Energy-Efficient Wireless Information Retrieval[1]

Yu Jiao

Computational Sciences and Engineering Division

Oak Ridge National Laboratory

jiaoy@ornl.gov

Telephone: 1-865-574-0647

Fax: 1-865-576-0003

Ali R. Hurson

Department of Computer Science and Engineering

The Pennsylvania State University

hurson@cse.psu.edu

Telephone: 1-814-865-9505

Fax: 1-814-865-3176

[1] The Office of Naval Research and the National Science Foundation under contracts N00014-02-1-0282 and IIS-0324835 in part have supported this work.

Abstract

In this paper, we propose an adaptive application-driven power management (AADPM) protocol for wireless information retrieval applications within the IEEE 802.11b infrastructure WLAN environment. Our goal is to minimize energy consumption while achieving low round trip time delay. We discuss the protocol and evaluate its effectiveness using the network simulator NS2. We also draw horizontal comparisons among a variety of PM methods reported in the literature. Experimental results show that, compared to the power save mode supported by 802.11b, AADPM reduces the network interface card energy consumption by 52% while only introducing 3% RTT delay.

Keywords: IEEE 802.11b, power management, mobile computing, information retrieval

1. Introduction

The proliferation of Internet search engines and digital libraries has enabled effective data and knowledge exchange around the world, and they have become an indispensable part of our everyday life. The wide deployment of wireless networks and the more and more affordable portable devices give us the freedom to access the vast information reservoir from anywhere, at any time. However, battery-powered systems are often confronted with the problem of delivering high performance with a limited power supply. When using the wireless network, the network interface card (NIC) may consume up to 10% of the total energy of a high-end portable computer and nearly 50% of the overall energy of a low-end handheld devices [12]. Thus, NIC energy conservation is critical to prolong battery life.

In this paper, we use the IEEE 802.11 infrastructure wireless LAN as the target mobile environment. We propose an adaptive application-driven (AAD) power management (PM) protocol that has online learning capability. Our goal is to minimize the NIC energy consumption while achieving high throughput for information retrieval applications. We evaluate the effectiveness of the proposed scheme by draw-

ing horizontal comparisons among a variety of PM strategies reported in the literature. We also investigate the impact of different parameters on the performance of AADPM.

The rest of the paper is organized as follows: section 2 presents the background and related work. Section 3 analyzes the NIC characteristics. Section 4 discusses the adaptive application-driven power management architecture and the protocol. Section 5 evaluates the proposed strategy and compares it with other PM methods. Finally, section 6 concludes our work.

2. Background and Related Work

The target network is the IEEE 802.11 infrastructure wireless LAN (as opposed to the IEEE 802.11 ad hoc network) [7], in which all stations communicate with an access point (AP). When the IEEE 802.11b power save mode (PSM) is enabled, the AP broadcasts a *Beacon* every *BeaconPeriod* (typically 100 ms). During the BeaconPeriod, the AP buffers all data destined for the stations. Each Beacon contains a traffic indication map (TIM) that shows whether a station has data buffered at the AP. IEEE 802.11b compliant wireless NICs support two power states: *awake* and *doze*. The NIC is fully powered while awake. In the doze state, the NIC is not able to transmit or receive and consumes very low power. When PSM is disabled, the NIC is in the continuous awake mode (CAM) and the AP forwards data packets destined for the mobile device without delay. With the PSM enabled, a mobile station can switch the NIC to doze state when it has no data to send or receive in order to conserve energy. The station wakes up the NIC periodically to listen to the Beacon. If the TIM indicates that there are data buffered at the AP, the mobile station must poll the AP and receive all outstanding data. Typically, a station listens to every Beacon, but it can also be configured to skip Beacons. Since its listening interval is fixed, this power management scheme is often referred to as PSM-static in the literature. Note that a mobile station cannot switch between CAM and PSM at will. It must notify the AP of the mode change intention. According to the 802.11b specification, a mode change can be carried out only after a successful frame exchange between the mobile station and the AP. A frame exchange typically includes a RTS (request-to-send) packet, a CTS (clear-to-send) packet, a data packet, and an ACK. Thus, it is not guaranteed that a mode change

between CAM and PSM will always be successful, and the overhead introduced by such a transition depends on the network condition. It is reported that a mode transition may take as long as 600 ms [2].

Research has shown that, while PSM-static does quite well in saving energy in some cases, it is not flexible enough to balance the energy consumption and performance [2,11]. Many dynamic power management (DPM) strategies [2,4,6,11,12,15,16] have been developed to improve the performance of PSM-static. PM schemes are mainly proposed at three levels of a computer system: the hardware level (MAC or Link layer) [6,11], the operating system level [15], and the application level [2,4,12]. PM policies at each level often use one of the following techniques: fixed timeout, idle period prediction, or stochastic modeling.

An example of hardware-level timeout-based DPM is the Cisco Aironet 802.11a/b/g series of wireless LAN adapters. They provide a DPM option, which switches the NIC between the PSM and CAM depending on network traffic [6]. The DPM component switches the NIC to CAM when retrieving a large number of packets and switches it back to PSM after the retrieval. Researchers reported that the DPM switches the NIC to CAM when more than one packet is waiting at the AP, and it switches back to PSM after approximate 800 ms without receiving a packet [2]. We refer to this approach as the fixed-timeout DPM (FTDPM) in the rest of this paper.

Krashinsky and Balakrishnan proposed the Bounded-Slowdown (BSD) protocol [11], which operates completely at the link layer with no higher-layer knowledge. Under the assumption that NIC mode transitions between the CAM and PSM mode are always successful and introduce negligible overhead, the authors mathematically proved that the BSD protocol guarantees that the observed request round trip time (RTT) is less than $(1+p)$ times longer than the RTT measured in the absence of PSM, where p is the maximum factor of RTT delay ($p > 0$). Experimental results show that in many cases the BSD protocol can significantly reduce the energy consumption of web page retrieval while guaranteeing the performance bound.

Stochastic models are often used to obtain optimal power management algorithms. Simunic et al. proposed a system-level time-indexed semi-Markov decision process (TISMDP) model and showed a factor of three in power savings with a low performance penalty for WLAN Telnet applications [15]. Stochastic model-based DPM strategies share two common drawbacks: First, the optimal solution can only be achieved if the real application follows the distribution assumed by the stochastic model; second, the computational overhead is often large. For example, the TISMDP algorithm has a time complexity of $O(N)$, where N is the maximum time index. Each of the $O(N)$ states requires evaluations of one double and two single integrals.

Compared with low-level DPM strategies, application-driven DPM approaches have the advantage of knowing the application's intention of network usage. This knowledge enables more aggressive power saving methods without sacrificing the performance. However, unlike low-level DPM, there is no one-size-fits-all solution to application level DPM problems because applications may exhibit drastically different network traffic patterns. Many application-level DPMs have been developed to embrace the energy/performance tradeoff.

Self-tuning power management (STPM) [2] is a system-level strategy that utilizes application-level knowledge. It applies a concept similar to but more sophisticated than the hardware level DPM of the Cisco Aironet LAN adapter. If an application's delay tolerance is less than the maximum latency of PSM, or the number or size of the upcoming transfer(s) is large, STPM will put the NIC in CAM. When no transfers are in progress, no application has specified a tolerance of delay less than the PSM's maximum latency, and the estimated idle period is long, STPM will switch the NIC back to PSM. The estimations of upcoming transfers and idle periods are based on the history of a group of requests that are closely related in time (150 ms). Results show that STPM can reduce the energy usage of a distributed file system by 21% compared to PSM, while reducing the RTT delay by 80%. However, STPM is not suitable for request/response applications that involve long user-think time (e.g. web browsing) or applications that have long server response times (e.g. distributed information retrieval), because they will not

generate enough traffic for STPM to switch the NIC to CAM. As a result, STPM will behave the same as PSM.

The protocol proposed by Kraves and Krishnan is a general application-level DPM, in which the mobile station acts as the master and the AP is the slave [12]. The slave only sends out data when the master commands it to do so. Both the master and slave can queue up data during non-transmit phases in which the NIC is suspended to save energy. The protocol uses a fixed *timeout period* to determine whether or not to suspend the NIC and uses the *sleep duration* parameter to control how long the NIC should be suspended. Performance evaluation of Web, joint project, and Email applications shows a power saving of 83% for communication at the cost of a delay that ranges from 0.4 s to 3.1 s.

Observing the regularity of packet arrival rates in multimedia streaming applications, Chandra proposed a simple history-based client-side DPM policy [4]. It predicts the NIC *sleep interval* by averaging the idle periods observed in the past.

When an NIC is in the doze mode, it is shut off from the rest of the network. The effectiveness of the DPM strategies discussed in this section depends on how well they predict the ongoing activities in the network during their connectivity blackout. Wake-on-wireless works as a nice complement to the existing DPM methods by using a low-power network to signal a mobile station when packets are waiting at the AP [16]. However, this technology is not yet widely available.

The system model that we consider (as shown in Figure 1) consists of a mobile device, a service provider, and a wireless network. In this paper, we focus on request/response type applications that involve remote execution, e.g. distributed information retrieval.

As shown in Figure 2, a request/response cycle in an information retrieval system typically includes the following four periods: sending a request to the server, waiting for server response, receiving a reply from the server, and user-think time before sending the next request. We call the waiting-for-server and user-think time periods idle periods because the mobile station does not send or receive data during that time.

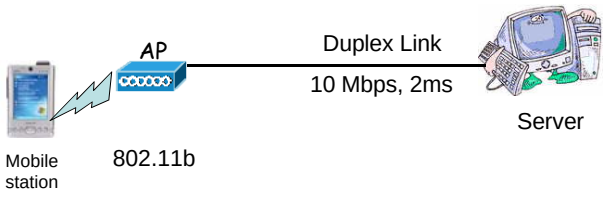


Figure 1. System model

As shown in Figure 2, a request/response cycle in an information retrieval system typically includes the following four periods: sending a request to the server, waiting for server response, receiving a reply from the server, and user-think time before sending the next request. We call the waiting-for-server and user-think time periods idle periods because the mobile station does not send or receive data during that time.

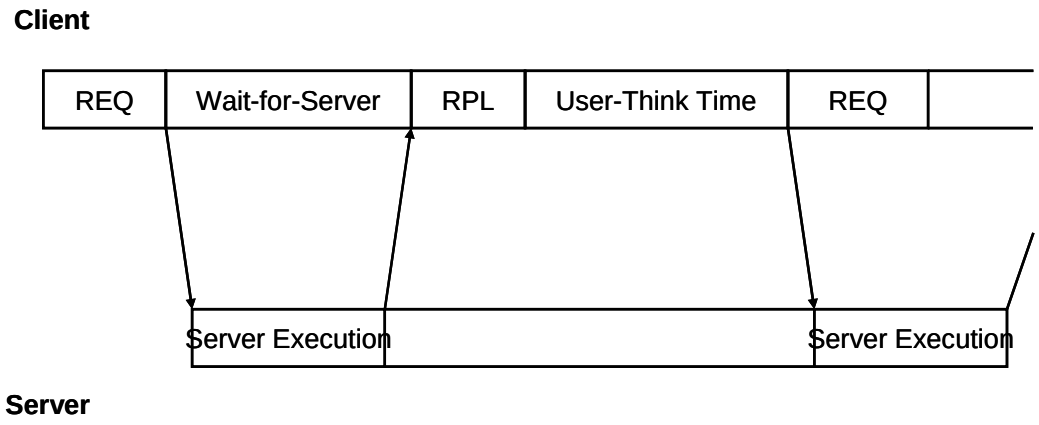


Figure 2. The request/response cycle

Our experimental results show that when no power management is employed, during a request/response cycle, more than 99% of the NIC energy is consumed during the idle period (Figure 3). Thus, in order to save energy, it is critical for a DPM method to identify the idle period and put the NIC to a low power state when appropriate. In general, DPM strategy developers are seeking answers to two fundamental questions:

- When does an idle period start?
- How long does the idle period last?

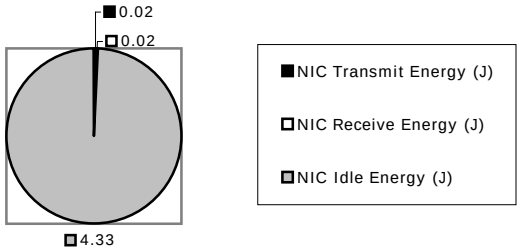


Figure 3. NIC energy consumption per request (NO_PM)

We propose an adaptive application-driven power management (AADPM) strategy with online idle period length distribution learning capability for information retrieval applications. AADPM differs from other DPM methods in several ways:

- Unlike the BSD protocol [11], it requires no changes to the existing IEEE 802.11b standard.
- It does not assume a particular request arrival pattern, nor does it make assumptions about the server response time. It learns the idle period length distribution online.
- It explicitly considers the power management mode transition cost when making transition decisions.
- It is designed to support NICs with multiple low-power states. AADPM automatically chooses the most suitable state according to the NIC's hardware characteristics.

3. NIC Characterization

IEEE 802.11b compliant NICs usually support two operational modes: awake and doze. When an NIC is transmitting, receiving, or idle, it is considered to be awake because it is ready to transmit or receive at anytime. When an NIC is in doze mode, it consumes very little power; however, it must transit to awake mode before sending or receiving data. Some NICs can even be suspended (consume no power) during long idle periods to further increase energy saving.

Mode transition from awake to doze typically consumes negligible time and energy, but the converse is not true. The lower power an operational mode consumes, the longer it takes to transit to the awake mode

(the wakeup delay or *Delay*). For example, Agere's WaveLAN card (Table 1) takes less than 0.75 ms to transit from doze to awake, but it takes about 600 ms if it is suspended [1]. A study shows that transition from a low-power mode to awake consumes the same amount of power as transmitting data [11] and therefore, switching the NIC to a low-power mode will save energy only if the energy saving in the low-power mode can amortize the energy required for the mode transition.

Table 1. WaveLAN NIC characteristics

Operational Mode	**Power (W)**	**Wakeup Delay (ms)**
Transmit	1.425	0
Receive	0.925	0
Idle	0.80	0
Doze	0.045	0.75
Suspended	0	600

Let $P_{transmit}$ and P_{low_power} denote the power usage of an NIC during transmission and when it is in a low-power mode, respectively. $E_{transition}$ ($E_{transition} = P_{transmit} \times Delay$) represents the energy consumption during the mode transition from a low-power state to awake. We define a break-even time t_{BE}, which is the minimum time that the NIC needs to stay in a low-power mode in order to achieve energy saving. If an NIC supports only one low-power state, the t_{BE} is calculated using equation (1).

$$t_{BE} = \frac{E_{transition}}{P_{transmit} - P_{low\_power}} \qquad (1)$$

When an NIC supports multiple low-power states and more than one of the states satisfy the equation (1), we choose the state that will minimize the value of E in equation (2), where i ($0 \le i < N$) is the low-power state, N is the total number of low-power states, $P_{low\_power\text{-}i}$ is the power consumed in state i, t_{idle} is the idle period length, and $Delay_i$ is the wakeup delay of state i.

$$E = P_{low-power-i} \times t_{idle} + P_{transmit} \times Delay_i \quad (2)$$

According to the hardware characteristics of the NIC card shown in Table 1, $t_{BE\text{-}doze}$ is 2 ms and $t_{BE\text{-}suspended}$ is 18.97 s. As a result, if $t_{idle} \le 2ms$, the NIC should remain in the idle state; if $2ms < t_{idle} \le 18.97s$, the NIC should use the doze mode; and if $t_{idle} > 18.97s$, the NIC should be suspended.

4. Adaptive Application-Driven Dynamic Power Management

4.1. Energy-Aware Applications

It is difficult for the operating system and hardware of a computer to know when an idle period starts, but the application can posses the precise knowledge if the application's server program provides an ACK upon receiving a request. We refer to such applications as energy-aware (EA) applications. An EA application notifies the start of the wait-for-server idle period to the PM component when the ACK of a request is received and it notifies the PM component of the beginning of a user-think-time idle period once a response is received. This answers the first fundamental question of DPM: when does an idle period start?

In order to answer the second fundamental question without *a priori* knowledge, we equipped the EA application with online idle period length probability distribution learning capability. More specifically, we use two histograms to record the lengths of server response delay and user-think time periods, respectively. The idle period length probability distributions can be calculated using these histograms, which in turn can generate probability-based idle period length predictions. In order to make the predictions, the AADPM component also needs the application to record the most recent power management mode change (between PSM and CAM) overhead, $t_{pm\_mode\_change\_overhead}$.

It should be emphasized that the NIC operational mode transition delay (or, wakeup delay), *Delay*, and the mobile station power save mode change overhead, $t_{pm\_mode\_change\_overhead}$, are two completely different concepts. The time required for an NIC to transition from a low power state to the fully powered state (active) is called the NIC wakeup delay (*Delay*). A mobile station has full control over its NIC's opera-

tional modes; however, it cannot perform the transition unilaterally. It must negotiate with the AP and reach an agreement.

By default, the AP assumes that the mobile station is always active (continuously awake mode, or CAM) and does not buffer data for it. When a mobile station decides to utilize the low power states of the NIC, it must notify the AP of its intention so that the AP will buffer data for it. This way, data packets will not be lost when the mobile station is in a low power mode. The IEEE 802.11b specification states that the transition between the CAM and PSM requires a successful frame exchange between the mobile station and the AP. We refer to this frame exchange overhead as the $t_{pm\_mode\_change\_overhead}$.

4.1.1. Building the histogram. The histogram building algorithm can be formally stated as follows: The set of all possible idle period lengths $(0, \infty)$ is partitioned into n intervals, where n is the number of bins in the histogram. Let l_i denote the starting point of interval i $(0 \le i < n)$. Bin i represents the idle period length range $[l_i, l_{i+1})$. If an idle period has a length greater than l_n, it belongs to bin $n-1$. Each bin has a counter C_i that indicates the number of idle period lengths that falls in the range of bin i.

A history window size ω is used to define the total size of the histogram – only the most recent ω idle periods are recorded ($\sum_{i=0}^{n-1} Ci = \omega$). A history linked list of size ω is needed to store the most recent ω idle period lengths in increasing order of time. When a new length is appended to the end of the linked list, the counter of the corresponding bin is incremented by 1. When the number of nodes in the linked list reaches ω, the oldest node is deleted from the head of the linked list and the counter in the corresponding bin is decremented by 1.

4.1.2. Calculate the probability distribution. $p(i)$ denotes the probability that the length of the next idle period will fall in the range of bin i. Equations (3) and (4) show the probability density function (PDF) and cumulative distribution function (CDF) of i, respectively.

$$p(i) = \frac{C_i}{\omega} \qquad (3)$$

$$P(i) = \frac{\sum_{j=0}^{i} C_j}{\omega} \qquad (4)$$

Figure 4 shows an example of the histogram, where $\omega = 10$, $n = 5$, and $l_{i+1} - l_i = 1$ second ($0 \le i < n$). Assume that the history of the previous 12 idle period lengths in increasing order of time is {1.5 s, 0.6 s, 0.8 s, 1.1 s, 0.5 s, 2.8 s, 1.3 s, 3.3 s, 5.6 s, 4.2 s, 1.5 s, 3.7 s}. On the top of Figure 4, the linked list records the last 10 history records. Based on the counter C_i, the PDF, p(i), and CDF, P(i), of each bin is shown in figure 4.

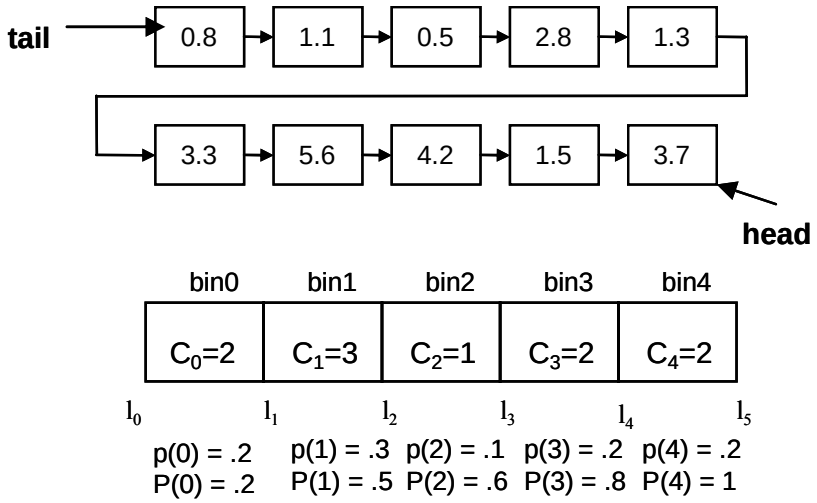


Figure 4. An example of the linked list and histogram

4.1.3. Predict the next idle period length. The predicted upcoming idle period length has direct impact on the energy/performance tradeoff: if the prediction is much shorter than the upcoming idle period, the energy saving is not maximized; if it is much longer than the upcoming idle period, the RTT increases and the performance degrades.

We use a user specified energy/performance tradeoff indicator *ep_ratio* as the guideline for choosing the proper idle period. The ep_ratio ranges from 0 to 1.0. It indicates the tradeoff between energy conservation and performance from the user point of view: 0 means maximum performance and 1.0 means maximum energy conservation. If no ep_ratio is specified, 0.5 will be used as default value, where equal

priority is given to energy conservation and performance. Figure 5 describes the algorithm for predicting the next idle period length.

1. Calculate $P(i)$ using equation (4).
2. Find the largest i, I, such that $P(I) \le ep\_ratio$.
3. The predicted next idle period length t_{idle} is calculated as

$$t_{idle} = l_I + (l_{I+1} - l_I)/2$$

Figure 5. The next idle period length prediction algorithm

Note that the ep_ratio is not a precise bound for RTT delay. It is used as a guide for the confidence level when a prediction is made. For instance, by setting the ep_ratio to 0.8, the user implies that 80% priority is given to energy saving, and 20% priority is given to performance. Thus, when making a prediction, the goal is to ensure that the probability that the next idle period is longer than the prediction is 0.2. In other words, the probability that this prediction will introduce no RTT delay is 0.2. For the example shown in Figure 4, when ep_ratio is 0.8, the predicted idle period length is 3.5 s.

4.1.4. Memory requirement analysis. Because applications may generate different network traffic patterns, we propose to use separate histograms for each application. Even for the same application, the server response time and user-think time are usually uncorrelated and exhibit very different characteristics [13] and therefore, we use separate histograms for the idle periods introduced by waiting-for-server and user-think time in order to improve the precision of predictions.

If the histogram is implemented using an array, the size (bytes) of a histogram with n bins and a history window size of ω is calculated by equation (5).

$$histogram\_size = \log\omega \times n/8 \qquad (5)$$

Assume that each node in the linked list is 12 bytes (one 8-byte double precision data entry and one 4-byte pointer). The size (bytes) of the linked list is given by equation (6).

$$linked\_list\_size = 12 \times \omega \qquad (6)$$

When n = 1024 and ω = 100, the size of a histogram and a linked list is 125 bytes and 1200 bytes, respectively. The memory required for each application is therefore about 1.5 KB. Contemporary handheld devices can easily meet this memory requirement.

4.1.5. Time complexity analysis. One of the important design criteria of online algorithms is the time efficiency. Complex algorithms may cause unacceptable performance and energy consumption overhead. One of the merits of our online learning method is its time efficiency. Building the histogram involves updating the linked list and the counter in a bin. The time complexity of insertion at the end and deletion from the head of a linked list is $O(1)$. Updating the histogram can be done in time $O(1)$. Calculating the CDF of i requires a linear traversal of the array and it is proportion to n (i.e., $O(n)$), where n is the number of bins. Predicting the next idle period length can be implemented using the binary search algorithm with the time complexity of $O(\log n)$. In summary, the time complexity of the online idle period length probability distribution learning and prediction algorithm (Figure 5) is $O(n)$.

4.2. AADPM Architecture and Protocol

In order to communicate with the application and gain control over the NIC driver, the ADDPM should be implemented as an operating system module. Figure 6 illustrates the AADPM architecture.

At the beginning of a wait-for-server idle period, the EA application discloses three hints to the AADPM: the beginning of the upcoming idle period, its estimated length (t_{idle}), and the last mode change overhead ($t_{pm\_mode\_change\_overhead}$). The AADPM obtains the break-even time t_{BE} and wakeup delay, *Delay*, of both doze and suspended states from the NIC. When making a mode change decision, the AADPM explicitly takes the mode change overhead into consideration. It switches the NIC from awake to the state that consumes the lowest power, in which the following holds: $(t_{idle} - t_{pm\_mode\_change\_overhead}) > t_{BE},$. If no such low-power state exists, the NIC stays awake (idle).

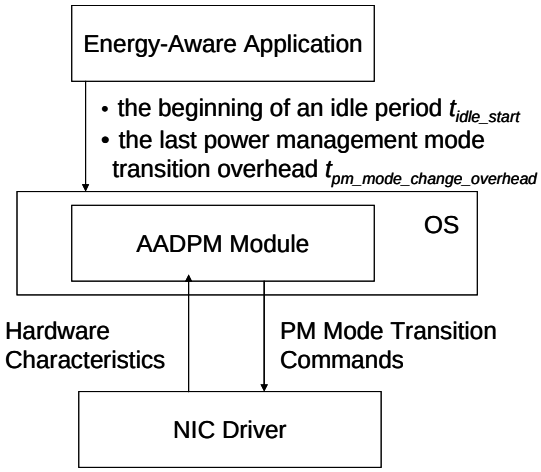


Figure 6. The AADPM architecture

If the NIC switches to the doze mode, the sleep duration is governed by t_{sleep} as in equation (8).

$$t_{beacon} = \left(\left\lfloor (t_{current} + t_{idle}) / t_{bp} \right\rfloor + 1\right) \times t_{bp} \qquad (7)$$

$$t_{sleep} = t_{beacon} - t_{current} - Delay \qquad (8)$$

where

$t_{current}$: the current time

t_{bp}: the beacon period

t_{beacon}: the time of the next beacon to listen

If the current idle period is the wait-for-server period, when the sleep duration t_{sleep} expires, the AADPM starts to act like PSM. It wakes the NIC up periodically to listen to the Beacon until pending data are detected. The AADPM then switches the NIC from power save mode to continuous awake mode and notifies the AP to stop buffering data. This way, we can minimize any further delay to the response packets.

After the EA application successfully receives a response, it informs the AADPM module of the start of a user-think-time idle period, predicted idle duration, and last mode change overhead. At this point, the NIC is in the continuous awake mode. Whether or not to transit to the power save mode and how long

the NIC should sleep are determined the same way as at the start of the wait-for-server idle period. If the NIC goes to the power save mode, upon termination of the sleep period, we can apply two schemes: (i) the NIC can be switched back to the continuous awake mode in order to minimize the wakeup delay for the next request (predictive wakeup), or (ii) let the NIC stay in the current low-power state until the arrival of the next request.

5. Evaluation

The effectiveness of AADPM in comparison to the standard IEEE 802.11b PSM, the FTDPM provided by Cisco Aironet NICs, and the bounded slowdown (BSD) protocol was simulated by using the network simulator NS2 [17].

5.1. Simulation Methodology

The NS2 version 2.26 that we used does not provide support for the IEEE 802.11b power save mode. Based on the 802.11b specification, we augmented this NS2 release with the following functions:

- Stations associate/ deassociate with the AP.
- The AP sends a Beacon every beacon period and the Beacon contains a TIM.
- The AP supports PSM by buffering packages for stations that are in doze mode.
- A station polls the AP if there are packets pending for it.
- The PM module is implemented as an OS component that can be accessed by the application and has control over the NIC operational modes.
- We implemented a library of PM strategies that include PSM, FTDPM, BSD, and AADPM.

We believe that our extension to the NS2 simulator and the PM library will significantly reduce the development and test cycles of future PM strategies.

The experiment setup includes a mobile station, an AP, and a server (Figure 1). The mobile station and the AP communicate using the 802.11 protocol. The AP and the server are connected through a duplex link that has a bandwidth of 10 Mbps and a latency of 2 ms.

The hardware characteristics of the Agere WaveLAN NIC are summarized in Table 1. We use these parameters throughout our simulation. Note that the energy consumed by the NIC is only part of the overall mobile device energy consumption. While PM strategies save energy for the NIC, they also inevitably increase the total execution time of requests. Thus, it is important to study the impact of a PM strategy on the overall mobile device energy consumption as well as its effect on the NIC energy consumption. It was measured in [2] that the base power of an HP iPAQ 3870 with 64 MB of DRAM and 32 MB of flash memory is 1.44 W. We used this parameter in our simulation.

An information retrieval application can be characterized by the following parameters: request size, server response time, server response size, and user-think time. In order to simulate more realistic scenarios and introduce variation to the request and response sizes, we used the request size model reported in [13], and the server response sizes are extracted from the UC-Berkley-home-IP trace [18]. Figure 7 shows the CDFs of the request size and response size.

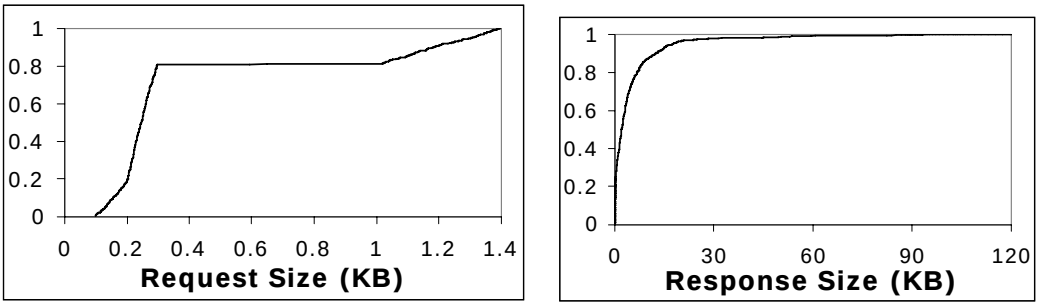


Figure 7. CDFs of the request size and response size

Our experience with Internet search engines and digital libraries has shown that although information servers may have different processing capabilities and the sizes of their databases vary, each search engine's response time is typically stable at a given workload. Based on the experimental results obtained from our mobile agent-based information retrieval prototype [9], we chose to model the server response time as a normal distribution with a mean value $\mu = 2.5$ s and a variance $\sigma = 0.2$.

The http traffic observed in the UC-Berkley-home-IP trace has a median user-think time of 15 s [13]. A recent Internet traffic characterization study has shown that the TCP flow arrival rate follows a Weibull regression model with the scale parameter ranging from 0.34 to 0.44 [14]. In our experiments, we used a three-parameter Weibull distribution, where the location, scale, and shape parameters are set to 15 s, 0.4, and 1.2, respectively.

During each simulation run, a mobile device executes 1,000 requests using a particular PM strategy or no PM at all (referred to as NO_PM in the following discussion). Three BSD variations with p = 1.0, 0.5, and 0.1 are simulated. For the AADPM strategy, we ran the simulation with different ep_ratios ranging from 0.1 to 1.0. By default, the AADPM does not use predictive wakeup. Unless otherwise noted, the AADPM response time and think time histograms both have histogram size n = 1024 and history window size ω = 20. Each bin in the response time and the think time histograms represents an interval of 10ms and 100 ms, respectively.

We use the average RTT delay, average NIC energy consumption, overall device energy consumption, and the AP buffer length as the performance metrics. In the following subsection, we compare the performance of different PM strategies and investigate the impact of histogram precision, history widow size, and predictive wakeup on the performance of AADPM. In each experiment, we first ran the simulator with NO_PM and recorded the RTTs. These RTTs are used as the basis for comparison because they represent the minimum RTT obtainable.

5.2. Performance Comparison of Different PM Strategies

Figure 8 compares the average RTT delay of different PM strategies. PSM incurs the longest RTT delay (37%) while FTDPM and BSD with p = 0.1 introduce the shortest RTT delay (2%). However, as we will see in Figure 10, these short delays are achieved at high costs of energy.

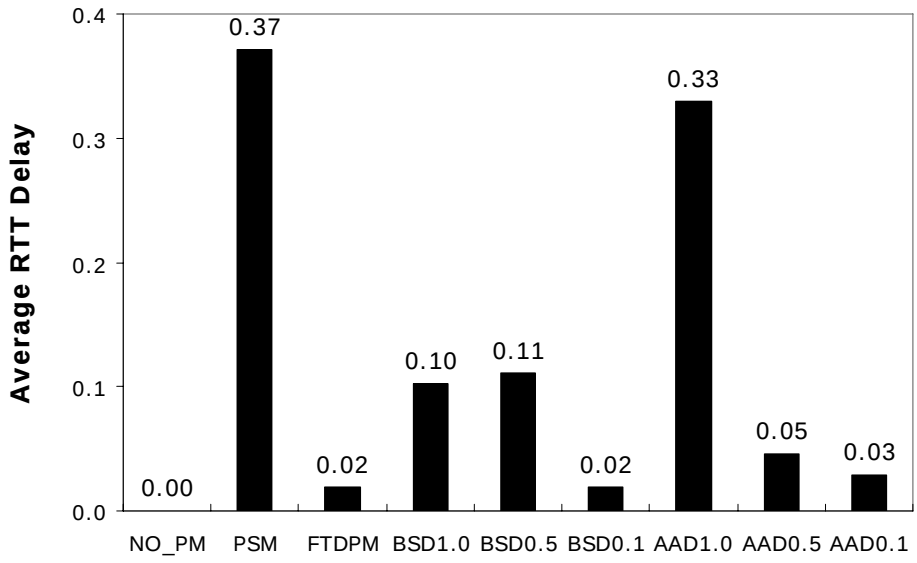


Figure 8. Average RTT delay per request

Intuitively, BSD0.5 has a lower performance bound than BSD1.0 and therefore, it has shorter sleep durations. As a result, we expect to see shorter RTT delays with BSD0.5. However, the results showed otherwise. A close examination of the experiment traces reveals that this is because the underlying transport protocol used in our experiments is TCP. In order to avoid TCP timeout, the longest BSD sleep duration is limited to 900 ms. In the experiments, the server response time follows a normal distribution with μ = 2.5 s. According to the BSD protocol, BSD1.0 and BSD0.5 can have maximum sleep duration (server response time × p) of 2.5 s and 1.25 s, respectively. However, due to the constraint of TCP timeout, both BSD1.0 and BSD0.5 will have the same maximum sleep duration – 900 ms. In addition, BSD0.5 causes more mode changes between the PSM and CAM than BSD1.0, and such mode changes introduce non-negligible overhead. As a result, BSD0.5 introduced slightly higher (less than 1%) average RTT delay than BSD1.0.

Smaller ep_ratio means that higher priority is given to the performance and therefore, we expect shorter RTT delays. The experimental results follow our expectation: when the ep_ratio is 1.0 (AAD1.0), the RTT delay of AADPM is 33%, and the delay is reduced to 3% when the ep_ratio is 0.1. It is interesting to notice that ADD0.5 and AAD0.1 achieved similar RTT delays and the delay is nearly 85% less than

AAD1.0. This is because the server response times fall into a small range of values. The predicted idle period durations are almost the same when ep_ratio is 0.1 and 0.5, and these predictions are more accurate than those made with ep_ratio set to 1.0.

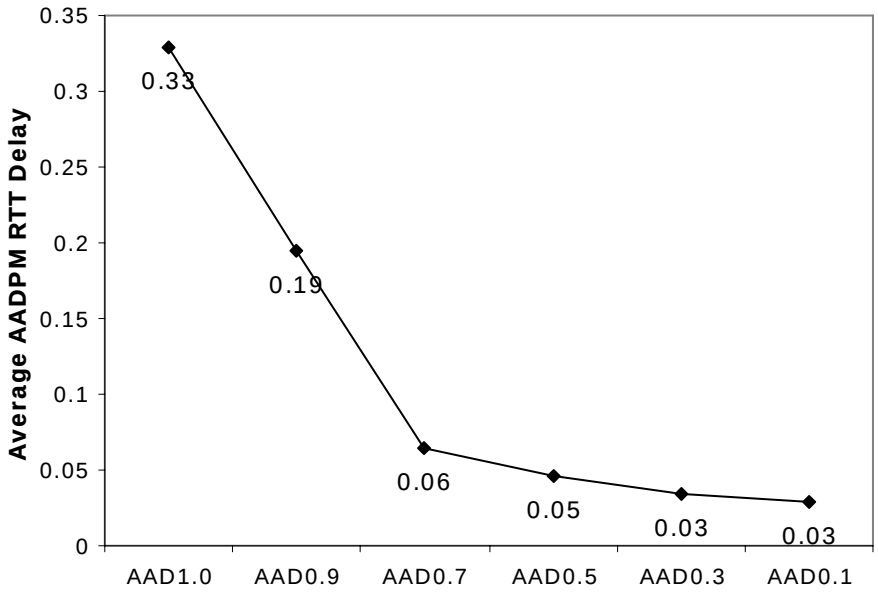


Figure 9. AADPM RTT delay with various ep_ratios

Figure 9 plots the average RTT delays of AADPM with various ep_ratio settings. These results suggest that the ep_ratio should not be statically chosen. If the response time distribution changes, the ratio should be adapted accordingly. One alternative is to use the user specified ep_ratio as the starting point. When it is time to make a prediction, multiple ep_ratios can be applied and the predicted values recorded. The AADPM module should choose to employ the ep_ratio that has generated the most accurate predictions based on its observation over a period of time.

An NIC consumes energy when transmitting, receiving, and being idle. Figure 10 presents a decomposed view of average NIC energy consumption per request. All the PM strategies consume similar amount of energy during transmission. PSM consumes the highest amount of energy for receiving packets because it listens to every Beacon.

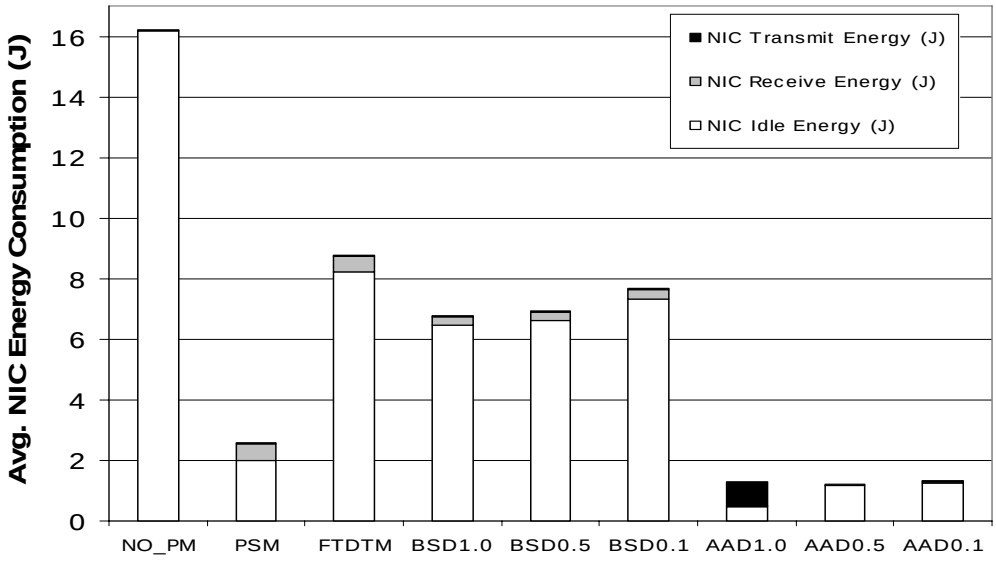


Figure 10. Average NIC energy consumption per request

The AADPM variations achieved the best energy saving among all PM strategies investigated. Compared with NO_PM, PSM, FTDPM, and BSD0.5, the proposed AADPM protocol with ep_ratio 0.5 (AAD0.5) reduced the NIC energy consumption by 92%, 52%, 86%, and 82%, respectively. Although FTDPM only introduced 2% RTT delay (Figure 8), its NIC energy consumption is also the highest among the PM strategies – only second to NO_PM.

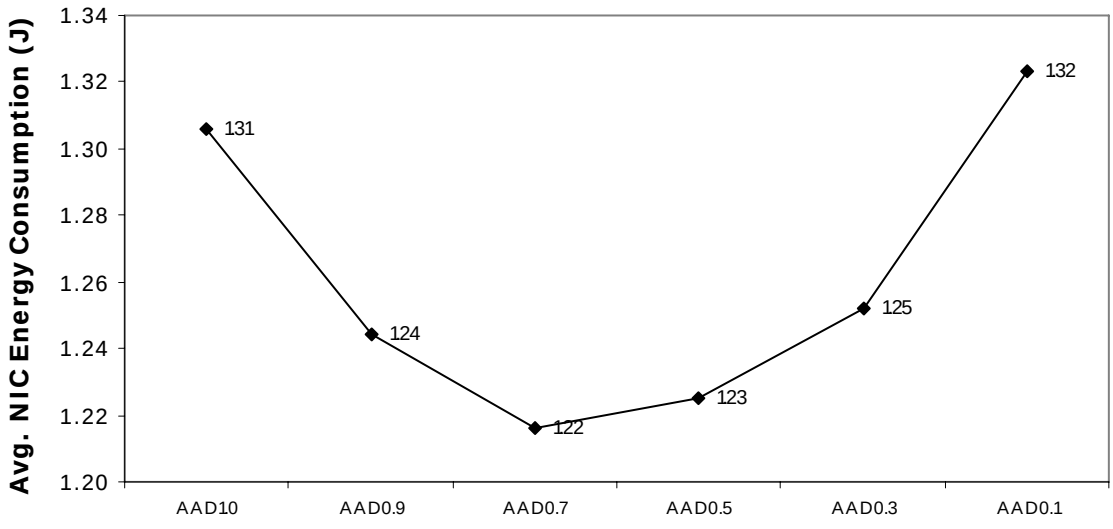


Figure 11. Average NIC energy consumption per request of AADPM with various ep_ratios

Figure 11 plots the average NIC energy consumption per request of AADPM with various ep_ratios. These results indicate that longer NIC sleep duration does not guarantee lower energy consumption – AAD1.0 has an average NIC energy consumption of 1.31 J while AAD0.7 consumed 1.22 J per request. This is because the performance of AADPM depends on the accuracy of the idle period prediction. If the prediction is much longer than the response time, the NIC wastes energy during the overslept period while it could have finished the query sooner. If the prediction is much shorter than the response time, however, the NIC consumes extra energy by staying idle. Since AAD0.7 generates the most accurate predictions, its NIC consumption is the lowest among the AADPM variations.

Next, we examine the impact of different PM methods on the overall mobile device energy consumption. Figure 12 shows that AADPM variations outperform all other PM strategies. More specifically, when compared with NO_PM, PSM, FTDPM, and BSD0.5, AAD0.5 reduces the overall energy consumption by 33%, 8%, 20%, and 21%, respectively.

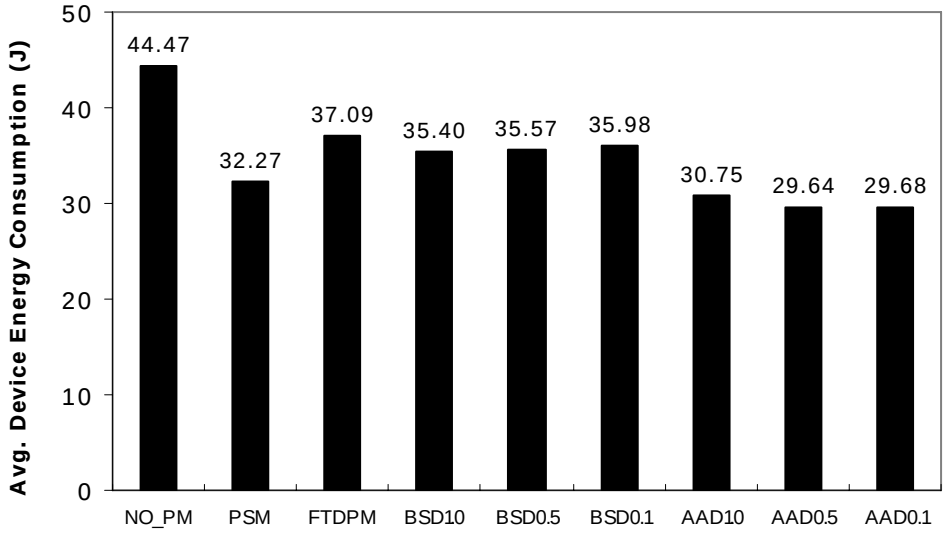


Figure 12. Average device energy consumption per request

One important effect of the PM strategy employment is the increased requirement on the AP buffer size. In reality, an AP has limited buffer space. Buffer overflow may cause packet losses, and consequently, it affects the RTT of requests. Unfortunately, this issue is often unexplored in PM studies.

Figure 13 compares the average and maximum AP buffer size of each PM strategy. It shows that all the methods investigated in our experiments require a maximum AP buffer size of 2 packets and an average buffer size less than 0.2 packets except PSM. At its peak, PSM requires the AP to buffer 20 packets and on the average it has 1.5 packets buffered.

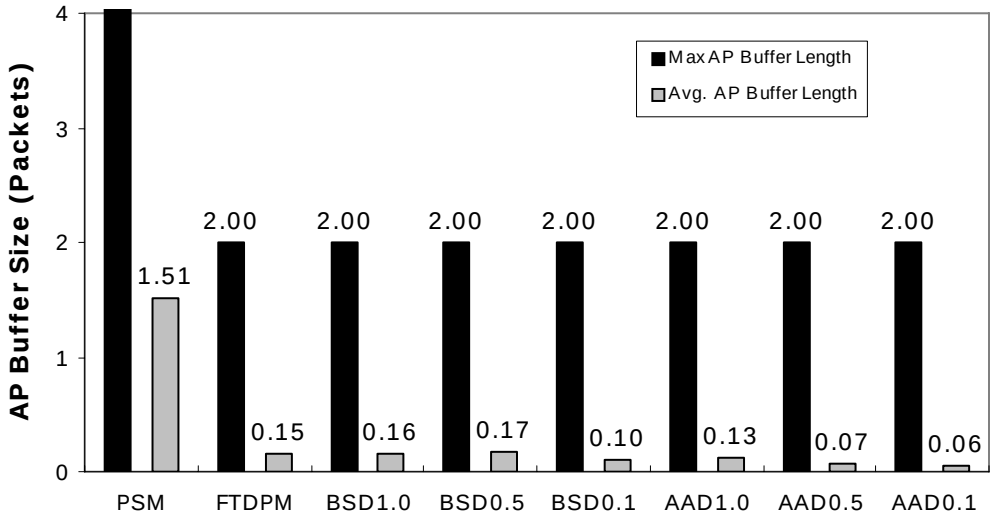


Figure 13. Average and maximum buffer size

5.3. Impact of AADPM Histogram Precision

All the experiments in subsection 5.2 used a server response time histogram with a precision of 10 ms and a user-think time histogram that has a resolution of 100 ms. Both histograms have the same length – 1024. We refer to this precision as the "normal precision" setting. In this subsection, we investigate the impact of the precision of the histogram on the overall AADPM performance. While keeping all other parameters the same as those in subsection 5.2, we increased the size of the histograms to 4096. We also changed the resolution of each bin in the think time histogram from 100 ms to 10 ms. In the following figures, it is referred to as the "high precision" setting.

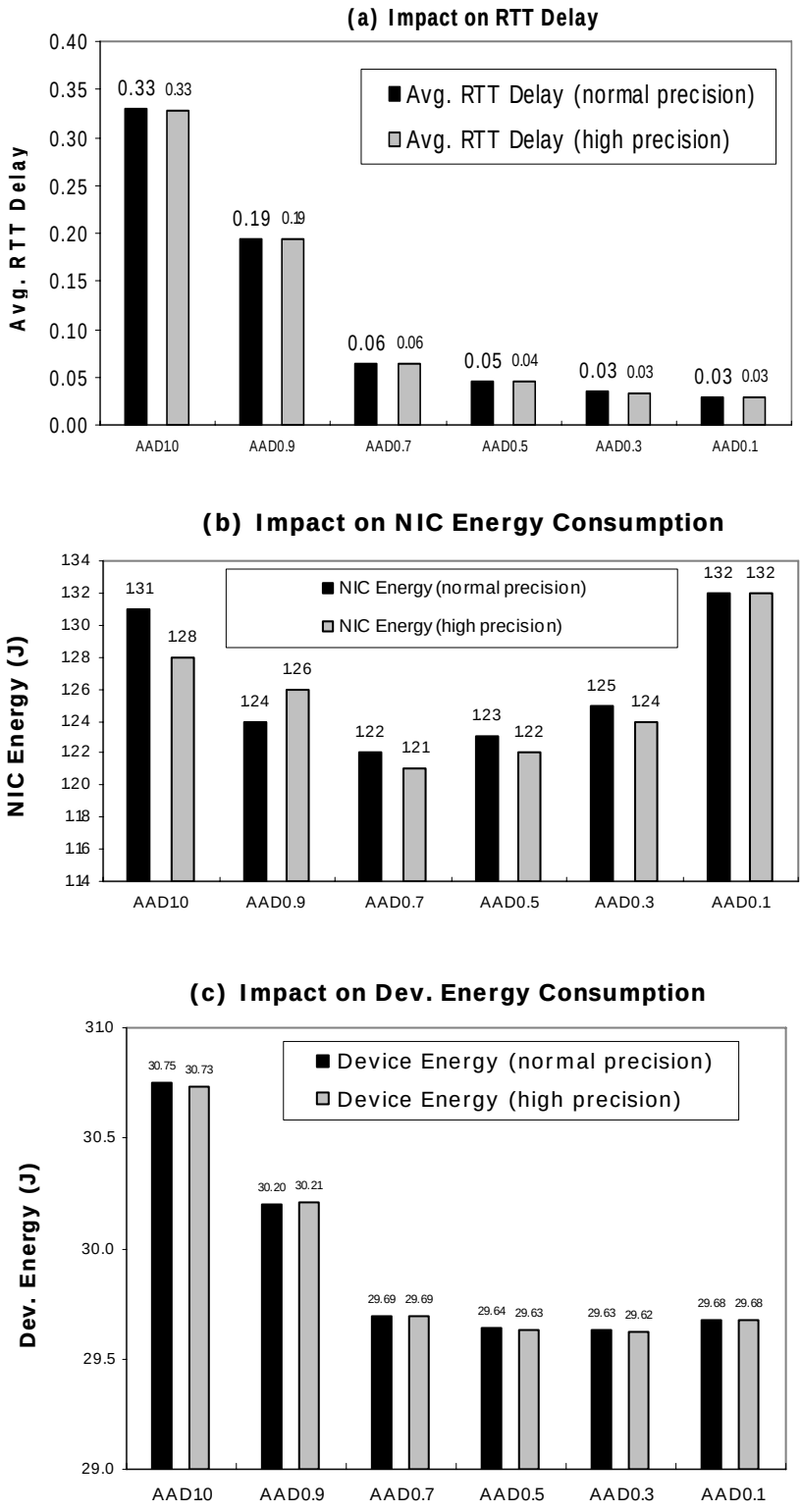


Figure 14. Impact of the precision of the histograms

Figure 14 shows the impact of the precision of the histograms on the RTT delay and the average NIC and device energy consumption per request. We can see that the performance gain from increasing the histogram size by a factor of 4 is minimal, if at all, with all ep_ratio settings. Thus, we conclude that the performance of AADPM cannot be improved significantly by increasing the histogram size.

5.4. Impact of AADPM History Window Size

In order to study the impact of the history window size (ω) on the AADPM performance, we vary ω from 10 to 100 while keeping all other parameters the same as those of subsection 5.2. Results indicate that AAD1.0 and AAD0.9 behave similarly, while AAD0.1, AAD0.3, AAD0.5, and AAD0.7 show strong resemblance. Thus, we only plot AAD0.9 and AAD0.3 in Figure 15.

As one can see from the above figures that there is no linear relationship between the history window size and the performance metrics – increased window size does not always result in the improvement of performance. The impact of ω also depends on the ep_ratio. When ω is increased from 10 to 20 (Figure 15a), the RTT delay of AAD0.3 is drastically reduced (87%), however, the delay of AAD0.9 is increased 5%. As we continue to raise the value of ω from 20 to 100, the RTT delay of AAD0.3 remains about the same, while AAD0.9 shows a gradual small increase in its RTT delay.

The variations of ω caused small fluctuations (< 0.5 J) in the average NIC (Figure 15b) as well as the overall device (Figure 15c) energy consumption per request for both AAD0.3 and AAD0.9.

We conclude that increasing the history window size alone cannot guarantee a better PM performance. Large history window size allows the PM component to observe longer history, while small window size reflects the situation changes more spontaneously. The value of ω should be chosen according to the nature of the application. Past traces can be used as training data to determine a suitable history window size for a particular application.

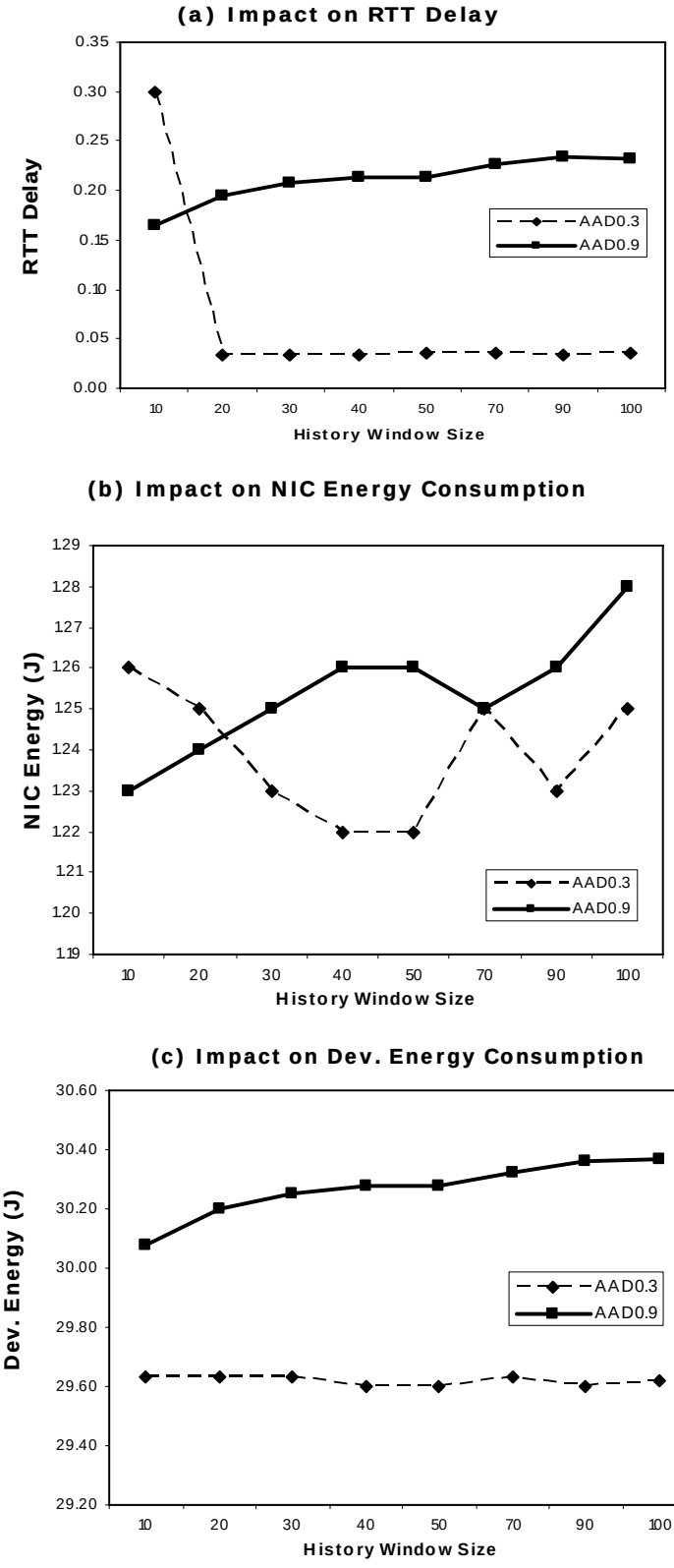
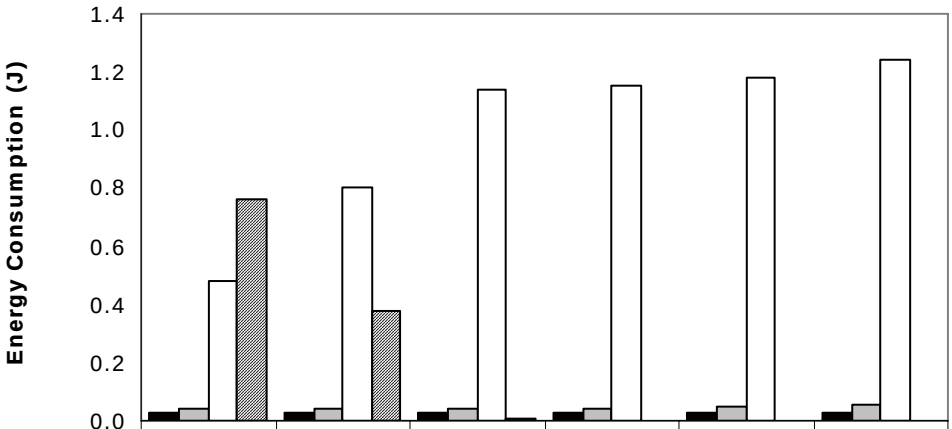


Figure 15. Impact of history window size

5.5. Impact of AADPM Predictive Wakeup

AADPM will suspend the NIC if it predicts that the upcoming idle period is longer than 18.97 s and by default, the NIC will remain suspended till the arrival of the next request. This means that once the NIC is suspended, it will be forced to wakeup when a request arrives and the request will encounter a delay of 600 ms. Figure 16 presents a decomposed view of the NIC energy consumption of AADPM with different ep_ratios.



	AAD1.0	AAD0.9	AAD0.7	AAD0.5	AAD0.3	AAD0.1
Transmit	0.03	0.03	0.03	0.03	0.03	0.03
Receive	0.04	0.04	0.04	0.04	0.05	0.05
Idle	0.48	0.80	1.14	1.16	1.18	1.24
Forced Wakeup	0.76	0.38	0.01	0.00	0.00	0.00

Figure 16. AADPM average NIC energy consumption per request

The forced wakeup energy in Figure 16 shows the average energy that the NIC consumed during the transition from being suspended to transmit. Results indicate that only when the ep_ratios are 1.0 and 0.9, are the suspended state of the NIC significantly utilized. This extra energy consumption also explains the noticeably longer RTT delays of AAD1.0 and AAD0.9 in Figure 9.

We can reduce both the RTT delay and the forced wakeup energy by predictive wakeup – during the user-think time, a suspended NIC will be reactivated (placed in CAM), once the predicted idle time has passed. We call the AADPM strategy with predictive wakeup AADPM+. Figure 17 compares the AAD1.0 with AAD1.0+ and AAD0.9 with AAD0.9+.

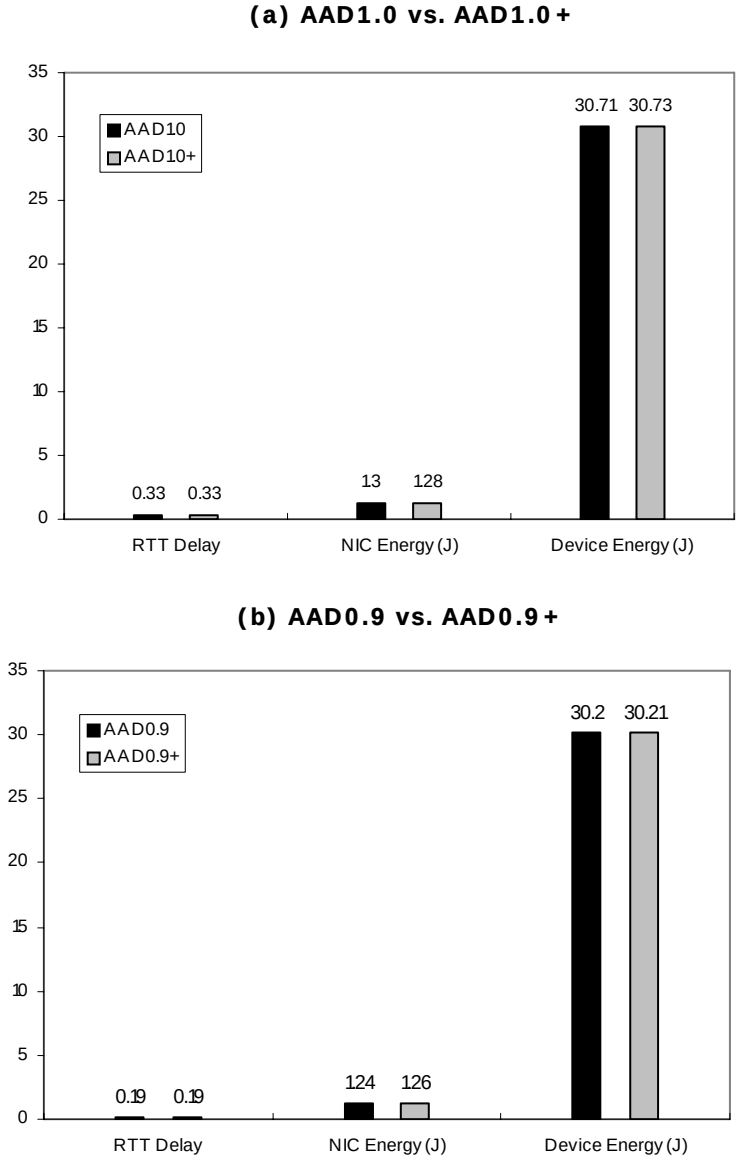
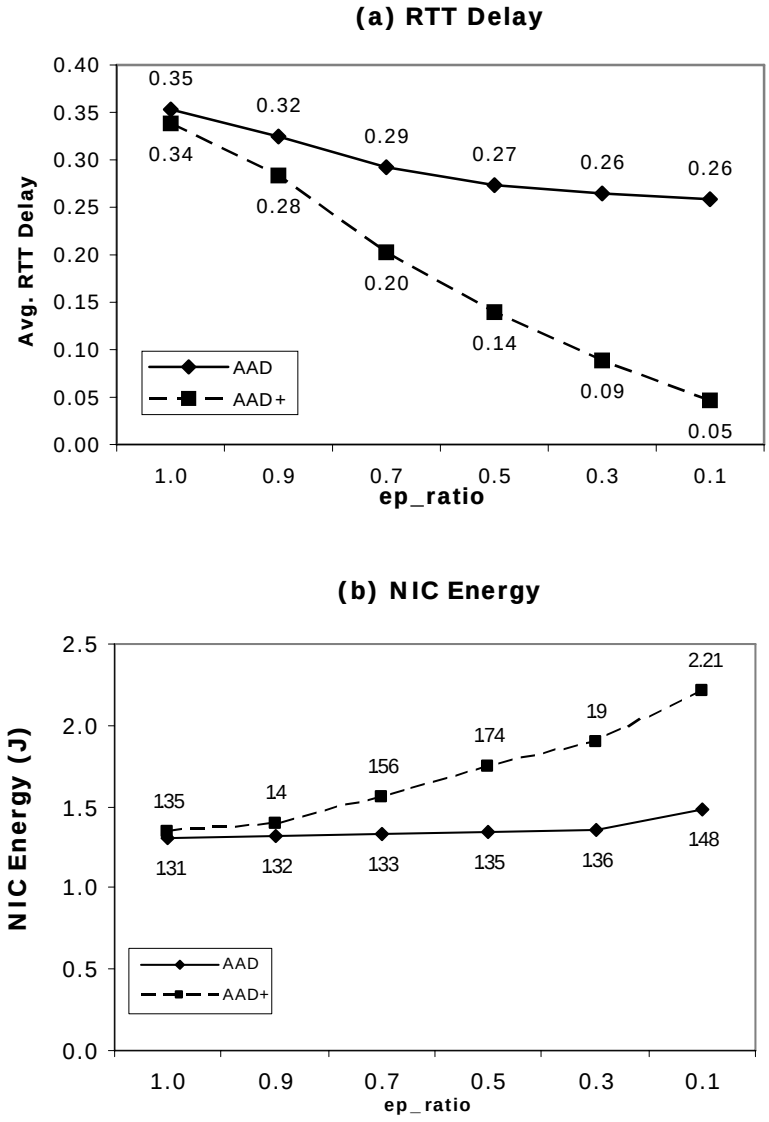


Figure 17. Impact of predictive wakeup

To our surprise, the predictive wakeup did not significantly improve the average RTT delay and NIC energy consumption, and it slightly worsens the over all device energy consumption. A close examination of the execution traces reveals that the reason is the user-think time distribution. Since we used a Weibull distribution with a location parameter of 15 s, only about 10% of the user-think times generated by the distribution are longer than 18.97 s. When the ep_ratios are 1.0 and 0.9, the AADPM component makes liberal predictions of the idle period that are often longer than the real values. Thus, the predictive

wakeup scheme does not have significant impact on the performance metrics. As the location parameter shifts to a higher value, however, the predictive wakeup method demonstrates improved performance, as shown below. We increased the location parameter from 15 s to 20 s and repeated the above experiments. Figure 18 plots the results.



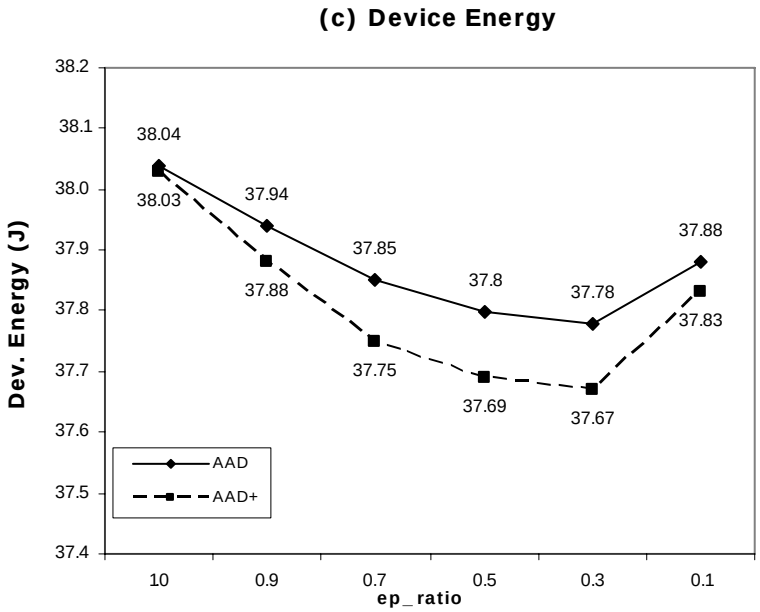


Figure 18. Impact of predictive wakeup with long user-think time

When the user-think time increases, the predictive wakeup method reduces the average RTT delay at the price of a higher NIC energy consumption. However, the average overall device energy consumption per request decreases under all ep_ratio configurations.

Each individual user may exhibit a different think time pattern, which can be used to determine whether or not to apply predictive wakeup. If the user profile indicates short think time (much shorter than the break-even time of the suspended state of the NIC), the AADPM predictive wakeup function should not be used. Otherwise, if a user tends to have long think times, predictive wakeup should be used to shorten RTT delays as well as to reduce the overall device energy consumption.

6. Conclusion

This paper analyzed the characteristics of distributed information retrieval applications and discussed the energy-efficiency issues of such applications in an IEEE 802.11b wireless LAN environment. We designed and evaluated an adaptive application-driven power management (AADPM) strategy, which can achieve high throughput while minimizing the NIC energy consumption.

We extended the network simulator NS2 to support the IEEE 802.11b power save mode and implemented a library of PM strategies that have been reported in the literature. We evaluated the proposed scheme through extensive simulation. We have also drawn horizontal comparisons among different PM methods in terms of the RTT delay, NIC energy consumption, overall mobile device energy consumption, and AP buffer size.

Our experimental results have shown that by knowing the application's intentions, AADPM achieved the best performance/ energy balance. Compared to NO_PM, AADPM (with ep_ratio 0.5) reduced the NIC energy consumption by 92% at the cost of a 5% RTT delay increase. When compared with PSM, AAD0.5 improved the RTT delay and energy saving by 87% and 52%, respectively. Results also indicate that low-level DPM strategies such as FTDPM and BSD can achieve good throughput by only conservatively putting the NIC to doze. However, the NIC energy consumption of these two dynamic methods is higher than the static PSM approach.

We also observed that if a PM strategy cannot balance the energy saving of the NIC and the RTT delay that it introduces, it may decrease the NIC energy consumption, but increase the overall mobile device energy consumption. We formally analyzed this problem in section 5.2 and an equation is given to determine whether or not a PM strategy is beneficial.

Histogram precisions, history window size, and the predictive wakeup technique have direct impact on the performance of the AADPM. Experiments show that simply increase the precision and the size of the history window size cannot guarantee shorter RTT delays and less energy consumption. Meanwhile, they increase the memory requirement. The predictive wakeup strategy is effective when the user-think time is often longer than the break-even time of the NIC's suspended state. It is recommended to take advantage of user profiling and use the past traces as the training data set in order to find out the optimal setting for these three parameters.

ADDPM is a powerful and easy-to-implement scheme. As part of our future research, we are working on implementing ADDPM on a personal digital assistant.

7. References

[1] Agere Systems Inc., *Wireless LAN PC Card (Extended)*, available from http://www.agere.com/client/docs/DS02115-1.pdf.

[2] M. Anand, E.B. Nightingale, and J. Flinn, "Self-Tuning Wireless Network Power Management", *Proc. of ACM MOBICOM*, San Diego, CA, September 2003, pp. 176-189.

[3] L. Benini, A. Bogliolo, and G.D. Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management", *IEEE Trans. on VLSI Systems*, 8(3), 2000, pp. 299-316.

[4] S. Chandra, "Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats", *Proc. of Multimedia Computing and Networking (MMCN)*, San Jose, CA, January 2002, pp. 85-99.

[5] J.-C. Chen, K.M. Sivalingam, and P. Agrawal, "Performance Comparison of Battery Power Consumption in Wireless Multiple Access Protocols", *ACM Wireless Networks*, 5(6), 1999, pp.445-460.

[6] Cisco Systems, Inc., *Cisco Aironet 802.11a/b/g Wireless LAN Client Adapters Installation and Configuration Guide*, 2001.

[7] IEEE Computer Society LAN MAN Standards Committee, *IEEE Std 802.11: Wireless LAN Medium Access Control and Physical Layer Specifications*, August 1999.

[8] S. Irani, S. Shukla, and R. Gupta, "Online Strategies for Dynamic Power Management in Systems with Multiple Power-Saving States", *ACM Trans. on Embedded Computing Systems*, 2(3), August 2003, pp. 325-346.

[9] Y. Jiao and A.R. Hurson, "Mobile Agents in Mobile Data access Systems", *Proc. of the 10th Intl. Conf. on Cooperative Information Systems (COOPIS)*, 2002.

[10] C.E. Jones, K.M. Sivalingam, P. Agrawal, and J.-C. Chen, "A Survey of Energy Efficient Network Protocols for Wireless Networks", *ACM Wireless Networks*, 7(4), July 2001, pp. 343-358.

[11] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown", *Proc. of ACM MOBICOM*, Atlanta, GA, July 2002, pp. 119-130.

[12] R. Kravets and P. Krishnan, “Application-Driven Power Management for Mobile Communication”, *ACM Wireless Networks*, 6(4), 2000, pp.263-277.

[13] B.A. Mah, “An Empirical Model of http Network Traffic”, *Proc. of IEEE INFOCOM*, Kobe, Japan, April 1997.

[14] X.Q. Meng, S. H. Y. Wong, Y. Yuan, and S.W. Lu, “Characterizing flows in large wireless data networks”, *Porc. Of ACM MOBICOM*, Philadelphia, PA, September 2004, pp. 174-186.

[15] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, “Event-Driven Power Management”, *IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, 20*(7), pp. 840-857.

[16] E. Shih, P. Bahl, and H. Balakrishnan, “Wake on Wireless: An Event –Driven Energy Saving Strategy for Battery Operated Devices”, *Proc. of ACM MOBICOM*, Atlanta, GA, July 2002.

[17] The VINT Project, *The NS Manual*, November 2001.

[18] UC Berkeley Home IP Web Traces, available from http://ita.ee.lbl.gov/html/contrib/UCB.home-IP-HTTP.html.