Smart Power-Saving Mode for IEEE 802.11 Wireless LANs

Daji Qiao
Iowa State University
Ames, IA 50011
daji@iastate.edu

Kang G. Shin
The University of Michigan
Ann Arbor, MI 48109
kgshin@eecs.umich.edu

***Abstract*— Static PSM (Power-Saving Mode) schemes employed in the current IEEE 802.11 implementations could not provide any delay-performance guarantee because of their fixed wakeup intervals. In this paper, we propose a smart PSM (SPSM) scheme, which directs a wireless station to sleep/wake up according to an "optimal" sequence, such that the desired delay performance is guaranteed with minimum energy consumption. Instead of constructing the sequence directly, SPSM takes a unique two-step approach. First, it translates an arbitrary user-desired delay performance into a generic penalty function. Second, it provides a generic algorithm that takes the penalty function as the input and produces the optimal station action sequence automatically. This way, the potentially-complicated energy-consumption-minimization problem subject to delay-performance constraints is simplified and solved systematically.**

Our simulation results show that, with a two-stair penalty function, SPSM achieves delay performance similar to the BSD (Bounded SlowDown) protocol under various scenarios, but always with less energy consumption, thanks to its capability to adapt to changes in the response-time distribution. Moreover, because of SPSM's two-step design feature, it is more flexible than BSD in the sense of being able to meet arbitrary user-desired delay requirement, e.g., providing soft delay-bound guarantees with power penalty functions.

I. Introduction

Battery-powered portable computing and communication devices, such as laptops and PDAs, have become increasingly popular and widely-deployed, but their usefulness is severely constrained by the limited amount of energy stored in the accompanying battery. In order to extend the battery life and hence the system operation time, it is very important to have a well-designed power-management scheme for each wireless communication device, which contributes to a significant percentage of the total energy consumption. This paper focuses on the IEEE 802.11 Wireless LANs and the infrastructure-based[1] network architecture, which currently dominates home, office environments, and public hotspots.

A. Power States and Power Management in 802.11 WLANs

The IEEE 802.11 [1] allows a wireless station to be in one of two different power states: *awake* and *doze*. In the awake state, a wireless station is fully powered and is ready to communicate with others at any time. In contrast, it consumes extremely low power in the doze state but cannot transmit/receive packets or sense the wireless channel. Transition from the doze state to the awake state takes a short duration of time [2], during which a wireless station consumes significantly higher power than being in the steady awake state [3].

There are two different power-management modes for an 802.11 wireless station: AM (Active Mode) or PSM (Power-Saving Mode). The AP (Access Point) keeps track of power-management modes for all the wireless stations in its cluster. It temporarily buffers the packets that are destined for PSM stations, and transmits them only at designated times. Every *tBeaconPeriod*, the AP transmits a Beacon frame, which carries a TIM (Traffic Indication Map) indicating the buffer status of all the PSM stations in its cluster.

A PSM station stays in the doze state for most of time and only wakes up to listen for selected Beacon frames with a *fixed* wakeup interval. For this reason, we call the current 802.11 PSM a *static* scheme. If the TIM carried in a Beacon frame indicates the presence of buffered packets for a station, it stays awake and issues PS-Poll frames to retrieve the buffered packets, one at a time, until all the packets are received; otherwise, the station goes back to sleep. On the other hand, if a PSM station itself wants to initiate a transmission, it may wake up at any time to do so without waiting for a Beacon frame. In contrast, an AM station always stays in the awake state, and hence, the AP transmits/relays the packets that are destined for AM stations directly without any extra delay.

Moreover, if there is any PSM station in its cluster, the AP buffers the broadcast/multicast packets, and transmits them immediately following a Beacon frame containing a special Delivery TIM (DTIM). The Beacon frames containing DTIMs are transmitted every *tDTIMPeriod*, which is a multiple of Beacon periods. Note that a PSM station is allowed to skip DTIM announcements if it is not interested in receiving broadcast/multicast packets.

B. Motivation and Contributions

From a networking perspective, a typical user's online activity, such as web-browsing, can be viewed as a sequence of request-response exchanges between the mobile user station and the Internet content server(s). So, a natural way to save energy is to operate a mobile wireless station in the power-saving mode as follows. After a wireless station sends a request, instead of staying in the awake state, being idle and waiting for the response packet, it enters the doze state and

[1] An infrastructure network includes an AP (Access Point) that provides both the connection to the wired network, if any, and the local relaying function between the wireless stations.

0-7803-8968-9/05/$20.00 (C)2005 IEEE

then takes actions (wake up or sleep) according to its PSM scheme. However, the problem with this approach is that, since the wireless station cannot communicate during its stay in the doze state, it is very likely that the response packet has to be buffered at the AP and delivered to the station at a later time, i.e., causing a potential response slowdown [4]. Obviously, the extra delay resulted from this approach is dictated by the station's action sequence.

In general, the less frequently a wireless station wakes up and/or the shorter time the station stays in the awake state, the less energy the station consumes, but more likely it will result in a larger extra delay. So, there is an inherent tradeoff between energy conservation and delay performance, and it is always desirable to find the station action sequence that satisfies a user-desired delay requirement while minimizing the energy consumption. In this paper, we propose a SPSM (*Smart* PSM) scheme, which is in sharp contrast to the static PSM schemes employed in the current 802.11 implementations. SPSM is a *generic* solution since it (1) translates a user-desired delay performance into a *generic* penalty function, and (2) provides a *generic* algorithm that takes the penalty function as the input and yields the optimal station action sequence automatically. This way, the potentially-complicated energy-consumption-minimization problem subject to delay-performance constraints is simplified and, more importantly, solved systematically.

C. Related Work

The authors of [4] presented a BSD (Bounded SlowDown) protocol, which is pioneering work on the tradeoff between minimizing energy consumption and reducing response delay with the IEEE 802.11 PSM. With BSD, after a wireless station sends a request, it stays awake for a certain period before entering the doze state. Then, it increases its wakeup interval gradually in a controlled manner until the response packet returns. The response slowdown is, therefore, bounded while energy is conserved. Notice that, since BSD implicitly assumes that the response packet may return soon after the request was made, it does not adapt dynamically to variation of the response-time distribution. Moreover, BSD is designed to guarantee a specific type of delay performance in bounding the response slowdown. So, one may naturally ask: *Is it possible to extend BSD to guarantee arbitrary user-desired delay performance?* Unfortunately, there has not been any good way to do this. Our proposed SPSM scheme deals with this problem from a different angle from BSD and provides a two-step solution. In fact, as we will show in Section V, the BSD protocol is one special case of SPSM, and can be derived with our approach by using a two-stair penalty function.

In [5], the authors used a TISMDP (Time-Indexed Semi-Markov Decision Process) model to derive an optimal policy for dynamic power management in portable systems. In [6], several application-specific policies were provided to put an idle WLAN device in the doze state. The authors of [7] implemented a STPM (Self-Tuning Power Management) module in the Linux kernel, which adjusts dynamically the power-management mechanism for 802.11 devices using application hints. The authors of [8] implemented a power-aware transport protocol by which a wireless station can judiciously suspend and restart its communication device, thus reducing the power usage of the communication device significantly. One common problem of the above schemes is that none of them could provide any delay-performance guarantee.

An alternate way to conserve energy is via TPC (Transmit Power Control) that allows an awake wireless station to transmit at the minimum required power level [9]–[12]. This is complementary to our proposed SPSM scheme that addresses a different problem of switching between the awake and doze states.

There have also been some studies on energy conservation in ad hoc wireless networks [13]–[15]. In [13], the authors proposed an enhancement to the power-management policy in 802.11-based ad hoc networks by dynamically changing the size of the ATIM (Ad hoc Traffic Indication Message) window independently for each wireless station. Three asynchronous power-management protocols were proposed in [14] for multi-hop networks by improving the current 802.11 PSM. The authors of [15] proposed a power-saving technique, called *Span*, for multi-hop ad hoc networks. Span adaptively elects coordinators among all nodes in the network. Elected coordinators stay awake and perform multi-hop routing within the network, while other nodes remain sleeping and check periodically whether they should wake up to become coordinators.

D. Organization

The rest of this paper is organized as follows. Section II gives the problem statement and introduces the proposed SPSM scheme. The details of SPSM are presented in Sections III and IV, which describe a simple algorithm to find the optimal station action sequence and a generic method for interpreting the user-desired delay performance, respectively. Section V presents and assesses the simulation results and, finally, the paper concludes with Section VI.

II. Smart Power-Saving Mode

Let t_0 and t_x denote the time points when a wireless station sends a request and when the response packet returns, respectively. The Beacon points after the request is sent are denoted by t_i ($i \geqslant 1$) and the interval between two adjacent Beacon points is *tBeaconPeriod*. During each Beacon interval $[t_i,\ t_{i+1})$ ($i \geqslant 1$), the wireless station is allowed to take any one of the three power-management-related actions, denoted by $\mathcal{A}_i$, in Table I.

Let *tDoze2Awake* denote the short doze-to-awake transition period. A wireless station consumes higher power ($\mathcal{P}_t$) than in the steady awake state ($\mathcal{P}_w$) during this period.

We are interested in a *station action sequence* ($\mathcal{S}$) that is in the form of

$$\mathcal{S} = \{\mathcal{A}_0,\ \mathcal{A}_1,\ \cdots,\ \mathcal{A}_i,\ \cdots\}, \tag{1}$$

where $\mathcal{A}_0 = w$ or s corresponds to the wireless station staying awake or going to sleep immediately after the request is sent, respectively.

TABLE I

THREE POWER-MANAGEMENT-RELATED ACTIONS DURING BEACON INTERVAL $[t_i, t_{i+1})$

Action	Description
Awaken (w)	The wireless station is fully powered (at the power level of $\mathcal{P}_w$) for the entire interval. If the response packet returns during this interval, the AP simply relays it without buffering.
Sleep (s)	The wireless station remains sleeping (at the power level of $\mathcal{P}_s$) for the entire interval. It skips the TIM announcement at t_i, and if the response packet returns during this interval, the AP buffers the packet for future delivery.
Alarm (a)	The wireless station wakes up at t_i for a short period of *tAlarmTime* to listen for the Beacon frame. If the TIM carried in the Beacon frame indicates the presence of buffered response packet(s) for itself, it generates PS-Poll frame(s) to retrieve the packet(s); otherwise, the station goes back to sleep.

Obviously, the less frequently a station wakes up and/or the shorter time a station remains awake, the less energy the station consumes, but more likely it will take the station longer time (incurring a larger extra delay) to retrieve the response packet. So, there is an inherent tradeoff. Fig. 1 shows a simple example to support the above statement. With station action sequence $\mathcal{S}_1$, the station wakes up at the least frequency of every eight Beacon intervals, which consumes the least amount of energy but always results in the largest extra delay ($\mathcal{D}_{xi}$) regardless when the response packet returns (t_{xi}). In contrast, by increasing the wakeup frequency (e.g., station action sequence $\mathcal{S}_2$) or the wakeup period (e.g., station action sequence $\mathcal{S}_3$), the extra delay is reduced at the expense of more energy consumption.

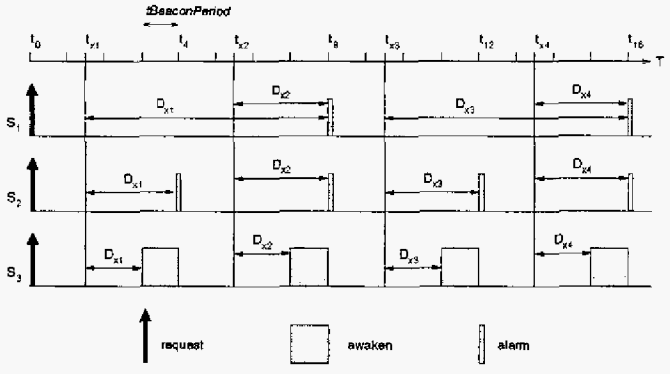


Fig. 1. Comparison of three station action sequences ($\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$)

It is always desirable to find the station action sequence that satisfies a user-desired delay requirement while minimizing the energy consumption. The simplest way to solve this energy-consumption minimization problem subject to delay-performance constraints is to construct the sequence directly based on careful examination of the system and thorough understanding of the relevant tradeoffs, e.g., the construction procedure of the BSD (Bounded SlowDown) protocol in [4]. However, it is not always easy or feasible to do so.

In this paper, we propose a SPSM (Smart Power-Saving Mode) scheme, which deals with this problem from a different angle. It is a two-step solution. First, it interprets a user-desired delay performance using a generic penalty function. Second, it provides a simple recursive algorithm that takes the penalty function as the input and produces the optimal station action sequence automatically. This way, a potentially-complicated problem is simplified and solved systematically.

The objective of SPSM can be formally described as follows. Given any user-desired delay performance, find the *optimal station action sequence* ($\mathcal{S}^*$) to minimize the corresponding expected weighted energy consumption. Here, the *expected weighted energy consumption* ($\mathcal{W}$) is a performance metric we introduce to evaluate (quantitatively) a station action sequence $\mathcal{S}$. It is defined as

$$\mathcal{W}(\mathcal{S}) = \int_{t_0}^{\infty} \mathcal{E}(t_x, \mathcal{S}) \cdot \mathcal{C}(\mathcal{D}(t_x, \mathcal{S})) \cdot f_{t_x} \cdot dt_x, \qquad (2)$$

where f_{t_x} represents the distribution of the response time t_x, and $\mathcal{E}(t_x, \mathcal{S})$ and $\mathcal{D}(t_x, \mathcal{S})$ are the corresponding energy consumption for awaiting the response packet and the resultant extra delay, respectively, when the station acts according to $\mathcal{S}$ to retrieve a response packet that returns at t_x. Note that we do not include the energy consumed to send the request or to receive the response packet as part of $\mathcal{E}$, as they are irrelevant to the power-management scheme adopted by the wireless station. $\mathcal{C}$ is a *penalty function* and different user-desired delay performances can be interpreted as, or translated into, different $\mathcal{C}$ functions.

III. A SIMPLE ALGORITHM TO FIND THE OPTIMAL STATION ACTION SEQUENCE

We now proceed to the second step of SPSM and investigate the problem of finding $\mathcal{S}^*$ by assuming the availability of the penalty function, which will be discussed in the next section. Also, we assume that the station has the knowledge of *tBeaconPeriod*, *tAlarmPeriod*, *tDoze2Awake*, and relevant power-usage information. Instead of exhaustively testing all the possible candidate sequences, we develop a novel recursive algorithm to simplify the search procedure.

A. Theorem and Corollary

Let $\mathcal{S}_i$ ($i \geqslant 0$) denote a sub-sequence of $\mathcal{S}$:

$$\mathcal{S}_i = \{\mathcal{A}_i,\ \mathcal{A}_{i+1},\ \cdots\}. \qquad (3)$$

It is called an *active station action sub-sequence*, and denoted by $\hat{\mathcal{S}}_i$, if it starts with an active action, i.e., w or a. For example, if $\mathcal{S} = \{s,\ a,\ s,\ w,\ \cdots\}$, then $\hat{\mathcal{S}}_1 = \{a,\ s,\ w,\ \cdots\}$ is an active sub-sequence, while $\mathcal{S}_2 = \{s,\ w,\ \cdots\}$ is not. We have the following theorem:

THEOREM. *If $\mathcal{S}^*$ is the optimal station action sequence that minimizes the expected weighted energy consumption, then any active sub-sequence of $\mathcal{S}^*$, denoted by $\hat{\mathcal{S}}_i^*$, is also optimal in the sense of minimizing*

$$\mathcal{W}_i(\hat{\mathcal{S}}_i) = \int_{t_i}^{\infty} \mathcal{E}(t_x, \hat{\mathcal{S}}_i) \cdot \mathcal{C}(\mathcal{D}(t_x, \hat{\mathcal{S}}_i)) \cdot f_{t_x} \cdot dt_x, \qquad (4)$$

or in other words,

$$\hat{\mathcal{S}}_i^* = \arg\min_{\hat{\mathcal{S}}_i} \mathcal{W}_i(\hat{\mathcal{S}}_i). \tag{5}$$

The theorem holds because if there exists another active sub-sequence $\hat{\mathcal{S}}_i'$ that results in a smaller value of $\mathcal{W}_i$, then by simply replacing $\hat{\mathcal{S}}_i^*$ with $\hat{\mathcal{S}}_i'$ in $\mathcal{S}^*$, the new $\mathcal{S}'$ will result in a smaller expected weighted energy consumption, which contradicts the assumption that $\mathcal{S}^*$ is optimal. Note, however, that this statement is true only with active sub-sequences. This is because replacing an active sub-sequence will only affect the delay and energy-consumption performances when the response packet returns after the starting time of the sub-sequence, which does not hold if a sub-sequence starts with a Sleep (s) action. Consequently, we have the following corollary:

COROLLARY. *If $\hat{\mathcal{S}}_i^*$ is optimal in the sense of minimizing $\mathcal{W}_i$, as described by Eqs. (4) and (5), then any active sub-sequence of $\hat{\mathcal{S}}_i^*$ is also optimal in a similar sense.*

B. Recursive Algorithm

Now, we describe in detail the algorithm to find the optimal station action sequence $\mathcal{S}^*$.

First, consider the general case of $\hat{\mathcal{S}}_i^*$ when $i \geqslant 0$. According to Corollary, we only need to check the candidate sequences in the form of $\hat{\mathcal{S}}_i = \{w/a,\ s,\ \cdots,\ s,\ \hat{\mathcal{S}}_j^*\}$ $(i < j \leqslant m(i))$, where $t_{m(i)}$ is the most adjacent mandatory wakeup point[2] after t_i.

Assume that $\hat{\mathcal{S}}_i = \{w,\ s,\ \cdots,\ s,\ \hat{\mathcal{S}}_j^*\}$ is selected. There are three possible scenarios as follows.

- If the response packet returns between t_i and t_{i+1}, i.e., $t_i \leqslant t_x < t_{i+1}$, the station only needs to stay awake for $(t_x - t_i)$ time and is able to receive the packet without any extra delay. Therefore, $\mathcal{D} = 0$.
- If the response packet returns between t_{i+1} and t_j, i.e., $t_{i+1} \leqslant t_x < t_j$, it is buffered at the AP. The station wastes e_w energy for being fully-powered during $[t_i,\ t_{i+1})$, then sleeps during $[t_{i+1},\ t_j)$, and will be notified of the buffered response packet after it listens for the TIM announcement at t_j (consuming e_a energy). e_w and e_a are

$$\begin{cases} e_w = \mathcal{P}_w \cdot \mathit{tBeaconPeriod}, \\ e_a = \mathcal{P}_w \cdot \mathit{tAlarmPeriod}. \end{cases} \tag{6}$$

  In this scenario, $\mathcal{D} = t_j - t_x$.
- If the response packet returns after t_j, i.e., $t_j \leqslant t_x$, the station wastes e_w energy to stay awake during $[t_i,\ t_{i+1})$ and then sleeps during $[t_{i+1},\ t_j)$. The resultant extra delay can be calculated in exactly the same way as when $\hat{\mathcal{S}}_j^*$ is determined.

[2] *Mandatory wakeup points* are defined as the Beacon points when a wireless station is required to wake up and communicate with the AP. For example, if a wireless station is mandated to listen for the DTIM announcements and receive the potential broadcast/multicast packets, the DTIM points are its mandatory wakeup points.

Recall that the extra energy consumed during the doze-to-awake transition is

$$e_t = (\mathcal{P}_t - \mathcal{P}_w) \cdot \mathit{tDoze2Awake}. \tag{7}$$

Hence, $\mathcal{W}_i$ can be calculated recursively as follows:

$$\begin{aligned} &\mathcal{W}_i(\{w,\ s,\ \cdots,\ s,\ \hat{\mathcal{S}}_j^*\}) \\ &= \int_{t_i}^{t_{i+1}} \mathcal{P}_w \cdot (t_x - t_i) \cdot \mathcal{C}(0) \cdot f_{t_x} \cdot dt_x \\ &+ \int_{t_{i+1}}^{t_j} (e_w + \mathcal{P}_s \cdot (t_j - t_{i+1}) + e_t + e_a) \\ &\qquad \cdot \mathcal{C}(t_j - t_x) \cdot f_{t_x} \cdot dt_x \\ &+ \left[\left(e_w + \mathcal{P}_s \cdot (t_j - t_{i+1}) + \left(1 - \left\lfloor \frac{t_{i+1}}{t_j} \right\rfloor\right) \cdot e_t\right)\right. \\ &\qquad \left. \cdot\, \mathcal{C}_j(\hat{\mathcal{S}}_j^*) + \mathcal{W}_j(\hat{\mathcal{S}}_j^*)\right], \end{aligned} \tag{8}$$

where

$$\mathcal{C}_j(\hat{\mathcal{S}}_j^*) = \int_{t_j}^{\infty} \mathcal{C}\left(\mathcal{D}(t_x, \hat{\mathcal{S}}_j^*)\right) \cdot f_{t_x} \cdot dt_x \tag{9}$$

is a special notation for simplicity. Term $\left(1 - \left\lfloor \frac{t_{i+1}}{t_j} \right\rfloor\right) \cdot e_t$ accounts for the fact that there is no doze-to-awake transition incurred when an awake wireless station takes an active action.

Similarly, with $\hat{\mathcal{S}}_i = \{a,\ s,\ \cdots,\ s,\ \hat{\mathcal{S}}_j^*\}$, we have

$$\begin{aligned} &\mathcal{W}_i(\{a,\ s,\ \cdots,\ s,\ \hat{\mathcal{S}}_j^*\}) \\ &= \int_{t_i}^{t_j} (e_a + \mathcal{P}_s \cdot (t_j - t_i - \mathit{tAlarmPeriod}) + e_t + e_a) \\ &\qquad \cdot \mathcal{C}(t_j - t_x) \cdot f_{t_x} \cdot dt_x \\ &+ \left[(e_a + \mathcal{P}_s \cdot (t_j - t_i - \mathit{tAlarmPeriod}) + e_t) \cdot \mathcal{C}_j(\hat{\mathcal{S}}_j^*)\right. \\ &\qquad \left. + \mathcal{W}_j(\hat{\mathcal{S}}_j^*)\right]. \end{aligned} \tag{10}$$

Therefore,

$$\hat{\mathcal{S}}_i^* = \arg\min_{\hat{\mathcal{S}}_i \in \mathbb{K}_i} \mathcal{W}_i(\hat{\mathcal{S}}_i), \tag{11}$$

where

$$\mathbb{K}_i = \left\{ \forall j : i < j \leqslant m(i),\quad \{w/a,\ \overbrace{s,\ \cdots,\ s}^{j-i-1},\ \hat{\mathcal{S}}_j^*\} \right\} \tag{12}$$

with

$$|\mathbb{K}_i| = 2 \cdot (m(i) - i). \tag{13}$$

Now, consider the special case of $\hat{\mathcal{S}}_M^*$, where t_M is a mandatory wakeup point that renders

$$\int_{t_M}^{\infty} f_{t_x} \cdot dt_x < \epsilon, \tag{14}$$

where ϵ is a significantly-small positive number. Since the possibility that the response packet returns after t_M is extremely low and hence negligible, we let

$$\mathcal{A}_M = a, \quad \text{and} \quad \begin{cases} \mathcal{W}_M = 0, \\ \mathcal{C}_M = 0. \end{cases} \tag{15}$$

By using this special case as the boundary condition, we have fully specified the computations of $\hat{\mathcal{S}}_i^*$ ($i \geqslant 0$) by Eqs. (8), (10), (11), and (15), and $\mathcal{S}^*$ is simply

$$\mathcal{S}^* = \arg\min_{\mathcal{S}\in\mathbb{K}} \mathcal{W}(\mathcal{S}), \tag{16}$$

where

$$\mathbb{K} = \left\{ \forall i : 0 \leqslant i \leqslant m(0), \quad \{\overbrace{s, \cdots, s}^{i}, \hat{\mathcal{S}}_i^*\} \right\} \tag{17}$$

with

$$|\mathbb{K}| = m(0) + 1, \tag{18}$$

and

$$\begin{aligned} &\mathcal{W}(\{s, \cdots, s, \hat{\mathcal{S}}_i^*\}) \\ &= \int_{t_0}^{t_i} (\mathcal{P}_s \cdot (t_i - t_0) + e_t + e_a) \cdot \mathcal{C}(t_i - t_x) \cdot f_{t_x} \cdot dt_x \\ &+ \left[\left(\mathcal{P}_s \cdot (t_i - t_0) + \left(1 - \left\lfloor \frac{t_0}{t_i} \right\rfloor \right) \cdot e_t \right) \cdot \mathcal{C}_i(\hat{\mathcal{S}}_i^*) \right. \\ &\quad \left. + \mathcal{W}_i(\hat{\mathcal{S}}_i^*) \right]. \end{aligned} \tag{19}$$

C. An Example

We give a simple example to illustrate the above-described recursive computation. Assume

- $\mathcal{P}_w = 0.925$ W, $\mathcal{P}_s = 0.045$ W, and $\mathcal{P}_t = 2 \cdot \mathcal{P}_w$.
- *tBeaconPeriod* = 100 ms, *tAlarmPeriod* = 2 ms, and *tDoze2Awake* = 250 μs.

Therefore, we have

$$e_w = 92.5 \text{ mJ}, \ e_a = 1.85 \text{ mJ}, \text{ and } e_t = 0.23125 \text{ mJ}.$$

Besides, assume that $t_1 - t_0 = 50$ ms and $m(i) = 5$ ($0 \leqslant i \leqslant 4$), meaning that the wireless station initiates a request in the middle of a Beacon interval and t_5 is the first mandatory wakeup point after the request. The simple CDF (Cumulative Distribution Function) for t_x used in this example is shown in Fig. 2. The desired delay performance is described by the

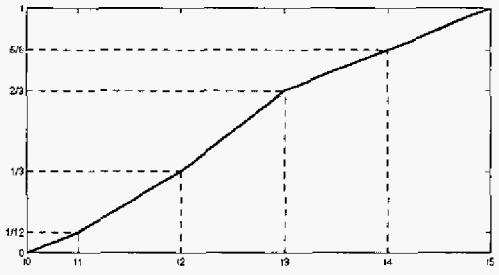


Fig. 2. A simple CDF (Cumulative Distribution Function) for t_x

following penalty function:[3]

$$\mathcal{C}(\mathcal{D}) = \begin{cases} 1 & \text{if } \mathcal{D} \leqslant t_x - t_0, \\ \infty & \text{otherwise.} \end{cases} \tag{20}$$

[3]More details about this penalty function will be discussed in the next section.

Table II lists the results of $\hat{\mathcal{S}}_i^*$'s ($0 \leqslant i \leqslant 5$) and the respective corresponding values of $\mathcal{W}_i$ and $\mathcal{C}_i$. We only recap

TABLE II
COMPUTATION RESULTS OF $\hat{\mathcal{S}}_i^*$'S

$\hat{\mathcal{S}}_i^*$		$\mathcal{A}_0$	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{W}_i$ (mJ)	$\mathcal{C}_i$
i	5	–	–	–	–	–	a	0	0
	4	–	–	–	–	a	a	1.39	1/6
	3	–	–	–	a	a	a	3.86	1/3
	2	–	–	a	a	a	a	8.81	2/3
	1	–	w	s	a	a	a	80.86	11/12
	0	w	w	s	a	a	a	167.58	1

the computation details of $\hat{\mathcal{S}}_4^*$. There are two candidates for $\hat{\mathcal{S}}_4^*$: $\{w, \hat{\mathcal{S}}_5^*\} = \{w, a\}$ and $\{a, \hat{\mathcal{S}}_5^*\} = \{a, a\}$. Since

$$\begin{aligned} \mathcal{W}_4(\{w, a\}) &= \int_{t_4}^{t_5} 0.925 \cdot (t_x - t_4) \cdot \frac{\frac{1}{6}}{t_5 - t_4} \cdot dt_x \\ &= 7.71 \text{ mJ}, \end{aligned}$$

and

$$\begin{aligned} \mathcal{W}_4(\{a, a\}) &= \int_{t_4}^{t_5} (1.85 + 0.045 \cdot 98 + 0.23 + 1.85) \\ &\quad \cdot \frac{\frac{1}{6}}{t_5 - t_4} \cdot dt_x \\ &= 1.39 \text{ mJ}, \end{aligned}$$

we have $\hat{\mathcal{S}}_4^* = \{a, a\}$. In fact, the optimal station action sequence is

$$\mathcal{S}^* = \arg\min_{\mathcal{S}\in\mathbb{K}} \mathcal{W}(\mathcal{S}) = \left\{\hat{\mathcal{S}}_0^*\right\} = \{\overset{\mathcal{A}_0}{\downarrow} w, \ w, \ s, \ a, \ a, \ \overset{\mathcal{A}_5}{\downarrow} a\}.$$

D. Implementation Issues

1) Feasibility of SPSM: The IEEE 802.11 standard [1] provides an MLME-POWERMGT.request primitive and includes a Power-Management bit in the Frame Control field of the MAC header for a wireless station to implement any *smart* power-management mechanism, including SPSM.

MLME-POWERMGT.request is generated by the SME (Station Management Entity) and has three arguments: *PowerManagementMode*, *WakeUp*, and *ReceiveDTIMs*. A wireless station may request a change of its power-management mode by using this primitive with *PowerManagementMode* set to the desired value (ACTIVE or POWER_SAVE). After that, the wireless station needs to inform the AP of the mode change through a frame exchange initiated by the station. The Power-Management bit of the frame sent by the station in this exchange indicates the power-management mode that the station will adopt after successful completion of the ongoing frame exchange. In addition, when a wireless station is in the power-saving mode, it may force its wireless network interface to wake up at any time by using this primitive with *WakeUp* set to True.

2) Estimating the Response-Time Distribution: As described in Section II, in order to determine the optimal station action sequence in SPSM, a wireless station needs to estimate the response-time distribution, and we apply an exponential moving average algorithm for this purpose.

Let f_{t_x} be the current estimate of the distribution. Assume that we observe a new response packet returning between t_μ and t_ν, where t_μ is either the request time t_0 or the most adjacent Beacon point when the wireless station is in the awake state before t_x, whichever is closer to t_x, and t_ν is the most adjacent Beacon point when the wireless station is in the awake state after t_x. Since a wireless station may initiate a request at any time during a Beacon interval, we have

$$t_1 - tBeaconPeriod \leqslant t_0 < t_1 \quad \text{and} \quad t_\mu < t_x \leqslant t_\nu$$
$$\Longrightarrow \quad \max(\mu - 1, 0) < \frac{t_x - t_0}{tBeaconPeriod} \leqslant \nu,$$

and hence, the new estimate of the distribution based on this observation is

$$f_{t_x}^{new} = \begin{cases} \frac{1}{[\nu - \max(\mu-1,0)] \cdot tBeaconPeriod} \\ \qquad \text{if} \quad \max(\mu-1,0) < \frac{t_x - t_0}{tBeaconPeriod} \leqslant \nu, \\ 0 \qquad \text{otherwise.} \end{cases} \tag{21}$$

Consequently, f_{t_x} is updated to

$$f_{t_x} = \alpha \cdot f_{t_x} + (1 - \alpha) \cdot f_{t_x}^{new}, \tag{22}$$

where α is a smoothing factor.

IV. A Generic Method for Describing User-Desired Delay Performances

We now return to the first step of SPSM and study various user-desired delay performances and their corresponding penalty functions. Note that a reasonable penalty function should be non-decreasing, and without loss of generality, we let $\mathcal{C}(0) = 1$.

A. Constant Penalty Function

If a user simply wants to minimize the energy consumption of its wireless station without any regard to delay performance, the corresponding penalty function is trivial:

$$\forall \mathcal{D}, \quad \mathcal{C}(\mathcal{D}) = 1. \tag{23}$$

B. Two-Stair Penalty Function

If a user is willing to accept certain response slowdown (specified by a bound Θ) but will not tolerate any additional delay beyond Θ, i.e., Θ is a *hard* delay bound, the corresponding penalty function is:

$$\mathcal{C}(\mathcal{D}) = \begin{cases} 1 & \text{if} \quad \mathcal{D} \leqslant \Theta, \\ \infty & \text{otherwise.} \end{cases} \tag{24}$$

With such a two-stair penalty function, we have

$$\mathcal{W}(\mathcal{S}) = \int_{t_x \in \mathbb{R}^-} \mathcal{E}(t_x, \mathcal{S}) \cdot 1 \cdot f_{t_x} \cdot dt_x + \int_{t_x \in \mathbb{R}^+} \mathcal{E}(t_x, \mathcal{S}) \cdot \infty \cdot f_{t_x} \cdot dt_x, \tag{25}$$

where $\mathbb{R}^-$ is the set of response times, given the station action sequence $\mathcal{S}$, each of them results in an extra delay that is equal to or smaller than Θ, and

$$\mathbb{R}^+ = \overline{\mathbb{R}^-}. \tag{26}$$

Clearly, due to the extreme penalty enforced on situations when the resultant extra delay exceeds Θ, in order to minimize $\mathcal{W}$, $\mathcal{S}^*$ must guarantee that

$$\mathbb{R}^+ = \emptyset, \tag{27}$$

i.e., the extra delay is bounded by Θ. Note that Θ can be given either by time units (as an absolute bound) or by percentage of the actual request-response turnaround time (as a relative bound), and we are more interested in the latter one, i.e.,

$$\Theta = \mathcal{B} \cdot (t_x - t_0), \tag{28}$$

where $\mathcal{B}$ is called the *slowdown factor*.

C. Power Penalty Function

On the other hand, if a user wants to exercise a *soft* delay bound on the response slowdown and, hence, is willing to tolerate late response returns (after the delay bound) as long as the energy consumption is kept low, the corresponding penalty function could be in the form of:

$$\mathcal{C}(\mathcal{D}) = 1 + \left(\frac{\mathcal{D}}{\Theta}\right)^z, \tag{29}$$

and the exponent value (z) reflects the extent to which the user is willing to tolerate the excessive delay. In general, this soft delay bound becomes *harder* as z increases, and in the extreme case when $z = \infty$, the power penalty function is equivalent to the two-stair penalty function.

D. Summary

Based on the above analysis, the qualitative descriptions of user-desired delay performances (using *acceptable response slowdown* and *excessive-delay tolerance level*) and their corresponding quantitative penalty functions are summarized in Table III.

TABLE III
User-Desired Delay Performances and Corresponding Penalty Functions

Acceptable Response Slowdown	Excessive-Delay Tolerance Level	Penalty Function
any	–	constant
Θ	*zero*	two-stair
	low	(high) power
	medium	(medium) power
	high	(low) power

V. Performance Evaluation

A. Simulation Setup

SPSM is evaluated using the ns-2 simulator [16]. We study the delay and energy-consumption performances of a wireless station in an infrastructure-based 802.11b system, and *tBeaconPeriod* and *tDTIMPeriod* are set to 100 ms and 1 s, respectively. Besides, *tAlarmPeriod* is set to 2 ms [4] and *tDoze2Awake* is set to 250 μs [2]. The wireless station is required to wake up at every DTIM point to receive the potential broadcast/multicast packets, i.e., the DTIM points are the mandatory wakeup points. Moreover, based on the power characteristics of the Orinoco 11b Client PC Card [17], we assume the power usage of the simulated wireless network interface to be $\mathcal{P}_w = 0.925$ W, $\mathcal{P}_s = 0.045$ W, and $\mathcal{P}_t = 2{\cdot}\mathcal{P}_w$. In fact, since we are only interested in how SPSM adapts the station action sequence to save energy, not the exact amount of energy savings, this assumption has little impact on the conclusions to be presented in this section.

During each simulation run, the wireless station requests 10,000 packets from Internet content server(s), which are separated by arbitrary-long user-thinking time. The request-response turnaround time consists of a relatively-stable server RTT (Round Trip Time) and a server response delay, which is modeled with a CDF (shown in Fig. 3) similar to the one used in [4].

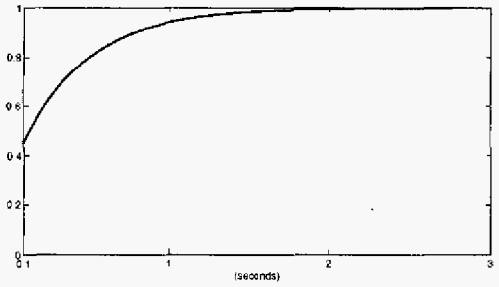


Fig. 3. CDF of the server response delay used in the simulation

We evaluate five SPSM schemes with various penalty functions, and compare them with two static PSM schemes (with various fixed wakeup intervals) and a realistic BSD implementation that (1) considers the mandatory wakeup points, and (2) rounds the sleep (wakeup) moment forward (backward) to the adjacent Beacon point when determining the station action sequence. Table IV lists the testing schemes.

The penalty functions for SPSM-2S and SPSM-Pz ($z = 1$, 2, 5, 10, 20) are, respectively,

$$\mathcal{C}(\mathcal{D}) = \begin{cases} 1 & \text{if } \mathcal{D} \leqslant \mathcal{B} \cdot (t_x - t_0), \\ \infty & \text{otherwise,} \end{cases} \tag{30}$$

and

$$\mathcal{C}(\mathcal{D}) = 1 + \left(\frac{\mathcal{D}}{\mathcal{B} \cdot (t_x - t_0)} \right)^z. \tag{31}$$

The slowdown factor is set to $\mathcal{B} = 0.2$ unless specified otherwise. Besides, the smoothing factor of the exponential moving average algorithm (to estimate the response-time distribution) is set to $\alpha = 0.9$.

TABLE IV
List of the testing schemes

Name	Station Action Sequence	Penalty Function
SPSM-2S	adaptive	two-stair
SPSM-P1	adaptive	linear
SPSM-P2	adaptive	power-of-2
SPSM-P5	adaptive	power-of-5
SPSM-P10	adaptive	power-of-10
SPSM-P20	adaptive	power-of-20
PSM-D	wakes up every *tDTIMPeriod*	–
PSM-B	wakes up every *tBeaconPeriod*	–
BSD	determined by the BSD protocol	–

The testing schemes are compared with each other in terms of

- *Average Response Slowdown* – the ratio of the observed request-response turnaround time to the actual request-response turnaround time;
- *Delay-Bound Miss Ratio*;
- *Per-Request Energy Consumption* – the energy consumed after a request is sent until the station is notified of return of the response packet.

Furthermore, for evaluation purpose, we also simulate the *benchmark* scenario when the station is always awake (no PSM) and is able to retrieve the response packet without any extra delay, i.e., the response slowdown is one. We conduct the simulation with various server RTTs.

B. Comparison of SPSM Station Action Sequences

Before discussing the simulation results, we first compare graphically the station action sequences of various SPSM schemes in Figs. 4 and 5, and all the sequences are obtained by assuming that the wireless station sends a request in the middle of a Beacon interval. Note that in both figures, a solid arrow and a wide (narrow) light-shaded bar represent the wireless station sending a request or taking an Awaken (Alarm) action, respectively, and we single out the mandatory wakeup points by dark-shading the Alarm bars.

Fig. 4 compares the station action sequences of various SPSM schemes when the server RTT is fixed at 10 ms. We have two observations. First, with a two-stair penalty function, SPSM-2S requires the wireless station to stay awake for 550 ms after sending the request and then to start sleeping and waking up every *tBeaconPeriod*. About one second after the request, the station doubles its wakeup interval, and so on. In fact, such an SPSM-2S station action sequence is identical to that of the BSD protocol because both mechanisms have the same design goals in bounding the delay performance to a 1.2× response slowdown. Second, compared with SPSM-2S, the SPSM-Pz station action sequences are less demanding, meaning that the wireless station is allowed to sleep earlier after the request and/or wake up less frequently. In general, as z increases, the penalty enforced on situations when the response packet returns after the delay bound goes up drastically, and

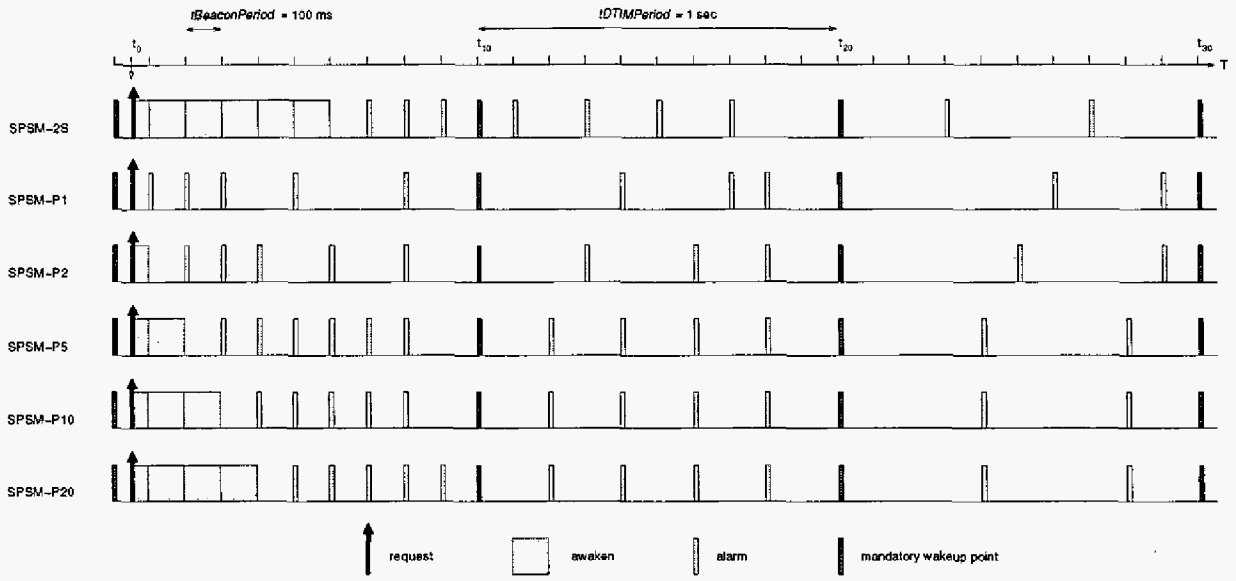


Fig. 4. Comparison of SPSM station action sequences with server RTT fixed at 10 ms

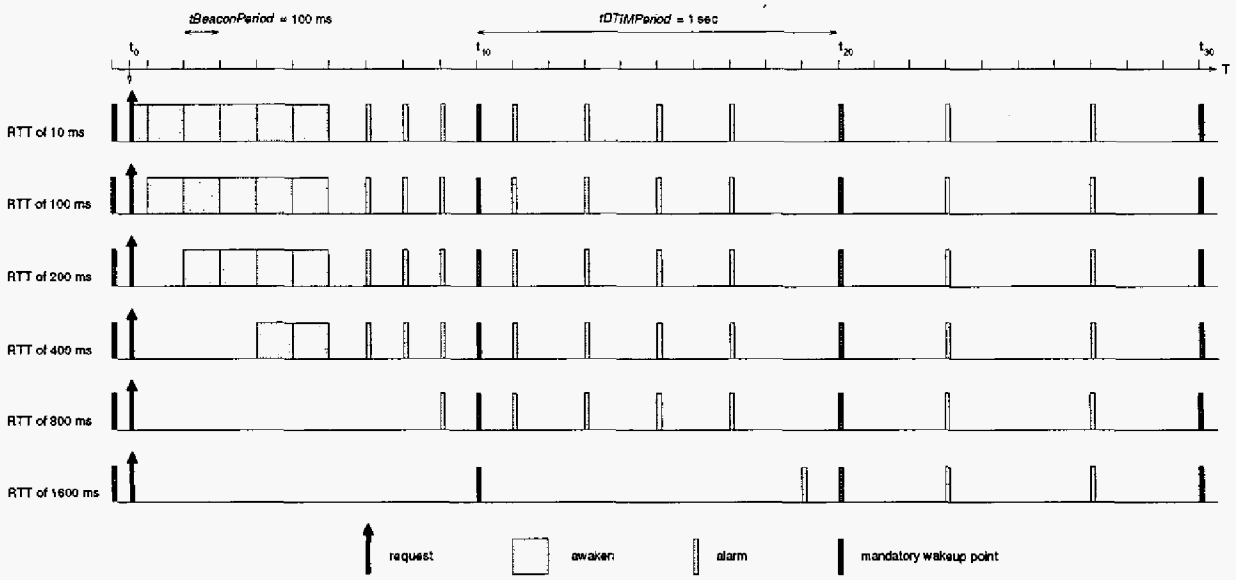


Fig. 5. Comparison of SPSM-2S station action sequences

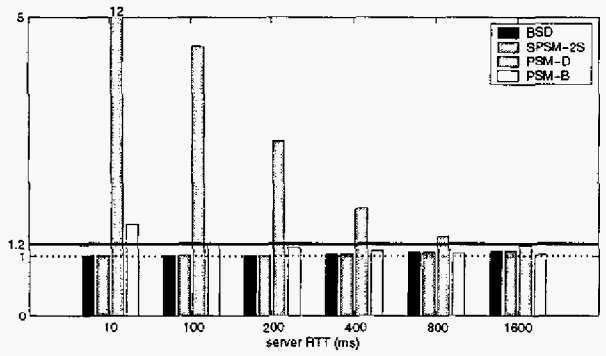
therefore, the resultant station action sequence appears more *like* that of SPSM-2S.

Fig. 5 compares the SPSM-2S station action sequences with various server RTTs. Apparently, SPSM is able to adapt to changes in the server RTT. For example, when the server RTT increases from 10 ms to 1600 ms, instead of staying awake for 550 ms after the request, the wireless station starts sleeping immediately and remains sleeping until the mandatory wakeup point, and thereafter, only wakes up occasionally. This is because SPSM is constructed in such a way that the response-time distribution is taken into consideration when determining the station action sequence. Therefore, as the server RTT changes, SPSM is able to rectify its station action sequence accordingly, and hence, it is a *smart* mechanism in contrast to the non-adaptive static-PSM and BSD protocols.

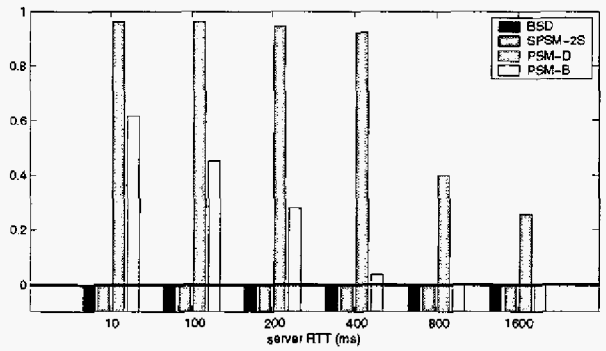
C. Simulation Results with Single Internet Content Server

In the first part of the simulation, we compare the testing schemes under a simple scenario where the wireless station communicates with a single Internet content server, i.e., the server RTT is not changing over time.

1) SPSM-2S vs. PSM-D, PSM-B, BSD: We first compare the delay performances of SPSM-2S, PSM-D, PSM-B, and BSD, and the simulation results of average response slowdown and delay-bound miss ratio are shown in Figs. 6(a) and (b), respectively. As expected, both SPSM-2S and BSD are able to guarantee the delay bound, while the static PSM schemes show worse delay performances. In fact, since a static PSM

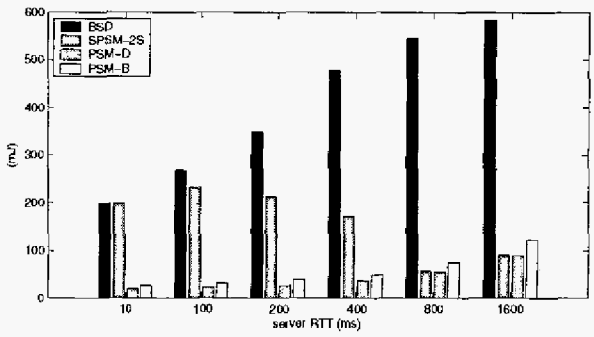


(a) average response slowdown

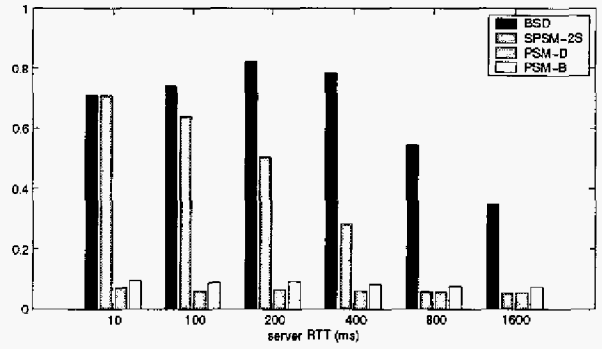


(b) delay-bound miss ratio

Fig. 6. Delay-performance comparison: SPSM-2S vs. PSM-D, PSM-B, and BSD



(a) per-request energy consumption



(b) normalized per-request energy consumption

Fig. 7. Energy-consumption comparison: SPSM-2S vs. PSM-D, PSM-B, and BSD

scheme uses a fixed wakeup interval, it could only provide the desired delay performance when the server RTT is larger than its wakeup interval over the slowdown factor. For example, in this simulation setup, PSM-B is able to provide the delay-bound guarantee when the server RTT is 800 ms or 1600 ms — evidenced by the corresponding zero delay-bound miss ratios — because both of them are larger than

$$\frac{tBeaconPeriod}{B} = \frac{100 \text{ ms}}{0.2} = 500 \text{ ms}.$$

On the other hand, PSM-D is not able to provide any delay-bound guarantee with all the simulated server RTTs because its wakeup interval (1 sec) is simply too large.

The energy-consumption performances of these four testing schemes are compared in Fig. 7, which plots the actual per-request energy consumption as well as its normalized value obtained by normalizing over that of the benchmark scenario. Recall that, in the benchmark scenario, the wireless station is always awake (no PSM) and is able to retrieve the response packet without any extra delay. It also consumes more energy than any PSM scheme. Using the normalized values, we can have a fair comparison of the energy-saving capabilities of the testing schemes.

We have the following three observations. First, PSM-D (PSM-B) is very energy-efficient because the wireless station goes to sleep immediately after the request and wakes up every *tDTIMPeriod* (*tBeaconPeriod*) to stay awake for a very short *tAlarmPeriod* (2 ms) for the TIM announcement. However, they are unable to provide the desired delay performance as we observed in Fig. 6.

Second, BSD consumes significantly more energy than the static PSM schemes because it requires a wireless station to stay awake for several Beacon intervals before sleeping. It is interesting to see that, with BSD, a wireless station consumes more energy as the server RTT increases (as shown in Fig. 7(a)), while its normalized value decreases (as shown in Fig. 7(b)). This may appear self-contradicting, but rather reasonable for the following reason. BSD determines the station action sequence regardless of the server RTT. As a result, the actually per-request energy consumption increases monotonically as the server RTT increases. But, at the same time, BSD allows a wireless station to keep increasing its wakeup interval while awaiting the response packet. Therefore, the energy saving compared with the benchmark scenario indeed gets larger. Similarly, since the station is required to stay awake or wake up more frequently during the early Beacon intervals, although it may take the station less time (and hence less energy) to fetch a packet from a server with a smaller RTT, its energy saving appears low (about 25%).

Third, SPSM-2S yields better energy-consumption performance than BSD, particularly when the server RTT is large. This is because, with SPSM-2S, a wireless station can adjust its station action sequence dynamically to the server RTT and,

hence, is able to avoid unnecessary energy wastes during the early Beacon intervals. As shown in the figure, when the server RTT is 800 ms or 1600 ms, SPSM-2S even shows comparable energy-consumption performance with static PSM schemes.

2) SPSM-2S vs. SPSM-Pz: We now compare the SPSM performances with various penalty functions (shown in Fig. 8) and the results are plotted in Fig. 9.

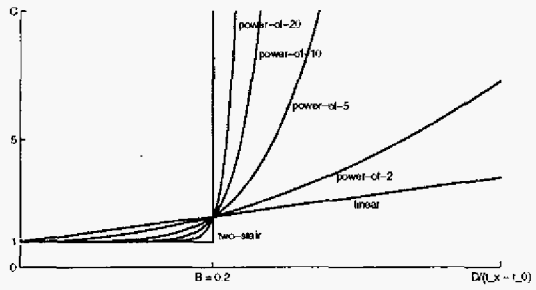


Fig. 8. Comparison of penalty functions

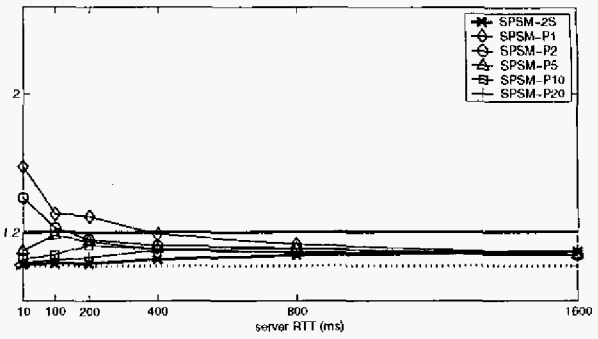
Clearly, due to different design philosophies of SPSM-2S and SPSM-Pz, which we have discussed in Section IV, only SPSM-2S (shown as $\times$ points in the figure) is able to bound the delay performance to a $1.2\times$ response slowdown, while all the SPSM-Pz schemes experience certain degrees of delay-bound violation. One interesting observation is that, with a high-enough-power penalty function, SPSM-Pz tends to yield a very low delay-bound miss ratio while saving considerably more energy than SPSM-2S. For example, on average, the delay-bound miss ratio of SPSM-P20 (shown as plus points in the figure) is less than 3% and its per-request energy consumption is about 26% lower than that of SPSM-2S in this simulation setup. In other words, a significant amount of energy can be saved at the expense of a slightly looser delay-bound guarantee.

We repeat the above simulation with other slowdown factors and get similar results, which are omitted due to space limitation.

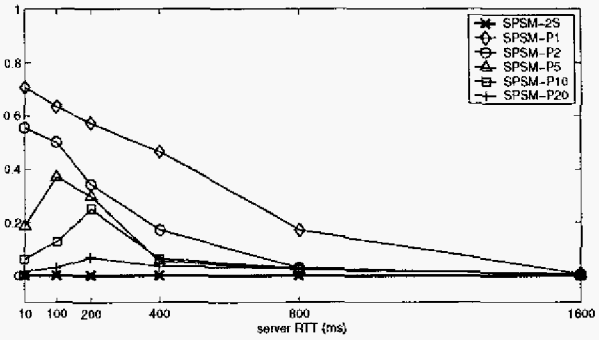
D. Simulation Results with Multiple Internet Content Servers

In the second part of the simulation, we consider a more realistic scenario where the wireless station communicates with multiple Internet content servers with different server RTTs. The 10,000 requests are divided into 200 groups, each with 50 consecutive requests, and the server RTT for each request group is selected arbitrarily from the set {10 ms, 100 ms, 200 ms, 400 ms, 800 ms, 1600 ms}. This is to emulate a typical user browsing pattern that the user stays with a website for a short period before switching to another. The simulation results (averaged over 25 different orderings of server RTTs) are listed in Table V.

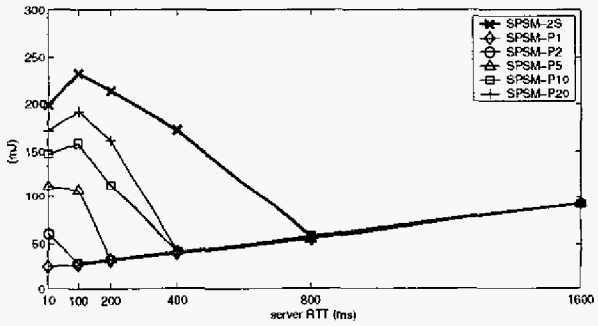
Similar to what we have observed under the single-server scenario, static PSM schemes consumes little energy while not being able to guarantee any delay bound. In contrast, BSD guarantees the desired delay performance at the expense of the highest energy consumption. For SPSM-2S, the results are



(a) average response slowdown



(b) delay-bound miss ratio



(c) per-request energy consumption

Fig. 9. Performance comparison of SPSM schemes

rather surprising: it yields a high energy consumption comparable to that of BSD, and some delay-bound miss events can be observed. Such a counterintuitive observation can be reasoned about as follows. First, SPSM is transparent to the higher-layer applications and is unaware of the user's browsing pattern, and hence the currently-selected station action sequence may not be optimal for a new packet request-response event. When the user switches to a new website with a smaller server RTT, it is likely that the first few response packets right after the switch may miss the delay bound. Second, SPSM adopts a simple exponential moving average algorithm to update the distribution of the response time. Combined with the two-stair penalty function that enforces no or extreme penalties, the wireless station tends to stick with a more conservative station action sequence even when the user starts browsing other

TABLE V

COMPARISON OF THE TESTING SCHEMES UNDER THE MULTIPLE-SERVER SCENARIO

Testing Scheme	Average Response Slowdown	Delay-Bound Miss Ratio	Per-Request Energy Consumption (mJ)
SPSM-2S	1.0431	0.0107	292.4311
SPSM-P1	1.2604	0.4267	48.9352
SPSM-P2	1.1792	0.2677	56.6781
SPSM-P5	1.1157	0.1565	80.5778
SPSM-P10	1.0796	0.0893	111.6051
SPSM-P20	1.0621	0.0297	131.1680
PSM-D	4.1058	0.7786	46.2409
PSM-B	1.4113	0.2776	63.2619
BSD	1.0399	0	403.7256

websites with larger server RTTs. On the other hand, SPSM-Pz is able to provide the user-desired *soft* delay-bound guarantee. For example, SPSM-P20 meets the user's low tolerance level on excessive delay while saving a significant amount of energy.

Finally, based on the above simulation results, we summarize in Table VI various user-desired delay performances and their corresponding selections of the energy-saving mechanism, which are consistent with the relation between delay performances and penalty functions we discussed in Section IV and listed in Table III.

TABLE VI

USER-DESIRED DELAY PERFORMANCES AND CORRESPONDING ENERGY-SAVING MECHANISMS

Acceptable Response Slowdown	Excessive-Delay Tolerance Level	Energy-Saving Mechanism
any	–	PSM-D
Θ	*zero*	SPSM-2S or BSD
	low	SPSM-P20
	medium	SPSM-P5/10
	high	SPSM-P2 or PSM-B

VI. CONCLUSIONS

By operating an IEEE 802.11-based wireless station in the power-saving mode (PSM), a user will inevitably experience a degraded delay performance during his/her online activity such as web-browsing. Static PSM schemes (with fixed wakeup intervals) in the current 802.11 implementations are very energy-efficient but cannot provide any delay-performance guarantee. In this paper, we propose a SPSM (Smart PSM) scheme, which provides (1) a generic method to interpret a user-desired delay performance using a penalty function, and (2) a generic algorithm that takes the penalty function as the input and generates automatically the optimal station action sequence that guarantees the user-specified delay performance while minimizing the energy consumption.

Our in-depth simulation shows that, with a two-stair penalty function, SPSM yields delay performance similar to BSD (Bounded SlowDown) under various scenarios, but with less energy consumption. Particularly, when the request-response turnaround time is large, SPSM even shows energy-consumption performance comparable to static PSM schemes. Moreover, SPSM is more flexible than BSD in the sense that it can meet arbitrary user-desired delay requirement, e.g., providing soft delay-bound guarantees with power penalty functions.

Our future work includes enhancing the estimation scheme for the response-time distribution and dealing with the challenging scenario when a wireless station *simultaneously* communicates with multiple Internet content servers with different server RTTs.

ACKNOWLEDGMENT

The work reported in this paper was supported in part by AFOSR under Grants F49620-00-1-0327 and F49620-01-1-0120.

REFERENCES

[1] IEEE 802.11, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Aug. 1999.

[2] A. Kamerman and L. Monteban, "WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band," *Bell Labs Technical Journal*, pp. 118–133, Summer 1997.

[3] P. J. Havinga and G. J. Smit, "Energy-Efficient TDMA Medium Access Control Protocol Scheduling," in *Proc. Asian International Mobile Computing Conference (AMOC 2000)*, Nov. 2000.

[4] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown," in *Proc. ACM MobiCom'02*, Atlanta, GA, Sep. 2002, pp. 119–130.

[5] T. Simunic, L. Benini, P. Glynn, and G. D. Micheli, "Dynamic Power Management for Portable Systems," in *Proc. ACM MobiCom'00*, Boston, MA, Aug. 2000, pp. 11–19.

[6] M. Stemm, P. Gauthier, D. Harada, and R. H. Katz, "Reducing Power Consumption of Network Interfaces in Hand-Held Devices," in *Proc. 3rd International Workshop on Mobile Multimedia Communications*, Princeton, NJ, Sep. 1996.

[7] M. Anand, E. B. Nightingale, and J. Flinn, "Self-Tuning Wireless Network Power Management," in *Proc. ACM MobiCom'03*, San Diego, CA, Sep. 2003, pp. 176–189.

[8] R. Kravets and P. Krishnan, "Application-Driven Power Management for Mobile Communication," *ACM Wireless Networks*, vol. 6, no. 4, pp. 263–277, 2000.

[9] D. Qiao, S. Choi, A. Jain, and K. G. Shin, "MiSer: An Optimal Low-Energy Transmission Strategy for IEEE 802.11a/h," in *Proc. ACM MobiCom'03*, San Diego, CA, Sep. 2003, pp. 161–175.

[10] S. Agarwal, S. V. Krishnamurthy, R. K. Katz, and S. K. Dao, "Distributed Power Control in Ad-Hoc Wireless Networks," in *Proc. IEEE PIMRC'01*, 2001, pp. 59–66.

[11] J.-P. Ebert and A. Wolisz, "Combined Tuning of RF Power and Medium Access Control for WLANs," *Mobile Networks & Applications*, vol. 5, no. 6, pp. 417–426, Sep. 2001.

[12] J. Gomez, A. Campbell, M. Naghshineh, and C. Bisdikian, "Conserving Transmission Power in Wireless Ad Hoc Networks," in *Proc. IEEE ICNP'01*, Nov. 2001, pp. 24–34.

[13] E.-S. Jung and N. H. Vaidya, "An Energy Efficient MAC Protocol for Wireless LANs," in *Proc. IEEE INFOCOM'02*, vol. 3, New York City, NY, Jun. 2002, pp. 1756–1764.

[14] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-Saving Protocols for IEEE 802.11-Based Multi-Hop Ad Hoc Networks," in *Proc. IEEE INFOCOM'02*, vol. 2, New York City, NY, Jun. 2002, pp. 200–209.

[15] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in *Proc. ACM MobiCom'01*, Rome, Italy, 2001, pp. 85–96.

[16] "The Network Simulator – ns-2," http://www.isi.edu/nsnam/ns/, online Link.

[17] *ORiNOCO 11b Client PC Card Data Sheet*, Proxim Corporation, 2003.