to do a decreasing amount of work (similar to thrashing in the case of multiple processors accessing the same shared resource). On the other hand, if the power manager sets the minimum duration of idle time before entering into the sleep mode to be long and yet no service requests arrives in that period, then the SP has unnecessarily wasted power by not going to sleep.

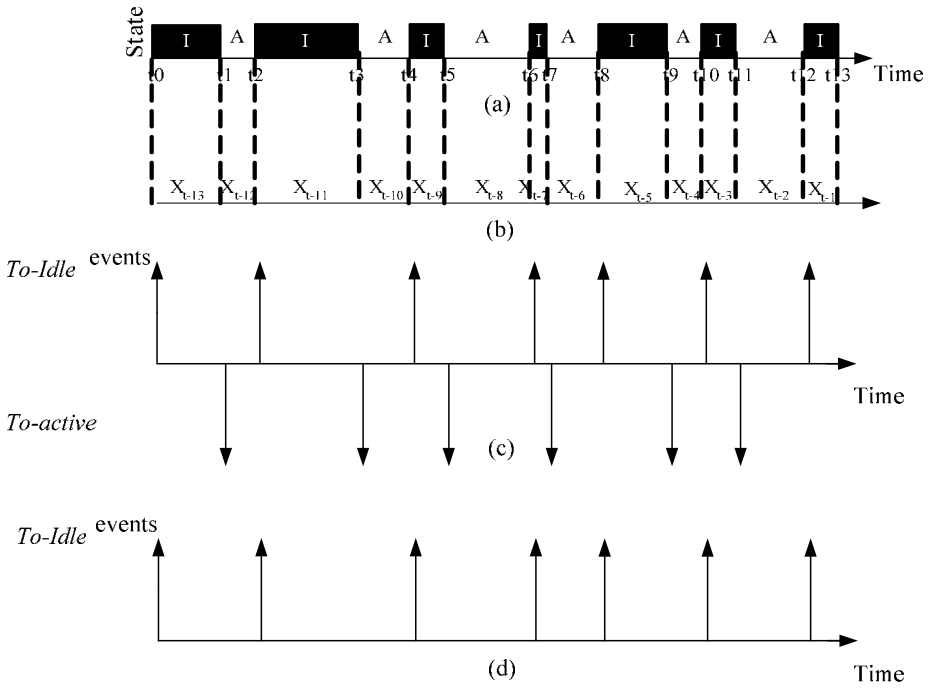



**Figure 1.** (a) Device states, (b) equivalent time series, (c) *to-idle* and *to-active* events, (d) *to-idle* events. I: idle, A: active.

In Figure 1(a), no sleep state is shown for the SP. This is because, in this example, there is no power manager to issue a command to the SP to enter its sleep state. In Figure 1(b), the corresponding time series comprising of the idle and active periods of the SP is illustrated. Figure 1(c) is an equivalent pictorial representation where the events have been denoted by up or down arrows. The up (down) arrows indicate the events whereby the SP state changes from active to idle (*to-idle* arrow) and idle to active (*to-active* arrow), respectively. In Figure 1(d), each one of the upward arrows only shows an occurrence of a *to-idle* event. Here, the arrow height does not correspond to the idle time length. If there are *n to-idle* events, we need *n*

memory locations for saving the event lengths. By concatenating the values of the events we will obtain a trace of the length of the *to-idle* events. We have utilized this trace for predicting the next value of the *to-idle* events.

As explained earlier, to have a successful DPM policy, it is essential to correctly predict the length of the idle time as soon as a to-idle event occurs. In fact, it is well-known that the minimum length of the idle time should be larger than a break-even time for any energy savings to take place. The break-even time is a function of the power dissipations in the idle and sleep states and the energy and latency overheads of transitions between the idle and sleep states [4].

In summary, since taking a device to the sleep state and bringing it back to the active state consumes some energy and has performance overhead, the DPM policy must determine if it is worth changing the state to sleep at all. The more accurate the prediction is, the higher the power saving that may be achieved. The discussion presented here can be easily extended to SP's with more than three states (leading to multiple break-even times depending on the transition.)

### *B. System Modeling*

First, we show how to construct a model for the entire system and then explain how the model can minimize the energy dissipation under different performance constraints. We have utilized a simple power management system, which includes four components: a Service Provider (SP), a Service Requester (SR), a Service Queue (SQ), and a Power Manager (PM) (see Figure 2.) The SR generates service requests for the SP while the SQ buffers the service requests. The SP services the requests while the PM monitors the states of SR, SQ, and SP and issues state-transition commands to the SP.

### *C. Previous DPM Works*

There have been many research efforts focused on proposing DPM methods which may be categorized into three major groups of timeout-based policy, stochastic policy, and predictive policy [7]. The main shortcoming of the timeout policies is that they waste power waiting for the timeout to expire which has motivated researchers to search for more effective techniques [7][8]. The stochastic policies model the arrival of the requests and the device power-state changes by a stochastic process such as Markov process (see, e.g., [9].) Several policies based on this method have been proposed to solve the policy optimization problem (see, e.g., [2] [10][11][12].) These methods may be divided to five major models which are discrete-time Markov decision model (see, e.g., [10]), continuous-time Markov model (see, e.g., [11]), continuous-time semi-Markov model (see, e.g., [12]), time-index semi-Markov model (see, e.g., [12]), and a method based on Markov decision process which has been proposed for non-stationary service requests [3][11]. Since Markov model is stationary and known, the distance of the Markov-based policies from the Oracle policy would be large for non-stationary workloads. In order to overcome this shortcoming, a heuristic method has been proposed in [3] which also cannot guarantee the global optimality in a non-stationary environment [3].

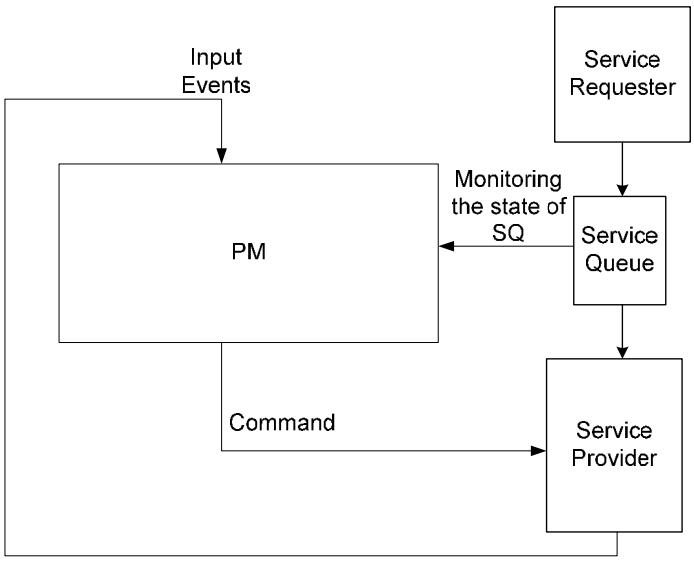



**Figure 2**. The model utilized for the entire system modeling.

In [13], the authors have modeled the non-stationary request process as a Markov-modulated process with a collection of modes which each mode corresponds to a particular stationary request process [13]. Optimal DPM policies are precalculated off-line for the modes using standard algorithms available for the stationary Markov decision processes (MDPs). The power manager then switches online among these policies to accommodate the stochastic mode-switching request dynamics using an adaptive algorithm to determine the optimal switching rule based on the observed sample path [13]. Since in the Hierarchical adaptive DPM they have modeled the service request behavior by concatenating some stationary process (for example 50 process), if the service requester induces a trace that has not been modeled in their precalculated policies and it could not be obtained by interpolating the policies, the system may not be able to make the correct decision. Using a wider range of modes, to model more types of input traces, needs larger state space and computation time/power to obtain the optimal policy. In addition, the stationary Poisson distribution has been assumed for the service requests. This may not be true for all the input traces. As will be described in this paper, the proposed wavelet based method, has the ability to model non-stationary workloads without assuming stationarity or any specific distribution. The method has the ability to automatically extract useful information from the input traces to find the active scales. It adapts itself to any changes in the behavior of the service requester by changing the active scales. It also should be noted that in the hierarchical adaptive DPM when a small (large) state space is used, the computational overhead will be smaller (larger) than that of the wavelet based method.

In the predictive policy, DPM will shut down the system as soon as the predicted length for the idle period is long enough to amortize the cost of shutting down and later reactivating the system [3]. Some of the predictive techniques are based on extensive off-line analyses of the usage traces which again would not be suitable for non-stationary request streams whose

statistical properties are not known as *a priori* [14]. To overcome this limitation some adaptive prediction policies have been proposed. As an example, the work proposed in [14] adopts an exponential average prediction scheme which has been only applied to the systems with a single sleep state and does not deal with resources with multiple sleep states [1]. Another predictive method which handles components with multiple sleep states has been proposed in [6]. This method, which is based on the decision learning tree [15], has the ability to adapt itself to the workload but it may not guarantee the globally optimum solution for the non-stationary input workloads [15]. To remedy these limitations for presenting a globally optimum solution for the non-stationary input workloads, the workload source (e.g., the user, also called SR) should be modeled as a non-stationary time series. In this work, we propose to use the wavelet transform for modeling and predicting the time series.

## III. WAVELET BASED DPM (WBDPM) POLICY

The final goal in DPM is to reduce the power dissipation of the system under different performance constraints by predicting the behavior of the service requester (SR). In DPM systems, when the service provider (SP) enters into the idle state, based on the arrival time of the next event (service request), the SP may change its state to lower power states. Since the power state change of the SP is based on the predicted idle time, the amount of the power saving is strongly depends on the accuracy of the prediction. In this model, the time series consist of the idle times. The prediction methods, including the approach presented in this paper which is based on the wavelet transform, aim at predicting the duration of the next idle time as accurately as possible. In the wavelet based DPM method, the next idle time is predicted based previous idle times. It should be noted that the proposed data driven method is adaptive and hence has the ability to adapt itself to any changes in the behavior of the system.

Since the devices in the system may face different workloads with non-stationary behaviors, a model which has the ability to accurately predict non-stationary time series is required. The Wavelet Transform has a two-dimensional representation (time/scale) of signals and, hence, can model both localized time and frequency behaviors of time series. In addition, in [18], it has been proven theoretically that the wavelet transform can model and forecast non-stationary time-series accurately. This makes the wavelet transform as a very suitable framework for DPM systems.

The wavelet transform presents a two dimensional representation of the time series [18]. The use of wavelet has proved successful in capturing local features of the observed data. The wavelet based forecasting method is a ***decomposition*** of the signal into a range of frequency scales to capture low and high frequency features of the signal. The prediction is based on a small number of coefficients on each of these scales. Although the wavelet based prediction uses a sparse modeling, but since it can be based on coefficients that are summaries or characteristics of large parts of the signal, it can predict the behavior with very good accuracies. The lower scales of the ***decomposition*** can capture the long-range dependencies with only a few coefficients, while the higher levels capture the usual short-term dependencies. Using the redundant or non-decimated wavelet transform has the advantage of being shift invariant. This means that by adding new samples to the data only a few calculations are needed to obtain the new value of the prediction. In addition, since the method is completely dynamic and data driven method, after each prediction, it will be updated by the actual value of the new event. Therefore, if the new data has come from a new situation, the prediction model can adapt itself to this new condition. In addition, since the forecasting value is obtained from a linear combination of the last $p$ events, it has the property of making the decision for the next event based on local behavior of the service request model. The main reason for these two characteristics (adapting to new conditions and

modeling based on local features of the model) by the wavelet transform is the fact that this transform is well-localized both in time and frequency domains while having the potential for naturally handling phenomena whose spectral characteristics change over time. This has enabled us in successfully predicting time series of the service provider (SP) idle times in a DPM system.

The idea of this work is to present an optimum forecasting method for the idle time duration based on a wavelet transform which in turn provides a two dimensional representation (time/scale) of the time series [18][26]. It has been shown that an optimum forecasting for a non-stationary time series which satisfies the local stationary constraint is guaranteed when the wavelet transform with the Mean Square Error (MSE) criterion is used. The MSE or Mean Square Prediction Error (MSPE)) is defined by $MSPE(\hat{X}_{t,T}, X_{t,T}) = E(\hat{X}_{t,T} - X_{t,T})^2$ where $\hat{X}_{t,T}$ are the predicted and $X_{t,T}$ the actual values, respectively. In the next section will show that the predictions coefficients, $b_{t-1-s,T}^{(t)}$, are obtained such that to minimize the Mean Square Prediction Error (MSPE) (for more information about MSPE, see [16][17][18][26].) In [18], the time-varying auto-covariance structure (non-stationary time series) was modeled rigorously by using the wavelet transform and the concept of "local stationary" random processes [17][18] (see Subsection III.F for the definition of local stationary.) The method was then extended in [20] to model the series whose auto-covariance changes very suddenly in time. Based on this model, it is possible to forecast the general non-stationary process [20] as it is needed in an accurate DPM policy.

Before describing the proposed method in detail, we briefly describe the framework of the DPM policy proposed in this work. First, we note that the time series associated with the events, in general, is a non-stationary process, which most of the times may be locally modeled by a class of stationary processes defined below. Second, for the local stationary process, one may use a linear predictor which optimally predicts the next member of the

series as a linear combination of the previous members. The coefficients for this predictor may be obtained from the wavelet theory [16]. Third, the coefficients are obtained by solving a linear system of equations containing the auto-covariance of the time series. The auto-covariance can be calculated using the wavelet spectrum and the auto-correlated wavelet [16]. The wavelet spectrum is carried out through the wavelet coefficients of the time series while the auto-correlated wavelet is determined using the wavelet basis [16].

### *A. Wavelet Framework*

A wavelet system is a set of building blocks to construct or represent a signal or function. In the wavelet expansion, a time-dependent function $f(t)$ may be expressed as a two-parameter system by [21]

$$f(t) = \sum_{k} \sum_{j} a_{jk} \psi_{jk}(t) \tag{1}$$

where both $j$ (scale index) and $k$ (translation index) are integers and $\psi_{jk}(t)$ are the wavelet expansion functions that usually form an orthogonal basis. Each function $\psi_{jk}(t)$ is constructed from a mother function $\psi(t)$ using

$$\psi_{jk}(t) = 2^{j/2} \psi(2^{j} t - k) \tag{2}$$

where $j$ and $k \in Z$ and $j < 0$. Based on the definition, for a given $j$, the function shift is $2^{-j}$ as $k$ is increased by one. As an example of the mother function, one can mention the Harr function which is depicted in Figure 3 [21]. The set of expansion coefficients $a_{jk}$ are called the discrete wavelet transform (DWT) of $f(t)$ and (1) is the inverse transform. The transform is a two-dimensional expansion set for some class of one-(or higher) dimensional signals. The wavelet expansion gives a time-frequency localization of the signals (through $j$ and $k$ indices) which means that most of the signal energy is well represented by a few expansion coefficients, $a_{j,k}$'s. Fourier analysis is appropriate for periodic signals or for signals whose statistical characteristics do not change with time [21]. It is the localizing property of wavelets that

allow a wavelet expansion of a transient event to be modeled with a sparse representation with a small number of coefficients [21]. This turns out to be very useful in our applications.

The standard wavelet transform discussed above can be categorized into two classes of decimated wavelets and non-decimated wavelets [21]. The decimated wavelets have exactly the same number of wavelet coefficients as the number of the samples in the input time series while in the non-decimated wavelet transform there are more wavelet coefficients in the input time series than the number of the samples in the input time series [21]. In this work, we make use of the theory presented in [18], which is developed for the discrete non-decimated wavelets to decompose local stationary processes.

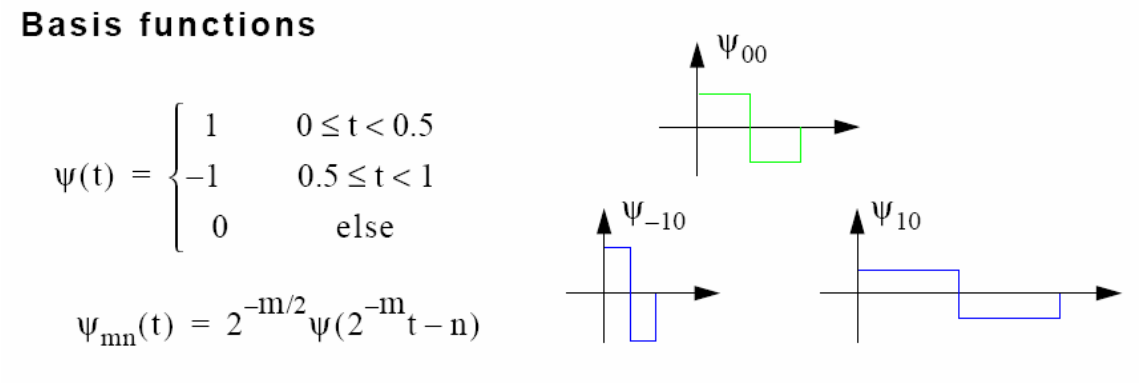



**Figure 3.** Haar basis function ($m = j$, $n = k$) [21].

### *B. Class of Local Stationary Processes and the Wavelet Spectrum*

The materials presented from here to the end of the section are mainly from [18] whose developed theory is used in this work and, therefore, only are other references mentioned. More details of the discussion may be found in this reference. First, let us present the definitions which are used in this work:

**Definition 1:** For $T$ observations of a series at points from 0 up to $T - 1$, it is possible to *rescale* the time in the interval [0,1) as is shown in Figure 4. This rescaling will help us to deal with defining, estimating, and predicting a function as a standard statistical problem.

**Definition 2:** The *discrete auto-correlation wavelet* function is defined as

$$\Psi_j(\tau) = \sum_{t=-\infty}^{t=\infty} \psi_{j0}(t)\psi_{j0}(t-\tau) \quad (3)$$

where $\tau \in Z$ and $j < 0$. The function $\Psi_j$ is called the discrete AutoCorrelation Wavelet (ACW) function at scale $j$ which inherits the localization properties from the wavelets. They are however symmetric about $\tau = 0$, that is $\Psi_j(\tau) = \Psi_j(-\tau)$, for all scales of $j$ and for all $\tau$'s.

**Definition 3:** A time series $X_{t,T}$ (where $t = 0, \ldots, T-1$ and $T > 0$) with a zero-mean is in the class of Local Stationary Wavelet (LSW) processes, if it can be expressed as

$$X_{t,T} = \sum_{j=-J}^{-1} \sum_{k=-\infty}^{\infty} w_{jk,T}\psi_{jk}(t)\xi_{jk} \quad (4)$$

where, $T$ is the number of samples in time series, $J$ is $\log_2 T$, $t$ is the time variable, $j$ and $k$ are the scale and the location parameters, respectively, $w_{jk,T}$'s are real coefficients, $\psi_{jk}(t)$'s form a set of non-decimated family of discrete wavelets, and $\xi_{jk}$ are random orthonormal increment sequence. For the sequence, the expected value ($E(\xi_{jk})$) is 0 and the covariance for all $j$, $l$, $k$, and $m$ ($\mathrm{Cov}(\xi_{jk}, \xi_{lm})$) is $\delta_{jl}\delta_{km}$ where $\delta_{pq} = 1$ if $p = q$ and 0 if not. If the unknown coefficients of $w_{jk,T}$ can be found, $X_{t,T}$ will be known as a Local Stationary process. It has been theoretically proven that $w_{jk,T}$ can be estimated using the wavelet spectrum of the time series. Let us denote the estimation of the $w_{jk,T}$ by $W_j(z)$ which is a time-varying quantity defined in rescaled time $z = k/T \in [0,1.)$ The wavelet spectrum of $X_{t,T}$, defined by $S_j(z) = W_j(z)^2$, is a unique parameter measuring the power of the process at a particular scale $j$ and time $z$.

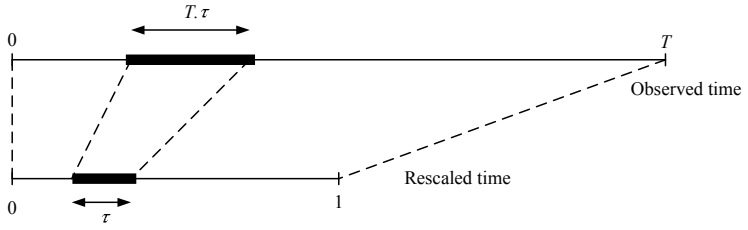



**Figure 4.** The rescaled time principle [18].

As mentioned before, the auto-covariance is used for obtaining the prediction coefficients. The Local Auto-CoVariance (LACV) of a LSW process is defined by $c_T(z,\tau) =$

Cov($X_{[zT],T}$, $X_{[zT]+\tau,T}$.) The relation between $c_T(z,\tau)$ and the wavelet spectrum, $S_j(z)$, of the LSW process $X_{t,T}$, as $T$ tends to infinity, is given by

$$\lim_{T\to\infty} c_T(z,\tau) = c(z,\tau) = \sum_{j=-\infty}^{-1} S_j(z)\Psi_j(\tau) \tag{5}$$

where auto-correlation wavelets $\Psi_j$ was given by (3). The representation is unique due to the fact that $\Psi_j$'s are linearly independent. Hence, $c(z,\tau)$ can be estimated by

$$c\left(\frac{k}{T},\tau\right) = \sum_{j=-J}^{-1}\left(\sum_{l=-J}^{-1} A_{jl}^{-1}\Psi_l(\tau)\right) I_j\left(\frac{k}{T}\right) \text{ where } k = 0, \ldots, t-1 \text{ and } \tau \neq 0 \tag{6}$$

Here, the elements of the Graam (**A**) and the wavelet periodogram (**I**) matrices are obtained from

$$A_{jl} = \left\langle \Psi_j, \Psi_l \right\rangle = \sum_{\tau} \Psi_j(\tau)\Psi_l(\tau) \tag{7}$$

and

$$I_j\left(\frac{k}{T}\right) = \left(\sum_{s=0}^{t-1} X_{s,T}\psi_{jk}(s)\right)^2 \tag{8}$$

From the theoretical point of view, if we use the wavelet periodogram (**I**) for estimating the auto-covariance, some biases will appear in the estimated values leading to some errors in the actual values of the auto-covariance. To obtain an asymptotically unbiased estimator, some corrections to the periodogram are necessary. As a preliminary estimator of the wavelet spectrum, the Corrected Wavelet Periodogram (CWP) can be obtained using

$$L_{j,T}\left(\frac{k}{T}\right) = \sum_{l} (A_T)_{jl}^{-1}\left(\sum_{t=0}^{T-1} X_{t,T}\psi_{lk}(t)\right)^2 \tag{9}$$

where $A_T$ is a $J \times J$ matrix.

### *C. Wavelet-based Prediction Theory*

Now, we wish to use $t$ observations of $X_{0,T},\ldots,X_{t-1,T}$ from an LSW process and predict the $t^{\text{th}}$ observation. For this, we consider a linear predictor as

$$\hat{X}_{t,T} = \sum_{s=0}^{t-1} b_{t-1-s,T}^{(t)} X_{s,T} \tag{10}$$

where coefficients $b_{t-1-s}^{(t)}$'s are determined to minimize the mean square prediction error $E(\hat{X}_{t,T} - X_{t,T})^2$. These coefficients can be obtained by solving

$$\sum_{m=0}^{t-1} b_{t-1-m,T}^{(1)} \left\{ \sum_{j=-J}^{-1} S_j\left(\frac{n+m}{2T}\right)\Psi_j(m-n)\right\} = \sum_{j=-J}^{-1} S_j\left(\frac{t+n}{2T}\right)\Psi_j(t-n) \tag{11}$$

for all $n = 0,\ldots, t-1$. Here, the superscript "1" specifies that we are considering the first predicted value obtained using the previous $t$ observations. In the wavelet approach, these observations could be used for more predictions. The prediction equation can be obtained using the relation between the wavelet spectrum and the local auto-covariance function given in (5.) Furthermore, the prediction equation can also be written as

$$\sum_{m=0}^{t-1} b_{t-1-m,T}^{(1)} c\left(\frac{n+m}{2T}, m-n\right) = c\left(\frac{n+t}{2T}, t-n\right) \tag{12}$$

The above predictor is asymptotically unbiased but is not consistent (its variance does not go to zero with $T$, [17]) and, therefore, it has to be smoothed by using, e.g., a Gaussian kernel smoother with the bandwidth of $g$ [17] (for the concept of the bandwidth, $g$, refer to [22].) It should be noted that in the above equations only $(t-1)$ values could be obtained for the auto-covariance from $t-1$ previous observations. Consequently, for forecasting the next event, extending and smoothing (see, e.g., [22]) the auto-covariance function, which is explained next, is unavoidable. Provided that the $i^{\text{th}}$ row of auto-covariance matrix (C) is given by $C_i = [c_{i1}, c_{i2}, \ldots, c_{in}]$, the $C_i(h)$ represents the smoothing function of $C_i$, and $h$ is an index which can have a value from 1 to $n+1$. To obtain $C_i(h)$, one can use

$$C_i(h) = \frac{\sum_{j=1}^{n} \omega(\frac{h-j}{n}, \sigma) c_{ij}}{\sum_{j=1}^{n} \omega(\frac{h-j}{n}, \sigma)}$$

where $\omega(x, \sigma)$ is given by

$$\omega(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2}$$

Note that the parameter $\sigma^2$ is the same as $g$ in the algorithm. For $h = 1, 2, \ldots, n$, $C_i(h)$'s are the smoothed coefficients of the $i^{th}$ row of the given $C$ matrix and $C_i(n+1)$ is the extended coefficient of the $C$ matrix.

In the next section, we will discuss a data-driven method for choosing the smoothing parameter [22]. Finally, note that for solving the equation of (12), we use the Cholesky decomposition method [23].

### *D. Forecasting Algorithm*

We now address the question of how to estimate the unknown non-stationary series in the system of equations given in (11.) In theory, the best linear predictor of $X_t$ is given by (10), where $b_t = \left(b_{t-1-s,T}^{(t)}\right)_{s=0,\ldots,t-1}$ satisfies the prediction equations (11.) In practice, each of the $t$ components of vector $b_t$ is estimated by using our estimator of local auto-covariance function (12) based on the observations $X_{0,T}, \ldots, X_{t-1,T}$ [17][18]. Hence, we have to compromise between the estimation error, potentially increasing with $t$, and the prediction error, which is a decreasing function of $t$ [18]. As a natural balancing rule, which works well in practice (cf. [17][18]), it is suggested to choose a number of $p$ such that

$$\hat{X}_{t,T}^{(p)} = \sum_{s=t-p}^{t-1} b_{t-1-s,T}^{(t)} X_{t-s,T} \qquad (13)$$

gives a good compromise between the theoretical prediction error and the estimation error.

To select the two parameters of the method which are the order $p$ and the bandwidth $g$, the following automatic procedure may be used. Estimate the auto-covariance $c(z,\tau)$ by smoothing over $k/T$ to achieve consistency where for the sake of simplicity, the same bandwidth $g$ for all $\tau$ is chosen. Also, only the last $p$ observations are incorporated into the predictor. $(g, p)$ is selected using the adaptive forecasting, *i.e.*, $(g, p)$ is gradually updated according to the success of prediction. First, move backward by $s$ observations and choose the initial parameters $(g_0, p_0)$ for forecasting $X_{t\text{-}s,T}$. Next, forecast $X_{t\text{-}s,T}$ using not only $(g_0, p_0)$ but also the eight neighboring pairs $(g_0 + \delta\varepsilon_g, p_0 + \varepsilon_p)$, for $\varepsilon_g, \varepsilon_p \in \{-1,0,1\}$ and $\delta$ fixed as shown in Figure 5. Since the exact value of $X_{t\text{-}s,T}$ is known, the nine forecasted values are compared with the exact value of $X_{t\text{-}s,T}$, and update $(g, p)$ to be equal to the pair which gives the best forecast. This updated pair, as well as its eight neighbors, will be used for forecasting $X_{t-s+1,T}$. The same procedure is continued until we reach $X_{t\text{-}1,T}$. The updated pair $(g_1, p_1)$ is used to perform the actual prediction, and it can be updated later if we wish to forecast the following members of the series such as $X_{t,T}$ and $X_{t+1,T}$.

| | | |
|---|---|---|
| $(g_0+\delta, p_0-1)$ | $(g_0+\delta, p_0)$ | $(g_0+\delta, p_0+1)$ |
| $(g_0, p_0-1)$ | $(g_0, p_0)$ | $(g_0, p_0+1)$ |
| $(g_0-\delta, p_0-1)$ | $(g_0-\delta, p_0)$ | $(g_0-\delta, p_0+1)$ |

**Figure 5.** The neighbors of pair $(g_0, p_0)$ for updating the algorithm parameters.

### *E. Test of Local Significance*

In the previous discussions, the corrected wavelet periodogram is used for estimating the wavelet spectrum. Now, the problem of testing the significance of the corrected wavelet periodogram over a given interval at a given scale is addressed. This test is important for practical purposes due to the fact that a scale of the wavelet spectrum can be active (*i.e.*, non-zero) at a given time and not active at another time, and this evolution corresponds to non-

stationary behavior of the process. Additionally, it is possible that only a few scales to be active significantly in the whole time and, as a consequence, the computations for predicting the time of the next event will be reduced drastically. More formally, the following criterion is used for identifying the active scales.

$$S_j(z) = 0 \text{ for a fixed scale } j < 0 \text{ and for all } z \in \Re, \tag{14}$$

where $\Re \subseteq (0,1)$ is an interval with non-zero interval. It is then possible to test if, for instance, the whole scale is "active" or not, or if it is non-zero before or after a fixed time point. To determine if a scale is significantly active, we should calculate its *p*-value as a criterion [18]. The *p*-value can be approximately calculated using

$$p = \exp\left( -0.125 \times \frac{\eta^2}{1 + (\eta \times 2^{j/2} \times \nu \times C_{\max}) / (n \times \sigma_{j,\Re,T})} \right),$$

where $v$ is a universal positive constant depending only on the wavelet mother function $\psi$, $C_{max}$ is the maximum eigenvalue of the covariance matrix, $n$ is length of the data, and

$$\eta = \Im_T = |Q_{j,\Re,T}| / \sigma_{j,\Re,T},$$

Here, $Q_{j,\Re,T}$ is the averaged corrected wavelet periodogram on the time interval of interest, $\Re$, and $\sigma_{j,\Re,T}$ is the standard deviation of $Q_{j,\Re,T}$. In the proposed DPM, first, the test can be run off-line (or on-line) to determine the active scales which should be used by the wavelet predictor. Since the approach proposed in this work is adaptive, if during the runtime, the prediction accuracy decreases, first the DPM automatically adjusts the values of $p$ and $g$ to improve the accuracy. If the prediction accuracy is not increased to the desired level using this adjustment, the test of local significance will be run again to determine the new active scales.

### *F. Test of Local Stationarity*

The time series is considered to be stationary in an interval if the wavelet spectrum of the time series does not change considerably in that interval. The condition is tested as explained here. Suppose that the wavelet spectrum for the scale $j$, denoted by $S_j$, at each time interval is well

approximated by the corrected wavelet periodogram (CWP), denoted by $L_j$. Consider $Q_{j,U,T}$ and $Q_{j,\Re,T}$ as the averages of the CWP of scale $j$ for given intervals of $U$ and $\Re$ where $U$ is any subset of the larger interval of $\Re$. If the absolute value of the difference between the two values of $Q_{j,\Re,T}$ and $Q_{j,U,T}$ is not significantly large, then the hypothesis of local stationarity of the wavelet spectrum $S_j(z)$ on $z \in \Re$ is passed [18].

## IV. DPM POLICY IMPLEMENTATION FOR PREDICTING THE IDLE TIME DURATION

In this section, the algorithm used for the DPM policy optimization guided by the wavelet-based idle time prediction algorithm is described. The SP device (HDD) which is considered in this study has three states which are idle, active, and sleep (low-power state.) After the device finishes servicing a request, it enters the idle state where the duration of the inactivity (idle time) is predicted. If the predicted time is greater than a minimum threshold i.e., the breakeven time ($T_{BE}$) AND if the delay constraints upon wakeup can be satisfied, the HDD will go to the sleep state (see Section IV.B). Otherwise, it will remain in the idle state. For this device, we need one threshold value for making the decision on changing to a sleep state. It should be noted that the device makes a transition from the low power state to the active state based on the performance constraint meaning that the transition may occur immediately after entering the request in the SQ or after a delay depending on the performance constraint. The same approach may be applied to the devices with more low-power states. The approach for the multi-low-power state SP is shown in Figure 6 where it is assume that the state $n$ has lowest power consumption. The parameters used in this chart will be defined in the following subsections.

We now explain how the DPM unit works. The DPM unit considers each transition from the active state to the idle state as an event occurrence. When an event occurs, the device

goes to the idle state and the DPM unit is activated to execute the prediction routine for forecasting the idle time duration, from which the transition to a low power state will be decided. Next, we describe the approach which has been taken for the calculation of the threshold value (i.e., the breakeven time.)

### *A.* Calculation of Threshold Values

The approach taken in the calculation of the threshold values is similar to the one discussed in [19]. To have a general expression for the breakeven times, assume that the system (device) has one active state, one idle state, and $n$ low power states ($n$ + 2 total states.) The low power states are denoted as $P = \{p_1, p_2, \ldots, p_n\}$, which are in descending power order, i.e., $p_n$ is the lowest power consuming sleep state. $p_0$ denotes the idle state power consumption. For such a system, it is essential to determine $n$ threshold values (one value for each low-power state.) The DPM policy utilizes these threshold values to make a decision for changing the state of the system from the idle state to a low power state. If the idle time is too short, the device must remain in the idle state. As will be seen, for a given device, the decision is made based on the idle state duration. For each idle interval, the energy consumed when the state changes from the idle state to the state $i$ is calculated from

$$E_i = t_{0\to i} \times p_{0\to i} + (t_{idle} - t_{0\to i} - t_{i\to 0}) \times p_i + t_{i\to 0} \times p_{i\to 0} \tag{15}$$

where $t_{0\to i}$ ($p_{0\to i}$) is the delay (power dissipation) from transiting from the idle state to low power state $i$, $t_{i\to 0}$ ($p_{i\to 0}$) is the transition time (power dissipation) for going from low power state $i$ to the active state, and $t_{idle}$ is the idle time. If the device does not go to low power state $i$, the consumed energy in the idle state is

$$E_0 = t_{idle} \times p_0 \tag{16}$$

To make a decision for changing from the idle state to low power state $i$, $E_0$ should be larger than $E_i$ [19]. Since for a device, except for $t_{idle}$, all other parameters are fixed, one can

obtain the idle time threshold value for making the transition from the idle state to the state *i*. Note that the lower power the state is, the more time the transition from the idle state to this state takes and the more power is consumed. Consequently, the threshold value for the idle time of this transition will be higher.

To determine the threshold of each state, $E_0$ is set equal to $E_i$ and is solved for $TH_{0,i}$ as

$$TH_{0,i} = T_{BEi} = \frac{p_{0\to i} \times t_{0\to i} + p_{i\to 0} \times t_{i\to 0} - p_i \times (t_{0\to i} + t_{i\to 0})}{p_0 - p_i} \tag{17}$$

If the predicted idle time is at least equal to $TH_{0,i}$, the device makes a transition to low power state *i* [19].

### *B. Delay Constraint*

Minimizing the average power dissipation of a system under a given delay constraint is the main objective of the DPM. The average delay that a request experiences from the time it enters the queue until it is serviced by the SP is related to the average number of waiting requests in the service queue. In a *stable* system where the queue is essentially not overflowed all the time, we must have a condition whereby the average request inter arrival time ($\rho$) is equal to or greater than the average service time of a request by the SP ($\sigma$), *i.e.*, $\rho \geq \sigma$.

As mentioned earlier, the predictor will be activated for predicting the length of the idle period as soon as the SP enters the idle state. Let $T_{BE,i}$ denote the $T_{BE}$ of state of *i*. The SP is commanded to enter the low power state of *i* only if the predicted idle time is greater than or equal to $T_{BE,i}$. In addition to the power saving check for going to a low power state, the delay constraints of the system must also be checked for both making transitioning from the idle state to a low power state and from a low power state to the active state. Next, we discuss these constraints.

***B.1. Changing the state of the SP from the idle to the $i^{th}$ low power state***

Let us denote the average request inter-arrival time immediately after a transition from some sleep state to the active state as $\rho_b$, the average request inter-arrival time as $\rho$, the average service time of a request by the SP as $\sigma$, and the maximum service queue size as $q_{max}$ denoting. Clearly, $\rho_b$ is less than $\rho$. If $\rho_b \geq \sigma$, then there is no problem since the current request will be serviced by the time when the next service request arrives. In this case, we do not need to include any additional delay constraint in the system. Consider the case where $\rho_b < \sigma$. We impose the condition that we can only have a service burst for up to $N_b$ requests during a time period $T_b$. In this case, $\rho_b = T_b/N_b$. To prevent a service queue overflow, we must have $N_b.(1 - \rho_b/\sigma) \leq q_{max}$. The requirement that $N_b.(1 - \rho_b/\sigma) \leq q_{max}$ is equivalent to $\rho_b \geq \sigma.(1 - q_{max}/N_b)$. If this condition is not satisfied, the service loss rate will become too high due to the queue overflow (we assume that some other controller in the system adjusts the request generation rate for the SP by stalling a pipeline, a system bus, etc.)

To determine if this delay constraint is satisfied, we should check if the maximum delay constraint will be met after the system wake-up when a burst of requests comes in. Furthermore, the latency for providing service to the last request in a burst of $N_b$ requests with an inter-arrival time of $\rho_b$ is equal to $T_{tr,i} + N_b.(\sigma - \rho_b)$. The transition from the idle state to the low power state $i$ is put in effect only if $T_{tr,i} + N_b.(\sigma - \rho_b) \leq D_{max}$ where $D_{max}$ is the maximum delay constraint for satisfying the performance constraint A decision to move into a low power state $i$ is made only if the breakeven time is satisfied *and* these delay constraints described above are met.

***B.2. Changing the state of the SP from the $i^{th}$ low power state to the active state***

Now the question is that how soon after a request arrives in the system queue, the SP should go to the active state. This is an important question since if the delay constraint is loose, the SP may choose to stay in the low power state for a while to save power, instead of

immediately making a transition to the active mode. Consider the case where the SP is in the low power state *i*, and the first request arrives. For given average burst rate (which is the inverse of the request inter-arrival times in the burst) and the expected number of requests in the burst following a wakeup from state *i*, the SP can wait for a maximum time (wakeup delay) of

$$T_{DW} = MIN(T_{DW,1}, T_{DW,2})$$

where

$$T_{DW,1} = D_{max} - (T_{tr,i} + \sigma)$$

and

$$T_{DW,2} = D_{max} - (T_{tr,i} + N_b.(\sigma - \rho_b))$$

and

$$D_{max} = \sigma.Prfc$$

where *Prfc* is the performance constraint and σ is the average service time of a request by the service provider (SP(σ)) [5].

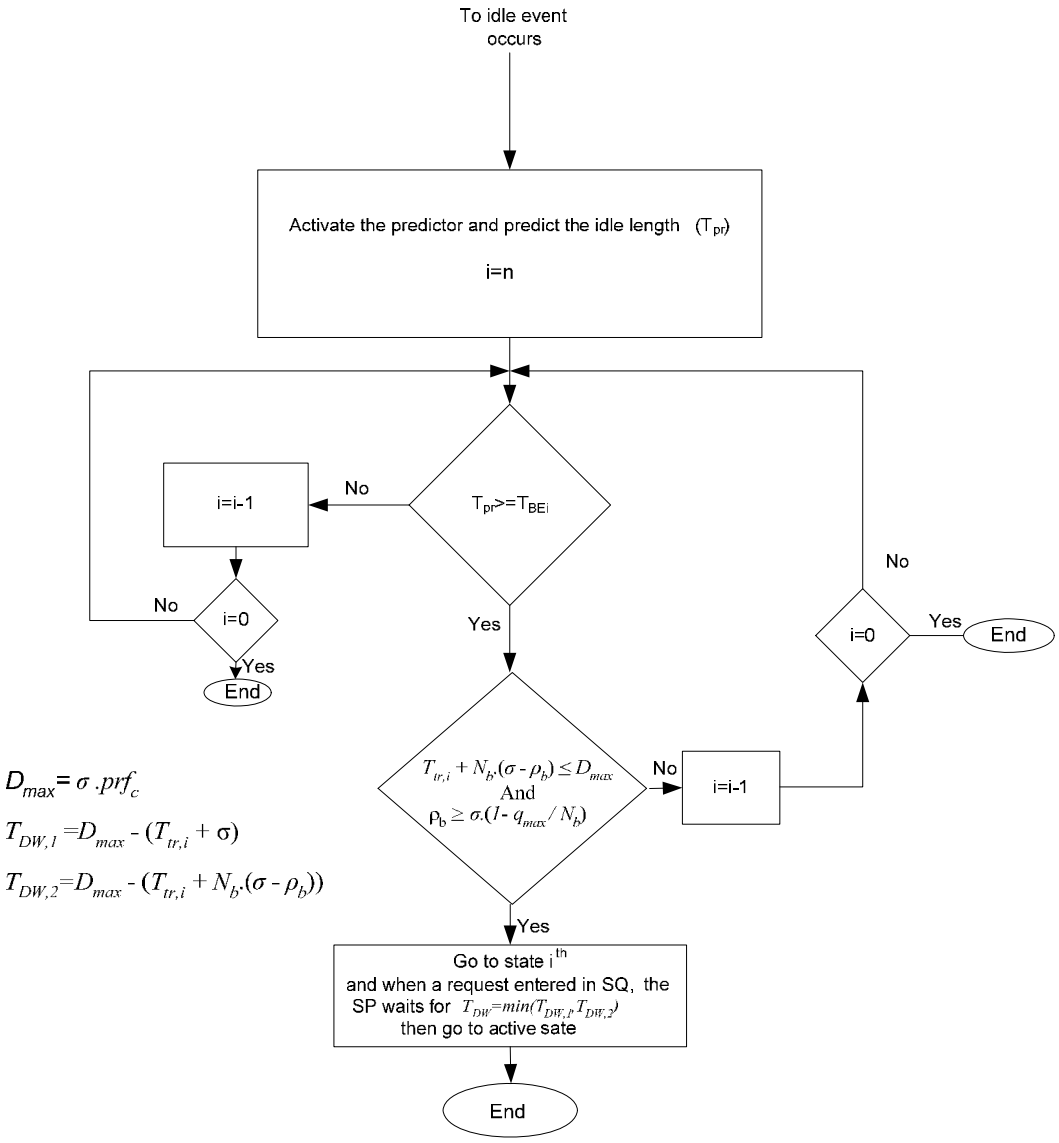



**Figure 6.** The flowchart illustrating the decision algorithm in the proposed method.

As abovementioned, it is possible, after the first request arrival, many others arrive in burst causing the queue to be overflowed and the deadlines to be missed. In these cases, to go to low power states, two predictors may be required where one of them should predict the length of the idle time assuming an empty queue while the second ought to predicts the request burst characteristics. The burst characteristics include the number ($N_b$) and the average inter-arrival time ($\rho_b$) of the service requests when we are in the low-power state of *i*. In this work, we have assumed that we are given the maximum bounds on $N_b$ and $\rho_b$ and based on them we

have designed the system including the queue size and the delay constraint such that no queue overflow or deadline miss occurs. Also, it should be mentioned since our DPM policy is adaptive, when a service request burst occurs, it affects the average idle time of the system prohibiting the system from going to a low-power state after the burst has occurred. This way, if the system is going into a mode with service request bursts, the system does not enter a low-power state which causes a queue overflow or deadline misses. The flowchart of making decision by DPM is illustrated in Figure 6 which is explained here. When a to-idle event occurs, the PM (Power Manger) is activated and the length of the idle time ($T_{pr}$) is predicted. Then the current state of the system, $i$ ($n \leq i \leq 1$) is considered to be $n$ which is the lowest (deepest) power state. In this step, the predicted idle length is compared with $T_{BEn}$ which is the break event time corresponding to the $n^{\text{th}}$ low power state. If the predicted value is smaller than the $T_{BEn}$, then the $n^{th}$ low power state will change to $(n-1)^{\text{th}}$ low power state and the procedure is repeated by comparing the predicted value with the $T_{BEn-1}$. The procedure continues until the predicted value become greater than the $T_{BEi}$ ($T_{pr} \geq T_{BEi}$) or $i$ becomes equal to 0. The latter case means that the idle length is not long enough for a low-power state transition and the SP remains in the idle state. When the former occurs, the PM will check that changing the state of the SP to the $i^{th}$ low power state can satisfy the $Prf_C$ constraint. For this, the PM should check if the maximum delay constraint will be met after the system wake-up and a burst of the requests comes in. This constraint is satisfied if $T_{tr,i} + N_b.(\sigma - \rho_b) \leq D_{max}$. Additionally, to prevent a service queue overflow, we must have $N_b.(1 - \rho_b/\sigma) \leq q_{max}$ which is equivalent to $\rho_b \geq \sigma.(1 - q_{max}/N_b)$. The power manager will check these two conditions before changing the state of the SP to the $i^{th}$ low power. If at least one of the constraints is not satisfied, the $i^{th}$ low power state will changes to $(i-1)^{\text{th}}$ and the process of checking these two constraints is repeated. This process continues until both constraints are satisfied or $i$ becomes equal to zero. If $i$ become zero again the SP will remain in the idle state. In the case that the

two constraints are satisfied the PM will changes the state of the SP to $i^{th}$ low power state. Finally, when the SP is in $i^{th}$ low power state, if a request enters into the service queue (SQ) the SP waits for $T_{DW} = \min(T_{DW,1}, T_{DW,2})$ before going to the active state.

### *C. Implementation of Online Prediction of Idle Time Length*

Now, the steps that should be taken in implementing the procedure for the prediction of the idle time length are described. The steps are illustrated as a flowchart which is given in Figure 7. Some of the steps are performed once while the others should be repeated. These steps, which are independent of the number of low power states, include

1- The behavior of the device is modeled according to the method proposed in the Section III.

2- Whenever the device makes a transition from the active state into the idle state, the prediction program is recalled to calculate the length of the idle time.

3- Since the significant scales have been determined in the runtime, only are they used in the computation. The autocorrelation function is computed and the matrix **A** is constructed using (7.)

4- Non-decimated wavelet coefficients of the significant scales are derived and its matrix **I** is constructed using (8.)

5- Using (6), the local auto-covariance function is derived and smoothed.

6- Equation (12) is constructed and is solved using the Cholesky decomposition, *b* coefficients are calculated, and then the length of the time is predicted.

7- The decision for the state transition is made based on the predicted time inter-arrival.

8- When the device makes a transition from the idle state to the active state, the actual idle time is determined and the error of the prediction is calculated. If the error is more than a definite value, the parameters ($p$, $g$) are updated.

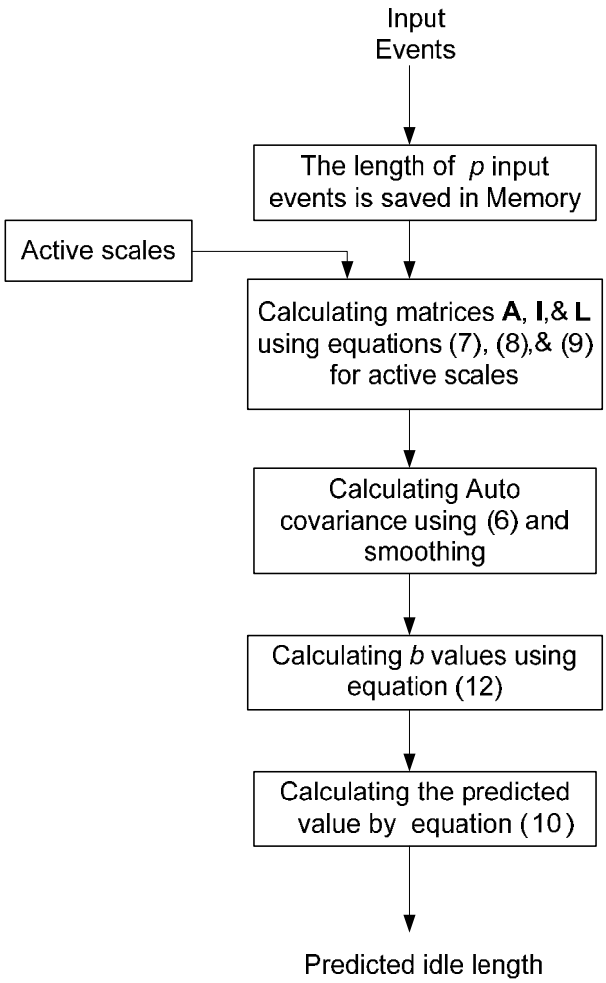



**Figure 7.** Flowchart for predicting the idle length in wavelet based DPM.

## V. RESULTS AND DISCUSSION

In order to assess the efficiency of the proposed algorithm for the DPM policy, we have implemented a simulator which determines the power consumption of the system under different DPM policies. The DPM algorithms implemented in the simulator includes Timeout [12], Oracle [12], Always On, continuous-time Markov decision processes [11], Predictive [6], Time Index Semi-Markov Decision Process (TISMDP) [12], Discrete Time Markov Decision Process (DTMDP) [10], and Non-stationary Markov Decision Process (NSMDP) [1] methods. The simulator applies the policies to some time series to make a decision for the state change and calculate the corresponding power consumption.

### *A. Application of Proposed Method to Hard Disks*

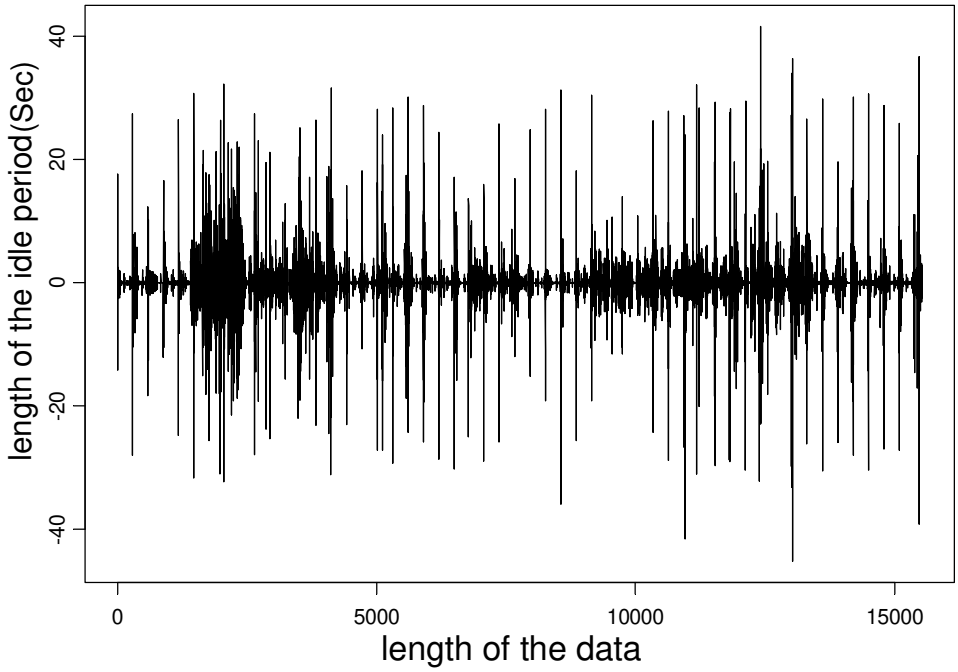
In order to evaluate the performance of the WBDPM policy to real devices, we applied the method to the hard disks of a laptop computer and a desktop computer. These devices, whose specifications are given Table I and Table II, have four states which are active, idle, standby, and shutdown [24][25]. The threshold values for these devices were computed as described in Section IV.A. The required experimental data for the test of various methods are collected by

Table I. Specifications of the PC hard disk [24].

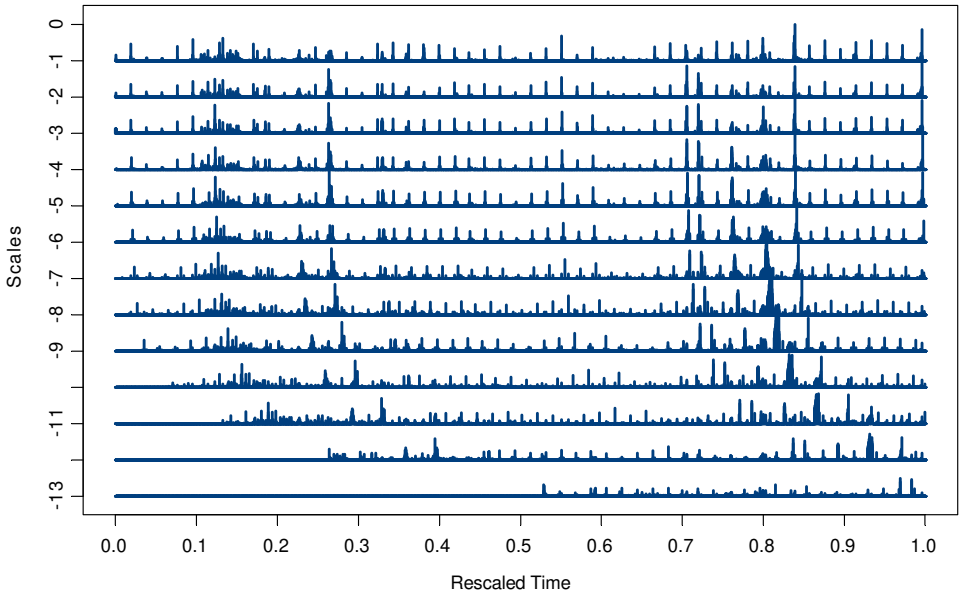
| *Samsung SV4002H* | | | | | | |
|---|---|---|---|---|---|---|
| State | Power(W) | $T_{tr1}$ | $T_{tr2}$ | $E_{tr1}$ | $E_{tr2}$ | Threshold |
| Active | 6 | - | - | - | - | NA |
| Idle | 4.8 | - | - | - | - | NA |
| Standby | 0.5 | 84mesc | 38msec | 46.31J | 8.21J | 12.68 |
| Sleep | 0.2 | 160msec | 62msec | 112.9J | 17.1J | 28.246sec |

Table II. Specifications of the laptop hard disk [25].

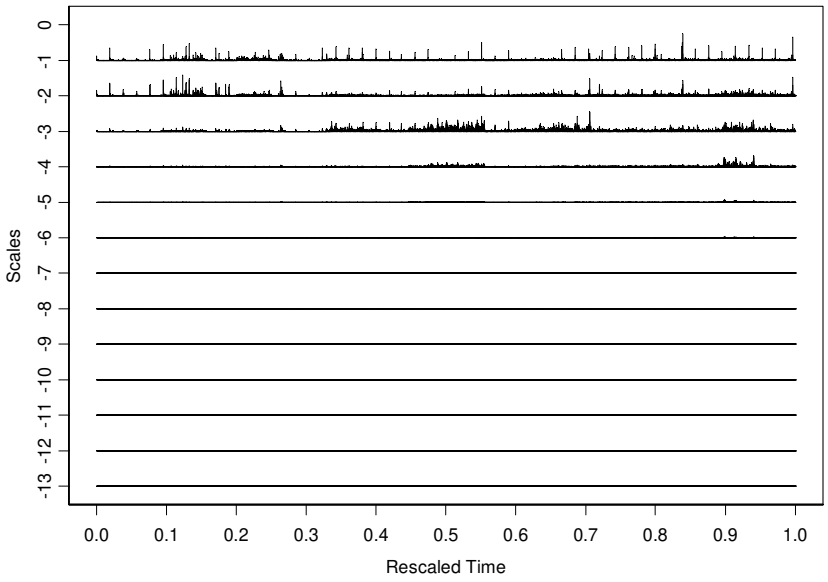
| *Toshiba MK4026GAX* | | | | | | |
|---|---|---|---|---|---|---|
| State | Power(W) | $T_{tr1}$ | $T_{tr2}$ | $E_{tr1}$(J) | $E_{tr2}$(J) | Threshold |
| Active | 2.5W | - | - | - | - | NA |
| Idle | 1.05W | - | - | - | - | NA |
| Standby | 0.25W | 75*msec* | 50*msec* | 6.12 | 2.4 | 10.673 sec |
| Sleep | 0.1W | 187*msec* | 70*msec* | 17.82 | 3.6 | 22.542 sec |


length of the idle period(Sec)
40
20
0
-20
-40
0
5000
10000
15000
length of the data

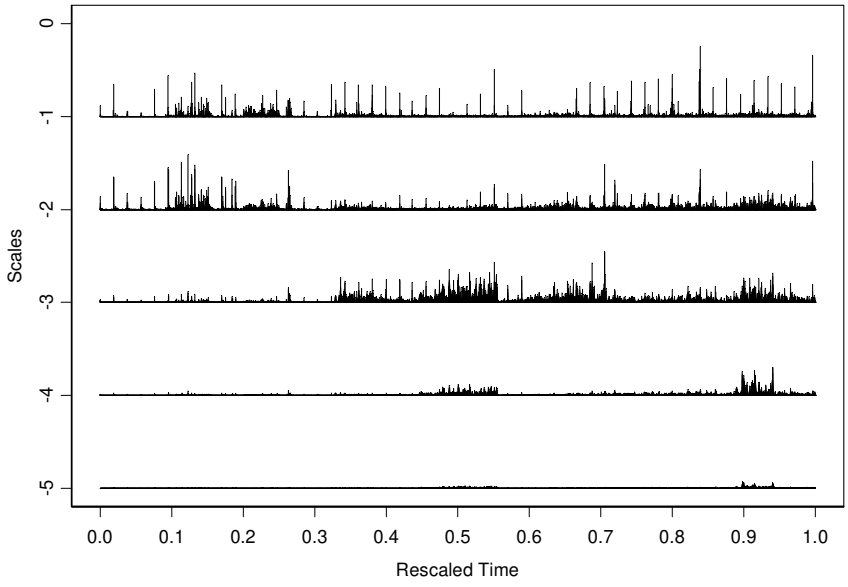
(a)


Scales
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-11
-13
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Rescaled Time

(b)




(c)




(d)

**Figure 8.** (a) Collected trace of hard desktop computer , (b) Uncorrected wavelet periodogram of the trace, (c) Corrected wavelet periodogram, (d) Magnified corrected wavelet periodogram for the five first scale

Table III. Results of the significance test for the hard disk of the desktop computed on $\Re$ = (0,1) for scales –1 to –13.

| Scale | $Q_{j,\Re,T}$ | $\sigma^2_{j,\Re,T}$ | Approximated *p*-value |
|---|---|---|---|
| –1 | 60.1 | 117 | 0.02 |
| –2 | 55 | 110 | 0.03 |
| –3 | 73.67 | 84.65 | $3.2\times10^{-4}$ |
| –4 | 35.29 | 121.36 | 0.27 |
| –5 | 24.93 | 92.04 | 0.429 |
| –6 | 18.75 | 104.89 | 0.65 |
| –7 | 8.26 | 40.42 | 0.809 |
| –8 | 4.30 | 18.46 | 0.891 |
| –9 | 3.16 | 12.54 | 0.905 |
| –10 | 2.42 | 15.36 | 0.95 |
| –11 | 2.12 | 13.88 | 0.96 |
| –12 | 1.18 | 10.64 | 0.98 |
| –13 | 1.23 | 24.08 | 0.99 |

Table IV. Results of the significance test for the hard disk of the laptop computed on $\Re$ = (0,1) for scales –1 to –13.

| Scale | $Q_{j,\Re,T}$ | $\sigma^2_{j,\Re,T}$ | Approximated *p*-value |
|---|---|---|---|
| –1 | 72.4 | 68.53 | $7\times10^{-5}$ |
| –2 | 54.36 | 71.92 | $5.8\times10^{-3}$ |
| –3 | 58.72 | 62.78 | $1\times10^{-3}$ |
| –4 | 43.56 | 76.4 | 0.044 |
| –5 | 32.1 | 88.7 | 0.234 |
| –6 | 24.6 | 65.7 | 0.316 |
| –7 | 21.3 | 92.83 | 0.542 |
| –8 | 14.26 | 62.67 | 0.666 |
| –9 | 10.51 | 44.92 | 0.735 |
| –10 | 8.16 | 38.34 | 0.804 |
| –11 | 7.32 | 30.76 | 0.81 |
| –12 | 5.6 | 28.76 | 0.872 |
| –13 | 4.2 | 21.62 | 0.903 |

monitoring the behavior of the hard disks for 24 hours. For collecting the data, we used a software package that monitors the states of the hard disks and record the time intervals for the idle and the active states. It should be mentioned that we have written a C++ program which uses ACPI of the windows for monitoring the state of the hard disk. The power-

performances of different DPM policies have been compared using the collected traces in a simulator.

The collected data from the hard disk of the PC and the laptop are shown in Figure 8(a) and Figure 9(a), respectively and their wavelet spectrum, without the correction by $\mathbf{A}^{-1}$, are given in Figure 8(b) and Figure 9(b), respectively. As discussed before, the wavelet periodograms are not appropriate for the local significant test, the local stationary test, and the prediction procedure. Their corrected wavelet periodograms are shown in Figure 8(c) and Figure 9(c) while Figure 8(d) and Figure 9(d) give five scales of the corrected wavelet periodograms. The results of the local significance test in the interval $\Re = (0,1)$ are given in Table III and Table IV. As observed from the tables, only scales –1, –2, and –3 are the active scales and the values of other scales are negligible. The information would lead us to use only the three first scales to reduce the computations.

In order to see if the collected data from the hard disks are non-stationary (which in that case may not be modeled by a stationary-based model) the stationary test is applied to these data. We use the same approach for the analysis of the data as the one taken in [18]. Here, for a better analysis, the data are partitioned into three segments as given in Figure 10(a) and Figure 11(a.) The test is applied to all three segments for scales –1 to –5. The results of the test are given in Table V and Table VI where the numbers in the tables are the minimum *p*-values obtained in each segment. To determine when a scale is absolutely or relatively dominant, we choose two thresholds. If the *p*-value is between 0.01 and 0.05, the scale is relatively dominant and if the *p*-value is less than 0.01, the scale is absolutely dominant. The absolutely dominant scale is of a high significance while the relatively dominant scale is of a low significance. The *p*-value of a scale is inversely proportional to its

Table V. Results of the local stationary test for the hard disk of desktop computed on segments 1, 2, and 3 according to Fig. 11(a) for scales –1 to –5.

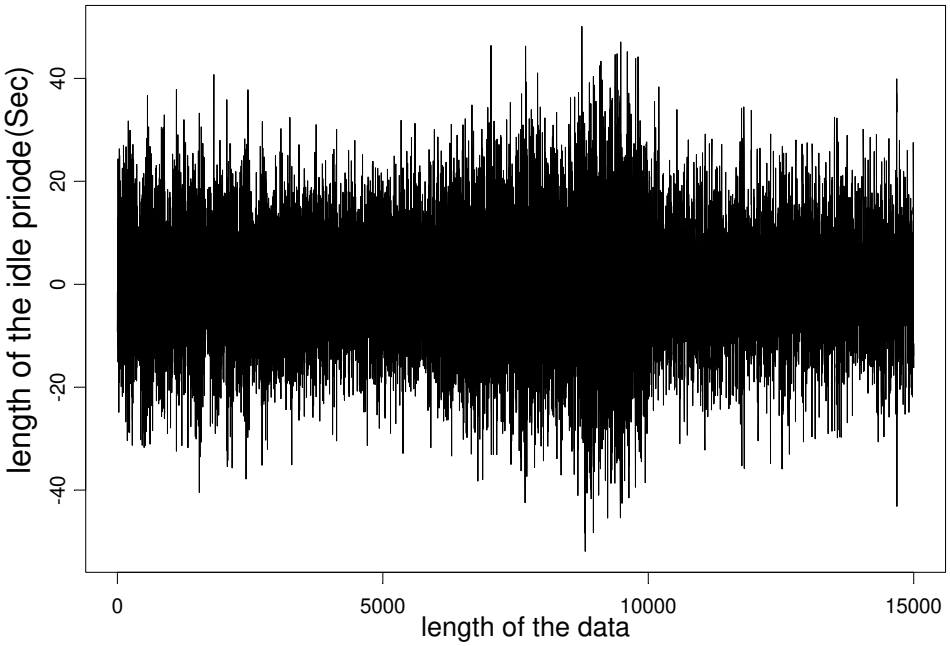
| Scale | *p*-values | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| –1 | 0.0128** | 0.058 | 0.032* |
| –2 | 0.008** | 0.06 | 0.04* |
| –3 | 0.024* | 0.0042** | 0.018** |
| –4 | 0.52 | 0.12 | 0.07 |
| –5 | 0.61 | 0.62 | 0.38 |

* Indicates a *p*-value 0.01 to 0.05, ** Indicates a *p*-value less than 0.01.

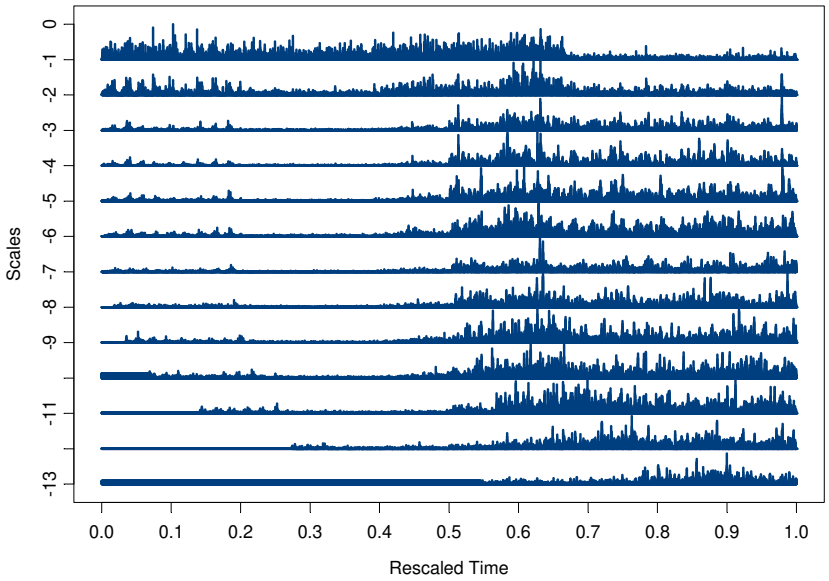
Table VI. Results of the local stationary test for the hard disk of the laptop computed on segments 1, 2, and 3 according to Fig. 12(a) for scales –1 to –5.

| Scale | *p*-values | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| –1 | $2\times10^{-5}$** | 0.0034** | 0.32 |
| –2 | 0.028* | 0.0012** | 0.042* |
| –3 | 0.44 | 0.015* | 0.0037** |
| –4 | 0.48 | 0.17 | 0.072 |
| –5 | 0.58 | 0.54 | 0.47 |

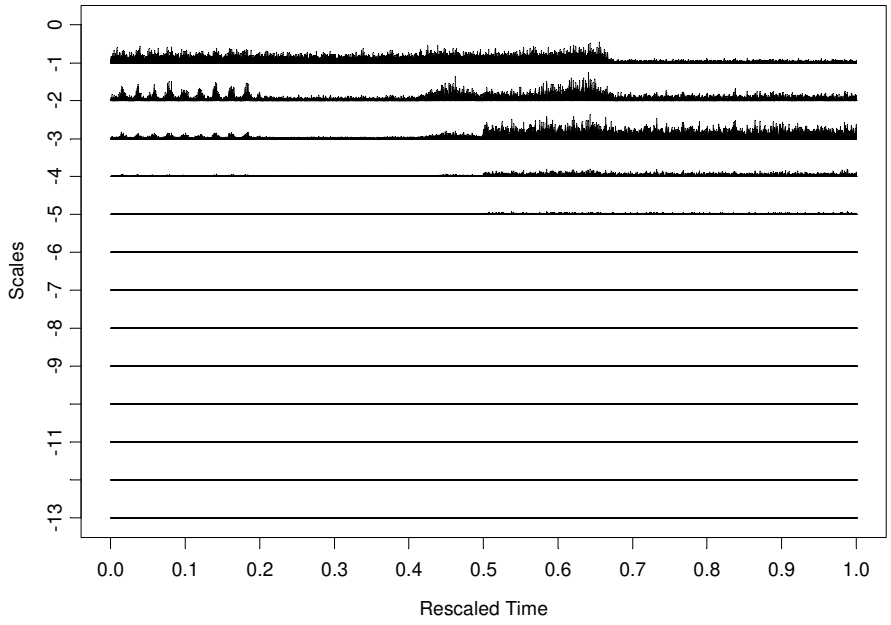
* Indicates a *p*-value 0.01 to 0.05, ** Indicates a *p*-value less than 0.01.


length of the idle priode(Sec)
40
20
0
-20
-40
0
5000
10000
15000
length of the data

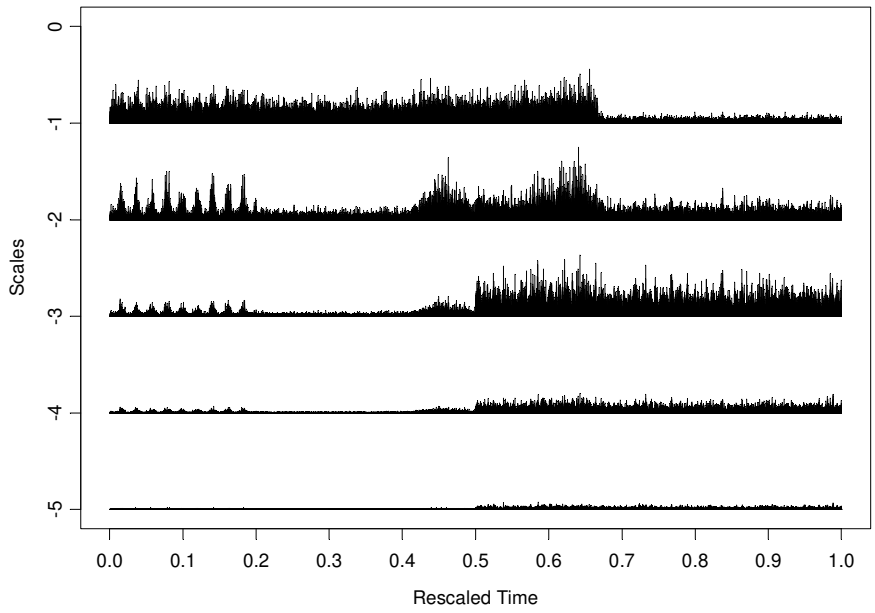
(a)


Scales
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-11
-13
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Rescaled Time

(b)

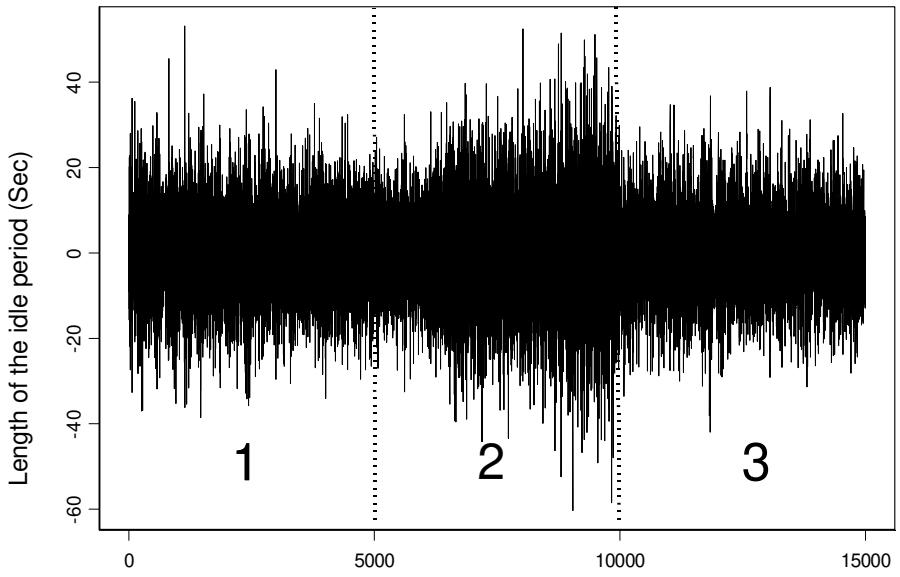



(c)

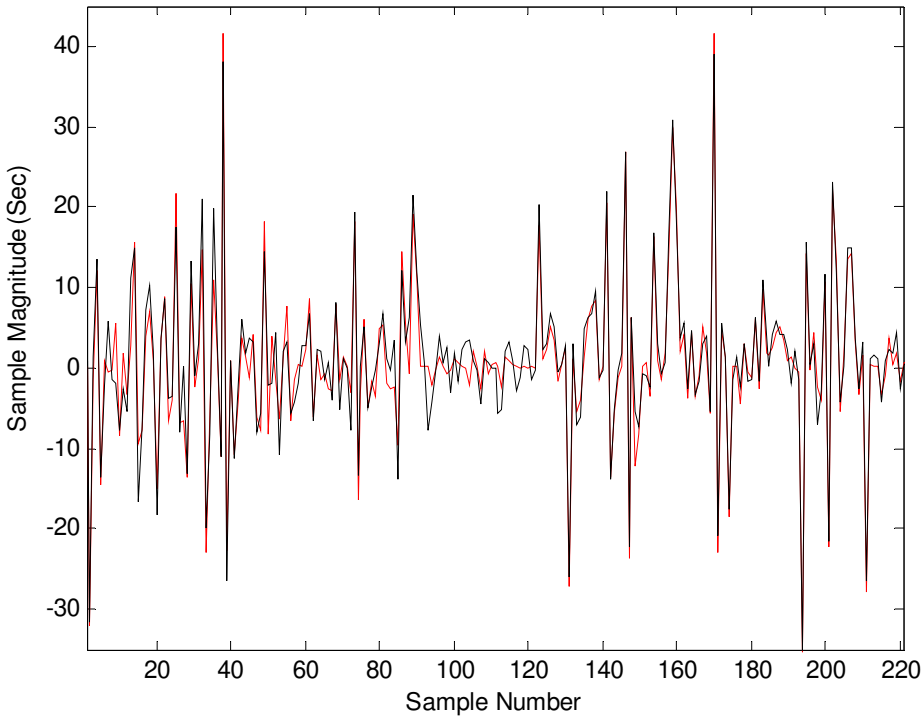



(d)

**Figure 9.** (a) Collected trace of hard of lap-top, (b) Uncorrected wavelet periodogram of the trace, (c) Corrected wavelet periodogram, (d) Magnified corrected wavelet periodogram for the five first scales

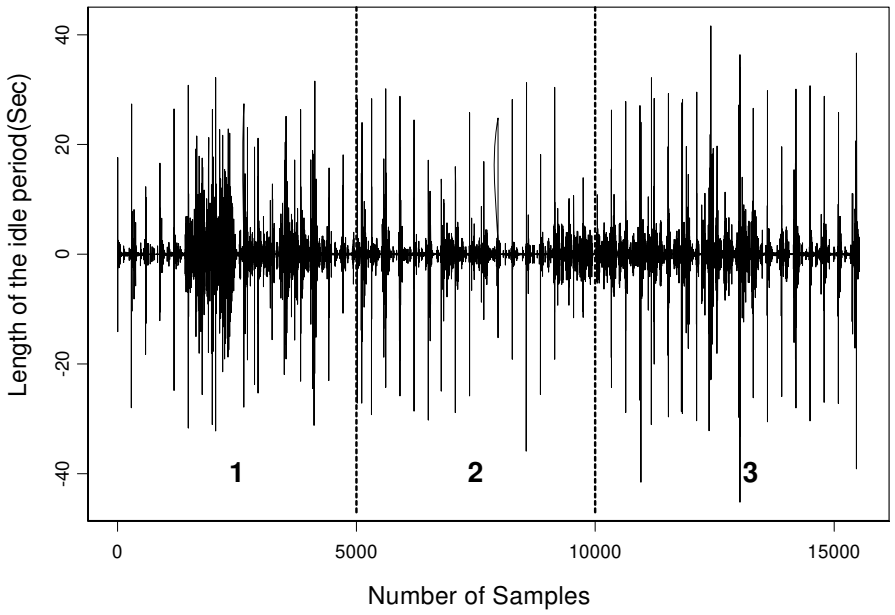



(a)

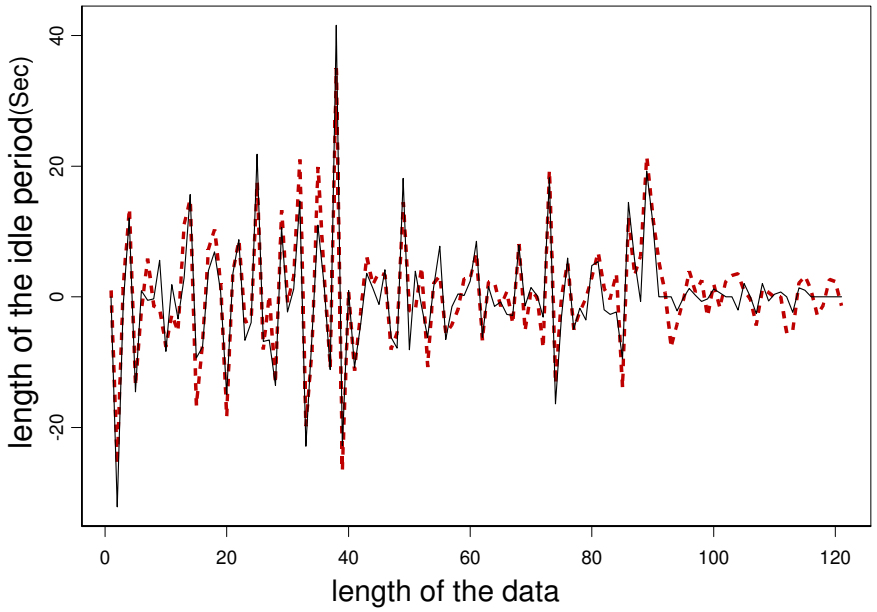



(b)

**Figure 10.** (a) Partitioning of trace collected from the HDD of a laptop to three segments, (b) black line: hard trace with the length of 120, red line: predicted sequence.




(a)




(b)

**Figure 11.** (a) Partitioning of trace collected from hard of a desktop computer to three segments, (b) Solid line: hard trace with length 120, doted line: predicted sequence

significance. Specifically, the scales having *p*-values near 1 are insignificant and may be neglected.

As is observed from Table V and Table VI, scale 1 is absolutely dominant in segment (1) and relatively dominant in segment (3) while it is insignificant in segment (2.) A similar discussion exists for the results presented in Table VI. Since the active scales are not the same in different intervals, the time series is non-stationary. After determining the significant scales, they will be used for predicting the length of the idle time. In the following, we will use the significant scales for predicting the behavior of the HDD. The prediction accuracies of the events with a length of 120 for the hard disks are shown in Figure 10(b) and Figure 11(b.) The values ($p$, $g$) are chosen optimally and updated during the prediction leading to an accuracy of 95%. In Figure 12, the prediction error has been plotted for 220 samples. As is evident from the figure, the error decreases as the sample number increases. One should note that for the DPM decision making, the required prediction accuracy should be such that the PM can make a correct decision for changing to a low power state.

As mentioned before, the DPM policies were designed to minimize the power consumption subject to the performance constraint expressed in terms of the upper bounds on the performance metric ($Prf_C$.) For our experiments, we used $Prf_C = 0.01$ which means that the normalized number of waiting request in the queue should be equal or less than 0.01. The number is defined as $N_W/N_T$ where $N_W$ and $N_T$ are the number of waiting requests in the queue and the total number of the incoming requests during time $T$, respectively. The reported parameters are defined in the following:

- **Pwr** (Power): It is the average power consumption (Watt.)
- **HR** (Hit Ratio): It is the ratio of the corrected predictions to the total predictions.
- **EN** (Energy): It is the total energy dissipation of the idle state normalized to the energy dissipation of the Oracle policy.

- **EF** (Efficiency): It is the ratio of the normalized power dissipation of the Oracle policy to that of other policies.
- **Ntr**: It is the total number of transitions from the low power states to the active state
- **Nwtr**: It is the number of wrong transitions causing a power and performance overhead.
- **Tss**: It is the average elapsed time in the low power states per transition (S.)

Ntr, is directly related to the performance penalty that has to be paid to wake up the SP. If these transitions are correct, then the power saving would justify the transitions power consumption based on the performance constraint. Nwtr, however, is a measure of the algorithm inefficiency which leads to both performance and power penalties. Tss can be seen as a measure of efficiency. For minimizing the power dissipation, it is essential to maximize the while minimizing the Ntr and Nwtr.

The results of the comparison among the DPM methods in reducing the energy dissipation for the same performance constraint are given in the Table VII, Table VIII and Table IX. The first two tables are for the non-stationary workloads while the third is for five concatenated stationary traces (this trace is produced by a pattern generator where each trace has a constant spectrum. As an example for concatenating a stationary trace to another trace see Appendix A.) As evident from the results, the proposed method has been reduced the energy consumption of the system about 28.2% to 35.1% for the non-stationary workloads compared to the best savings which is for the NSMDP. In the case of the concatenated stationary process, the energy saving is about 11.32% when compared to NSMDP. The lower energy saving was expected due to fact that the wavelet based technique outperforms other techniques considerably when the workload is non-stationary. It should be noted that the proposed method has only between 4 and 7.7% less energy saving when compared to the oracle policy.

Table VII. Comparison of the power dissipation and the performance evaluated by the simulation of different DPM policies for the traces collected from hard disk of the desktop.

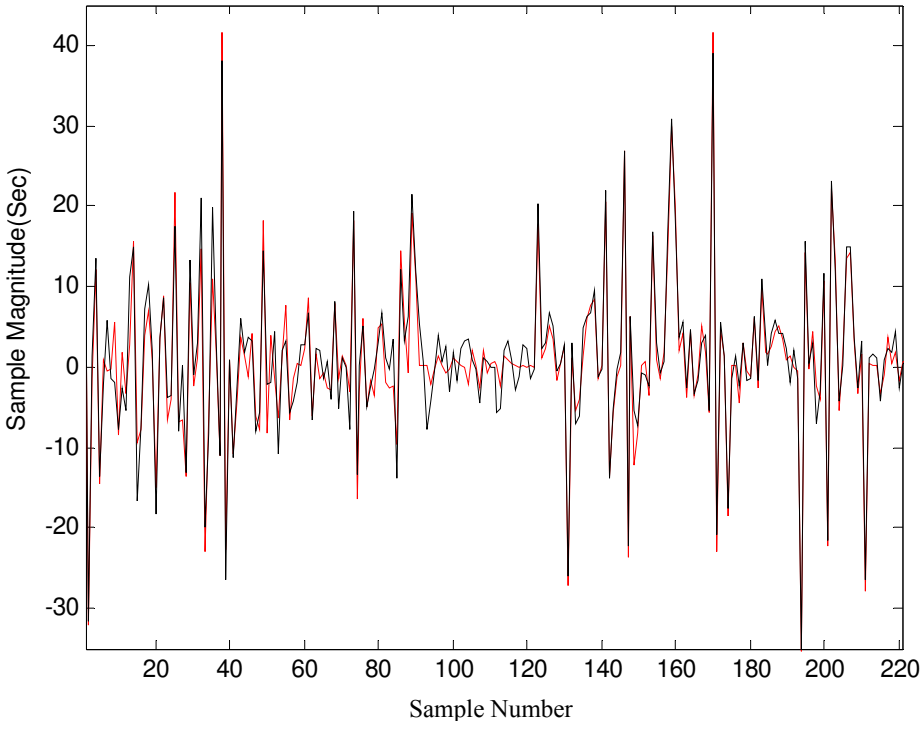
| Algorithm | *Pwr* (W) | *HR* (%) | *EN* (J) | *EF* | *Ntr* | *Nwtr* | *Tss* |
|---|---|---|---|---|---|---|---|
| Oracle | 0.42 | 100 | 1 | 1 | 522 | 0 | 1170 |
| WBDPM | 0.432 | 97.2 | 1.077 | 0.928 | 428 | 12 | 1010 |
| NSMDP | 0.5 | 82 | 1.38 | 0.724 | 422 | 76 | 916 |
| TISMDP | 0.55 | 71 | 1.75 | 0.571 | 586 | 170 | 1390 |
| Adaptive Learning Tree (ADLT) | 0.58 | 68 | 2.24 | 0.446 | 437 | 140 | 878 |
| timeout | 0.45 | 81 | 1.142 | 0.87 | 273 | 52 | 644 |
| DTMDP | 1.67 | 59 | 3.51 | 0.28 | 434 | 178 | 752 |
| Always on | 4.8 | --- | 12 | 0.083 | --- | --- | --- |
| CTMDP | 5.2 | 22 | 15 | 0.066 | 271 | 212 | 248 |

Table VIII. Comparison of the power dissipation and the performance evaluated by the simulation of different DPM policies for the traces collected from hard disk of the laptop.

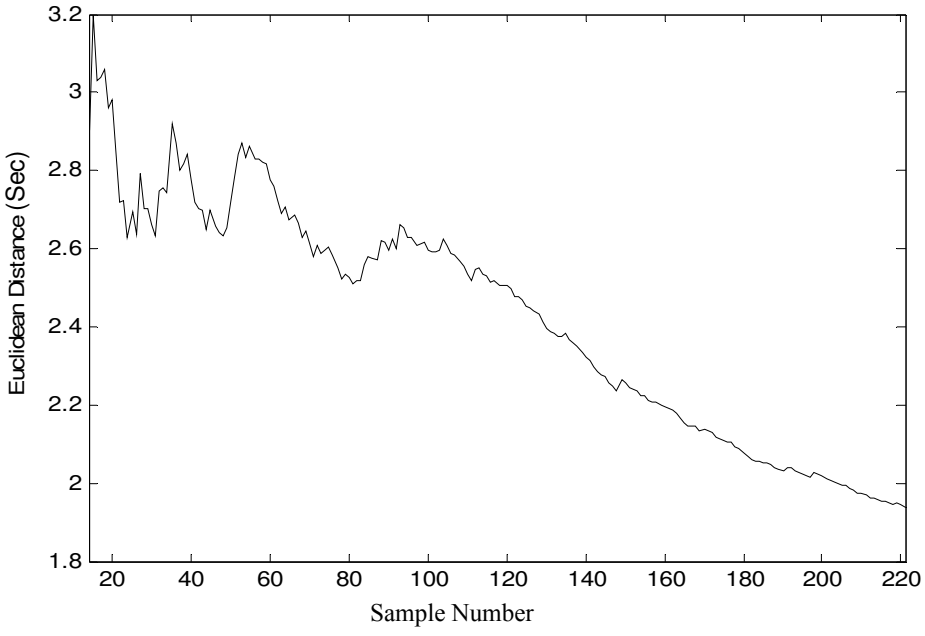
| Algorithm | *Pwr* (W) | *HR* (%) | *EN* (J) | *EF* | *Ntr* | *Nwtr* | *Tss* |
|---|---|---|---|---|---|---|---|
| Oracle | 0.175 | 100 | 1 | 1 | 382 | 0 | 820 |
| WBDPM | 0.179 | 98.2 | 1.040 | 0.962 | 444 | 8 | 830 |
| NSMDP | 0.21 | 75 | 1.404 | 0.712 | 136 | 34 | 890 |
| TISMDP | 0.31 | 67 | 1.757 | 0.571 | 236 | 78 | 620 |
| Adaptive Learning Tree(ADLT) | 0.42 | 58 | 2.153 | 0.465 | 202 | 85 | 719 |
| timeout | 0.187 | 78 | 1.250 | 0.803 | 127 | 28 | 289 |
| DTMDP | 0.72 | 52 | 2.38 | 0.42 | 416 | 200 | 410 |
| Always on | 0.95 | 0 | 5.5 | 0.18 | 0 | 0 | 0 |
| CTMDP | 1 | 18 | 8.3 | 0.12 | 171 | 141 | 74 |

Table IX. Comparison of different DPM policies for five concatenated stationary traces.

| Algorithm | *Pwr* (W) | *HR* (%) | *EN* (J) | *EF* | *Ntr* | *Nwtr* | *Tss* |
|---|---|---|---|---|---|---|---|
| Oracle | 0.154 | 100 | 1 | 1 | 268 | 0 | 620 |
| WBDPM | 0.158 | 96.8 | 1.042 | 0.974 | 218 | 7 | 640 |
| NSMDP | 0.161 | 97.2 | 1.16 | 0.954 | 392 | 11 | 615 |
| TISMDP | 0.174 | 84 | 1.28 | 0.87 | 150 | 24 | 572 |
| Adaptive | 0.188 | 79 | 1.49 | 0.78 | 180 | 38 | 654 |
| 12s timeout | 0.21 | 82 | 1.7 | 0.63 | 177 | 32 | 347 |
| 24s timeout | 0.24 | 71 | 2.17 | 0.44 | 203 | 59 | 484 |
| DTMDP | 0.220 | 78 | 1.78 | 0.57 | 186 | 41 | 520 |
| Always on | 0.74 | --- | 4.8 | - | - | - | - |
| CTMDP | 0.192 | 83 | 1.61 | 0.77 | 258 | 44 | 670 |




(a)




(b)

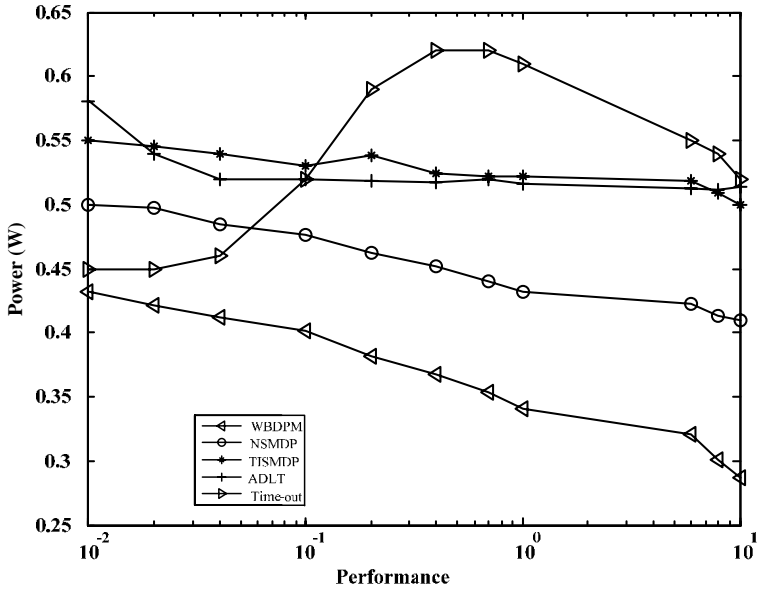
Figure 12: (a) HDD actual (black line) and predicted sequence (red line) traces, (b) Euclidean distance between the predicted values and the actual values for Figure 12(a). Average Error ($i$) = (Euclidean distance between the signal and the predicted signal from first sample to the $i^{th}$ sample)/$i$ [18].

### *B. Power-Performance Tradeoff*

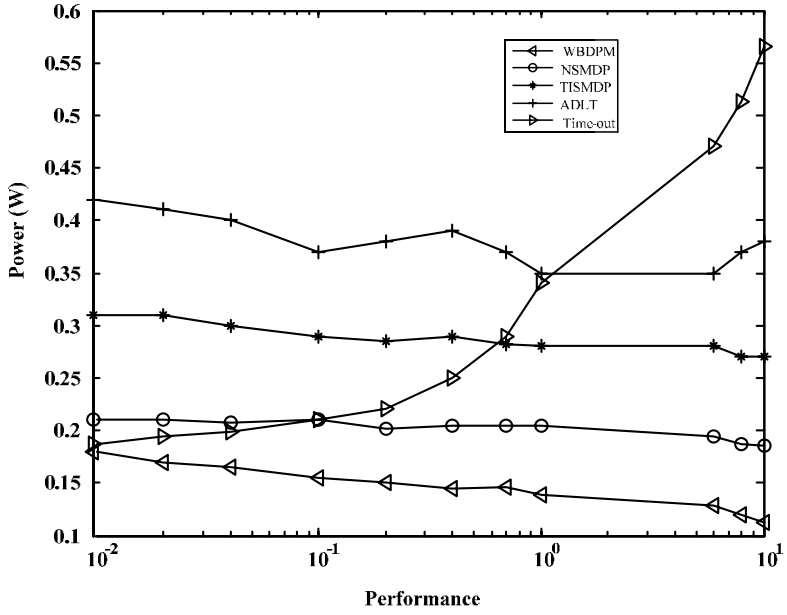
To compare the ability of the DPM policies in reducing the power dissipation of the hard disks under different performance constraints, the power-performance characteristics for the hard disks are plotted in Figure 13. It should be noted that the input traces of the hard disk was tested using the test of local stationary where it was observed that the input trace was fully non-stationary. As the results show, for a given performance, the proposed technique leads to a less power consumption. The improvement in the DPM policy may be attributed to the dynamic nature of the technique which is more suitable for non-stationary behaviors.

Regarding the time-out policy, it should be noted that this policy has a single parameter $T_{out}$ and is not a robust method for different performance constraints. In this policy, the DPM unit switches the SP from "idle" to "sleep" after the SP has been in the "idle" state (the SQ is empty) for a time period of $T_{out}$. The DPM unit switches the SP from the "sleep" to "active" immediately after a request arrives. In our experimental results, we have used different values for $T_{out}$ to study the performance and the power characteristics of the "time out" policy, and the optimum values for $T_{out}$ which minimizes the power consumption subject to the performance constraint of $Prf_C = 0.01$ has been chosen.

From Figure 13(a) and Figure 13(b), it is evidence that the proposed WBDPM outperforms the other methods including stochastic, adaptive, and time-out methods. This is due to the non-stationary feature of the method and its dynamic behavior of the system. The WBDPM can always trade the performance for the power and is robust for non-stationary applications. Also, it should be noted that the stochastic methods as well as the adaptive learning tree method, similar to the WBDPM, can always trade the performance for the power. Nevertheless, the heuristic policies such as the time-out policy can not provide a valid




(a)



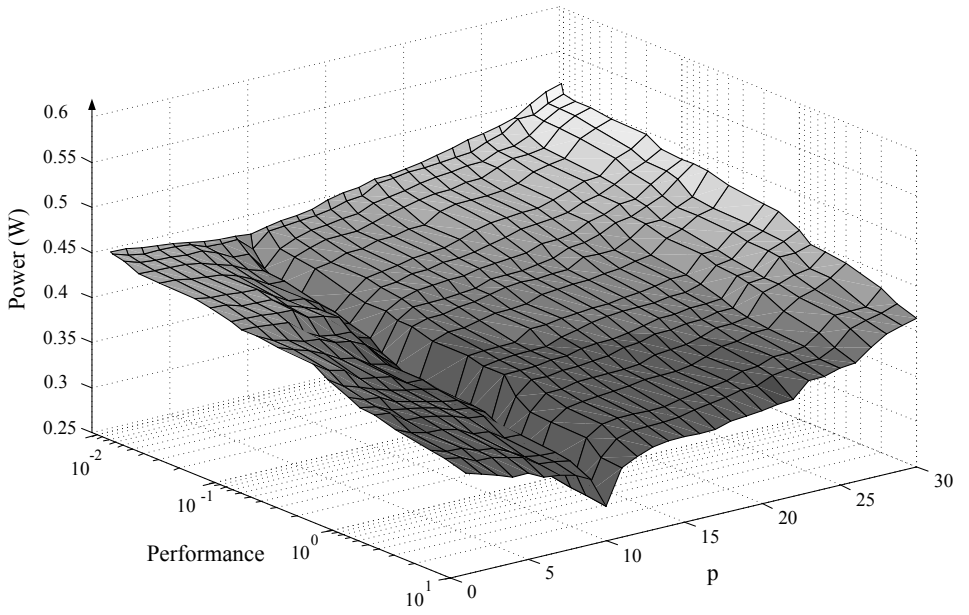

(b)

**Figure 13.** Power consumption as a function of the performance of the hard disk of (a) the desktop (b) the laptop.

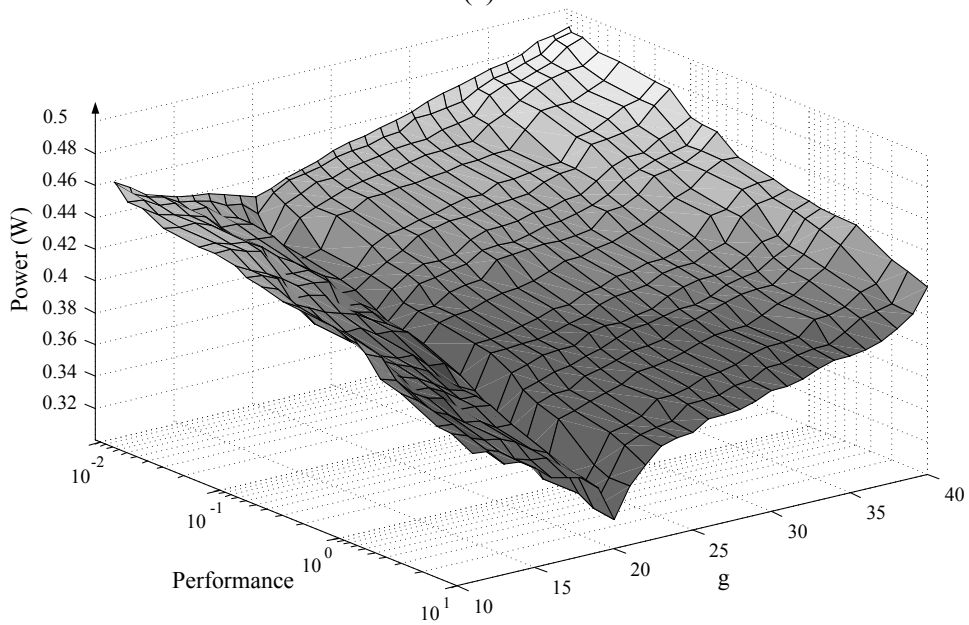
power-performance trade-off. In all the experiments, decreasing the time-out threshold increase the average latency for the request. On the other hand, decreasing the time-out threshold does not essentially decrease the average power consumption [5]. It should be noted that for some applications, there could be an optimal time-out threshold which could minimize the average request latency and the power consumption. However, if the system has different applications with different associated optimal time-out thresholds, then time-out policy may not work very well.

When the workloads are greatly non-stationary, the predictions made by the methods that assume stationary or piecewise stationary workloads would be less accurate. Adaptive learning tree uses an idle grouping method for classification the length of the idle period, and then predicting it. In fact, the method uses some stationary methods whose statistical properties are known in advance and try to extract the idle time duration using the models [3]. The main shortcoming of the method is that, not only it is not possible to provide a large library of the stationary methods, but there could be a workload with non-stationary behavior which is not similar to any of the known stationary work load. Besides, even if such data is provided, the processing time for making the decision time will be long which may prohibit performing it at the runtime. In these cases, a simple method such as the time-out policy may work more efficiently than these methods if the time is selected optimally. It, however, should be noted that the time-out policy is not a robust method for different performance constraints as is discussed here. Note that the results shown in Figure 13 reveal that when the performance constraint decreases, the efficiency of the time-out policy decreases significantly compared to other methods. Also, note that for stationary or piecewise stationary workloads, the performance of the time-out policy will decrease drastically compared to the previous methods. As an example, the results for this type of workloads are presented in Table IX,

where the upper bound of the performance is 0.01. As is inducted from these results, the performance of the time-out policy is the worse among the DPM policies.




(a)




(b)

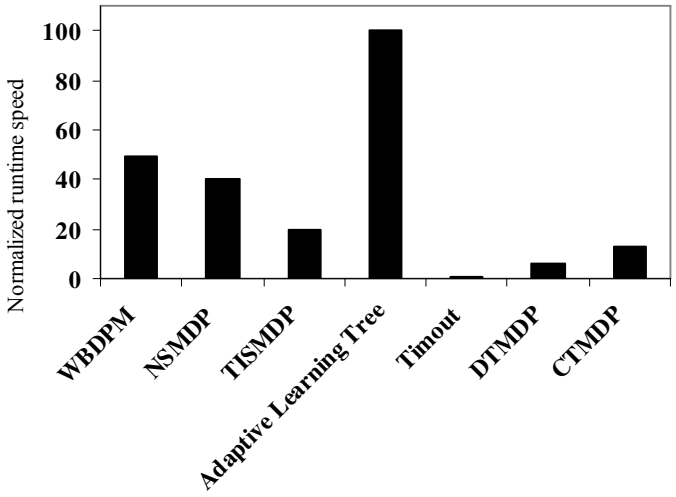
Figure 14. Power dissipation of the desktop hard disk versus the Performance and (a) *p*, (b) *g*.

### *C. Optimum Values of p and g versus Performance*

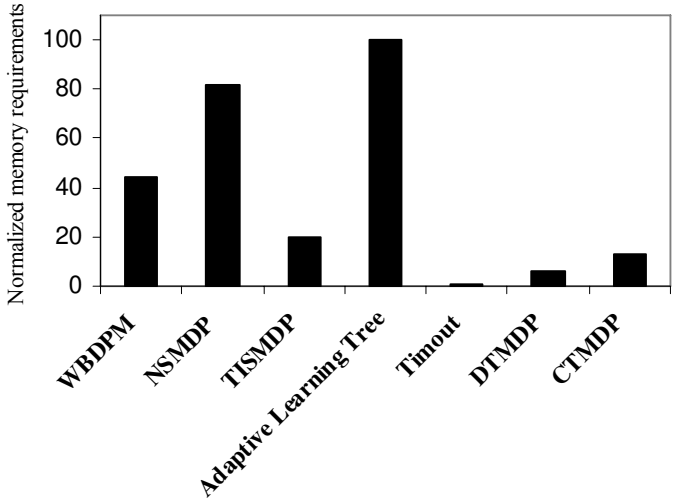
The 3D plots of the power versus the performance and $p$ ($g$) is given in Figure 14(a) (Figure 14(b).) The plots show that the variations of the optimum values of these parameters as functions of the performance. It is evident from figures that for different performance constraints, the $p$ and $g$ have almost the same optimal values. If, however, for any reason the prediction accuracy decreases, there is a routine included in the DPM unit which finds and updates the optimum values of these parameters.

### *D. Runtime Speed and Memory Requirement.*

The runtime speed and the memory requirement of the DPM policies have been compared with those of others in Figure 15. The algorithms were implemented using a laptop computer with Pentium 4 CPU (2GHz) which had 512MB of RAM. As discussed in Section I, the power consumed in the processing unit of a system normally would be smaller than those of other parts such as the display and the hard disk drive of the system. Therefore, a DPM policy with a higher computation power but with a higher accuracy could lead to more power savings and hence would be attractive. The computational overhead of the proposed method is not too high. This was achieved by using the Haar wavelet which has a very low computational overhead. Also, as was illustrated before (test of local significance), only a few scales in the spectrum are active and other scales can be neglected. In addition, a major portion of the computations is common between two successive predictions of the idle times and only a small amount of the computation should be repeated in each step of the prediction. The common parts of the calculation include the calculation of the wavelet coefficients, the auto-coloration coefficients, and the smoothing step. For example, when a new point of event




(a)




(b)

**Figure 15**. Comparison of normalized (a) runtime speed (b) memory (in the range of sub-KB) requirement for different DPM algorithms respect to adaptive learning tree.

is concatenated to the event sequence, all the calculated wavelet coefficients are valid and are used for the wavelet spectrum evaluation. In this case, only one new wavelet coefficient should be calculated for each level.

To further evaluate the accuracy of the proposed model, we generated fifty non-stationary sequences and applied them to different DPM algorithms. To generate these sequences, fifty

non-stationary spectra (such as the variations of the spectrum shown Figure 17(a) in Appendix A concatenated with white noise) were used. The power dissipation of each time sequence is calculated by averaging ten independent time series realizing each non-stationary spectrum. Figure 16 shows the results of the comparison of the power dissipations when different DPM policies are employed. It is observed that the power dissipation of the system managed by the proposed method is close to that of the Oracle policy.

## VI. SUMMARY AND CONCLUSION

In this work, a wavelet based dynamic power management (WBDPM) policy was proposed. On contrary to the previous approaches, a stationary behavior was not presumed for the service requests to the device and, hence, the idle time prediction accuracy was improved. To determine if the service request behavior of the device is a local stationary process, a test called local stationary was utilized. Using another test called the test of local significance, it was shown that only a few scales in the wavelet spectrum were active which enabled us to ignore insignificant scales. This led to a sparse representation of the SR behavior improving the efficiency of the technique. The technique used two parameters which can be adjusted during the runtime based on the data (data-driven) to have more accurate idle times predictions. The effectiveness of the approach was tested on the hard disk drives (HDD) of a laptop and a desktop computer revealing a very good accuracy for the idle time prediction of the method under different performance constraints.

## ACKNOWLEDGMENTS

The authors wish to thank Dr. Sébastien Van Bellegem of Université catholique de Louvain (Institut de statistique) for the valuable help regarding the wavelet implementation.

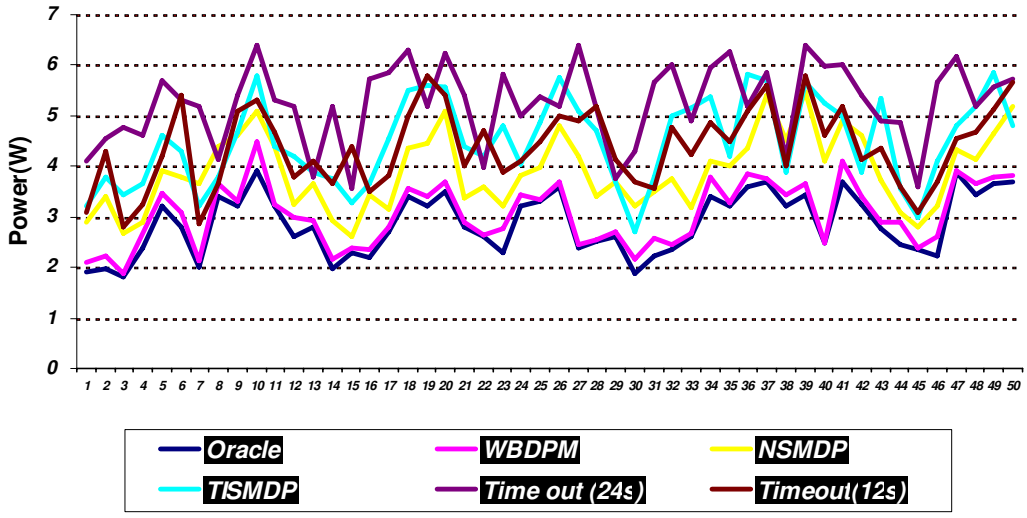



**Figure 16.** A comparison of the power dissipations in different policies. The x-axis shows the partition number out of 50 in time series.

## REFERENCES


[1] E. Chung, L. Benini, and G. De Micheli, "Dynamic power management for nonstationary service requests," in *Proceedings of Design and Automation in Europe*, pp. 77–81, 1999.

[2] G. Paleologo, L. Benini, A. Bogliolo, and G. D. Micheli, "Policy Optimization for Dynamic Power Management," *IEEE Transactions on Computer Aided Design of Integrated Circuit and System*, vol. 18, no. 6, pp. 813-833, June, 1999.

[3] Eui-Young Chung, Luca Benini, Alessandro Bogliolo, Yung-Hsiang Lu, and Giovanni De Micheli "Dynamic Power Management for Nonstationary Service Requests," *IEEE Transactions on Computers*, vol. 51, no. 11, pp. 1345-1361, November 2002.

[4] Y.-H. Lu, E.-Y. Chung, T. Simunic, L. Benini, and G. D. Micheli. "Quantitative Comparison of Power Management Algorithms," *in Proceedings of Design Automation and Test in Europe*, March 2000, Paris, France, pp. 20-26.

[5] Q. Qiu, Q. Wu, and M. Pedram, "Stochastic modeling of a power-managed system: Construction and optimization," in *Proc. Int. Symp. Low Power Electronics and Design*, August 1999, pp. 194–199.

[6] E. Chung, L. Benini, G. De Micheli, "Dynamic Power Management Using Adaptive Learning Tree," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design,* pp. 274-279, 1999.

[7] Y.-H. Lu and G. De Micheli, "Comparing System-Level Power Management Policies," *IEEE Transactions on Design & Test of computer,* vol. 18, no. 2, pp. 10-19, March-April 2001, pp. 10-19.

[8] E. Chung, L. Benini, G.D. Micheli, "Dynamic Power Management Using Adaptive Learning Tree," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design,* pp. 274-279, 1999.

[9] T. Simunic, L. Benini, and G. De Micheli, "Dynamic Power Management of Portable Systems," in *Proc. ACM Int. Conf. Mobile Computing and Networking*, pp. 11–19, August 2000.

[10] D. Ramanathan and R. Gupta, "System Level On-Line Power Management Algorithms," in *Proc. Design Automation and Test in Europe IEEE Conference and Exhibition,* pp. 606-611, March 2000.

[11] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time markov decision processes," in *Proc. ACM/IEEE Design Automation Conf.,* pp. 555–561, June 1999.

[12] T. Simunic, L. Benini, P. Glynn, G. De Micheli, "Event Driven Power Management," *IEEE Transaction on Computer Aided Design of Integrated Circuit and System*, vol. 20, no. 7, pp. 840-857, July 2001.

[13] Zhiyuan Ren, Bruce H. Krogh and Radu Marculescu, "Hierarchical Adaptive Dynamic Power Management," *IEEE Transactions on Computers*, vol. 54, no. 4, April 2005.

[14] C.-H. Hwang and A. Wu, "A predictive system shutdown method for energy saving of event-driven computation," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 28–32, Nov. 1997.

[15] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning,* 81-106, 1986.

[16] S. Van Bellegem, P. Fry_zlewicz, and R. von Sachs, "A wavelet based model for forecasting non-stationary processes," in *J-P. Gazeau, R. Kerner, J-P. Antoine, S. Metens and J-Y. Thibon (Eds..) GROUP 24: Physical and Mathematical Aspects of Symmetries,* Conference Series Number 173, Bristol 955-958, 2003.

[17] P. Fryzlewicz, S. Van Bellegem, and R. von Sachs, "Forecasting non-stationary time series by wavelet process modeling," in *Annals of the Institute of Statistical Mathematics,* vol. 55, no. 4, pp. 737-764, 2003.

[18] S. Van Bellegem, "Adaptive Methods for Modeling, Estimating and Forecasting Local Stationary Processes," *Thesis, Univ. Catholics De Louvain Institute de Statistics,* December, 2003. Available online at http://edoc.bib.ucl.ac.be:81/ETD-db/collection/available/BelnUcetd-12102003-125105/unrestricted/thesis.pdf.

[19] Qingbo Zhu, Francis M. David, Christo F. Devaraj, Zhenmin Li, Yuanyuan Zhou, Pei Cao. "Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management," *Proceedings of the 10th International Symposium on High Performance Computer Architecture*, Page: 118, 2004

[20] R. von Sachs, G. P. Nason, and G. Kroisandt, "Adaptive estimation of the evolutionary wavelet spectrum," Technical Report 516, Department of Statistics, Stanford, 1997. http://www.stats.bris.ac.uk/pub/reports/Wavelets/StanTechRep516.ps.gz.

[21] C.S. Burrus, R.A. Gopinath, H. Guo, *Introduction to Wavelets and Wavelet Transform, A Primer*, Prentice Hall, New Jersey, 1998.

[22] A.W. Bowman and A. Azzalini, *Applied Smoothing Techniques for Data Analysis*, *Clarendon Press,* 1997.

[23] J. E. Gentle, *Numerical Linear Algebra for Applications in Statistics*, *Springer-Verlag, Berlin,* pp. 93-95, 1998.

[24] http://product.samsung.com/cgi-bin/nabc/product/b2c_product_detail.jsp?eUser =&prod_id =SV4002H&selTab=Specifications.

[25] http://www.toshiba-europe.com/storage/Index.asp?page=PCI&nav=ISH_PRS&model= MK4026GAX.

[26] O. Renaud, J.-L. Starck, and F. Murtagh, "Wavelet-based forecasting of short and long memory time series" Department d'econometrie, University of Genève, 2002.04. Available online at http://www.unige.ch/ses/metri/cahiers/2002_04.pdf.

# Appendix A

## An Example for Demonstrating the Technique

To demonstrate the wavelet based DPM technique, we use an example of a time series whose spectrum is shown in Figure 17(a.) The series may be synthesized, for example, by concatenating a time modulated process ($j$ = –1) and a simple stationary process ($j$ = –5) which is depicted in Figure 17(a.) The sample size is $T$ = 1000. This non-stationary time series corresponds to the wavelet process generated using the spectrum given by

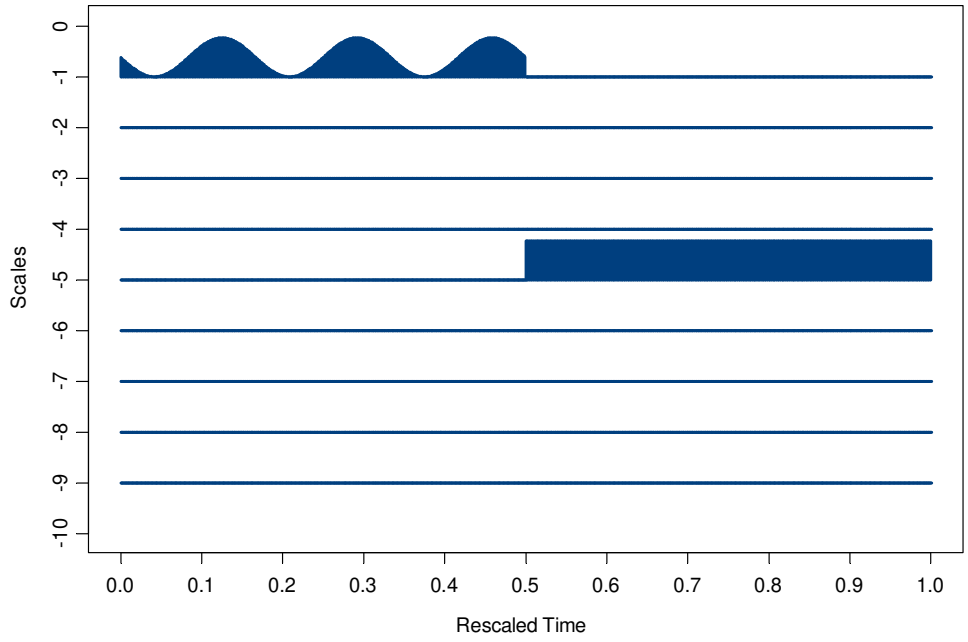
$$S(z) = \begin{cases} 0.25 + \cos^2(3\pi z + 0.25\pi) \text{ if } z \le 0.5 \text{ and } j = -1 \\ 1 \text{ if } z \ge 0.5 \text{ and } j = -5 \end{cases} \quad (18)$$

where the boundary between the two concatenated processes is at time $z$ = 0.5. For our purpose of having a non-stationary process, the boundary point may be placed at any point in the rescaled time. The spectrum of this process shows that it is non-stationary in one segment of time ([0,0.5]) and stationary in the other segment ([0.5,1].) Since at each point of the time, only one scale is nonzero, the wavelet spectrum $S(z)$ is considerably sparse. Figure 17(c) also show the average of 100 uncorrected wavelet periodograms ($I_{j,k}$) obtained using (8) from 100 independent series obtained from the realizations of the spectrum $S(z.)$ When compared to the spectrum given in Figure 17(a), it is understood that the uncorrected wavelet periodogram given in Figure 17(c) is biased and does not have a good accuracy for the estimation of the wavelet spectrum. The periodogram obtained by averaging the 100 corrected wavelet periodograms ($L_{j,k}$) is depicted in Figure 17(d.) This periodogram is an unbiased estimation of the spectrum with a fair accuracy. Figure 17(e) shows a single corrected wavelet periodogram (CWP) from one realization of $S(z)$ which is computed using (9).

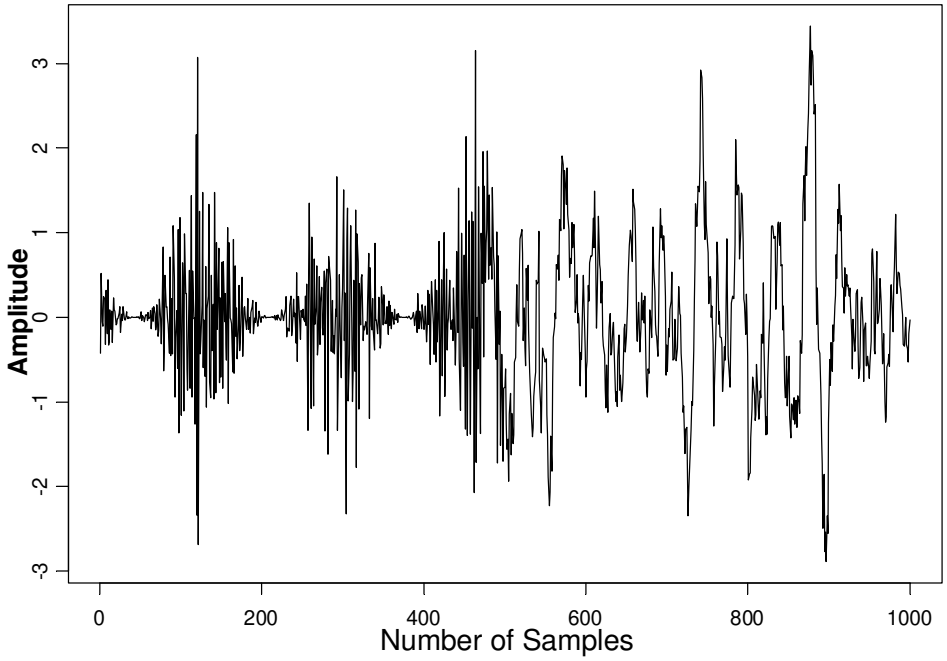
Now the local significance and the stationary tests are performed on the 100 time series. First, we apply the test of local significance to the CWP. For each of the 100 series, we have

computed the test statistics defined by $\Im_T = |Q_{j,\Re,T}|/\sigma_{j,\Re,T}$ for scales $j = -1$ to $j = -9$. The computation is performed on two sub-intervals in $\Re$. In the test statistics, $Q_{j,\Re,T}$ is the averaged corrected wavelet periodogram on interval $\Re$ and $\sigma_{j,\Re,T}$ is the standard deviation of $Q_{j,\Re,T}$ [18]. Using the test statistics $\Im_T$, a parameter "*p*-value" indicating the significance of each scale can be calculated [18]. A box plot of the 100 *p*-values obtained from the 100 independent time series in intervals $\Re = (0,1)$, $\Re = (0,0.5)$, and $\Re = (0.5,1)$ are illustrated in Figure 18. Based on the theoretical discussions of [18], these figures show that very small *p*-values are obtained for the regions with non-zero spectrum and large *p*-values correspond to the sparse regions.

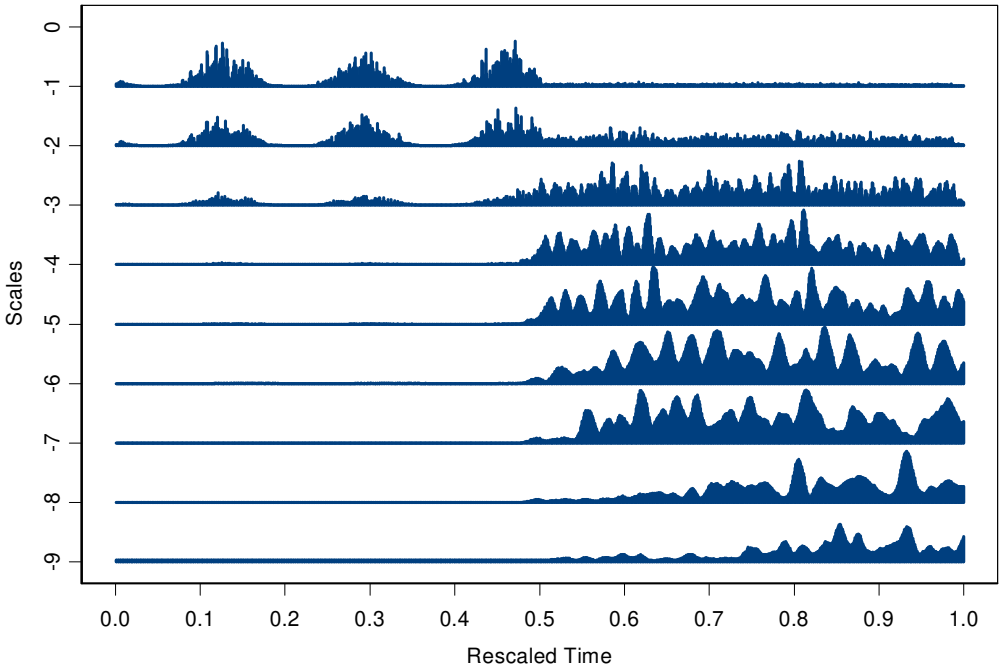
As the next experiment, to compare the prediction accuracy of the proposed method with those of other methods, we used ten independent time series ($T = 1000$) obtained from the spectrum given in Figure 17(a.) These series are then converted to a pattern as entry requests for the SR in the system managed with DPM. The prediction accuracy of the WBDPM is compared to Oracle, NSMDP, TISMDP, and Timeout policies. For the comparison purpose, we partitioned the spectrum $S(z)$ into twenty equal segments where in each segment, we generate ten series with the length of fifty samples based on the corresponding part of the spectrum. Then, for each of the ten series, the prediction accuracies for the abovementioned methods are calculated and averaged to obtain the prediction accuracy of the method. The result of this comparison is given in Figure 19. As it is evident from the figure, the proposed method has a higher accuracy compared to previous methods.


Scales
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Rescaled Time

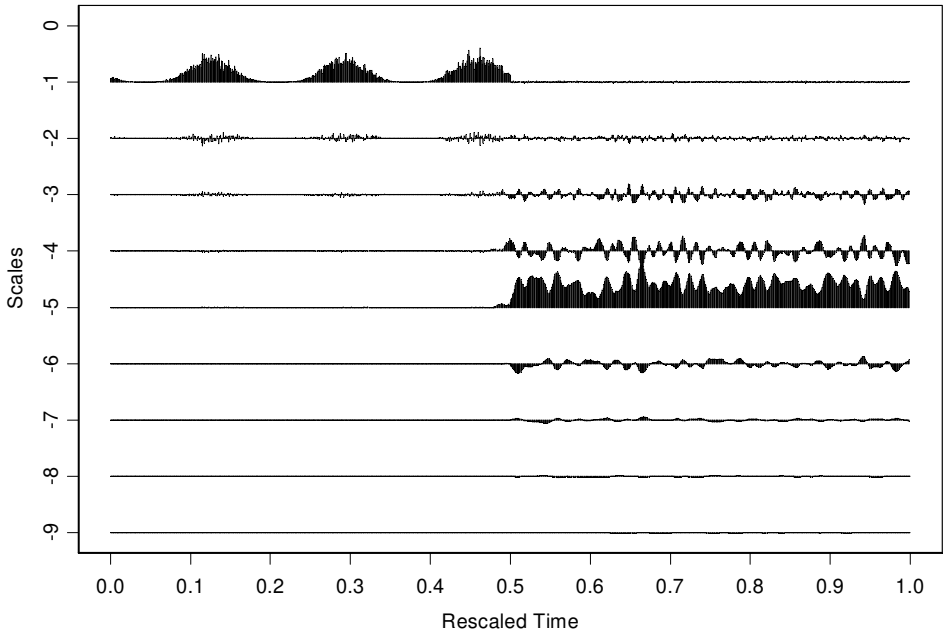
(a)


Amplitude
3
2
1
0
-1
-2
-3
0
200
400
600
800
1000
Number of Samples

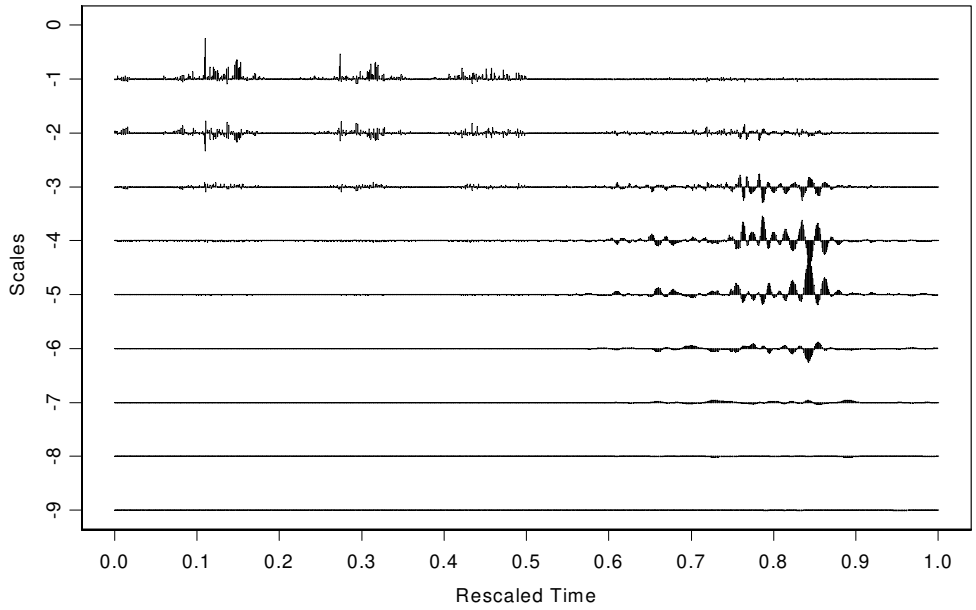
(b)


Scales
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Rescaled Time

(c)

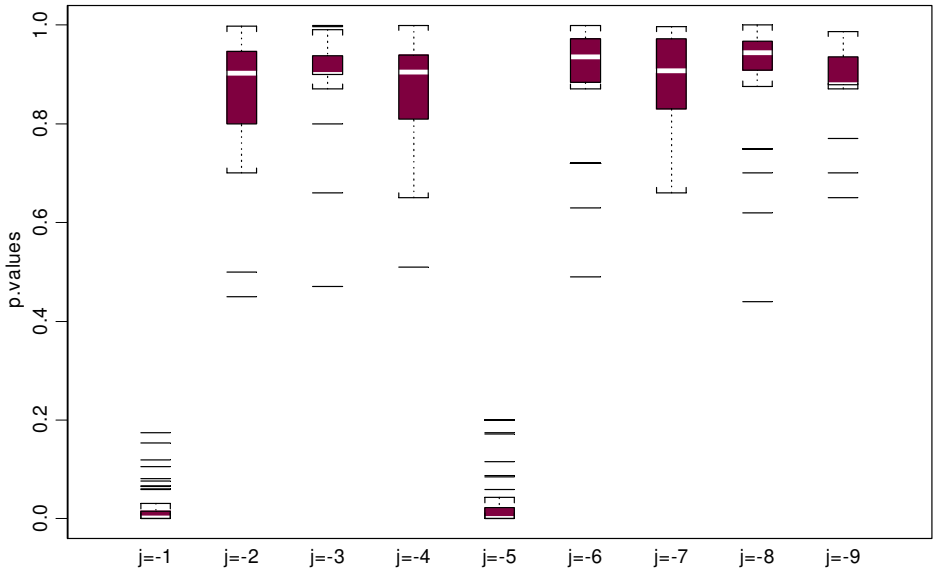

Scales
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Rescaled Time

(d)

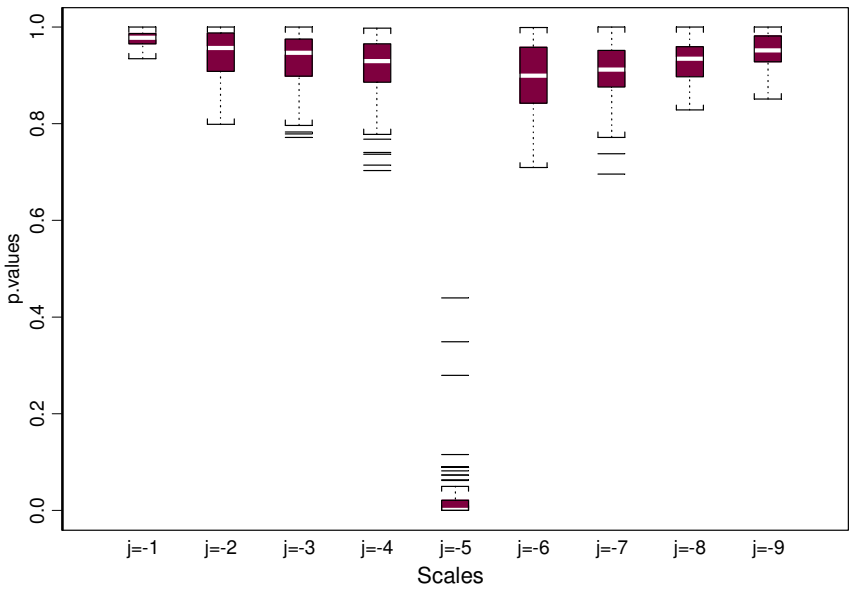



(e)

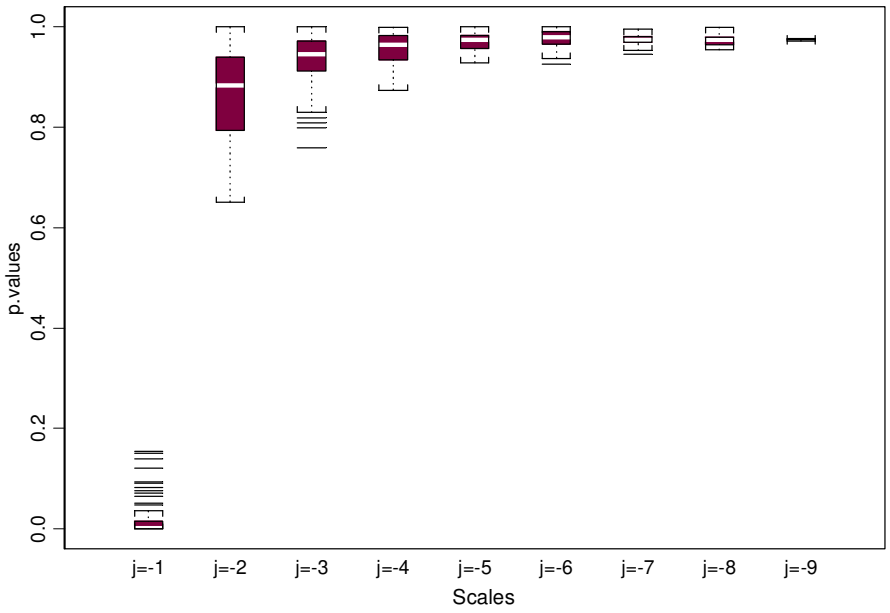
**Figure 17.** (a) The wavelet spectrum $S(z)$, (b) one realization of Spectrum $S(z)$, (c) the mean of uncorrected wavelet periodograms from 100 independent realization of $S(z)$, (d) the mean of corrected wavelet periodograms from 100 independent realization of $S(z)$, (e) The corrected wavelet periodograms of a single realization of $S(z.)$


p.values
0.0
0.2
0.4
0.6
0.8
1.0
j=-1
j=-2
j=-3
j=-4
j=-5
j=-6
j=-7
j=-8
j=-9

(a)


p.values
0.0
0.2
0.4
0.6
0.8
1.0
j=-1
j=-2
j=-3
j=-4
j=-5
j=-6
j=-7
j=-8
j=-9
Scales

(b)




(c)

**Figure 18.** (a) the *p*-value of the test of local significance for realization of $S(z)$ computed on $\Re = (0,1)$, (b) *p*-value of test of local significance for realization of $S(z)$ computed on $\Re=(0,0.5)$, (c) *p*-value of test of local significance for $S(z)$ computed on $\Re = (0.5,1.)$

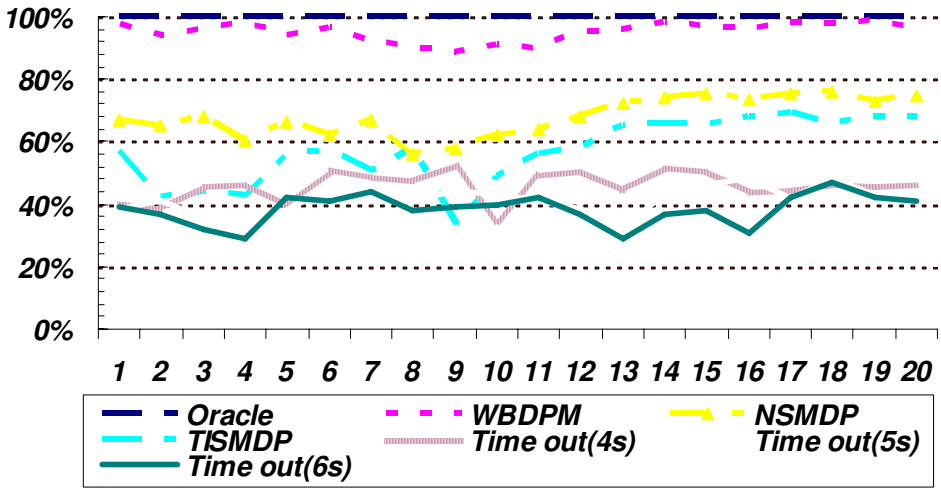



**Figure 19.** Comparison of the prediction accuracies of various approaches.

## Appendix B

### An Example for Showing the Concepts in Section III

Assume that we choose the $p = 5$ (the history size). Also, for the sake of simplicity, suppose that we have only two significance scales. For this case, assume that the lengths of the last $p$ idle events are [0.2, 10, 20, 5, 3] (sec). To make a mean zero sequence from this input trace, we subtract the mean of the data from each sample which to a zero-mean trace as [-7.44, 2.36, 12.36, -2.64, -4.64] (sec).

| Sequence of idle event lengths (*sec*) | | | | | | Zero-mean sequence obtained from the idle events (*sec*) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.2 | 10 | 20 | 5 | 3 | → | -7.44 | 2.36 | 12.36 | -2.64 | -4.64 |

It should be noted that, in theory, the mean of the sequence from time = 0 to time = ∞ must be zero. The mean of the last $p$ is not necessarily equal to the mean of the complete sequence. For practical implementation purposes, we use the mean of the last $p$ samples as the mean of the sequence. The Graam matrix "$A$" is independent of the input data and is only dependent on the mother wavelet function. The matrix should be calculated using (7) as

$$A_{jl} = \left\langle \Psi_j, \Psi_l \right\rangle = \sum_{\tau} \Psi_j(\tau)\Psi_l(\tau)$$

As mentioned earlier, we assume that there are only two active scales. This leads to

$$A = \begin{bmatrix} 1.5 & 0.75 \\ 0.75 & 1.75 \end{bmatrix}_{J\times J}$$

Using (9), we have

$$L_{j,T}\left(\frac{k}{T}\right) = \sum_{l} (A_T)^{-1}_{jl} \left( \sum_{t=0}^{T-1} X_{t,T}\psi_{lk}(t) \right)^2$$

The corrected wavelet periodogram ($L$) will be given by

$$L = \begin{bmatrix} 20.83 & 20.84 & 22.51 & 75.54 & -42.3 \\ 22.36 & 22.28 & 21.64 & -1.083 & 87.27 \end{bmatrix}_{J\times P}$$

Using the $L$ matrix and the following formula, the Local Auto covariance (C matrix) is obtained from (6)

$$c\left(\frac{k}{T}, \tau\right) = \sum_{j=-J}^{-1} \left( \sum_{l=-J}^{-1} A^{-1}_{jl} \Psi_l(\tau) \right) L_j\left(\frac{k}{T}\right)$$

which leads to

$$C = \begin{bmatrix} 49.904945 & -1.684186 & -17.859106 & -10.751326 & 0.000000 \\ -1.684186 & 49.961922 & -1.573939 & -17.947172 & -10.795948 \\ -17.859106 & -1.573939 & 50.018524 & -1.462349 & -18.035637 \\ -10.751326 & -17.947172 & -1.462349 & 50.074749 & -1.349425 \\ 0.000000 & -10.795948 & -18.035637 & -1.349425 & 50.130569 \end{bmatrix}_{P\times P}$$

Using matrix **C** and (12) yields

$$\sum_{m=0}^{t-1} b^{(1)}_{t-1-m,T} c\left(\frac{n+m}{2T}, m-n\right) = c\left(\frac{n+t}{2T}, t-n\right)$$

The $b$ coefficients will be calculated using (10) as

$$b = [-0.2798692 \;\; -0.2712338 \;\; -0.4321834 \;\; -0.5386909 \;\; -0.2530397]_{1\times P}$$

Finally, the next event will be predicted based $b$ values using

$$\hat{X}_{t,T}^{(p)} = \sum_{s=t-p}^{t-1} b_{t-1-s,T}^{(t)} X_{t-s,T}$$

Therefore, the next (zero mean) idle length is $X6$ = -1.3 and, hence, $\hat{X}6$ (= $X6$ + *mean*) becomes = 8.94sec.