Computer Networks 56 (2012) 548–565

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

An adaptive hybrid dynamic power management algorithm for mobile devices

Hung-Cheng Shih, Kuochen Wang [*]

Department of Computer Science, National Chiao Tung University, 1001, University Road, Hsinchu 30010, Taiwan

ARTICLE INFO

Article history:
Received 29 October 2010
Received in revised form 10 August 2011
Accepted 12 October 2011
Available online 20 October 2011

Keywords:
Hard disk
Mobile device
Power management
Self-similarity
WLAN NIC

ABSTRACT

We propose a novel power efficient *adaptive hybrid dynamic power management* (AH-DPM) algorithm. To adapt well to bursty request arrival patterns with self-similarity and a service provider (SP, i.e., hard disk or WLAN NIC, in this paper) with multiple inactive states, the proposed AH-DPM first derives the average idle time of the SP in the bursty (ON) period and non-bursty (OFF) period separately. Then, to achieve better power saving, we use the average idle time in the ON period to adjust the timeout value more precisely and use the average idle time in the OFF period to decide which inactive state the SP should be switched to. Experimental results based on real traces show that, for the hard disk, the average power consumption of the proposed AH-DPM is better than that of the Adaptive Timeout (ATO), Machine Learning (ML), Predictive, Static Timeout (STO), and Stochastic algorithms. In addition, the average response time of the proposed AH-DPM algorithm is still lower than that specified in a typical hard disk specification. As to the WLAN NIC, experimental results show that the average power consumption of the proposed AH-DPM is comparable to that of the Oracle (theoretically optimal), ATO, and Predictive algorithms, and is better than that of the ML, STO, and Stochastic algorithms. However, the average packet transmission delay of the proposed AH-DPM is better than that of the ATO and Predictive algorithms. Therefore, by providing a better tradeoff between average power consumption and average response time (or average packet transmission delay), the proposed AH-DPM algorithm is very feasible for extending the battery lifetime of ever increasing mobile devices that are equipped with hard disks and WLAN NICs.

1. Introduction

In recent years, mobile devices such as smart phones are getting more pervasive and popular due to a wider spread of wireless internet. Because mobile devices have the characteristic of mobility, their power sources must rely on battery power. Due to increasing demands of users on performance and functionality of mobile devices, the power consumption of these devices will enlarge. Battery lifetime in mobile devices can be prolonged in two ways: increasing battery capacity per unit weight and reducing power consumption with minimal performance loss [1]. Since the battery capacity per unit weight has only improved by a factor of two to four over the last 30 years while the computational power of digital ICs has increased by more than four orders of magnitude, reducing the power consumption of components, such as hard disks and WLAN NICs, in mobile devices becomes a vital research issue. That is, in order to extend the battery lifetime, managing the power consumption of components in a mobile device is essential. Since not all components in a mobile device are active at the same time, we can switch some components to low power consumption states when they are idle for a certain period of time. The concept of dynamically switching between different states with

[*] Corresponding author. Tel.: +886 3 5712121x56613; fax: +886 3 5721490.

E-mail addresses: hcshih@cs.nctu.edu.tw (H.-C. Shih), kwang@cs.nctu.edu.tw (K. Wang).

1389-1286/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2011.10.005

different power consumption levels, called *dynamic power management* (DPM), has been introduced to the design of components in mobile devices. With this capability, we can dynamically switch a component's current working state to a low power consumption state for power saving.

1.1. Overview of the DPM

DPM is an effective approach for mobile devices to reduce power consumption without significantly degrading their performance. The DPM shuts down components when they are not being used and wakes them up when necessary [2]. With careful observation of components' state transition patterns, the DPM can predict when an idle period will likely occur. The operating system (OS) in a DPM-enabled mobile device has a module called *Power Manager* (PM). The PM is responsible for monitoring all components in a mobile device and controlling the working state of each component [1]. The PM has several power management policies, possibly one policy for one component according to its working pattern. These policies are used by the PM to decide at what time and which state a component should be transferred to. It is important that power management policies have to be adaptive because arrival patterns of service requests are usually non-stationary [16]. If we use a fixed power management policy for all possible arrival patterns, the effect of power management will not be good.

Fig. 1 is an example of a component's working pattern. When the PM decides to switch the state of the component from busy to idle for power saving, the PM has to consider the extra power consumption needed during a state transition. Take the transition from *Busy 1* to *Idle 1* as an example. Because the length of *Idle 1* is long enough, the PM decides to switch the component to a deeper sleeping state. Since switching to a deeper sleeping state will consume more state transition energy compared to switching to a shallower sleeping state, the PM must carefully evaluate the power consumption when performing a state transition. When the PM decides to switch a component to a new state, the component will not enter the new state immediately because performing a state transition takes time. The time spent during a state transition is called *state transition time*. The power consumed by a component during a state transition is a waste because there will be no request served during the state transition. In order to compensate the extra power dissipation caused by the state transition, the time which the component stays in an inactive state must be long enough. The minimum inactive time required to compensate the extra power consumed during the state transition is called *break-even time*, which is denoted as T_{BE} [1]. In general, the goal of the DPM is maximizing power saving while meeting the response time (delay) requirement of a component.

1.2. Self-similarity characteristic of hard disk and WLAN NIC workloads

A self-similar stochastic process is a stochastic process that all statistical properties remain unchanged at various observation time scales. That is, the stochastic process "looks the same" if one zooms in time "in and out" in the process [3]. The observed shape of a self-similar stochastic process in a time scale of milliseconds would be still similar to that in a time scale of seconds, hours, or even days. According to the studies of Xiang et al. [4] and Gomez et al. [5,6], both hard disk access patterns and WLAN NIC traffic are *bursty* and *self-similar*. That is, the patterns of hard disk access and WLAN NIC traffic are bursty no matter how long we observe them. Bursty arrival patterns with self-similarity can be modeled by the ON-OFF model [5,6]. In ON (bursty) periods, the hard disk (WLAN NIC) idle time is short compared with that in OFF (non-bursty) periods. If we compare the hard disk (WLAN NIC) idle time with the break-even time, we can see that there will be periods that the hard disk (WLAN NIC) idle time are shorter than the break-even time. We define these periods as ON periods. The other periods with the hard disk (WLAN NIC) idle time longer than or equal to the break-even time are called OFF periods. In the OFF periods, the hard disk (or WLAN NIC) has to be switched to a low power consumption state in order to save power.

1.3. Motivation and main contribution of this work

Based on the observation that hard disk access and WLAN NIC traffic are bursty and self-similar, it motivates us to propose a DPM algorithm to reduce the power consumption of these two components and to extend the battery of mobile devices. Related work on dynamic power management (DPM) mostly focuses on hard disks and handles only one inactive state. The uniqueness of our work is that we have developed a DPM algorithm to predict the working states of components, such as WLAN NICs as well

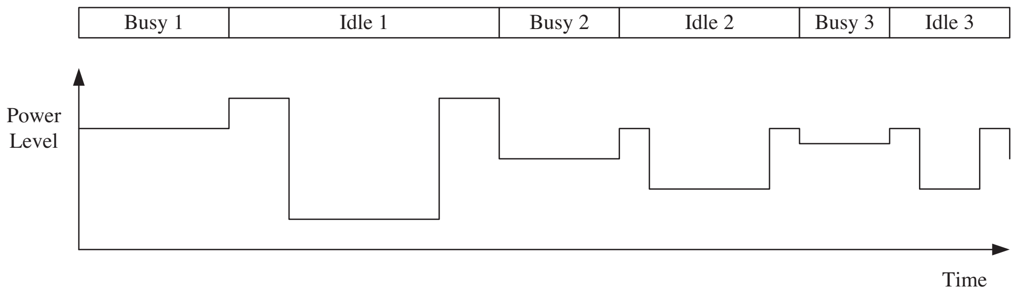


Fig. 1. An example of a component's working pattern.

as hard disks, embedded in mobile devices that have multiple inactive states. The proposed AH-DPM algorithm can adapt to the bursty request arrival patterns with self-similarity of the components in order to enhance power saving, while not affecting the average transmission delay or average response time too much. The proposed DPM algorithm handles the lengths of idle time in ON and OFF periods separately in order to adjust the timeout value more precisely and decides which inactive state the SP should be switched to, and thus it achieves better power saving. That is, the proposed AH-DPM algorithm can fully utilizes the self-similarity characteristic of disk access (or WLAN access) to predict request arrival patterns and adaptively adjust the timeout value and select an appropriate inactive state to switch to.

The main contribution of this paper is that the proposed AH-DPM algorithm can provide a better tradeoff between average power consumption and average response time (or average packet transmission delay) for hard disks and WLAN NICs and thus it is very feasible to mobile devices for extending their battery lifetime.

The remaining of this paper is organized as follows. Section 2 reviews related work of DPM algorithms. Section 3 depicts our design approach and shows the flowcharts and pseudo codes of the proposed DPM algorithm. Experimental setup, experimental results, and discussion are presented in Section 4. We give concluding remarks in Section 5.

2. Related work

Four categories of DPM policies have been proposed: *timeout*, *predictive*, *stochastic*, and *machine learning* policies [1]. Fig. 2 classifies existing DPM algorithms based on these four categories. In the following, we briefly introduce the characteristics of each category.

2.1. Timeout-based algorithms

Timeout-based algorithms can be divided into two classes: *static timeout* (STO) and *adaptive timeout* (ATO) [1]. The STO scheme turns off a component after a fixed period of idle time. Because the timeout value is fixed, the STO scheme, shown as a dotted block in Fig. 2, is not a DPM algorithm. In this scheme, the user has to decide the best timeout period manually. The ATO scheme is more efficient because it changes the timeout value according to the latest idle time. There are several adaptive timeout algorithms. In [7], it adjusts the timeout value by using the ratio of the length of the previous idle period divided by the wakeup delay. If the ratio is small, the timeout value is increased. If the ratio is large, the timeout value is decreased.

In [8], the authors proposed an OS power management technique called *PowerNap* that modifies the timing mechanism of the OS to achieve better power saving [8]. They observed that when the OS is idle, the widely used *periodic timing* (PT) scheme, which a timer will issue interrupts to the OS periodically, will cause unnecessary power dissipation [8]. The solution to this phenomenon is to eliminate the periodic timer tick whenever the OS is idle. A scheme called *Work Dependent Timing* (WDT) was proposed, which will switch the system to a low power state when there is no task to execute [8]. The WDT will determine the nearest timeout value, write it into the hardware timer, and switch the system state to a low power consumption state. When the timer expires, the hardware will issue a hardware interrupt to wake up the whole system [8]. Generally speaking, timeout schemes have two main advantages. They are general and the throughput of serving requests can be guaranteed simply by increasing the timeout value [9]. They also have two main disadvantages. They waste a lot of energy because of waiting the timeout value to expire and they always result in performance penalty when components wakeup [9].

2.2. Predictive-based algorithms

The predictive-based algorithms can be classified into two categories: *predictive shutdown* and *predictive wakeup* [9]. They were proposed to deal with the disadvantages of timeout schemes [9]. The predictive shutdown scheme predicts the length of an idle period when the PM detects that a component is going to enter the idle state. If the PM assesses that the length of the idle period will be longer than the break-even time [9], the component will be switched to a lower power consumption state immediately to eliminate the unnecessary waste of energy usually caused by timeout schemes. The predictive wakeup scheme predicts the expiration of an idle period. If the PM predicts that the idle period of a component is going to be ended in a short time, the component will be

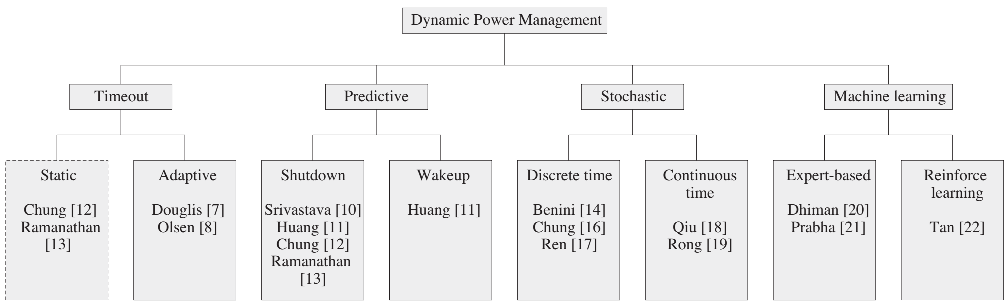


Fig. 2. The classification of existing DPM algorithms.

switched to an active state to avoid an incoming request waiting the component to switch from an inactive state to an active state.

Representative predictive-based algorithms are reviewed as follows. Srivastava et al. [10] proposed two approaches which belong to predictive shutdown [9] for a component. The first approach uses regression analysis to arrive at a model for predicting the length of idle periods [10]. The second approach is based on the observation of the phenomenon that a long duration of an active state is followed by a short duration of an idle state with a very high probability, and the probability of an idle state followed by a short duration of an active state is fairly evenly distributed [10]. In this case, the component will be shut down when the PM observes that an idle period is about to begin. These two approaches strongly rely on offline analysis of the component behavior; thus they are not adaptive.

Huang et al. [11] addressed three predictive methods: predictive shutdown using exponential average, correction of prediction misses, and pre-wakeup. In the predictive shutdown, the formula of exponential average is as follows:

$$I_{n+1} = \alpha \cdot i_n + (1 - \alpha) \cdot I_n \tag{1}$$

where I_{n+1} is the new predicted value, I_n is the last predicted value, i_n is the latest idle period, and α is a constant attenuation factor in the range between 0 and 1 [11]. In the correction of prediction misses, there are two sub-issues: under-prediction and over-prediction. Under-prediction happens when a long idle period occurs after a series of short, uniformly distributed idle periods. Over-prediction happens when a short idle period occurs after a series of long, uniformly distributed idle periods. The former situation is resolved by setting a watchdog to periodically monitor the current idle period. The latter situation is resolved by adding a saturation condition to the original algorithm. The pre-wakeup scheme is used to deal with the performance penalty due to the wakeup delay. This can be accomplished by predicting the occurrence of the next wakeup signal [11].

Chung et al. [12] proposed a DPM method using an adaptive learning tree [12]. Using the tree, the PM can accurately predict the most appropriate low-power sleep state at the start of an idle period [12]. They also proposed an enhanced scheme which adopts a fixed timeout filter in order to eliminate the unnecessary shutdown when a very short idle period occurred [12]. Ramanathan et al. [13] used the previous request inter-arrival time τ to predict the next idle period [13]. If τ is greater than the shutdown threshold k, the component will be shut down immediately because the algorithm assumes that the next idle time will be greater than k time units [13]. If τ is less than k, it keeps the component idle for a period of k unless a new request arrives [13]. This approach is similar to the algorithm proposed by Huang et al. [11] and it is a combination of STO and predictive shutdown algorithms [13]. Nevertheless, the above predictive-based algorithms suffer from the prediction accuracy of the length of idle time.

2.3. Stochastic-based algorithms

The stochastic-based algorithm proposed by Benini et al. [14] uses stochastic processes to model the behaviors of the Service Requester (SR), Service Queue (SQ), and Service Provider (SP). The overall system architecture for the DPM is shown in Fig. 3 [14]. The SR will send a request to the SP. When the SP is busy and if requests keep coming, incoming requests will be stored in the SQ. If the SQ is full, incoming requests will be discarded. The Power Manager (PM) observes the status of the SR, SQ, and SP, and it decides which command, such as shutdown, wakeup, or state-transition, will be sent to the SP. The probability models used to describe the behaviors of the SR, SQ, and SP are the main issue of the stochastic-based schemes. The more precise the probability models that describe the SR, SQ, and SP are, the more accurate the state transition decisions made by the PM will be; thus more energy of the system can be saved.

The following are representative stochastic-based algorithms. In [14], it models request arrivals and state transitions as stationary discrete-time Markov processes. The assumptions of this approach are as follows [14]:

1. The arrival of service requests can be modeled by an m-memory Markov chain. The m-memory Markov model has 2^m states, one for each possible sequence of consecutive bits.
2. The state transition delays in the SP can be modeled as random variables with a geometric distribution.
3. Model parameters and cost functions are available and accurately measured before optimization.

However, the above constraints are not likely to occur in real life. First, in most cases, the arrivals of user requests, the state transition delays, and even the service time, are non-stationary [15]. Second, the characteristic of discrete time causes additional cost for the PM because the PM must periodically wakeup to do computation. The first drawback can be dealt with using a non-stationary stochastic process, and the second drawback can be handled by changing the time from discrete to continuous [16,18].

For the first drawback, Chung et al. [16] proposed an approach using a non-stationary stochastic process to

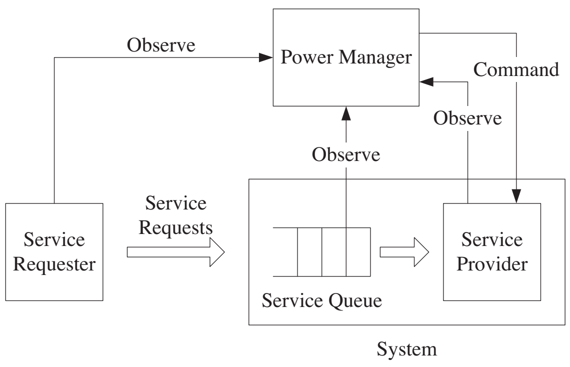


Fig. 3. Overall system architecture for the DPM [14].

Table 1
A qualitative comparison of representative DPM algorithms.

	Static timeout	Adaptive timeout [5]	Predictive [11]	Stochastic [14]	Machine learning [20]	AH-DPM (proposed)
Timeout value adjustment	Static	Adaptive	Adaptive	None	Adaptive	Adaptive
Time complexity	$O(1)$	$O(1)$	$O(1)$	$O((xa)^3L)$	Depend on experts	$O(1)$
Space complexity	$O(1)$	$O(1)$	$O(1)$	$O(xa)$	Depend on experts	$O(1)$
Offline calculation	No	No	No	Yes	Depend on experts	No
Manual configuration	Yes	Yes	Yes	No	Depend on experts	No
Suitable for bursty arrival patterns with self-similarity	No	No	No	No	Depend on experts	Yes

model the arrival distribution of user requests. They proposed a mechanism called sliding window to keep historical data. The sliding window is limited in length and hence recent historical data are kept in order to reflect recent user request behavior. Because the distribution of user requests is non-stationary, the decision table of the SP must be recalculated in every period. To overcome this drawback, the authors used table lookup and interpolation to calculate the decision table to avoid the recalculation. Ren et al. [17] modified the approach in [16] and introduced a multi-mode model using a Markov-modulated stochastic process to model the non-stationary arrival process of service requests [17]. The advantage of these two approaches is that the request arrival distribution of the SR can be adapted to any distribution. But the disadvantages are an enormous amount of memory usage and computation power required. If we apply these two approaches to several components in a mobile device, we have to derive a request arrival distribution for each component in advance and it will be time consuming and inconvenient.

As to the second drawback, Qiu et al. [18] proposed a continuous-time Markov decision process to decrease the computation of the PM. In this approach, the decision is made on an event arrival, such as a user request arrival, the SP starting to serve a user request, and the SP finishing a user request. Rong et al. [19] extended the work in [18] to model a battery-powered portable system by introducing and incorporating a new continuous-time Markovian decision process model of the battery source [19]. There are some disadvantages in this approach. First, the computation complexity both in time and space are high because of the characteristic of continuous-time based policy optimization. Second, the experiment was based on a continuous-time Markov process, that means that the inter-arrival time of user requests, the switching time of the SP, and the state transition of the SQ are exponential distributed, which are not quite realistic in the real world.

2.4. Machine learning algorithms

Several researchers applied machine learning to learn the request arrival patterns of the SR. Dhiman et al. [20] and Prabha et al. [21] proposed expert based machine learning algorithms. An expert based machine learning algorithm selects the best DPM policy from a set of DPM policies. These policies are called experts. Each expert has a weight value which indicates the expert's priority and is adjustable by the machine learning algorithm. The weight value will be adjusted in every idle period and the expert with the highest weight value in the current idle period will be used to control an embedded system during the next idle period. However, the performance of an expert based machine learning algorithm is highly dependent on chosen experts. In [22], the authors proposed a reinforcement learning based algorithm. The algorithm is based on the Q-learning algorithm, which was originally designed to find a policy for a Markov Decision Process, to learn the arrival request patterns of the SP [22]. The authors modified the Q-learning algorithm to solve the DPM problem and speed up the algorithm's convergence time by updating more than one Q values simultaneously. The time complexity of the modified Q-learning algorithm is $O(|SP| \times |A|)$, and the space complexity is $O(|SP| \times |SQ| \times |SR| \times |A|)$, where $|SP|$, $|SQ|$, and $|SR|$ are the number of states of the SP, SQ, and SR, respectively, and $|A|$ is the number of commands. Note that the time and space complexities of the modified Q-learning algorithm are higher than those of the proposed AH-DPM algorithm, which are constant in both time and space complexity, which will be explained later.

In summary, a qualitative comparison of major DPM algorithms, including the proposed AH-DPM, is shown in Table 1. The time complexity and space complexity of the proposed AH-DPM algorithm are both $O(1)$ since our algorithm is a hybrid of adaptive timeout and predictive algorithms. Let p be the number of states in the SP, q be the queue length in the SQ, and r be the number of states in the SR, and let $x = p \times q \times r$ [14]. The time complexity of the stochastic algorithm proposed by Benini et al. [14] is $O((xa)^3L)$, where L is the bit length of input data [23] (usually 32 bits in modern computer systems) and a is the number of commands that the PM can issue to the SP [14]. The space complexity of the stochastic algorithm is $O(xa)$ [23]. The stochastic algorithm needs offline calculation because the system state transition matrix must be calculated in advance. Because the machine learning algorithm uses experts, which are a set of DPM algorithms, the time and space complexities, offline calculation, and manual configuration are dependent on the selected DPM algorithms. In addition, STO, ATO, and predictive algorithms need to configure some initial values manually. Furthermore, to the best of our knowledge, none of previous works is suitable for bursty arrival patterns with self-similarity. They are more suitable for stationary request arrival patterns.

3. Proposed AH-DPM algorithm

3.1. The design of the proposed algorithm

To obtain better power saving of the SP, the proposed AH-DPM algorithm handles the average idle time in the bursty (ON) and non-bursty (OFF) periods in the request arrival pattern separately. We derive the average idle time of the SP in the ON and OFF periods separately using exponential average. All parameters used in the proposed algorithm are defined in Table 2.

The proposed AH-DPM algorithm returns two values: the next state, S_{SP_next}, that the SP should be switched to and the timeout value, $T_{timeout}$, that the SP should wait before being switched to the next state. There are three main ideas in the proposed AH-DPM algorithm: (1) keeping track of the average idle time in the ON period, T_{on_avg}, and in the OFF period, T_{off_avg}, separately, (2) using the average idle time in the ON period to adjust the timeout value more precisely and using the average idle time in the OFF period to decide which inactive state the SP should be switched to, and (3) comparing the most recent idle time, T_{idle}, with the break-even time, T_{BE}, to determine whether the expected period of the request arrival pattern, P_{expect}, is *ON_period* or *OFF_period*. Three cases will be monitored by the proposed AH-DPM algorithm:

1. When a request arrives, based on P_{expect} and the comparison between T_{idle} and T_{BE}, there are three situations required to be taken care of:
   (a) If P_{expect} equals to *ON_period*, calculate new T_{on_avg} using new T_{idle}. Then, new $T_{timeout}$ is obtained from the maximum value among old $T_{timeout}$, T_{idle}, and T_{on_avg}. Note that if $T_{idle} > T_{BE}$, our algorithm will not choose T_{idle} as the new timeout value to avoid long busy wait.
   (b) If P_{expect} equals to *OFF_period* and T_{idle} is smaller than T_{BE}, this situation is called *prediction miss* because in the OFF period, T_{idle} should be larger than T_{BE}. In this case, the calculations described in *situation a* will be performed since P_{expect} should be in the ON period.
   (c) If P_{expect} equals to *OFF_period* and T_{idle} is larger than or equal to T_{BE}, this situation is called *prediction hit*. T_{off_avg} will be calculated using new T_{idle}.

   After handling either of the above three situations, P_{expect} will be set to *ON_period*, S_{SP_next} will be set to idle state, and S_{SP_next} along with timeout value of *zero* will be returned. The reason for returning zero as the timeout value is to wake up the SP immediately whenever there is a request arrived and the SP is in an inactive state.
2. When the SP has finished serving a request and the SQ is empty, the SP may be inactivated since there is no request to be served. The next state of the SP, S_{SP_next}, will be chosen by the algorithm *InactiveState*. Then, S_{SP_next} and $T_{timeout}$ will be returned.
3. When $T_{timeout}$ expires, the SP will be switched to the state indicated by S_{SP_next} and P_{expect} will be set to *OFF_period* to indicate that the request arrival pattern is now in the OFF period.

Table 2
The parameters used in the proposed AH-DPM algorithm.

Parameters	Description
T_{on_avg}	Average idle time in the ON period
T_{off_avg}	Average idle time in the OFF period
T_{idle}	Most recent idle time
$T_{timeout}$	Timeout value
T_{BE}	Break-even time
$S_{SP\_current}$	Current state of the SP
S_{SP_next}	Next state of the SP
P_{expect}	Expected period of the request arrival pattern, either *ON_period* or *OFF_period*

In summary, the benefits of the proposed AH-DPM algorithm are that our algorithm can adapt to bursty request arrival patterns with self-similarity and a service provider (SP, i.e., hard disk or WLAN NIC in this paper) with multiple inactive states using the following two steps to achieve better power saving. First, it derives the average idle time of the SP in the bursty (ON) period and non-bursty (OFF) period separately. Second, it uses the average idle time in the ON period to adjust the timeout value more precisely and uses the average idle time in the OFF period to decide which inactive state the SP should be switched to. In other words, the derivation of the new timeout value described in *situation a* of case 1 above is to set the timeout value long enough to prevent the SP from an unexpected state transition to an inactive state and to keep the timeout value short enough to decrease the length of the busy wait period during the ON period.

3.2. The flowchart and pseudo codes of the proposed algorithm

Figs. 4 and 5 show the flowchart and the pseudo codes of the proposed AH-DPM algorithm, respectively. If a request arrives and the SP is in the ON period, T_{on_avg} and $T_{timeout}$ are updated accordingly, as shown from line 7 to line 11 in Fig. 5. On the other hand, if a request arrives and the SP is in the OFF period, two circumstances must be considered. First, if T_{idle} is larger than or equal to T_{BE}, the algorithm has made a correct prediction that the SP is actually in the OFF period. This circumstance is called a *prediction hit* and T_{off_avg} will be updated accordingly, as shown at line 17 in Fig. 5. Second, if T_{idle} is smaller than T_{BE}, this is called a *prediction miss* since in the OFF period T_{idle} must be larger than T_{BE}. In this case, T_{on_avg} and $T_{timeout}$ are updated instead of T_{off_avg}, and P_{expect} is set to *ON_period*, as shown from line 21 to line 25 in Fig. 5. Once a request is served and the SQ is empty, the SP becomes idle. Then, the SP will be switched to an inactive state S_{SP_next} with timeout value $T_{timeout}$. S_{SP_next} will be assigned using algorithm *InactiveState*, as shown in Fig. 6. Algorithm *InactiveState* is used to determine which inactive state the SP will enter. It calculates T_{BE} of each inactive state, and save it into *BE_List*. *BE_List* is an array that stores T_{BE} of each inactive state. *BE_List* is then sorted in the descending order. T_{off_avg} will be compared with each T_{BE}. If T_{off_avg} is greater than or equal to a certain T_{BE}, the inactive state corresponding to the previous T_{BE} in *BE_List* will be returned to algorithm *AH-DPM*, as shown from line 6 to line 14 in

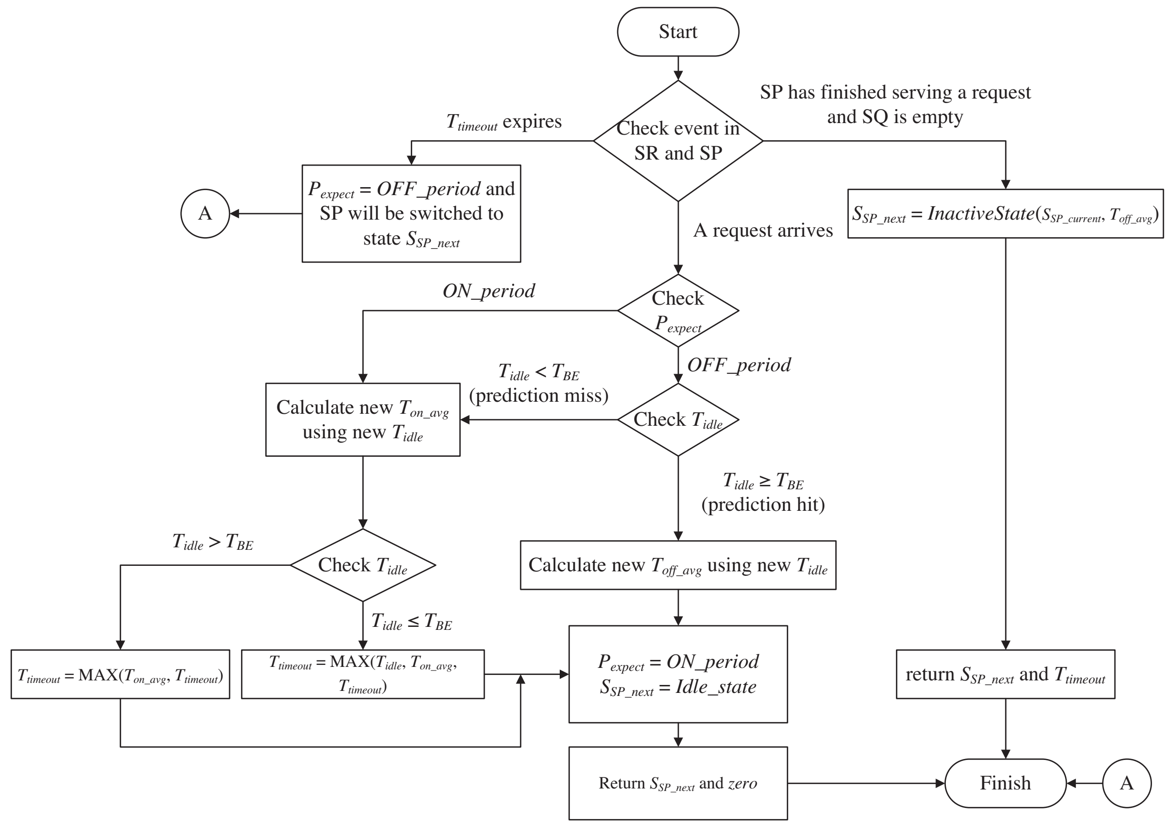

Start
Check event in SR and SP
$T_{timeout}$ expires
SP has finished serving a request and SQ is empty
P_{expect} = OFF_period and SP will be switched to state S_{SP_next}
A
S_{SP_next} = InactiveState($S_{SP\_current}$, T_{off_avg})
A request arrives
Check P_{expect}
ON_period
OFF_period
Calculate new T_{on_avg} using new T_{idle}
$T_{idle} < T_{BE}$ (prediction miss)
Check T_{idle}
$T_{idle} \geq T_{BE}$ (prediction hit)
Calculate new T_{off_avg} using new T_{idle}
$T_{idle} > T_{BE}$
Check T_{idle}
$T_{idle} \leq T_{BE}$
$T_{timeout}$ = MAX(T_{on_avg}, $T_{timeout}$)
$T_{timeout}$ = MAX(T_{idle}, T_{on_avg}, $T_{timeout}$)
P_{expect} = ON_period S_{SP_next} = Idle_state
return S_{SP_next} and $T_{timeout}$
Return S_{SP_next} and zero
Finish
A

Fig. 4. The flowchart of the proposed AH-DPM algorithm.

Fig. 6. If $T_{timeout}$ expires, the SP will be switched to state S_{SP_next} and P_{expect} will be set to *OFF_period*.

4. Experimental results and discussion

4.1. Experimental setup

We compare our algorithm (AH-DPM) with the Oracle algorithm (Oracle), the static timeout algorithm (STO) with timeout value of 30 s [2], the adaptive timeout algorithm (ATO) of Douglis et al. [5] with parameters $(\alpha_m, \beta_m, \rho) = (0.5, 1.5, 0.1)$ [2] and initial timeout value of 30 s [2], the predictive algorithm (Predictive) of Huang et al. [11] with parameters $\alpha = 0.3$ and $c = 2$ [2], the stochastic (Stochastic) algorithm of Benini et al. [14], and the machine learning (ML) algorithm of Dhiman et al. [20]. Remind that the Oracle algorithm is a theoretically optimal algorithm because it knows the arrival time of all requests; therefore, the algorithm can determine exactly when and to which state the SP should be switched for power saving. The Stochastic algorithm does not need to set any initial value, but the optimal decision policies must be calculated in advance. The ML algorithm uses STO, ATO, Predictive, and Stochastic algorithms as its experts. The parameters of each expert used in the ML algorithm are same as those listed above. Since the Oracle algorithm is aware of all the requests issued from the beginning to the end, it can come out with the most power saving results. The *AlwaysOn* algorithm is defined as “always keeping the SP in the active state,” and represents the worst case of average power consumption and the best case of average response time (average packet transmission delay) in hard disk (WLAN NIC) experiments.

In the hard disk experiments, we used a typical hard disk specification of Hitachi Travelstar 5K100 as an example SP specification, as shown in Tables 3 and 4 [24]. The hard disk request traces were collected for a week by monitoring the ATA commands sent and received by libATA drivers under Fedora 12 [25], kernel version 2.6.32.11 [26]. The trace characteristics of hard disk requests for each day of the week are listed in Table 5. We compare the *average power consumption* and *average response time* for the hard disk among these DPM algorithms. The average power consumption is measured in *Watt*, and it is defined as total energy consumed divided by total elapsed time. The average response time is measured in *millisecond* and it is defined as the average elapsed time between a request arrived and the request having been served, which includes the queuing delay and service time of a request.

As to the WLAN NIC experiments, we used the specification of the Intel PRO/Wireless 3945ABG (802.11g) card, which is shown in Table 6 [27]. However, the state transition time and state transition power of the Intel PRO/Wireless 3945ABG card are not available. We used the state transition time listed in [28]. The state transition power

```
algorithm $AH\text{-}DPM(S_{SP\_current}, T_{idle})$
{
  if(a request arrives)
  {
    if($P_{expect}$ == ON_period)
    {
      $T_{on\_avg} = \alpha \times T_{idle} + (1 - \alpha) \times T_{on\_avg}$
      if($T_{idle} > T_{BE}$)
        $T_{timeout}$ = MAX($T_{on\_avg}, T_{timeout}$)
      else
        $T_{timeout}$ = MAX($T_{idle}, T_{on\_avg}, T_{timeout}$)
    }
    else if($P_{expect}$ == OFF_period)
    {
      if($T_{idle} \geq T_{BE}$)  /* prediction hit */
      {
        $T_{off\_avg} = \alpha \times T_{idle} + (1 - \alpha) \times T_{off\_avg}$
      }
      else  /* prediction miss */
      {
        $T_{on\_avg} = \alpha \times T_{idle} + (1 - \alpha) \times T_{on\_avg}$
        if($T_{idle} > T_{BE}$)
          $T_{timeout}$ = MAX($T_{on\_avg}, T_{timeout}$)
        else
          $T_{timeout}$ = MAX($T_{idle}, T_{on\_avg}, T_{timeout}$)
      }
    }
    $P_{expect}$ = ON_period
    $S_{SP\_next}$ = Idle_state
    return $S_{SP\_next}$ and zero
  }
  else if(SP has finished serving a request and SQ is empty)
  {
    $S_{SP\_next}$ = InactiveState($S_{SP\_current}, T_{off\_avg}$)
    return $S_{SP\_next}$ and $T_{timeout}$
  }
  else if($T_{timeout}$ expires)
  {
    $P_{expect}$ = OFF_period
  }
}
```

Fig. 5. The proposed AH-DPM algorithm.

from sleep to idle is set to twice of the power consumption in idle state, and the state transition power from idle to sleep is set to the power consumption in sleep state [29]. The state transition characteristics of the WLAN NIC are listed in Table 7. In addition, the real traces of the WLAN NIC were captured using Wireshark [30], version 1.2.6, under Fedora 12 [25]. The trace characteristics of WLAN NIC packets for each day of the week are listed in Table 8. The *average power consumption* and *average packet transmission delay* are metrics of the WLAN NIC for performance evaluation among these DPM algorithms. The definition of the average power consumption is identical to that for the hard disk and the average packet transmission delay is measured in *millisecond*, and it is defined as the queuing delay plus the packet transmission time.

Finally, we also evaluate the *prediction miss rate* and the *inactivation ratio*, which will be defined later. The prediction miss rate can reflect the average power consumption and the inactivation ratio can reflect the average response time in the hard disk and the average transmission delay in the WLAN NIC. There are two cases of prediction miss: *false positive prediction miss* and *false negative prediction miss*.

```
algorithm $InactiveState(S_{SP\_current}, T_{off\_avg})$
{
    calculate each $T_{BE}$ which relates to each inactive state and $S_{SP\_current}$, and save them into $BE\_List$
    sort $BE\_List$ in descending order
    $S_{ret} = S_{SP\_current}$
    for(index = 0; index < the size of $BE\_List$; index++)
    {
        if($T_{off\_avg}$ >= $BE\_List$[index])
        {
            $S_{ret}$ = the inactive state corresponding to $BE\_LIST$[index]
            break;
        }
    }
    return $S_{ret}$
}
```

Fig. 6. Inactive state decision algorithm.

Table 3
State transition time and power consumption specifications of Hitachi Travelstar 5K100 hard disk [24].

From	To	Time (s)	Power consumption (Watt)
Sleep	Performance Idle	3.5	3.8
Standby	Performance Idle	2.5	3.8
Low Power Idle	Performance Idle	0.3	2.0
Active Idle	Performance Idle	0.02	2.0
Performance Idle	Standby	0.35	2.0
Performance Idle	Sleep	0.35	2.0

Table 4
Power consumption specification of Hitachi Travelstar 5K100 hard disk [24].

State	Power consumption (Watt)
Performance Idle	2.0
Active Idle	1.1
Low Power Idle	0.65
Read	2.0
Write	2.0
Seek	2.5
Standby	0.2
Sleep	0.1

The false positive prediction miss occurs in a situation that the PM inactivates the SP (i.e., switching the SP to a lower power consumption state), but the length of the inactive period is shorter than the break-even time. That is, the SP should not be inactivated, but it is inactivated. The false negative prediction miss occurs in a situation that the PM keeps the SP in active state, but the length of the idle period is longer than the break-even time. That is, the SP

Table 6
Power consumption specification of Intel PRO/wireless 3945ABG WLAN NIC [27].

State	Power consumption (Watt)
Transmit	1.8
Receive	1.4
Idle	0.15
Sleep	0.03

Table 7
State transition time and power consumption of the WLAN NIC [27,28].

From	To	Time (μs)	Power consumption (Watt)
Sleep	Idle	250	0.3
Idle	Sleep	80	0.03

Table 5
Trace characteristics of hard disk requests.

Day of week	Number of requests	$\overline{T_{RI}}$	$\sigma_{T_{RI}}$
Sunday	148175	0.5826582132	28.066745610
Monday	273503	0.3158121481	7.576979263
Tuesday	90081	0.9588597164	13.238787090
Wednesday	190393	0.4537697663	7.577420349
Thursday	532259	0.1623176762	3.813604001
Friday	128420	0.6725806881	10.876428390
Saturday	46820	1.8449750100	18.263384880
A week	1409651	0.4289868597	11.847667333

$\overline{T_{RI}}$: Average request inter-arrival time (s).
$\sigma_{T_{RI}}$: Standard deviation of the request inter-arrival time.

Table 8
Trace characteristics of WLAN NIC traffic.

Day of week	Number of requests	$\overline{T_{PI}}$	$\sigma_{T_{PI}}$
Sunday	68812	1.2524797900	13.271983260
Monday	24697	3.0321168550	43.423190660
Tuesday	48308	1.6348761390	218.246429500
Wednesday	115253	0.7495722815	10.870117770
Thursday	183500	0.4701362994	8.127996827
Friday	13013	6.5880108190	73.593723700
Saturday	8164	10.4842361800	93.448664130
A week	461747	1.2648289645	73.980365017

$\overline{T_{PI}}$: Average packet inter-arrival time (s).
$\sigma_{T_{PI}}$: Standard deviation of the packet inter-arrival time.

should be inactivated, but it is not inactivated. Both cases will result in extra power consumption penalty.

The definition of the prediction miss rate $R_{prediction\_miss}$ is as follows:

$$R_{prediction\_miss} = \frac{N_{fp\_miss} + N_{fn\_miss}}{N_{prediction}} \quad (2)$$

where N_{fp_miss} is the number of false positive prediction miss, N_{fn_miss} is the number of false negative prediction miss, and $N_{prediction}$ is the total number of predictions made by the PM. The definition of the inactivation ratio $R_{inactivate}$ is defined as follows:

$$R_{inactivate} = \frac{N_{inactivate}}{N_{prediction}} \quad (3)$$

where $N_{inactivate}$ is the number of switching the SP from the active state to an inactive state (a lower power consumption state). When the SP is in an inactive state and a request arrives, the SP must switch to the active state in order to serve the request. Since existing DPM algorithms, except the Oracle algorithm, did not actually implement the pre-wakeup mechanism, an incoming request will be queued in the SQ waiting for the SP to be switched from an inactive state to the active state. Therefore, it results in queuing delay. Since each SP inactivation will lengthen the queuing delay of the next incoming request, the higher the inactivation ratio is, the longer the average response time of the hard disk and the average packet transmission delay of the WLAN NIC are.

4.2. Experimental results

Fig. 7 shows the experimental results of the average power consumption of the hard disk for each DPM algorithm on each day of a week. In Fig. 7, we found that the proposed AH-DPM algorithm is always better than the other algorithms except the Oracle algorithm under the request trace of each day in a week. From Table 5 and Fig. 7, we found that average power consumption is proportional to number of requests, except the case on Tuesday. The average power consumption on Tuesday is still higher in spite of a lower number of requests. This is because of a higher inactivation ratio on Tuesday, as shown in Fig. 9. Since a higher inactivation ratio implies a higher

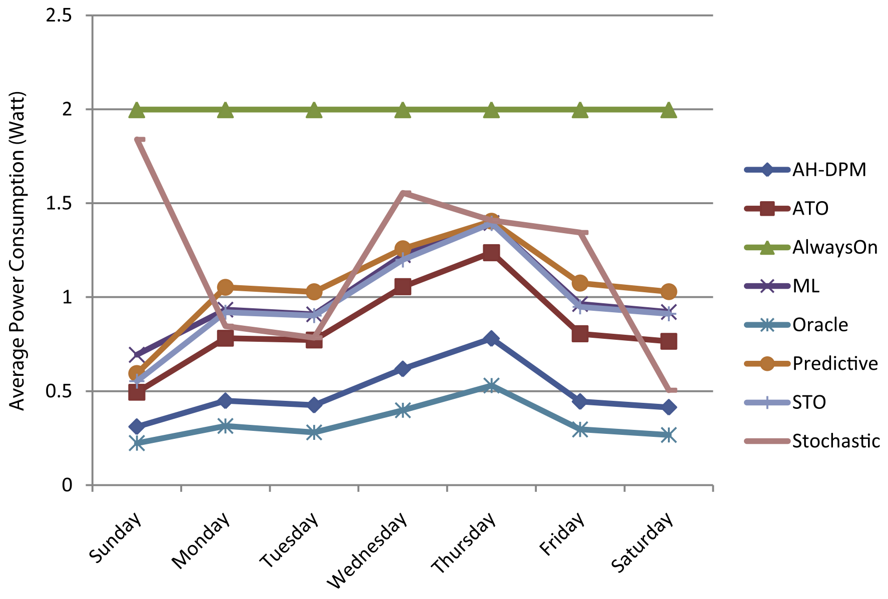


Fig. 7. Comparison of the average power consumption of the hard disk.

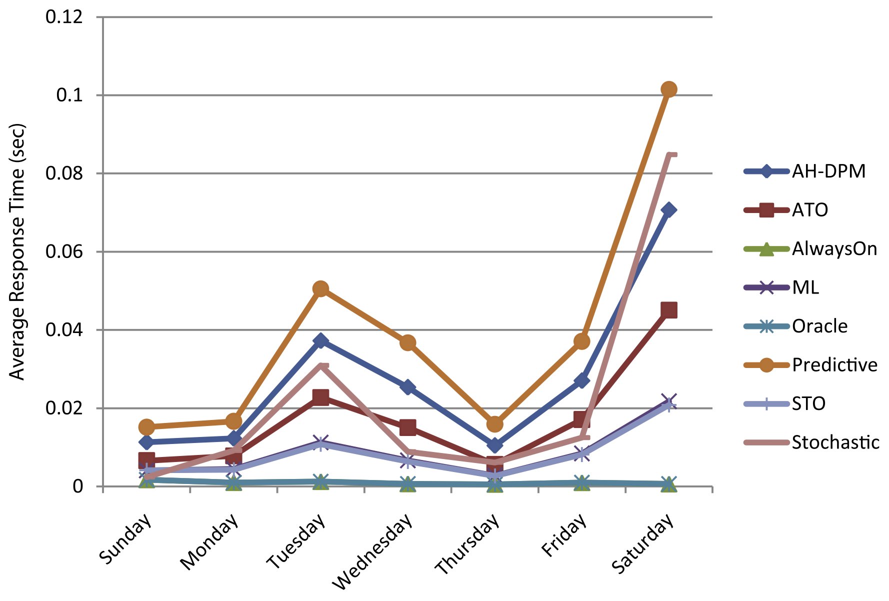


Fig. 8. Comparison of the average response time of the hard disk.

number of state transitions, the power consumption increases due to frequent state transitions. As to the average response time, Fig. 8 shows that the proposed AH-DPM algorithm is larger than the ATO, AlwaysOn, ML, Oracle, and STO algorithms, and is shorter than the Predictive algorithm. From Figs. 8 and 9, we found that average response time is proportional to inactivation ratio. This is because that if the SP is in an inactivated state and there is an incoming request, the SP must be switched to the active state to serve the request. Since no existing DPM algorithms has the ability to switch the SP to the active state in advance, the request must be queued in the SQ to wait for the SP to be switched from an inactive state to the active state. Therefore, a larger inactivity ratio will result in longer response time.

Figs. 10 and 11 are the comparison of average power consumption and average response time in a week for the hard disk, respectively. In Fig. 10, we observed that the average power consumption of the proposed AH-DPM is 33.90% worse than that of the Oracle algorithm, and is better than that of the ATO, ML, Predictive, STO, and Stochastic algorithms by 69.40%, 101.68%, 113.69%, 95.33%, and 278.37%, respectively. That is, the proposed AH-DPM algorithm performed the best except the Oracle algorithm

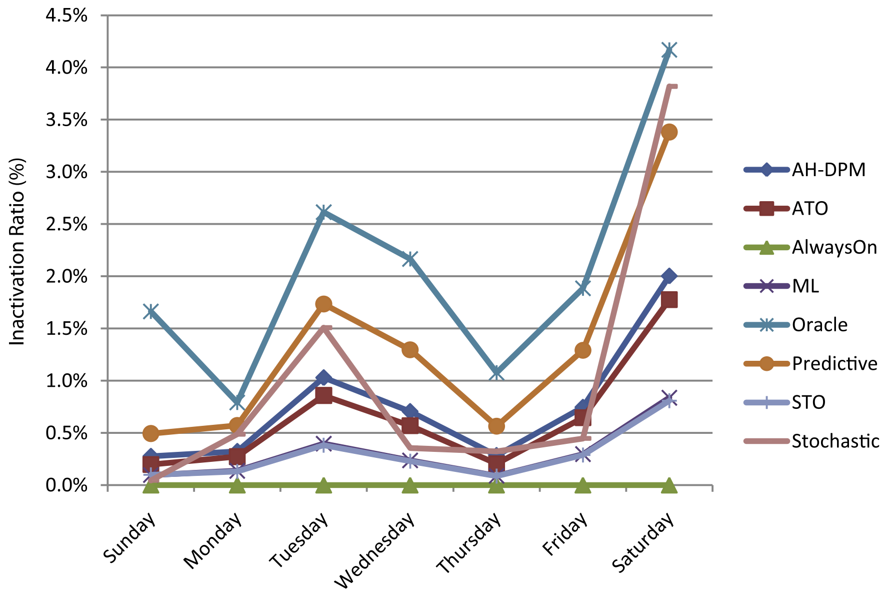


Fig. 9. Comparison of the inactivation ratio of the hard disk.

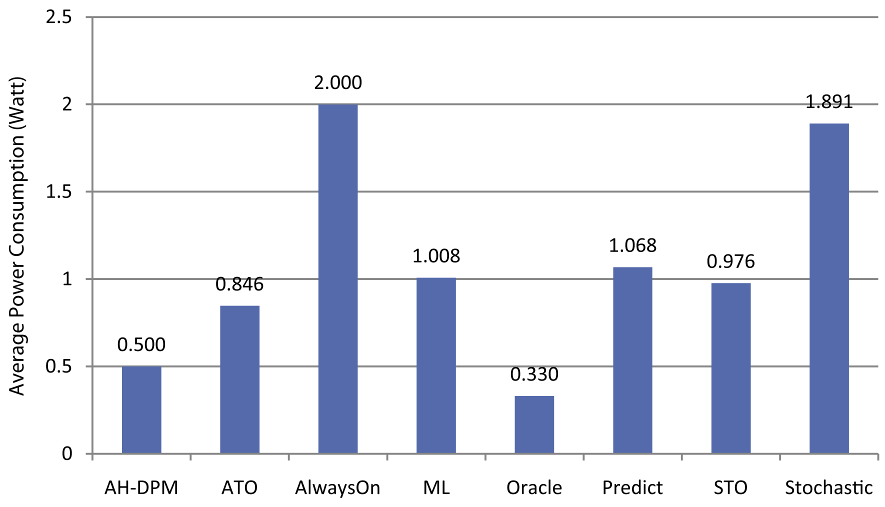


Fig. 10. Comparison of the average power consumption of the hard disk in a week.

in terms of average power consumption. Remind that the Oracle algorithm is theoretically optimal. Since the *prediction miss* will cause extra power consumption, we derive the prediction miss rate of each algorithm. In Fig. 12, we found that the prediction miss rate of the proposed AH-DPM algorithm is the lowest compared to that of the other DPM algorithms except the Oracle algorithm. The results are in accordance with those illustrated in Fig. 10. Although the average response time of the proposed AH-DPM algorithm in Fig. 11 is not the best, it is still lower than that of the Predictive algorithm by 43.03% and it is also lower than the average disk access time specified in a hard disk specification [24]. According to this specification [24], the average response time for read/write one byte from/to the hard disk is about 20.5 ms (command overhead + average seek time + average latency + average disk-buffer data transfer rate). Therefore, the average response time of the proposed AH-DPM algorithm, which is 18.022 ms as shown in Fig. 11, is lower than 20.5 ms.

In the following, we illustrate that the *inactivation ratio* of the hard disk is also in accordance with its average response time. In Fig. 13, in term of the inactivation ratio, we observed that the proposed AH-DPM algorithm is larger than the ATO, ML, and STO algorithms, and is smaller than the Predict and Stochastic algorithms. A larger inactivation ratio means that the PM inactivates the SP more aggressive; however, the penalty is longer average response time, as shown in Fig. 11.

We also evaluate the average power consumption and average packet transmission delay on each day of a week for each DPM algorithm. In Fig. 14, we found that the average power consumption of the proposed AH-DPM is comparable to that of the ATO, Oracle, and Predictive algorithms, and is better than that of the ML, STO, and Stochastic algorithms. However, in terms of average packet transmission delay, the proposed AH-DPM algorithm is better than the ATO and Predictive algorithms, as shown in Fig. 15.

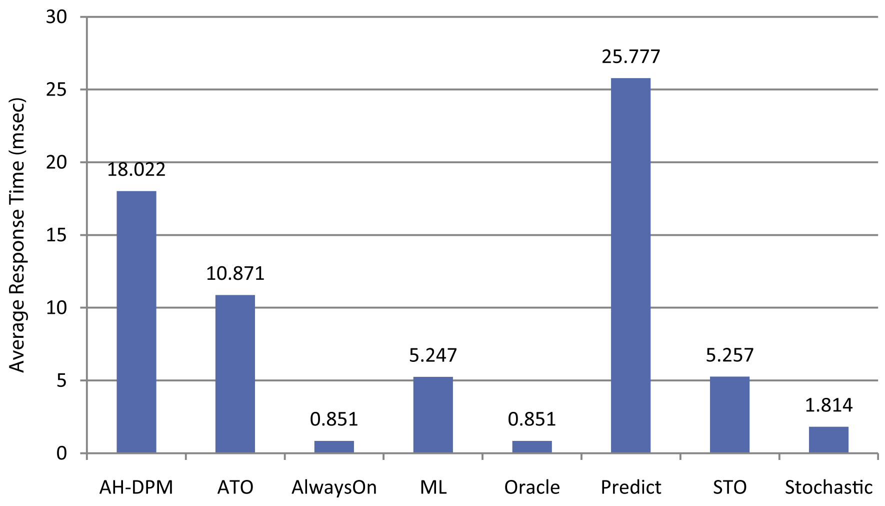


Fig. 11. Comparison of the average response time of the hard disk in a week.

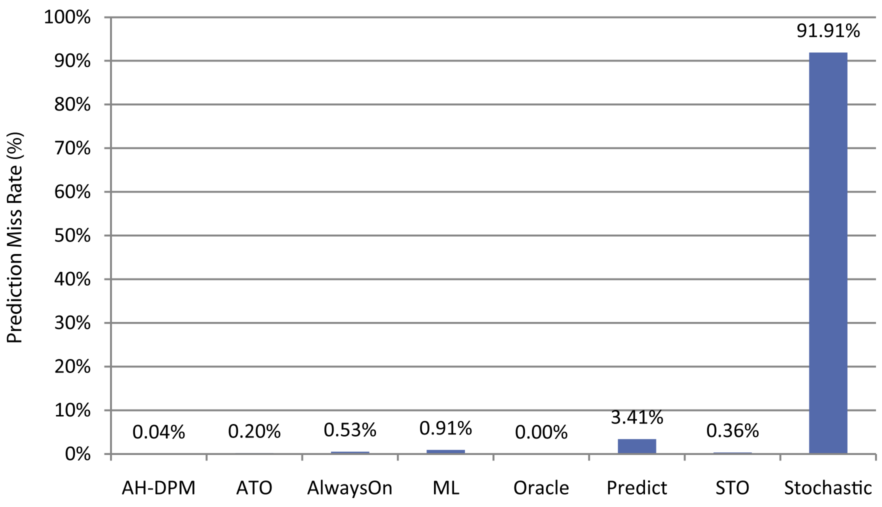


Fig. 12. Comparison of the prediction miss rate of the hard disk in a week.

Figs. 16 and 17 illustrate the average power consumption and average packet transmission delay of the WLAN NIC in a week for different DPM algorithms, respectively. In Fig. 16, we observed that the average power consumption of the AH-DPM algorithm is comparable to (0.7% more than) that of the ATO, Oracle, and Predictive algorithms, and is better than that of the ML, STO, and Stochastic algorithms by 172.34%, 89.14%, and 372.68%, respectively. Note that although the prediction miss rate of the proposed AH-DPM algorithm is the best, as shown in Fig. 18, compared with the other algorithms except the Oracle algorithm, the average power consumption of the AH-DPM algorithm is still slightly larger than that of the ATO and Predictive algorithm. This is because that the average timeout value of the AH-DPM algorithm is larger than that of the ATO and Predictive algorithms, as illustrated in Fig. 19. That is, the proposed AH-DPM has a higher prediction hit rate that can reduce more power consumption (see Fig. 16) and decrease more packet transmission delay (see Fig. 17); however, the minor penalty is a longer timeout value that causes slightly larger power consumption.

As to the average packet transmission delay, Fig. 17 shows that the proposed AH-DPM algorithm is better than the ATO and Predictive algorithms by 23.22% and 25.18%, and is worse than the ML, STO, and Stochastic algorithms by 41.37%, 41.05%, and 38.20%, respectively. Moreover, Fig. 20 illustrates the inactivation ratio of the WLAN NIC in a week for each DPM algorithm. We found that the inactivation ratio of the proposed AH-DPM is larger than that of the ATO and Predict algorithms and is smaller than that of the ML, STO, and Stochastic algorithms. The results of the inactivation ratio are again in accordance with those of the average packet transmission delay (see Fig. 17).

In summary, considering the tradeoff between average power consumption and average response time (or average packet transmission delay), the proposed AH-DPM

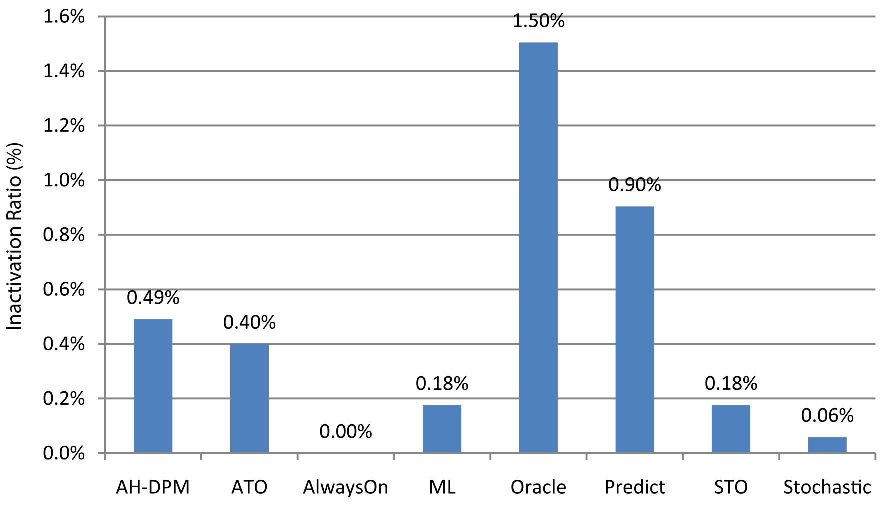


Fig. 13. Comparison of the inactivation ratio of the hard disk in a week.

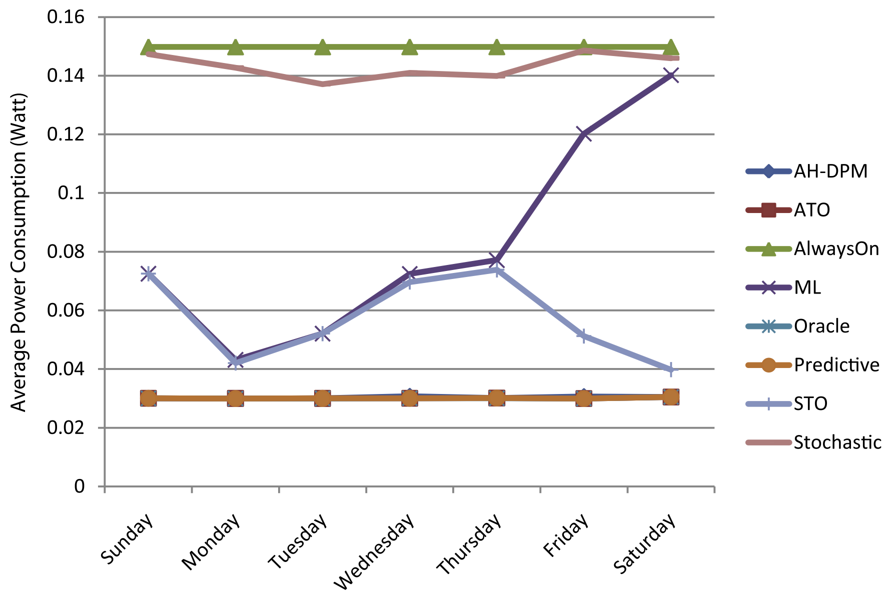


Fig. 14. Comparison of the average power consumption of the WLAN NIC.

algorithm performs the best among the DPM algorithms, except the Oracle algorithm, for hard disks and WLAN NICs. Note that the proposed algorithm derives the average idle time in the ON period and OFF period separately. It uses the average idle time in the ON period to determine an appropriate timeout value, which means to set the timeout value long enough to decrease the false positive prediction miss rate and to set the timeout value short enough to decrease the false negative prediction miss rate. By decreasing these two types of prediction miss rates, the proposed algorithm can reduce unnecessary power consumption of the SP. In addition, the proposed algorithm can also adapt to the SP that has multiple inactive states by using the average idle time in the OFF period to decide which inactive state the SP should be switched to. Such adaptation can further decrease the power consumption of the SP because we can choose a better inactive state according to the average idle time in the OFF period. The longer idle time of the SP in the OFF period, the deeper inactive state the SP can be switched to.

4.3. Discussion

In Fig. 8, we observed that the average response time on Tuesday and Saturday are higher than that in the other days. This is because that the average request inter-arrival time on Tuesday and Saturday are longer than those on the other days, as shown in Table 5. A longer average request

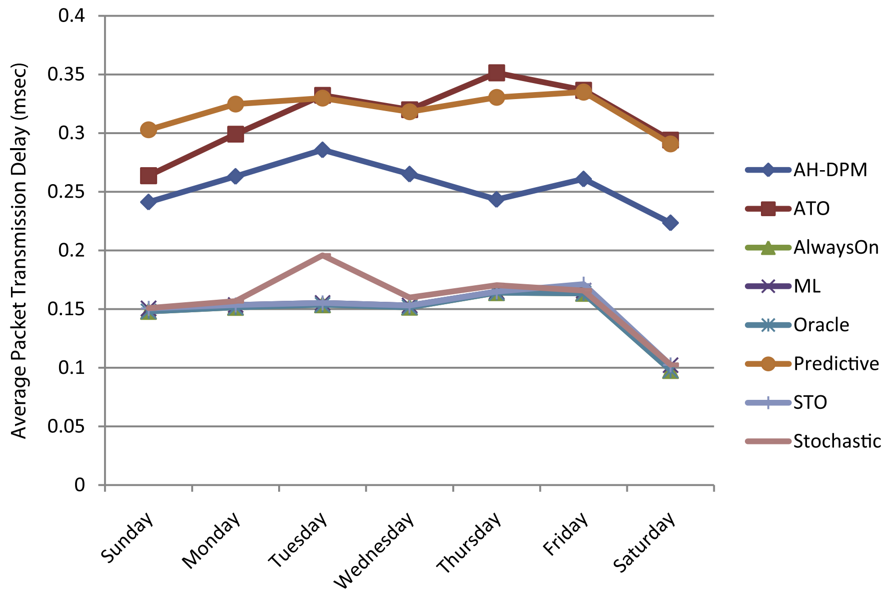


Fig. 15. Comparison of the average packet transmission delay of the WLAN NIC.

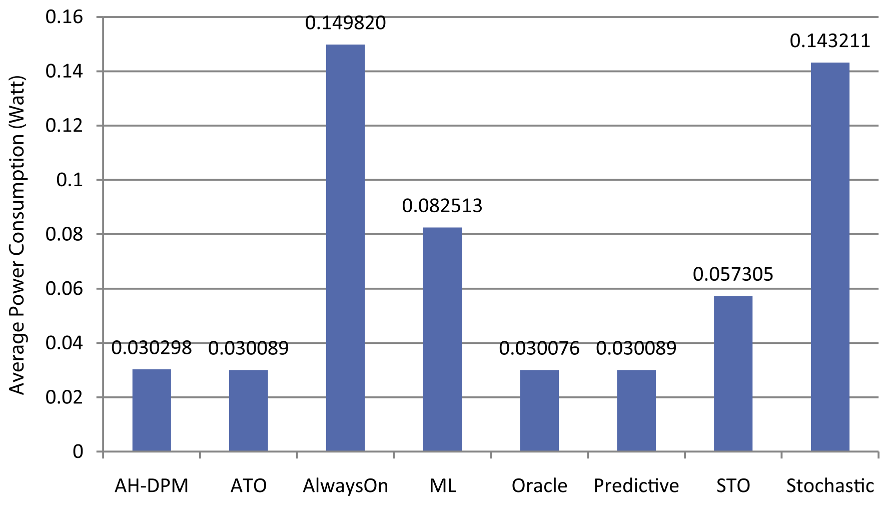


Fig. 16. Comparison of the average power consumption of the WLAN NIC in a week.

inter-arrival time implies a higher probability to inactivate a component. Because of lacking the pre-wakeup mechanism for each DPM algorithm, except the Oracle algorithm, the SP will only be waked up when a request arrived. Therefore, the average response time will increase due to the state transition time penalty while waking up the SP.

The performance of the ATO algorithm [7] is highly correlated with the SP's state transition time from an inactive state to the active state, called *wakeup state transition time*, and the request arrival pattern. The ATO algorithm will increase the timeout value if the wakeup state transition time of the SP is larger than the latest idle time of the SP, and will decrease the timeout value otherwise. If the wakeup state transition time is smaller (larger) than the idle time in a period with bursty request arrivals, the ATO algorithm will decrease (increase) the timeout value rapidly. For the hard disk, since the wakeup state transition time, which are 3.5 s from *Sleep* to *Performance Idle* and 2.5 s from *Standby* to *Performance Idle* (see Table 3), is larger than the idle time of the SP during a bursty request arrivals period, it will cause the timeout value to be increased rapidly. A larger timeout value means a lower probability to inactivate the SP, which results in higher power consumption of the ATO algorithm. As to the WLAN NIC, the wakeup state transition time of the ATO algorithm, which is 80 μs (see Table 7), is smaller than the idle time of the SP during a bursty request arrivals period, and it causes the timeout value to be decreased rapidly. A smaller timeout value causes a larger inactivation ratio as well as a larger false positive prediction miss rate, which result in longer packet transmission delay.

The Predictive algorithm [11] also suffers from the bursty request arrivals pattern. This is because that the Predictive algorithm uses Eq. (1) (refer to Section 2) to predict the idle time. If the predicted idle time is longer than the break-even time, the SP will be inactivated. Otherwise, the SP will remain in the active state. When a request arrives, the PM will recalculate and update the predicted idle time. If the request arrival pattern is bursty and the idle time of the SP in the bursty period is shorter than

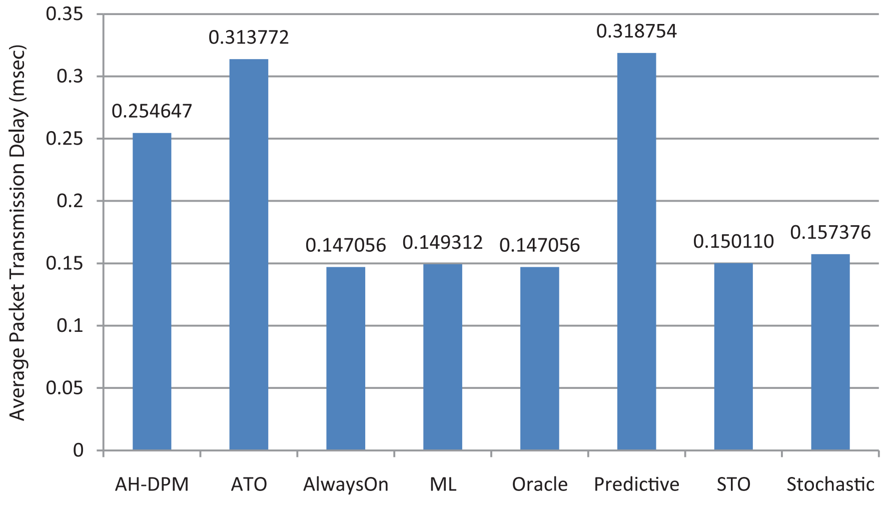


Fig. 17. Comparison of the average packet transmission delay of the WLAN NIC in a week.

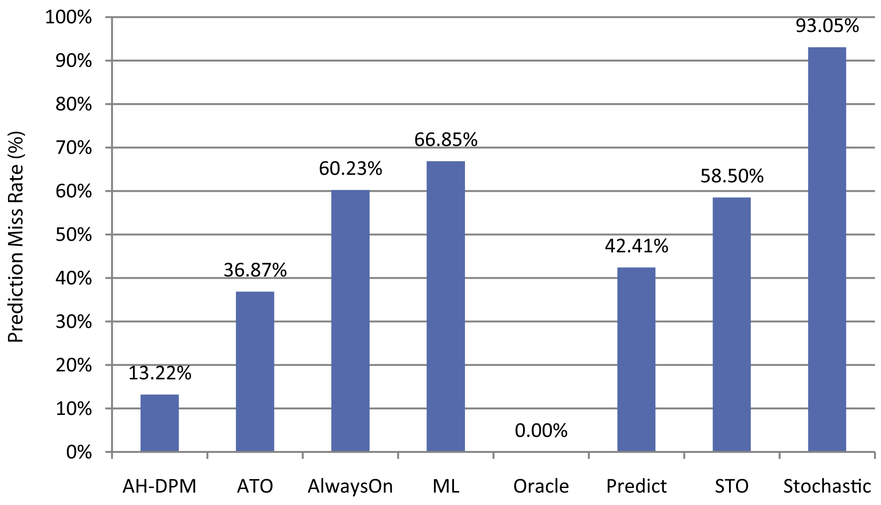


Fig. 18. Comparison of the prediction miss rate of the WLAN NIC in a week.

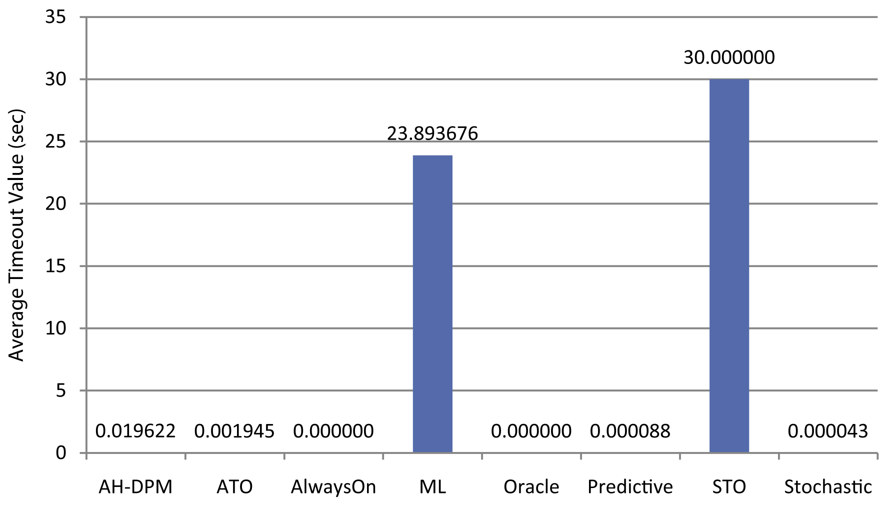


Fig. 19. Comparison of the average timeout value of the WLAN NIC in a week.

the break-even time, the average idle time will be shortened rapidly and will be smaller than the break-even time. When the average idle time is shorter than the break-even time, the SP will remain in the active state until the average idle time becomes larger than the break-even time, even if the next idle time is longer than the break-even time. This situation of the SP remaining in the active state while the actual idle time is longer than the break-even time will cause extra power consumption. Although the Predictive algorithm uses a watchdog mechanism to compensate the prediction inaccuracy caused by the bursty request arrivals pattern, the busy wait period caused by the watchdog mechanism will also result in extra power consumption.

The ML algorithm [20] has a drawback that is caused by the calculation of a weight factor for each expert. The weight of each expert is calculated as follows:

$$w_i^{t+1} = w_i^t \beta^{l_i^t} \tag{4}$$

where w_i^t is the weight factor of expert i, β is a chosen value between 0 and 1, which was assigned to 0.75 in the experiments [20], and l_i^t is the joint loss factor, which is given by:

$$l_i^t = \alpha \times l_{ie}^t + (1 - \alpha) \times l_{ip}^t \tag{5}$$

where l_{ie}^t and l_{ip}^t are the loss factors corresponding to energy savings and performance delay for an expert i [20]. Because β is between 0 and 1 and l_i^t is always positive, the weight will approach to zero after a series of calculation, which will cause the underflow problem. If this problem happens, the weight of each expert will be the same and the result of selecting an operational expert will be the same.

Finally, the performance of the Stochastic algorithm is not good because the request arrival pattern of the SR is bursty. With the bursty request arrival patterns of the hard disk and WLAN NIC, the state transition policies calculated by the Stochastic algorithm will tend to keep the SP in the

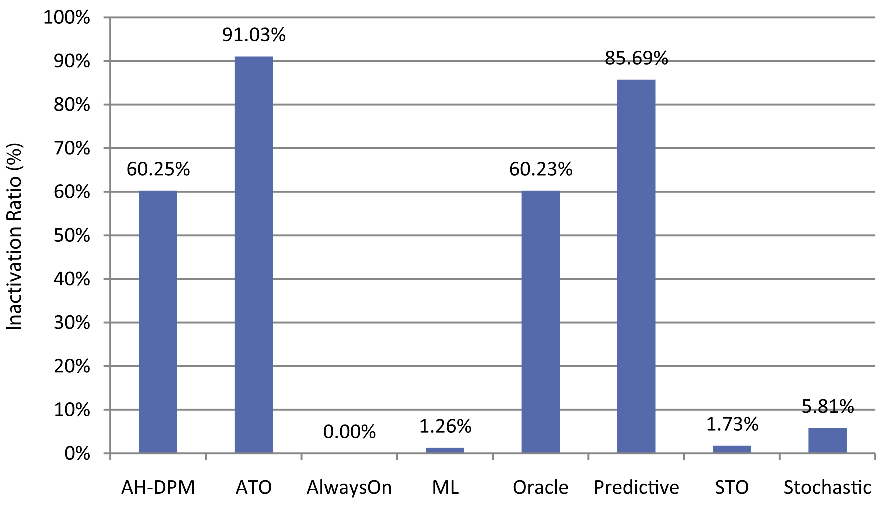


Fig. 20. Comparison of the inactivation ratio of the WLAN NIC in a week.

active state and it will result in high average power consumption.

5. Conclusions

In this paper, we have presented a new power efficient *adaptive hybrid dynamic power management* (AH-DPM) algorithm that can adapt to the self-similar workloads with bursty nature of the SPs (hard disk and WLAN NIC) in mobile devices to lengthen their battery lifetime. To adapt to bursty request arrival patterns with self-similarity of hard disks or WLAN NICs, the proposed AH-DPM first derives the average idle time of the SP in the bursty (ON) period and non-bursty (OFF) period separately. Then, to achieve better power saving, we use the average idle time in the ON period to adjust the timeout value more precisely and use the average idle time in the OFF period to decide which inactive state the SP should be switched to. Experimental results based on real traces of a hard disk have shown that the average power consumption of the proposed AH-DPM is better than that of the ATO, ML, Predictive, STO, and Stochastic algorithms by 69.40%, 101.68%, 113.69%, 95.33%, and 278.37%, respectively. Nevertheless, the proposed AH-DPM algorithm did not sacrifice the average response time of the hard disk too much, which is still lower than the average disk access time specified in a hard disk specification. In addition, experimental results based on real traces of a WLAN NIC have also shown that the average power consumption of the proposed AH-DPM is comparable to that of the ATO, Oracle, and Predictive algorithms, and is better than that of the ML, STO, and Stochastic algorithms by 172.34%, 89.14%, and 372.68%, respectively. As to the average packet transmission delay, the proposed AH-DPM algorithm is better than that of the ATO and Predictive algorithms by 23.22% and 25.18%, and is worse than that of the ML, STO, and Stochastic algorithms by 41.37%, 41.05%, and 38.20%, respectively. In summary, the experimental results have supported that the proposed AH-DPM algorithm can provide a better tradeoff between average power consumption and average response time (or average packet transmission delay) for hard disks and WLAN NICs, and thus it is really feasible to ever increasing mobile devices for extending their battery lifetime.

Acknowledgement

This work was supported by the NSC96-2628-E-009-140-MY3 and NSC99-2221-E-009-081-MY3.

References

[1] Y.-H. Lu, G.D. Micheli, Comparing system-level power management policies, IEEE Des. Test 18 (2001) 10–19.

[2] Y.-H. Lu, E.-Y. Chung, T. Simunic, L. Benini, G.D. Micheli, Quantitative comparison of power management algorithms, in: Proceedings of the Conference on Design, Automation and Test in Europe, ACM, Paris, France, 2000, pp. 20–26.

[3] K. Zhang, X. Ge, C. Liu, L.Xiang, Analysis of frame traffic characteristics in IEEE 802.11 networks, in: Proceedings of the 4th International Conference on Communications and Networking in China, IEEE, Wuhan, China, 2009, pp. 1–5.

[4] L. Xiang, X. Ge, K. Zhang, C. Liu, A self-similarity frame traffic model based on the frame components in 802.11 networks, in: Proceedings of the 2009 International Conference on Computational Science and Engineering, IEEE Computer Society, vol. 02, 2009, pp. 955–960.

[5] M.E. Gomez, V. Santonja, Self-similarity in I/O workload: analysis and modeling, in: Proceedings of the Workload Characterization: Methodology and Case Studies, IEEE Computer Society, 1998, p. 97.

[6] M.E. Gomez, V. Santonja, Analysis of self-similarity in I/O workload using structural modeling, in: Proceedings of the 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE Computer Society, 1999, p. 234.

[7] F. Douglis, P. Krishnan, B.N. Bershad, Adaptive Disk Spin-down Policies for Mobile Computers, in: Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing, USENIX Association, 1995, pp. 121–137.

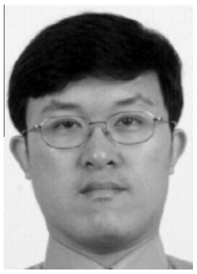
[8] C.M. Olsen, C. Narayanaswami, PowerNap: an efficient power management scheme for mobile devices, IEEE Transactions on Mobile Computing 5 (2006) 816–828.

[9] L. Benini, A. Bogliolo, G.D. Micheli, A survey of design techniques for system-level dynamic power management, IEEE Trans. Very Large Scale Integr. Syst. 8 (2000) 299–316.

[10] M.B. Srivastava, A.P. Chandrakasan, R.W. Brodersen, Predictive system shutdown and other architectural techniques for energy efficient programmable computation, IEEE Trans. Very Large Scale Integr. Syst. 4 (1996) 42–55.

[11] C.-H. Hwang, A.C.-H. Wu, A predictive system shutdown method for energy saving of event-driven computation, ACM Trans. Des. Autom. Electron. Syst. 5 (2000) 226–241.

[12] E.-Y. Chung, L. Benini, G.D. Micheli, Dynamic power management using adaptive learning tree, in: Proceedings of the 1999 IEEE/ACM International Conference on Computer-Aided Design, IEEE Press, San Jose, California, United States, 1999, pp. 274–279.
[13] D. Ramanathan, S. Irani, R.K. Gupta, An analysis of system level power management algorithms and their effects on latency, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 21 (2002) 291–305.
[14] L. Benini, A. Bogliolo, G.A. Paleologo, G.D. Micheli, Policy optimization for dynamic power management, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 18 (1999) 813–833.
[15] S. Irani, S. Shukla, R. Gupta, Online strategies for dynamic power management in systems with multiple power-saving states, ACM Trans. Embed. Comput. Syst. 2 (2003) 325–346.
[16] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, G.D. Micheli, Dynamic power management for nonstationary service requests, IEEE Trans. Comput. 51 (2002) 1345–1361.
[17] Z. Ren, B.H. Krogh, R. Marculescu, Hierarchical adaptive dynamic power management, IEEE Trans. Comput. 54 (2005) 409–420.
[18] Q. Qiu, Q. Qu, M. Pedram, Stochastic modeling of a power-managed system-construction and optimization, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 20 (2001) 1200–1217.
[19] P. Rong, M. Pedram, Battery-aware power management based on Markovian decision processes, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 25 (2006) 1337–1349.
[20] G. Dhiman, T.S. Rosing, Dynamic power management using machine learning, in: Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design, ACM, San Jose, California, 2006, pp. 747–754.
[21] V.L. Prabha, E.C. Monie, Hardware architecture of reinforcement learning scheme for dynamic power management in embedded systems, EURASIP J. Embedded Syst. (2007). pp. 1–6.
[22] Y. Tan, W. Liu, Q. Qiu, Adaptive power management using reinforcement learning, in: Proceedings of the 2009 International Conference on Computer-Aided Design, ACM, San Jose, California, 2009, pp. 461–467.
[23] Y. Ye, Interior Point Algorithms: Theory and Analysis, first ed., Wiley-Interscience, New York, 1997.
[24] Hitachi Global Storage Technologies, "Hard Disk Drive Specification Hitachi Travelstar 5K100 2.5 inch ATA/IDE hard disk drive". Available at: <http://www.hitachigst.com/tech/techlib.nsf/techdocs/98F75D1B2B25C42F86256EBC0065A2F9/$file/T5K100_sp.pdf>.
[25] Red Hat. Inc., "Fedora project". Available at: <http://fedoraproject.org/>.
[26] Linux Kernel Organization, Inc., "The linux kernel archives". Available at: <http://www.kernel.org/>.
[27] Hewlett-Packard Development Company, L.P., "The Intel PRO/Wireless 3945ABG (802.11a/b/g) Card Specification". Available at: <http://h18000.www1.hp.com/products/quickspecs/12510_div/12510_div.PDF>.
[28] L.Y. Zhang, Y. Ge, J. Hou, Energy-efficient real-time scheduling in IEEE 802.11 wireless LANs, in: Proceedings of the 23rd International Conference on Distributed Computing Systems, IEEE Computer Society, 2003, p. 658.
[29] E.-S. Jung, N.H. Vaidya, Improving IEEE 802.11 power saving mechanism, Wirel. Netw. 14 (2008) 375–391.
[30] Wireshark Foundation, "Wireshark". Available at: <http://www.wireshark.org/>.



Hung-Cheng Shih received the B.S. degree and the M.S. degree in the Department of Computer Science from National Chiao Tung University, Hsinchu, Taiwan, in 1997 and 2002. He is currently a Ph.D. student in the Department of Computer Science, National Chiao Tung University. His research interests include mobile computing, wireless Internet, and power management.

Kuochen Wang received the B.S. degree in Control Engineering from the National Chiao Tung University, Taiwan, in 1978, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Arizona in 1986 and 1991, respectively. He is currently a Professor/Director in the Institute of Computer Science and Engineering, National Chiao Tung University. He was a Deputy Director of the Computer and Network Center at this university from June 2007 to July 2009. He was a Visiting Scholar in the Department of Electrical Engineering, University of Washington from July 2001 to February 2002. From 1980 to 1984, he was a Senior Engineer at the Directorate General of Telecommunications in Taiwan. He served in the army as a second lieutenant communication platoon leader from 1978 to 1980. His research interests include wireless (ad hoc/sensor/VANET) networks, mobile (cloud) computing, and power management for multimedia portable devices.