# Statistically Optimal Dynamic Power Management for Streaming Data

Nathaniel Pettis, *Student Member*, *IEEE*, Le Cai, *Student Member*, *IEEE*, and Yung-Hsiang Lu, *Member*, *IEEE*

**Abstract**—This paper presents a method that uses data buffers to create long periods of idleness to exploit power management. This method considers the power consumed by the buffers and assigns an energy penalty for buffer underflow. Our approach provides analytic formulas for calculating the optimal buffer sizes without subjective or heuristic decisions. We simulate four different hardware configurations with MPEG-1, MPEG-2, and MPEG-4 formats as a case study. Our results indicate that the optimal buffer size varies significantly for different data formats on different hardware. Simulation results indicate that 16 MB buffers are sufficient for MPEG-1, MPEG-2, and MPEG-4 video streams from a microdrive or a network card, but transfers from an IDE disk require buffer sizes ranging from 16 MB to 176 MB, depending on each video's statistical properties.

**Index Terms**—Dynamic power management, optimal buffering, quality-of-service, streaming media.

## 1 Introduction

Dynamic power management (DPM) [1] changes a component's power state when it is idle or underutilized. For example, when a hard disk is idle, the disk can be shut down ("turned off" or "asleep") to save power. The main challenge in DPM is to correctly predict when a component is idle because shutting down and restarting a component takes time and additional energy. Hence, a component should be shut down only if the overhead can be amortized across a long period of idleness with sufficient energy savings. Some methods use scheduling to explicitly create idle periods [2], [3]. Some other methods model the interactions between a service provider and a service requester as stochastic processes [4], [5], [6]. In these methods, a queue is inserted between the provider and the requester. The provider inserts data into the queue as needed by the requester, which processes the data. These methods shut down the provider when the queue is empty and is expected to remain empty for an extended period of time.

Some applications are better represented by a producer-consumer model, such as streaming video over the Internet or music playback from a local disk. In these systems, a producer generates data that is consumed at a regular interval. However, data sizes may vary between requests. The main difference between the producer-consumer model and the requester-provider model is that the data producer proactively buffers data for the consumer, whereas the provider is reactive and cannot produce data before the requester sends a request. This relationship allows the producer-consumer model to use prefetching, a common technique for performance improvement. Prefetching may also facilitate DPM. For example, a video player may prefetch many frames and store them in a buffer. The consumer retrieves the frames and displays them on the screen, subject to the timing constraints (frames/s). The producer may be shut down to save power when a sufficient number of frames are stored in the buffer. Before the buffer empties, the disk is turned on ("awakened") to fill the buffer. As multimedia devices become increasingly portable, dynamic power management techniques become more important.

More data should be stored in the buffer to amortize the overhead of turing on the producer [7]. However, memory consumes a nontrivial amount of energy in a computer system. When the power consumption of the buffer is considered, a large buffer may no longer be advantageous. Hence, we need to find an optimal buffer size that is large enough to amortize the overhead of DPM but small enough to actually save power.

This paper builds upon our previous work using buffers [8], [9], [10]. These works derive the optimal buffer size for constant production rates and uncertain consumption rates with a subjective penalty for quality-of-service (QoS) violations due to buffer underflow. This paper extends our previous work in three ways:

1. We remove subjectivity from optimal buffer sizes by restructuring the problem. The new method allows system designers to specify a quantitative and objective measure for QoS and solve for the optimal buffer size that satisfies the criterion.
2. The energy savings for the statistically optimal policy are determined by simulation for MPEG-1, MPEG-2, and MPEG-4 video trace playback from four different hardware configurations. We use disks and network cards as the examples of producers in this paper, but our method is applicable to any two interacting components with a buffer between them. We use MPEG videos as examples. Other types of

• *The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906.*
*E-mail: {npettis, lc, yunglu}@purdue.edu.*

*Manuscript received 13 Sept. 2004; revised 19 July 2005; accepted 7 Dec. 2005; published online 22 May 2006.*
*For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0300-0904.*

data streams, such as audio or other video formats, can also benefit from our approach. The results suggest that the optimal buffer size is not necessarily large and is available in desktop or handheld computers.

3. We analyze the sensitivity of the optimal buffer size and energy savings to changes in hardware and consumer parameters. We also predict the amounts of energy savings in future technology with lower awakening energy or memory power.

## 2 Background

### 2.1 Dynamic Power Management

DPM methods have been proposed to manage different types of components, such as processors [2], [3], [11], hard disks [12], [13], network cards [14], and sensor nodes [15], [16]. The basic concept of DPM is predicting when a component is idle or underutilized. Then, the component is shut down or its performance is scaled down. The main differences among existing methods are their approaches to predicting idle periods.

Shutdown algorithms predict when a component is idle and shut down the component [1], [17]. Krishnan et al. [18] compare the decision to spin down a hard disk with the decision to buy merchandise. Spinning the platters continuously consumes energy; this is analogous to paying rent regularly. The disk is shut down if it remains idle for an extended period of time. This is similar to buying the merchandise and keeping it for a long duration. Hwang and Wu [12] use exponential averages of recent idle period lengths to predict the length of future idle periods. If the predicted length is longer than the *break-even time* of the component being managed, the component is shut down. The break-even time of a component is the minimum length of an idle period required to save energy [1]. Ramanathan and Gupta [13] present an adaptive algorithm that improves prediction by assuming that the lengths of idle periods follow a probability distribution function. Benini et al. [4] use stationary Markov models to predict idleness. Request arrivals are modeled as stochastic processes; the method uses stochastic optimization to determine when to shut down a component. Simunic et al. [14] use time-indexed semi-Markov models to predict the idleness of wireless network cards. Chung et al. [5] present a strategy that handles idle periods with nonstationary distributions. The previous methods use discrete-time Markov models. The drawback is that management decisions occur only periodically. Qiu et al. use continuous-time Markov models [6] and generalized stochastic Petri nets [19] to make DPM decisions asynchronously and guarantee QoS. Simunic et al. [20] add time indices so that management decisions can be triggered by time events.

In many applications, devices do not remain idle long enough to shut down. Gurumurthi et al. [21] propose dynamically changing disk speeds to match workload. It is not always possible to match a disk's speed with a stream's bit rate because the rate varies from one video to another. Even for the same video, the rate is not a constant due to the variations of scenes and frame types. Research has also been conducted to extend idle periods and facilitate shutdown.

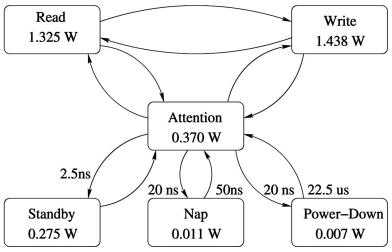


Fig. 1. Power model for 128-Mbit Rambus Direct RDRAM. Memory may be shut down in 16 MB blocks.

Bertozzi et al. [7] study a method with data buffering for a network card. All network packets are inserted into a buffer while the network card remains in a low-power state. When the buffer reaches its capacity, the network card is awakened and all data in the buffer is transmitted. When the buffer empties, the network card returns to a low-power state. Their results show the minimum buffer size as a function of the network bandwidth and indicate that buffers should be as large as possible. By using large buffers, the awakening overhead energy is amortized across a longer interval, increasing power savings. Ramachandran and Jacome [22] employ dedicated caches and application partitioning to produce idleness. Streaming data is prefetched into a dedicated cache; frequently used constants and variables are stored in a scratchpad memory to limit the number of external memory references.

### 2.2 Dynamic Memory Management

Dynamic Random-Access Memory (DRAM) is organized into sets of rows and columns called banks. Modern DRAMs provide low-power modes where each bank may be put into a different mode. The size of each bank and the number of low-power modes vary by technology. Bank sizes of 16 MB with at least one low-power mode are typical for both synchronous DRAM (SDRAM) and Rambus DRAM (RDRAM). A bank must be in the active state to provide data. Nonactive banks may be put into states where data is retained or where data is destroyed. If a bank is put into a low-power state where data is retained, the bank must merely be awakened to provide data. If the bank is put into a state where data is destroyed, then an access to that bank is handled like a page fault, where data must be recovered from a storage device. Subsequently, data should only be destroyed when the idleness exceeds the cost of storing and recovering data from storage. In general, the lower the power mode, the longer the time required to return to the active state where data requests may be served.

Fig. 1 shows the power states for a 128 megabit RDRAM [23]. Each state is represented by a vertex. The power consumed in each state is labeled within the vertex. State transitions are represented by arrows. The latency for each state transition is provided near the arrow. If a transition arrow does not have a latency, the transition is assumed to occur immediately. RDRAM has three active states: read, write, and attention. The device is active but not currently serving a request in the attention state. Upon receiving a read or a write request, the device transitions to the

appropriate state. The device supports three low-power states: standby, nap, and power-down. All of these states retain data but consume less power than the attention state. However, they all incur transition latency. The standby state has a 2.5 ns latency to *enter* the state, but responds immediately to requests. The nap and power-down states require 20 ns to enter their states and 50 ns and 22.5 $\mu$s to respond to requests, respectively. Many memory devices are configured to enter the standby state immediately after serving requests. When transitioning between states, the power consumption is equal to that of the attention state. Since most memory buses are synchronous, a significant delay may cause bus errors, requiring operating system attention to restart the operation [24]. These errors may occur whether memory power management is handled in software or in hardware. Hence, time and power overhead are associated with state transitions.

A significant amount of research has been dedicated to creating opportunities for memory power management. Lebeck et al. [25] model a power-aware RDRAM, based on the Direct RDRAM specification, where the current power state can be controlled by a power manager. They use this model to create power-aware page allocation schemes that adjust the memory power state based on the required amount of physical memory. Pisharath and Choudhary [26] reduce energy consumption within the memory hierarchy by using "Energy Saver Buffers." These buffers hold data waiting to be written back to the next highest level of memory until the memory is returned to an active state. Delaluz et al. [24] demonstrate that instructions for memory state transitions may be inserted at compile-time to help determine when to change power states. Their work also adopts an RDRAM model. Hu et al. [27] propose a model called *cache decay*, where cache lines are put into a sleep state after a preset timeout interval. Hu et al. note that cache lines are usually subject to bursty activity shortly after being loaded and remain inactive for an extended time before being overwritten by new data. Cache decay exploits this behavior by counting the cycles between accesses and powering down inactive cache lines. While data is obliterated by deactivating the individual cache lines, simulation results indicate that miss rates resulting from sleeping cache lines are acceptably low. They also suggest that cache decay is useful at all levels of the memory hierarchy, provided that timeouts increase to avoid high miss rates. Chen et al. [28] use garbage collectors within the Java Virtual Machine to reduce the amount of live memory within the heap. Ramachandran and Jacome [22] turn off banks of main memory while decoding MPEG-2 video to save power.

### 2.3 Quality-of-Service

Researchers have employed many techniques for handling quality-of-service (QoS). Qiu et al. [29] define a network-centric QoS in terms of delay, jitter, and loss rate. They provide techniques for optimizing power consumption by specifying a buffer size and requirements on any two of the three features. Chase et al. [30] adopt an economic model that accepts bids for resources, guaranteeing the best available resources to the highest bidder. Flinn and Satyanarayanan [31] define QoS by the term *fidelity*. The output quality is changed based upon the user's preferences and energy required to finish a task successfully. For example, a video may decrease in resolution or bitrate as batteries discharge, allowing the video to complete before losing power. We adopt a simple description of QoS by defining it in terms of data availability. That is, QoS is maintained if and only if the necessary data is immediately readable upon request, possibly in a buffer. If the data is located on a sleeping producer, the data cannot be immediately provided; thus, a QoS violation occurs.

### 2.4 Optimal Reordering Problem

The Optimal Reordering Problem is a well-studied problem in the field of operations research [32]. This problem involves choosing the number of products to order to minimize the expense due to storage space and shipping overhead. For example, consider a computer retailer where customers buy computers at a demand that varies from month to month. Reordering stock includes a fixed processing cost and a shipping cost proportional to the amount of stock. Storage space is required for all unsold stock and has a cost. Consumers who cannot receive a computer at the time of service become disappointed and may not return for future purchases.

Parallels exist between the Optimal Reordering Problem and DPM. Consider an application to be the customer trying to collect data (computers) from a buffer (store). The application consumes data at a variable rate dependent upon scheduling and file size. Awakening a storage device (factory) entails fixed energy overhead and static power while data is buffered. The buffered data requires memory space and its associated static power. If an application cannot retrieve data from a buffer due to underflow, execution is delayed until the storage device awakens to provide more data. Since the application may have performance guarantees, a penalty must be assigned to indicate unsatisfactory performance. This is similar to the customers' dissatisfaction described in the previous example. Therefore, a strong relationship exists between the Optimal Reordering Problem and DPM.

### 2.5 Overview of MPEG

Our study focuses upon MPEG video because of MPEG's popularity, but our method can easily be extended to other streaming media formats. The Moving Pictures Experts Group (MPEG) encoding defines compression techniques for video and audio media [33]. A unit of a video is the frame. Each frame is divided into a grid of macroblocks. MPEG videos consist of three types of frames: intrapicture frames, predicted frames, and bidirectional frames. These types are referred to as I-frames, P-frames, and B-frames, respectively. I-frames represent complete pictures and serve as a reference point for the other two frame types. P-frames predict the motion of pixels by vectors representing the future position of macroblocks that change from the previous I or P-frame. Since they consist of less information, P-frames are smaller in data size than I-frames. B-frames contain motion prediction between the previous and the next I or P-frame. While bidirectional prediction increases the computational complexity of decoding, its use increases signal-to-noise ratio and further decreases data size. An

TABLE 1
Mathematical Symbols Used throughout Derivation

| Symbol | Section | Definition | Unit |
|---|---|---|---|
| $\alpha$ | 3.1 | Production data rate | MB/s |
| $\beta$ | 3.1 | Consumption data rate | MB/s |
| $S$ | 3.1 | Prefetch amount | MB |
| $Q$ | 3.1 | Buffer size | MB |
| $t$ | 3.2 | Period length | s |
| $p_b$ | 3.2 | Buffer access power | W |
| $p_{b,mb}$ | 3.2 | Buffer retention power | W/MB |
| $p_p$ | 3.2 | Producer power | W |
| $k$ | 3.2 | Producer awakening energy | J |
| $w$ | 3.3 | Awakening point | MB |
| $\gamma$ | 3.3 | Expected consumption rate | MB/s |
| $\lambda$ | 3.3 | Producer awakening delay | s |
| $\psi(\beta)$ | 3.3 | Probability density for $\beta$ | |
| $\Psi(\beta)$ | 3.3 | Cumulative distribution for $\beta$ | |
| $\rho$ | 3.4 | Underflow energy penalty | J/MB |
| $f(w)$ | 3.4 | Average energy penalty | J |
| $\zeta$ | 3.5 | Probability of underflow | |
| $T_{be}$ | 4.1 | Break-even time of producer | s |

I-frame and the set of frames before the next I-frame are defined as a group-of-pictures (GoP). The number of frames in a GoP is usually constant in a video.

### 2.6 Contributions

1. An important contribution of our work is the ability to analytically determine the necessary changes in buffer size as the work environment changes. Existing studies provide only heuristic rules for estimating buffer sizes [7], [11], [34] or require extensive simulations to determine the sizes [22].
2. Our work considers the power of the buffer and computes the buffer size for optimal power savings. Very little of the existing work considers the power of the buffer. Bertozzi et al. [7] propose a heuristic buffering technique where the buffer size is as large as reasonably possible and quality-of-service is evaluated by the probability of underflow. When the power of the buffer is considered, large buffers may not be optimal.
3. We evaluate the effectiveness of our DPM policy by simulation with MPEG-1, MPEG-2, and MPEG-4 video traces on four hardware configurations. Bertozzi et al.'s technique is used to buffer data for transmission over a wireless network. Since their application exhibits similar locality to streaming media, we use their heuristic methodology for comparison. We also compare our DPM policy against other heuristic buffer sizes to demonstrate the ability of our technique to handle variations in workload and hardware gracefully.
4. We evaluate the sensitivity of optimal buffering to variation in several hardware parameters and comment on the effectiveness of buffering techniques as more advanced hardware becomes available.

## 3 Prefetching for Power Management

The following section develops analytical expressions for optimal buffer sizes with DPM. Table 1 has been included as a reference for the symbols used throughout the derivations. The first column denotes the symbol. The second column contains the section where the symbol is first defined. The third column holds the definition for the symbol within the context of the derivation. The last column states the physical units for the symbol.

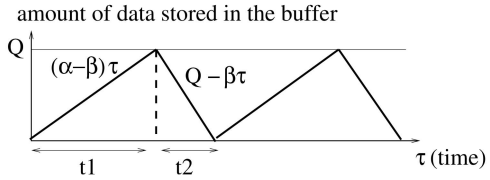


Fig. 2. The amount of data stored in the buffer for deterministic production and consumption rates.

### 3.1 Problem Description

Fig. 2 illustrates the basic problem for power management using a data buffer. The producer stores data into the buffer at rate $\alpha$ (MB/s) and the consumer removes data from the buffer at rate $\beta$ (MB/s). During $t_1$, the amount of data stored in the buffer grows at rate $\alpha - \beta$. At the end of $t_1$, the producer is shut down. In total, $\alpha t_1$ MB are produced; let $S$ be this amount. Let $Q$ be the maximum amount of data stored in the buffer. The consumer continues removing data, so the amount of data in the buffer declines at rate $\beta$. It takes $t_2$ to remove all the data in the buffer. The producer is turned on at the end of $t_2$ to restart data production. Our goal is to find the values of $S$, $Q$, $t_1$, and $t_2$ to achieve the maximum power savings of the producer.

**Assumption 1 ($\alpha$ is a constant).** *The production rate is dependent upon the bandwidth of the producer. This assumption is reasonable for many applications, such as a CD-ROM. In this case, the production rate is limited by the data bus and the CD-ROM's bandwidth. While some conditions may lead to slight fluctuations in $\alpha$ (i.e., rereading a sector after detecting an error), we use the average production rate for our derivations. The effect of $\alpha$ on buffer parameters and energy savings is covered in greater detail in Section 5.*

**Assumption 2 ($\beta$ is a constant).** *The consumption rate is dependent upon the application that requires the streaming data. This is a simplifying assumption when calculating buffer sizes for deterministic consumption rates and will be relaxed in Section 3.3.*

**Assumption 3 ($\alpha > \beta$).** *If the consumption rate $\beta$ exceeds the production rate $\alpha$, the buffer never fills.*

There are three components consuming power: the producer, the buffer, and the consumer. Since the consumer continues removing the data from the buffer, its power consumption is not affected by DPM. Our analysis does not consider the consumer's power. Two additional assumptions are used to calculate average power consumption.

**Assumption 4 (Buffer retention power is proportional to stored data).** *The buffer's power consists of two components: a static component representing access power and a dynamic component representing the amount of data within the buffer. Modern memory may be shut down in banks (as described in*

*Section 2.2). For derivation, we assume that the retention power consumption is proportional to the amount of data stored in the buffer with single-byte granularity. We discuss the impact of this simplifying assumption in Section 3.7.*

**Assumption 5 (Producer can be turned on with no delay).** *The producer incurs no delay when changing power states. This is a simplifying assumption for calculating buffer sizes for deterministic consumption rates and will be relaxed in Section 3.3.*

### 3.2 Buffer Size and Period Length

Suppose the producer's average power is $p_p$ during $t_1$. The producer consumes no power during $t_2$ because it is turned off. The power consumed by the buffer is composed of two parts: a component $p_b$ representing access power (W) and a component $p_{b,mb}$ for the power required to retain a unit size of data (W/MB). The energy overhead to turn on and off the producer is $k$. Let $T = t_1 + t_2$; this is the length of a period. Since the required buffer size is $Q$, the power consumed by the filled buffer is $p_b + Qp_{b,mb}$. From Fig. 2, we can see that $Q = (\alpha - \beta)t_1 = \beta t_2$ and obtain the relationships $t_1 = \frac{\beta}{\alpha}T$, $t_2 = \frac{\alpha-\beta}{\alpha}T$, and $Q = \frac{\alpha-\beta}{\alpha}\beta T$. The energy consumed by the producer and the buffer in each period is $p_p t_1 + p_b T + p_{b,mb} QT + k$. The average power is

$$\frac{p_p t_1 + p_b T + p_{b,mb} QT + k}{T} = \frac{p_p \frac{\beta}{\alpha} T + p_b T + p_{b,mb} \frac{\alpha-\beta}{\alpha}\beta T^2 + k}{T} = p_p \frac{\beta}{\alpha} + p_b + p_{b,mb} \frac{\alpha-\beta}{\alpha}\beta T + \frac{k}{T}.$$

By taking the first derivative with respect to $T$, we can find the condition for the minimum average power:

$$T = \sqrt{\frac{k\alpha}{p_{b,mb}\beta(\alpha-\beta)}}, \quad (1)$$

$$Q = \sqrt{\frac{k\beta(\alpha-\beta)}{p_{b,mb}\alpha}}. \quad (2)$$

The formulas indicate that when the DPM overhead ($k$) is large, the length of the period should be longer ($\propto \sqrt{k}$). The buffer size ($Q$) is also larger. On the other hand, when the buffer power per unit data ($p_{b,mb}$) is large, the buffer size should be smaller ($\propto \frac{1}{\sqrt{p_{b,mb}}}$). The period and the buffer size are also affected by the producing and consuming rates ($\alpha$ and $\beta$). The memory's access power $p_b$ has no effect on the optimal buffer size. The amount of prefetched data is

$$S = \alpha t_1 = \sqrt{\frac{k\beta\alpha}{p_{b,mb}(\alpha-\beta)}}. \quad (3)$$

The average power consumption after inserting the buffer is $p_p \frac{\beta}{\alpha} + p_b + \sqrt{\frac{p_{b,mb}k\beta(\alpha-\beta)}{\alpha}}$. The average power without the buffer is $p_p$. The saved power is

$$p_p - p_p \frac{\beta}{\alpha} - p_b - \sqrt{\frac{p_{b,mb}k\beta(\alpha-\beta)}{\alpha}} = p_p(1 - \frac{\beta}{\alpha}) - p_b - \sqrt{\frac{p_{b,mb}k\beta(\alpha-\beta)}{\alpha}}.$$

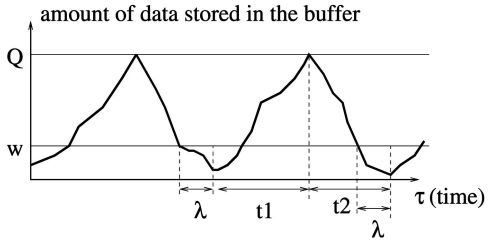


Fig. 3. When $\beta$ is a random variable, the amount of data stored in the buffer does not increase or decrease at constant rates.

If this is positive, adding the buffer and shutting down the producer can save power. The percentage of power savings is $((1 - \frac{\beta}{\alpha}) - \frac{p_b}{p_p} - \frac{1}{p_p}\sqrt{\frac{p_{b,mb}k\beta(\alpha-\beta)}{\alpha}}) \times 100\%$.

### 3.3 Uncertain Consumption Rate and Awakening Delay

The previous section assumes a constant consumption rate, $\beta$, and no awakening delay. In reality, the consumption rate may vary and the awakening delay may be significant.

**Assumption 6 (The awakening delay, $\lambda$, is a constant).** *Let $\lambda$ be the awakening delay for the producer. The awakening delay is a component-specific parameter. The delay ranges from several milliseconds for a processor to a few seconds for a hard disk. The optimal state transition and buffer size can be determined by comparing the energy consumptions between states and choosing the transition with the lowest energy.*

**Assumption 7 (The consumption rate, $\beta$, is a random variable).** *Suppose $\psi(\beta)$ is the probability density function of $\beta$ and $\Psi(\beta)$ is the cumulative distribution function, $\Psi(\beta) = \int_0^\beta \psi(\eta)d\eta$. Let $\gamma$ be the expected value of $\beta$, $\gamma = \int_0^\infty \psi(\beta)\beta d\beta$.*

**Assumption 8 ($\alpha > \gamma$).** *If the expected consumption rate $\gamma$ exceeds the production rate $\alpha$, the buffer will eventually underflow.*

Fig. 3 shows the new strategy when $\beta$ is uncertain and $\lambda$ is nonzero. The producer is awakened when $w$ MB of data remain in the buffer. The value of $w$ is called the *awakening point*. After $\lambda$, the producer starts producing data at rate $\alpha$ and stores the excess data into the buffer.

**Assumption 9 ($\lambda \leq t_2$).** *We intend to preawaken the device to prevent underflow. $\lambda$ represents the awakening time of the producer. If $\lambda > t_2$, the producer must awaken before it is asleep.*

Suppose $S$ MB of data is produced, $S = \alpha t_1$. In steady-state operation, the consumer removes all prefetched data in a period. Hence, the expected length of a period is $\frac{S}{\gamma} = t_1 + t_2$. When the producer starts producing at the beginning of $t_1$, the expected amount of data in the buffer is $w - \lambda\gamma$. The producer fills the buffer at the expected rate of $\alpha - \gamma$. After $t_1$, the expected amount of data in the buffer is $Q = w - \lambda\gamma + (\alpha - \gamma)t_1$. This represents the total buffer size.

### 3.4 Energy Penalty for Underflow

A large $w$ keeps more data in the buffer after $\lambda$ so more energy is consumed by the buffer. On the other hand, a smaller $w$ increases the probability of buffer underflow. Buffer underflow means that the consumer cannot obtain the data it needs. For video prefetching, buffer underflow is manifested by disrupted frames. Hence, $w$ should be sufficiently large to prevent underflow. We quantify the quality-of-service (QoS) by assigning an energy penalty when buffer underflow occurs. Let $\rho$ be the energy penalty for each MB of underflowed data. This value represents the reduction in service; it is dependent upon application. If underflow is unacceptable, $\rho$ is very large. The advantage of using an energy penalty for QoS is that we can explicitly trade off between the energy consumed by the buffer and buffer underflow. The penalty occurs only if the consumed data exceeds $w$, i.e., $\lambda\beta > w$, over the interval $\lambda$. The amount of underflow is the data consumption in excess of $w$ multiplied by the probability of data consumption at that level. Let $f(w)$ be the penalty energy: $f(w) = \rho \int_{\frac{w}{\lambda}}^{\infty} \psi(\beta)(\lambda\beta - w)d\beta$. In the next subsection, we explain how to use $\rho$ to specify the probability of underflow.

### 3.5 Optimal Buffering

The total expected energy consumption in a period is the sum of the energy consumed by the producer: $p_p t_1 + k$, the buffer: $p_b(t_1 + t_2) + p_{b,mb}((w - \lambda\gamma) + (\alpha - \gamma)t_1)(t_1 + t_2)$, and the underflow penalty: $f(w)$. The expected length of a period is $t_1 + t_2 = \frac{S}{\gamma}$. Also, $S = \alpha t_1$ because all prefetched data is consumed within a period in steady-state operation. We use $S$ instead of time to simplify the optimization process. The expected power consumption in one period is $\frac{p_p t_1 + k}{t_1 + t_2} + p_b \frac{t_1 + t_2}{t_1 + t_2} + \frac{p_{b,mb}((w-\lambda\gamma)+(\alpha-\gamma)t_1)(t_1+t_2)}{t_1+t_2} + \frac{f(w)}{t_1+t_2}$. Therefore, the expected power consumption $p_e$ in one period is:

$$p_e = \frac{p_p\gamma}{\alpha} + \frac{k\gamma + f(w)\gamma}{S} + p_b + p_{b,mb}(w - \lambda\gamma) + p_{b,mb}(\alpha - \gamma)\frac{S}{\alpha}. \tag{4}$$

We find the partial derivatives with respect to $S$ and $w$ to compute the values that minimize $p_e$:

$$\frac{\partial p_e}{\partial w} = p_{b,mb} - \frac{\rho\gamma \int_{\frac{w}{\lambda}}^{\infty} \psi(\beta)d\beta}{\lambda S} = 0, \tag{5}$$

$$\frac{\partial p_e}{\partial S} = \frac{p_{b,mb}(\alpha - \gamma)}{\alpha} - \frac{k\gamma + f(w)\gamma}{S^2} = 0. \tag{6}$$

The condition to have a minimum average power is $\frac{\partial^2 p_e}{\partial S^2}\frac{\partial^2 p_e}{\partial w^2} - (\frac{\partial^2 p_e}{\partial S \partial w})^2 > 0$; otherwise, the solutions $S$ and $w$ represent a saddle point of $p_e$. We can simplify the expression as

$$2\frac{k\gamma + f(w)\gamma}{S^3}\frac{\rho\gamma\psi(w/\lambda)}{\lambda^2 S} - \left(\frac{\rho\gamma \int_{w/\lambda}^{\infty} \psi(\beta)d\beta}{\lambda S^2}\right)^2 = \frac{\rho\gamma^2}{\lambda^2 S^4}\left[2\psi\left(\frac{w}{\lambda}\right)(k + f(w)) - \rho\left(\int_{\frac{w}{\lambda}}^{\infty} \psi(\beta)d\beta\right)^2\right].$$

Since this is not necessarily positive, a minimum average power may not exist. Suppose a minimum does exist. Let $S^*$ and $w^*$ be the solutions for this minimum average power. Since $Q = w^* - \lambda\gamma + (\alpha - \gamma)t_1$ and $\alpha t_1 = S^*$, we can also compute the required buffer size $Q^*$. The derivation is similar to finding the optimal reorder points as explained in [32], where ordering cost, holding cost, and shortage cost are analogous to producer power, buffer power, and underflow penalty, respectively.

$$\int_{\frac{w^*}{\lambda}}^{\infty} \psi(\beta)d\beta = \frac{p_{b,mb}\lambda S^*}{\rho\gamma}, \tag{7}$$

$$S^* = \sqrt{\frac{\alpha\gamma(k + \rho\int_{\frac{w^*}{\lambda}}^{\infty} \psi(\beta)(\lambda\beta - w^*)d\beta)}{p_{b,mb}(\alpha - \gamma)}}. \tag{8}$$

We define the probability of underflow as the likelihood that the amount of data consumed during $\lambda$ is greater than $w$. This can be translated directly into a QoS specification. Let $\zeta$ be the probability of underflow. Given a value of $w$ and the consumption rate $\psi(\beta)$, the probability of underflow is:

$$\zeta = \int_{\frac{w^*}{\lambda}}^{\infty} \psi(\beta)d\beta. \tag{9}$$

A system designer can specify a quantitative value for $\zeta$ based upon the specific application. A low value for $\zeta$ indicates high QoS. In (9), $w^*$ is the only unknown parameter, indicating the minimum awakening point that can satisfy the requirement $\zeta$. Therefore, the value of $w^*$ can be computed numerically. A system designer can determine $\rho$ from $\zeta$. We derive two closed form equations from (7)-(8) to solve for $S^*$ and $\rho$, leading to a result for the optimal buffer size $Q^*$.

$$S^* = \frac{\rho\gamma\zeta}{p_{b,mb}\lambda} = \sqrt{\frac{\alpha\gamma(k + \rho\int_{\frac{w^*}{\lambda}}^{\infty} \psi(\beta)(\lambda\beta - w^*)d\beta)}{p_{b,mb}(\alpha - \gamma)}},$$

$$\left(\frac{\rho\gamma\zeta}{p_{b,mb}\lambda}\right)^2 = \frac{\alpha\gamma\left(\rho\int_{\frac{w^*}{\lambda}}^{\infty} \psi(\beta)(\lambda\beta - w)d\beta + k\right)}{p_{b,mb}(\alpha - \gamma)},$$

$$a\rho^2 + b\rho + c = 0, \tag{10}$$

where $(a, b, c)$ are defined as:

$$a = \left(\frac{\gamma\zeta}{p_{b,mb}\lambda}\right)^2,$$

$$b = \frac{-\alpha\gamma}{p_{b,mb}(\alpha - \gamma)}\int_{\frac{w^*}{\lambda}}^{\infty} \psi(\beta)(\lambda\beta - w^*)d\beta,$$

$$c = \frac{-\alpha\gamma k}{p_{b,mb}(\alpha - \gamma)},$$

$$S^* = \frac{\gamma\zeta}{p_{b,mb}\lambda}\rho, \tag{11}$$

$$Q^* = w^* - \lambda\gamma + (\alpha - \gamma)\frac{S^*}{\alpha}. \tag{12}$$

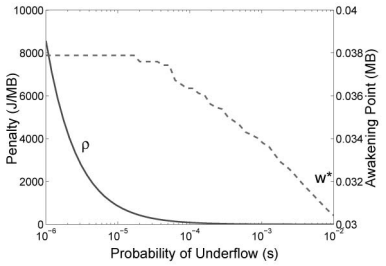


Fig. 4. Underflow penalty $\rho$ and awakening point $w^*$ as probability of underflow $\zeta$ changes.

The value of $\rho$ we find for $w^*$ allows us to minimize $p_e$ with respect to $S^*$. This is because our equations are derived from (5)-(6), which solve for points $S^*$ and $w^*$ that minimize $p_e$. Because $w^*$ is derived from the underflow criterion $\zeta$, we can analytically quantify QoS in our optimization.

### 3.6 Analysis

The density function $\psi(\beta)$ is nonnegative, so $\int_{\frac{w}{\lambda}}^{\infty} \psi(\beta)d\beta$ decreases as $w$ grows. From (7), we note that when $p_{b,mb}$ increases, $w^*$ should decrease. This is intuitive: When $p_{b,mb}$ is large, the buffer memory consumes more power. Thus, $w^*$ should be small to keep the amount of data in the buffer lower. When $\rho$ increases, $w^*$ also increases to reduce the probability of underflow. When $\gamma$ is large, the consumer removes data from the buffer quickly. As a result, $w^*$ should be larger. Because $\int_{\frac{w^*}{\lambda}}^{\infty} \psi(\beta)d\beta \leq 1$, $\frac{p_{b,mb}\lambda S^*}{\rho\gamma}$ must not exceed one. Fig. 4 details how $\rho$ and $w^*$ change with $\zeta$ for one of our sample videos (V2 using an IBM microdrive). Complete details about the videos and experimental setup are explained in Section 4.1. As $\zeta$ increases, the QoS requirement is reduced. The penalty $\rho$ decreases because underflow is less important; similarly, $w^*$ decreases. When $\zeta$ decreases, the QoS requirement becomes more stringent. Subsequently, the values of $\rho$ and $w^*$ increase to reduce the probability of underflow.

Comparing (3) and (8), we find two differences. The constant rate $\beta$ is replaced by the expected rate $\gamma$. $S^*$ also includes the underflow penalty. When $\rho$ is larger, the penalty is higher. As a result, $S^*$ increases to amortize the cost of QoS violations across a larger interval. Finally, the buffer size $Q^*$ in (12) is different from $Q$ in (2) because the producer starts filling the buffer when there are $w^* - \lambda\gamma$ MB of data in the buffer. $Q^*$ includes this expected amount of data. When the density function $\psi(\beta)$ is known, we can calculate $w^*$ and $S^*$.

Existing work indicates that a producer's power savings can be maximized by choosing as large a buffer as possible [7]. Our mathematical model in (4) verifies that increasing $S^*$ reduces the impact of the awakening energy $k$. The producer's average power $p_p$ remains unaffected by changing the buffer size. Given this observation and the fact that the producer's power consumption is often one or two orders of magnitude greater than the buffer's retention power, a simple solution to the buffer insertion problem is to prefetch the entire data stream. This solution minimizes the producer's total energy consumption because it is active for the minimum amount of time to produce the data. However, the calculations reveal the fallacy of this approach. The producer's average power $p_p$ is multiplied by $\frac{\gamma}{\alpha}$. In many cases, $\alpha$ is much larger than $\gamma$. For example, an MPEG-1 video may have a bitrate of $\gamma = 50$ kB/s (400 kbits/s), whereas an IBM 2.5'' TravelStar disk has $\alpha = 17.6$ MB/s. Subsequently, the producer's average power is multiplied by 0.003, reducing the producer's average power over a period by two orders of magnitude. Conversely, the buffer's retention power is multiplied by $\frac{\alpha-\gamma}{\alpha} = 0.997$ and the size of the buffer, indicating that the buffer consumes more power than the producer over a period. The simple solution only maximizes power savings if $\gamma$ is near $\alpha$ or the producer's average power greatly exceeds the buffer's retention power and the stream is short. The optimal buffering technique will yield the same results as the simple solution under these conditions. We believe that our work is the first study with an analytic solution to calculate the optimal amount of prefetched data and buffer sizes for power management.

### 3.7 Modifications for More Practical Hardware Models

The previous subsections derive an optimal buffer size and awakening point for a memory model where memory may be shut down with single-byte resolution. This assumption is not consistent with current memory technology. As explained in Section 2.2, SDRAM and RDRAM have a minimum resolution of 16 MB. To account for this discrepancy, we note that the average power $p_e$ is convex with respect to the buffer size $Q$. This trend is evident from the mathematics. Note that each function exhibits a strict global minimum; this is the optimal buffer size computed from (10)-(12). Since $p_e$ is convex, the optimal discrete buffer size must be either $n\lfloor\frac{Q^*}{n}\rfloor$ or $n\lceil\frac{Q^*}{n}\rceil$, where $n$ represents the resolution of a single bank ($n = 16$ MB, in our case). That is, all other discrete buffer sizes will consume a higher $p_e$ than the two neighboring buffer sizes. To find the optimal discrete buffer size from our equations, the optimal buffer size is selected assuming single-byte resolution using (10)-(12). Then, the average power of the neighboring discrete buffer sizes are computed using (4). The discrete buffer size with the lowest average power is selected. Hence, our optimal technique can locate the optimal buffer size, even when memory must shut down in discrete quantities.

Many producers still consume small amounts of power in the off state. We assume that this quantity is zero in our derivation to simplify explanation. We now illustrate how to include this power within the optimal buffer calculations. Recall that the producer's energy consumption for a complete period is $p_p t_1 + k$. Now, suppose that the power consumption in the off state is $p_{p,off}$. Then, the producer's total energy over $T$ becomes:

$$\begin{aligned} E_p &= p_p t_1 + p_{p,off} t_2 + k \\ &= p_p t_1 + p_{p,off}(T - t_1) + k \\ E_p &= (p_p - p_{p,off})t_1 + p_{p,off} T + k. \end{aligned} \tag{13}$$

Both of the power terms can be merged into the other terms for computing $p_e$ and do not alter our final equations (10)-(12). However, we must consider the additional terms when determining if power management can save energy.

TABLE 2
Hardware Parameters

| | Microdrive | IDE | LAN | WAN |
|---|---|---|---|---|
| $\lambda$ | 0.238 s | 4.5 s | 1.996 s | 1.996 s |
| $p_p$ | 0.221 W | 2.166 W | 0.500 W | 0.500 W |
| $\alpha$ | 1.824 MB/s | 17.6 MB/s | 5.876 MB/s | 0.443 MB/s |
| $k$ | 0.151 J | 17.1 J | 0.333 J | 0.333 J |
| $T_{be}$ | 0.559 s | 7.89 s | 0.666 s | 0.666 s |

*Local and wide-area networks use the same device but have different $\alpha$.*

## 4 Experiments

### 4.1 Simulation Setup

The effectiveness of our DPM policy is evaluated by a MATLAB simulation. This simulation models two hard disks, two network cards, and an RDRAM buffer for playback of an MPEG video trace. The video traces are constructed using a modified version of the MPlayer [35] media viewer (version 1.0pre4-3.3.3) and consist of frame type and size. An IBM 1 GB microdrive, an IBM 2.5'' TravelStar IDE disk, and a Netgear PCMCIA network card act as producers that generate data for a Rambus RDRAM buffer. The parameters for the producers are extracted from manufacturer datasheets and physical measurements in our previous work [8], as shown in Table 2. We obtain the bandwidth of each hard disk by using the command hdparm in Linux (`hdparm -t`). We measure the power of the hard disk without applying any workload to determine the static power of the hard disk. To obtain dynamic power, we measure the disk's power consumption under different data read rates (MB/s). After subtracting the static power, we divide the differences by the rates. We use the average value of these quotients as the dynamic energy for reading 1 MB of data. Together, these two measures can be used to compute the producer's average power $p_p$ when prefetching data. We determine $k$ by measuring the energy consumed by the hard disk when it is turned on. We will refer to the IBM microdrive as "Microdrive" and the TravelStar disk as "IDE."

To determine the characteristics for the Netgear PCMCIA network, we perform measurements with a National Instruments data acquisition card. We send a stream of packets over the network to measure the average power $p_p$. To find $k$, we measure the energy between the wake-up command and the first packet's transmission. We consider two applications for this network card: a local-area network (LAN) and a wide-area network (WAN). The production rate of each case and hardware parameters of the network card are provided in Table 2. To find the production rate of each case, we transferred many large files over the network and determined the average data transmission rate. The WAN is much slower than the LAN because their packets must traverse more hops before arriving at their destination. We choose a 0.01 percent probability of underflow ($\zeta$) for our experiments.

We use the memory model described in Section 2.2 for RDRAM. When banks of memory are idle, they are reduced to the nap state. We select the nap state because transition back to the active state incurs little overhead with respect to the power-down state. We set the memory's static power to the average of the read access power and write access power. This models the scenario where one bank of memory is being read half the time (reading data from buffer) and written half the time (prefetching data into buffer). While the access pattern may vary, we choose this static power because it represents near worst-case access power. That is, inserting memory idle periods will reduce access power consumption, leading to a subsequent increase in the energy savings and effectiveness of our methodology.

All measured energy savings for our method and comparison methods are referenced against a system without power management using MPlayer. The reference system includes a small memory buffer to handle variations in data production. This configuration is similar to how media players are actually implemented. In particular, MPlayer uses a one-frame buffer when reading from local devices and a buffer of less than 1 MB when reading from network devices. Hence, the power for a single bank of memory must be attributed to the reference system, despite the absence of power management.

Table 3 contains the characteristics of all the MPEG-1, MPEG-2, and MPEG-4 videos used in our study, sorted by type and bit rate. These videos represent a broad range of scenes, including different levels of motion. Since the videos are encoded using different tools, the videos exhibit different resolutions, frame rates, compression, and GoP formats. Consequently, the selected videos provide a representation of the workload many streaming applications may expect.

Although we simulate each video on each device, we note that the bit rates for some videos exceed the production rate of some devices. For example, the MPEG-2 videos cannot be played over the wide-area network because the I-frames of the lowest bit rate MPEG-2 video ($\gamma = 438$ kB/s) exceed the production rate ($\alpha = 443$ kB/s). Although $\alpha > \gamma$, the producer cannot prefetch quickly enough to display each I-frame in time. Each I-frame underflows, leading to large underflow penalties. However, each I-frame would underflow on the reference case as well. We will note each video where the hardware is incapable of supporting full QoS without power management with the symbol "–" in Table 3 and Table 4.

Our prefetching technique requires knowledge of the probability distribution $\psi(\beta)$. We adopt Chung et al.'s approach [5], where statistical information is embedded at the beginning of each video. The size of each frame in bytes in sequential order is sufficient to generate an exact probability density function. Since frame sizes are integer values, we can use radix sort to generate $\Psi(\beta)$ in linear time. The increase in video size is negligible as each frame size can be represented by four bytes. This equates to an additional 120 bytes per second in a 30 frames per second video. In (9)-(12), the integrals represent the amount of data consumed during $\lambda$. The function $\psi(\beta)$ can also be computed in linear time from the sequential ordering of frames. Given the frame rate, we can compute the number of frames ($n_f$) that occur during $\lambda$. Then, we iterate through the frames and compute the size of each set of $n_f$ sequential frames. This data can be used to generate $\psi(\beta)$.

TABLE 3
Average Power and Total Energy Consumption for Each Video Using MPlayer on the Reference System

| Id | Format | Resolution | Size | Bitrate | Microdrive Energy | IDE Energy | LAN Energy | WAN Energy |
|---|---|---|---|---|---|---|---|---|
| v1 | MPEG-1 | $160 \times 112$ | 59.0 MB | 0.40 Mbps | 1887 J | 4178 J | 2218 J | 2218 J |
| v2 | MPEG-1 | $352 \times 240$ | 187.3 MB | 0.64 Mbps | 3740 J | 8281 J | 4399 J | 4399 J |
| v3 | MPEG-1 | $352 \times 240$ | 13.6 MB | 0.68 Mbps | 258 J | 570 J | 303 J | 303 J |
| v4 | MPEG-1 | $320 \times 240$ | 44.7 MB | 0.99 Mbps | 576 J | 1276 J | 678 J | 678 J |
| v5 | MPEG-1 | $352 \times 240$ | 64.4 MB | 1.14 Mbps | 721 J | 1597 J | 849 J | 849 J |
| v6 | MPEG-1 | $320 \times 240$ | 99.8 MB | 1.16 Mbps | 1097 J | 2429 J | 1292 J | 1292 J |
| v7 | MPEG-2 | $720 \times 480$ | 630.8 MB | 3.50 Mbps | 2291 J | 5069 J | 2711 J | — |
| v8 | MPEG-2 | $704 \times 528$ | 180.5 MB | 4.01 Mbps | 571 J | 1264 J | 677 J | — |
| v9 | MPEG-2 | $720 \times 480$ | 712.9 MB | 4.21 Mbps | 2150 J | 4757 J | 2549 J | — |
| v10 | MPEG-2 | $320 \times 240$ | 1452.4 MB | 8.01 Mbps | 2281 J | 5047 J | 2730 J | — |
| v11 | MPEG-2 | $704 \times 528$ | 360.9 MB | 8.02 Mbps | 568 J | 1252 J | 677 J | — |
| v12 | MPEG-2 | $704 \times 528$ | 541.1 MB | 12.0 Mbps | 569 J | 1239 J | 677 J | — |
| v13 | MPEG-4 | $320 \times 240$ | 28.0 MB | 0.11 Mbps | 3255 J | 7207 J | 3823 J | 3823 J |
| v14 | MPEG-4 | $352 \times 240$ | 51.3 MB | 0.20 Mbps | 3254 J | 7205 J | 3823 J | 3823 J |
| v15 | MPEG-4 | $720 \times 480$ | 166.4 MB | 0.61 Mbps | 3482 J | 7710 J | 4095 J | 4095 J |
| v16 | MPEG-4 | $720 \times 480$ | 436.5 MB | 0.80 Mbps | 6961 J | 15413 J | 8190 J | 8190 J |
| v17 | MPEG-4 | $720 \times 480$ | 262.9 MB | 0.81 Mbps | 4177 J | 9248 J | 4914 J | 4914 J |
| v18 | MPEG-4 | $720 \times 480$ | 328.2 MB | 1.21 Mbps | 3477 J | 7699 J | 4095 J | 4095 J |

TABLE 4
Summary of Buffer Sizes, Awakening Points, and Energy Savings for All Videos

| | Microdrive | | | IDE | | | LAN | | | WAN | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | $Q^*$ | $w^*$ | Savings | $Q^*$ | $w^*$ | Savings | $Q^*$ | $w^*$ | Savings | $Q^*$ | $w^*$ | Savings |
| v1 | 16 | 0.021 | 9.3 | 32 | 0.235 | 57.3 | 16 | 0.107 | 26.2 | 16 | 0.107 | 23.4 |
| v2 | 16 | 0.036 | 9.1 | 48 | 0.383 | 56.7 | 16 | 0.179 | 26.0 | 16 | 0.179 | 21.6 |
| v3 | 16 | 0.037 | 9.1 | 48 | 0.420 | 57.3 | 16 | 0.194 | 26.1 | 16 | 0.194 | 21.4 |
| v4 | 16 | 0.044 | 8.7 | 48 | 0.569 | 57.1 | 16 | 0.267 | 25.8 | 16 | 0.267 | 19.0 |
| v5 | 16 | 0.050 | 8.6 | 64 | 0.660 | 56.8 | 16 | 0.298 | 25.7 | 16 | 0.298 | 17.9 |
| v6 | 16 | 0.052 | 8.6 | 64 | 0.682 | 56.1 | 16 | 0.305 | 25.7 | 16 | 0.305 | 17.7 |
| v7 | 16 | 0.309 | 6.4 | 96 | 4.370 | 52.9 | 16 | 2.004 | 24.0 | — | — | — |
| v8 | 16 | 0.158 | 5.9 | 112 | 2.312 | 53.0 | 16 | 1.039 | 23.7 | — | — | — |
| v9 | 16 | 0.235 | 5.7 | 112 | 3.184 | 52.5 | 16 | 1.509 | 23.6 | — | — | — |
| v10 | 16 | 0.271 | 2.3 | 144 | 4.542 | 48.6 | 16 | 2.006 | 21.1 | — | — | — |
| v11 | 16 | 0.307 | 2.6 | 144 | 4.600 | 48.9 | 16 | 2.068 | 21.0 | — | — | — |
| v12 | 16 | 0.450 | 0.2 | 176 | 6.886 | 46.1 | 32 | 3.096 | 18.5 | — | — | — |
| v13 | 16 | 0.037 | 9.6 | 16 | 0.159 | 58.0 | 16 | 0.072 | 26.4 | 16 | 0.072 | 25.6 |
| v14 | 16 | 0.045 | 9.5 | 32 | 0.229 | 57.6 | 16 | 0.104 | 26.3 | 16 | 0.104 | 24.9 |
| v15 | 16 | 0.174 | 9.1 | 48 | 1.029 | 56.7 | 16 | 0.486 | 26.1 | 16 | 0.486 | 21.8 |
| v16 | 16 | 0.126 | 8.9 | 48 | 0.767 | 56.4 | 16 | 0.361 | 25.9 | 16 | 0.361 | 20.4 |
| v17 | 16 | 0.167 | 8.9 | 48 | 0.957 | 56.3 | 16 | 0.464 | 25.9 | 16 | 0.464 | 20.4 |
| v18 | 16 | 0.190 | 8.5 | 64 | 1.225 | 55.9 | 16 | 0.605 | 25.6 | 16 | 0.605 | 17.4 |

*Buffer size $Q^*$ and awakening point $w^*$ are in units of MB. Energy savings are given in percent (%).*

## 4.2 Optimal Buffering

Fig. 5 exemplifies the difference in power consumption between a buffered configuration and our reference configuration based on MPlayer's implementation. The upper plot indicates power consumption for each system over time. The dashed line represents the average power consumption of the reference configuration over a single period. The solid line indicates the power consumption of the buffered system. The lower plot shows energy consumption over time. Without prefetch, the IDE disk is kept on continuously. The disk and a single bank of RDRAM consumes average power of 3.5 W. With the buffer, the producer's power varies. As the buffer is filled from the producer, the overall power consumption is larger than the reference case until point A. This is because more power is consumed,

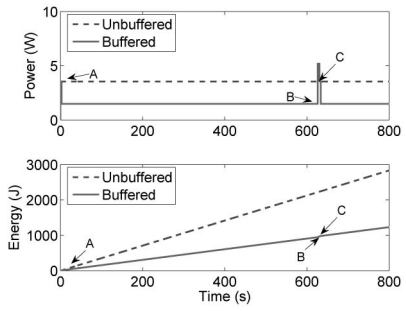


Fig. 5. Comparison of power and energy in buffered and reference systems for video V2 using the IDE disk.

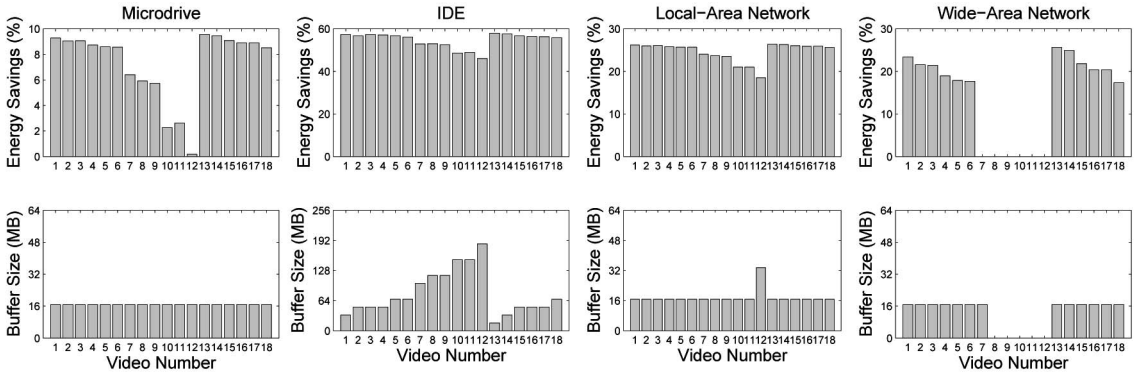


Fig. 6. Power savings and discrete buffer sizes for statistically optimal DPM on each hardware configuration.

retaining data in the buffer. Subsequently, the energy consumption exceeds that of the unbuffered system. At point A, the disk is turned off, reducing the power dissipation. The power is reduced until point B as the buffered data is consumed. When the buffer reaches its awakening point at point B, the power dissipation increases for a short time due to the overhead associated with spinning up the hard disk. After the IDE disk awakens at point C, one period has been completed and the power savings between the two systems can be evaluated. Since the buffered system's energy curve is beneath the reference case, the buffering technique saves energy.

Recall from Section 4.1 that our experimental results are referenced to the implementation of MPlayer, which does not use power management techniques. We will refer to this case as the "reference system" or, simply, the "reference." Table 3 shows the average power and total energy consumed by the reference for each video on each hardware configuration. These figures are presented to provide a sense of scale for our energy saving results.

The optimal buffer energy savings and buffer sizes for all videos are shown in Fig. 6. Note that each buffer size is a multiple of 16 MB because we assume RDRAM with 16 MB memory resolution. The energy savings of our technique vary based upon the statistical properties of individual videos. The optimal buffer sizes $Q^*$ are 16 MB for all videos on the microdrive, LAN, and WAN, excluding video V12 on the LAN. Because the bit rate of V12 is significantly higher than the other videos, a larger buffer is required to facilitate power management. In general, the buffer sizes increase as the video data rate increases. This is consistent with (11), where $\gamma$ increases. The value of $w^*$ ranges between 21 kB (video V1) and 3.096 MB (video V12) for these devices, increasing with the consumption rate. The value of $w^*$ must be increased to maintain the same $\zeta$ as the lower consumption rate. The awakening time $\lambda$ also plays a role in computing $w^*$. Since more frames are consumed during a longer $\lambda$, the device must have a higher $w^*$ to provide the same QoS. Energy savings range between 9.6 percent (video V13) and 0.2 percent (video V12) with an average of 7.3 percent for the microdrive when compared to the reference case. For the LAN, savings range from 26.4 percent (video V13) to 18.5 percent (video V12) with an average of 24.6 percent. WAN savings are between 25.4 percent (video V13) and 17.4 percent (video V18) with an average of 21.0 percent. The WAN energy savings do not include the MPEG-2 videos because their bit rates are too high to play on the reference configuration without underflow. Energy savings decrease as the consumption rate increases because a longer portion of each period is devoted to prefetching data. The values of $Q^*$ and $w^*$ for the microdrive and network card can be achieved on a portable device such as an iPod or PocketPC.

As with the microdrive and network card, the energy savings for the IDE disk decrease as the consumption rate increases. However, the optimal buffer size $Q^*$ varies significantly across video formats. The MPEG-1 videos indicate optimal buffer sizes between 32 MB (video V1) and 64 MB (videos V5 and V6). The MPEG-2 videos require buffers from 96 MB (video V7) to 176 MB (video V12), whereas the MPEG-4 videos have optimal buffer sizes between 16 MB (video V13) and 64 MB (video V18). The same general trend holds—higher bit rates require larger buffers. Similarly, we observe that $w^*$ increases with bit rate from 159 kB (video V13) to 6.89 MB (video V12). Energy savings range from 58.0 percent (video V13) to 46.1 percent (video V12) with an average of 54.7 percent. While these buffer sizes are typically too large for embedded systems, the buffer sizes are within reasonable size limits for personal computers.

### 4.3 Oracle Buffering

When comparing DPM policies, it is often useful to know the highest possible power savings. An *Oracle DPM* is a policy that consumes the minimum amount of energy, given complete a priori knowledge of a video's frames and hardware parameters. The Oracle DPM uses this information to determine the most energy efficient method possible, setting an upper bound on power savings. This analysis is performed globally, avoiding local decisions that may produce inefficiencies elsewhere. While our method is statistically optimal, it does not consider global optimization of incomplete periods (i.e., periods that do not prefetch the full $S^*$ MB of data). An example is shown in Fig. 7. Fig. 7a depicts the power consumption of our statistically optimal DPM over a complete video. Each period's prefetch is determined using statistical information, but a few frames remain at the beginning of the final period. The producer must be awakened again to complete the video. In this case, it would be more energy efficient to prefetch the small

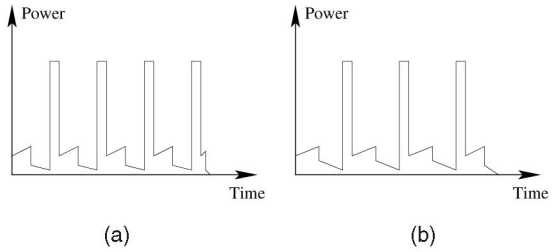


Fig. 7. (a) Statistically optimal DPM only considers statistical properties of a video. (b) Oracle DPM avoids excess periods by optimizing globally with a priori knowledge.

amount of data in the previous period to avoid awakening the producer again. However, our DPM policy has only statistical knowledge of future frames' sizes, so it cannot account for this situation. The Oracle DPM scheme in Fig. 7b can avoid the unnecessary power consumption by sizing each prefetch such that the additional period is not needed to finish the video because it has complete knowledge of the video. As videos increase in length, the waste due to a short period is reduced. Finding the oracle power savings is difficult because it requires global optimization in the number of periods, the prefetch size, and the awakening point for each period. Since the consumption rate varies, the prefetch size and awakening point should also adjust accordingly. The number of periods is affected by the prefetch size; however, as shown in Fig. 7, the prefetch size is affected by the number of periods. Hence, multiple iterations are needed to find this oracle power savings. Since this is impractical for long videos, we do not consider the oracle power savings.

### 4.4 Heuristic Buffering

Since finding the Oracle DPM power savings is difficult, we compare our method with heuristically sized buffers to support our claim that our buffer size is optimal. Particularly, we compare against Bertozzi et al. [7], who claim that buffers should be as large as possible to maximize energy savings. We consider the energy consumption of 16 MB, 32 MB, 48 MB, 64 MB, 80 MB, 96 MB, and 112 MB buffers for the microdrive, LAN, and WAN. We use slightly larger buffers (up to 256 MB) for the IDE disk because larger buffer sizes are necessary to achieve optimal energy savings. Bertozzi et al. determine an awakening point based upon the probability of underflow. Thus, we assign their DPM policy the same awakening point as our policy for valid comparison. We will discuss the affect of the awakening point on energy savings shortly.

Fig. 8 shows the average energy savings for the heuristic buffer sizes for each hardware configuration. This figure illustrates two key points. First, our optimal buffer size is capable of finding the maximum energy savings for each video. A simple comparison indicates that our optimal buffer size exhibits energy savings greater than or equal to each of the heuristic buffer sizes. The optimal energy savings are equal to the 16 MB buffer sizes for the microdrive and WAN because the optimal buffer sizes are all 16 MB. The energy savings are nearly equal for the 16 MB buffer LAN because all but one video require 16 MB buffers. The second point is that larger buffer sizes do not necessarily save more energy when the memory energy is considered. The mathematical justification is explained in Section 3.6 and confirmed by the data in Fig. 8.

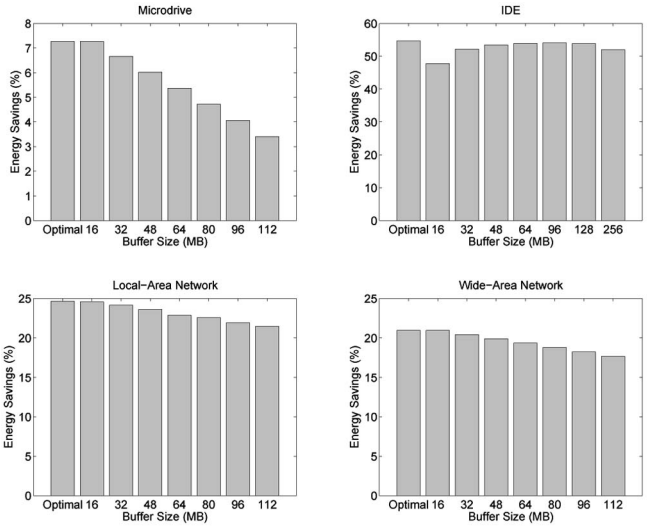


Fig. 8. Energy savings for each heuristic buffer size with optimal awakening point $w^*$ for each hardware configuration.

We now consider the effect of awakening point $w^*$ on energy savings. Fig. 9 plots the average energy savings for each heuristic buffer size over a range of awakening points. We observe that no energy savings are achieved for small awakening points. Awakening points less than $w^*$ experience large amounts of underflow. Subsequently, these buffers pay a high QoS cost. Larger buffer sizes reduce the impact of QoS penalty by shutting down the device less frequently and amortizing the cost across a longer interval. For example, the 80 MB and 96 MB buffers are capable of saving energy for the microdrive for 256 kB awakening points, unlike the smaller buffers sizes. Once the buffers exceed $w^*$, the power savings for the values of $Q$ remain relatively constant because underflow is no longer a significant factor in energy consumption. In this region of

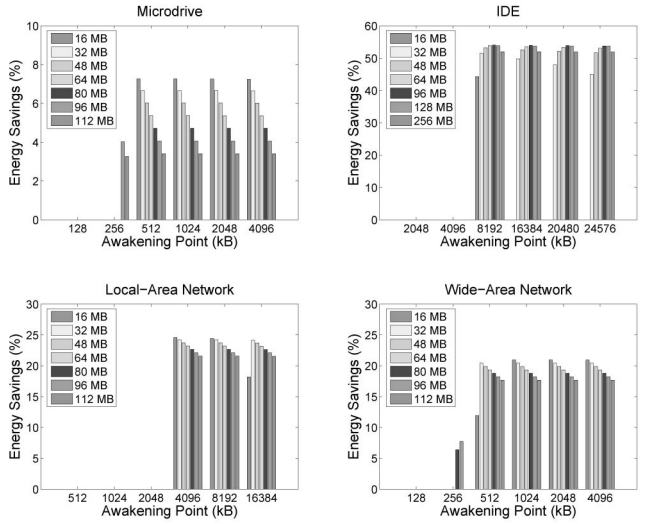


Fig. 9. Energy savings for each heuristic buffer size with optimal awakening point $w^*$ for each hardware configuration.

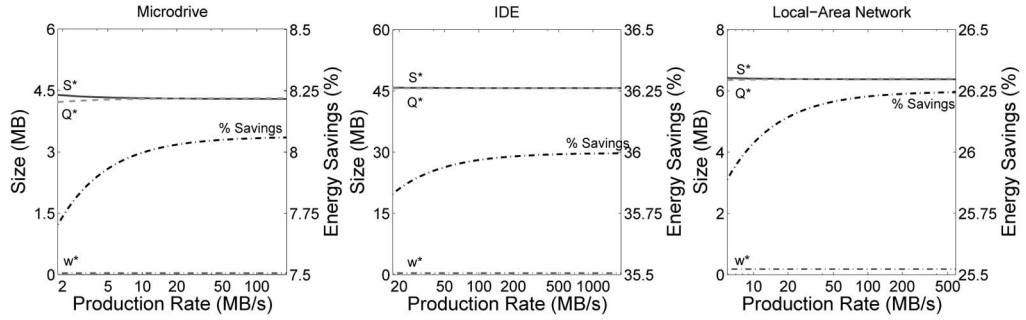


Fig. 10. Buffer size, prefetch size, awakening point, and energy savings for each hardware configuration as a function of $\alpha$.

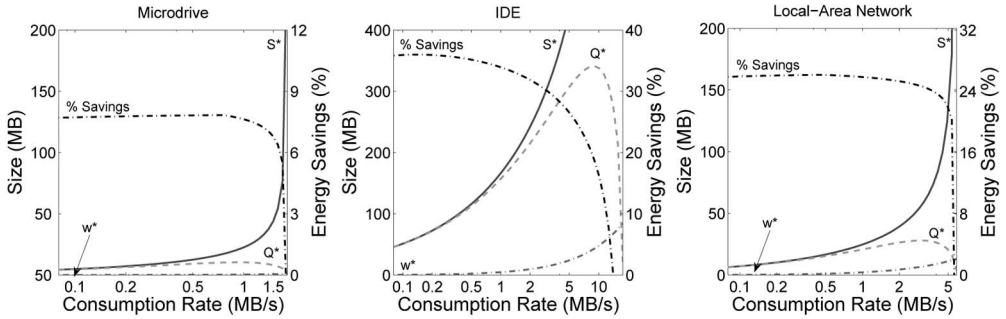


Fig. 11. Buffer size, prefetch size, awakening point, and energy savings for each hardware configuration as a function of $\gamma$.

the graph, the balance between awakening overhead and buffer power consumption is key. Smaller buffers incur more cost due to changing power states, whereas larger buffers consume more memory power. This trade-off can be observed by comparing the values of $Q^*$ at a fixed $w^*$. Consider the IDE disk for a 16,384 kB (16 MB) awakening point. The average optimal value of $Q^*$ lies near 75 MB. Consequently, the energy savings increase from 16 MB to 96 MB and decrease from 96 MB to 256 MB. The best energy savings for the 256 MB buffer are near 50 percent, whereas the optimal buffer saves 54.7 percent energy with respect to the reference system. This demonstrates that large buffers are not necessarily beneficial when the buffer's energy is considered.

The value of our analytical policy becomes evident by considering that we can select the most effective buffer size and awakening point based on data and hardware properties. Many mobile systems operate with constrained system resources. By computing the optimal buffer size, valuable memory resources are not wasted by oversizing the buffer. Likewise, battery life is not wasted by selecting too small or too large a buffer to maximize energy savings. Furthermore, our technique is not limited to a single data type or hardware configuration. We use MPEG videos in our experiments, but no constraints prevent our optimal buffering technique from being applied to other streaming data formats. As data types change, our method is capable of identifying the optimal point without additional measurement or computationally expensive simulation. Our generic solution can also be applied to any hardware configuration where the power consumption properties and production bandwidth are known. Hence, changes in hardware, application, and data sets can be effectively handled quickly by using our statistically optimal DPM policy.

## 5 Sensitivity to Parameters

This section explores the changes of the buffer parameters and expected energy savings for changes in various system parameters. Since our statistically optimal DPM policy is analytical, the effects of technological advancement can be determined analytically. The *sensitivity* of a system is the ratio of change in a function $y$ to incremental changes in a parameter $x$ [36]. The sensitivity can be calculated by finding $S_x^y = \frac{\partial y}{\partial x}\frac{x}{y}$. For example, the sensitivity of $S^*$ to variations in $\alpha$ is:

$$\begin{aligned}
\frac{\partial S^*}{\partial \alpha} &= \frac{1}{2}\left(\frac{2\alpha\gamma(k+f(w))}{p_{b,mb}(\alpha-\gamma)}\right)^{-\frac{1}{2}}\left(\frac{-2\gamma^2(k+f(w))}{p_{b,mb}(\alpha-\gamma)^2}\right)\\
\frac{\partial S^*}{\partial \alpha} &= \frac{-\gamma S^*}{2\alpha(\alpha-\gamma)}\\
S_\alpha^{S^*} &= \frac{\partial S^*}{\partial \alpha}\frac{\alpha}{S^*} = \frac{-\gamma}{2(\alpha-\gamma)}.
\end{aligned} \tag{14}$$

From this result, we observe that the sensitivity of $S^*$ for a given $\alpha$ depends only upon the expected consumption rate $\gamma$ and the production rate $\alpha$ itself. By substituting for the values of $\alpha$ and $\gamma$, we can determine how $S^*$ varies for small variations of $\alpha$. Variations may occur due to many factors, such as fragmentation and bus contention. In each case, the value of $\alpha$ seen by the prefetching operation decreases slightly. This analysis cannot be undertaken with heuristically sized buffers.

The values of $S^*$, $w^*$, $Q^*$, and energy savings for different hardware parameters are included in Fig. 10, Fig. 11, Fig. 12, and Fig. 13. We do not show the WAN configuration because the results are similar to the LAN configuration. The sizes of $S^*$, $w^*$, and $Q^*$ are indicated on the left vertical axis. Energy savings are measured on the right vertical axis.

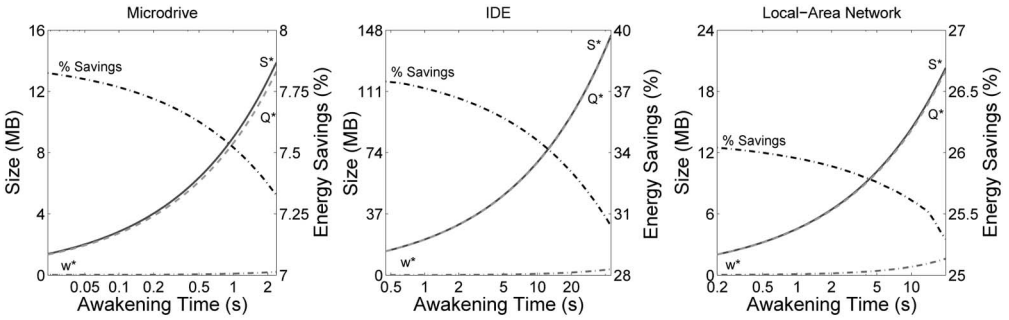


Fig. 12. Buffer size, prefetch size, awakening point, and energy savings for each hardware configuration as a function of $\lambda$.

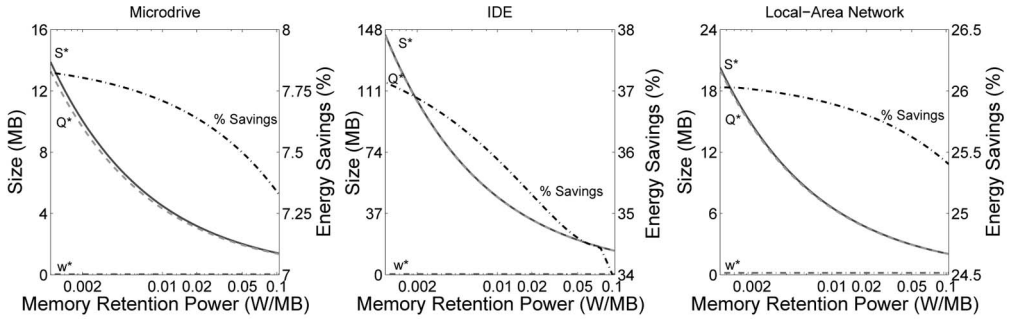


Fig. 13. Buffer size, prefetch size, awakening point, and energy savings for each hardware configuration as a function of $p_{b,mb}$.

These figures are constructed using video V2 by changing the desired parameter and analytically solving for the buffer parameters. We choose V2 because it is a long video and provides many frames for statistical sampling. Each horizontal axis is represented on a logarithmic scale. These results do not consider the 16 MB memory shutdown resolution. These figures demonstrate how changing system parameters affects the performance of our DPM policy. Since curves on some configurations are difficult to distinguish, we describe a single configuration for each figure and note any differences between the described configuration and other cases. As technology progresses with increasing producer bandwidth, increasing video quality, decreasing awakening times, and memory retention power, our statistically optimal DPM technique remains a practical scheme for saving power in streaming video systems.

Increases in $\alpha$ represent different bandwidths for storage devices. Fig. 10 shows the buffer parameters $S^*$, $w^*$, and $Q^*$ and the associated energy savings as the production rate $\alpha$ increases. Observe the microdrive configuration. The prefetch size $S^*$ decreases slightly as $\alpha$ increases. Note that this curve matches the results from (14); the slope of $S^*$ decreases as $\alpha \to \infty$. The buffer size $Q^*$ increases slightly as the production rate increases because data can be accumulated more quickly with respect to the consumption rate. The buffer must be enlarged to handle the increased capacity. The value of $Q^*$ exceeds $S^*$ for large values of $\alpha$. When $\alpha \gg \gamma$, $Q^* \approx w - \lambda\gamma + \frac{S^*\alpha}{\gamma}$. Since $\frac{\alpha}{\gamma} > 1$ and $w - \lambda\gamma \approx 0$, $Q^*$ becomes larger than $S^*$. The value of $w^*$ is independent of $\alpha$ because $w^*$ is derived from $\psi(\beta)$ and $\zeta$. The expected power savings increase with $\alpha$ because the producer can sleep for a longer part of a period. Such results demonstrate that our DPM policy will become more effective in future applications. However, the savings asymptotically approach a value of 8.06 percent because an increasingly significant part of a period is occupied with draining the buffer. Therefore, a point of diminishing returns exists where increasing $\alpha$ makes no significant increase in energy savings.

Increases in the expected consumption rate $\gamma$ denote increases in video quality. Fig. 11 depicts the changes in $S^*$, $w^*$, $Q^*$, and energy savings as $\gamma$ increases toward $\alpha$. This limit exists due to Assumption 8 in Section 3.3. For this figure, we consider the IDE configuration. As $\gamma$ approaches $\alpha$, the prefetch size $S^*$ increases rapidly. When the difference between $\alpha$ and $\gamma$ is small, the buffer accumulates data slowly. Subsequently, more data is required to reach the optimal amount of sleep time for the producer. The awakening point $w^*$ increases with the change in $\gamma$ because more data is consumed during $\lambda$. To maintain a constant QoS, $w^*$ must increase. The buffer size $Q^*$ increases for small values of $\gamma$ and decreases as $\gamma$ approaches $\alpha$. Increases in buffer size are necessary to handle increased prefetch sizes at low $\gamma$. As $\gamma$ becomes larger, less data accumulates in the buffer. Thus, $Q^*$ decreases for $\gamma > 9.09$ MB/s. Energy savings are reduced as $\gamma$ approaches $\alpha$ because a larger part of a period is allocated to prefetching. Eventually, the energy savings become zero, indicating that no prefetching scheme can save energy. The other configurations exhibit similar trends. Increasing video quality requires longer prefetches, but the buffer size $Q^*$ remains feasible for personal computing devices.

As storage devices have become smaller, from 3.5″ to 2.5″ to 1″ microdrives, the amount of time to awaken them from low-power states has decreased. Fig. 12 shows the changes in $S^*$, $w^*$, $Q^*$, and energy savings as the awakening

time $\lambda$ varies. Since changes in awakening time reduce awakening overhead, we change $k$ proportionally to $\lambda$. This maintains a constant awakening power, rather than energy. For this analysis, we describe the microdrive configuration. As $\lambda$ decreases, all of the buffer parameters decrease. The awakening point $w^*$ decreases because less data is consumed during the shorter $\lambda$. Because the energy overhead for awakening the producer decreases, a shorter time is required to amortize the energy of changing power states. Consequently, prefetch size $S^*$ decreases. Because $Q^*$ is a function of both $S^*$ and $w^*$, the buffer size decreases as well. Therefore, decreases in awakening time reduce the buffer parameters to levels suitable for embedded devices as well as personal computers. Energy savings continue to increase as $\lambda$ decreases because the producer can spend a longer part of a period sleeping. However, the energy savings reach a point of diminishing returns similar to those observed for increases in $\alpha$ in Fig. 10; the power savings asymptotically approach 7.88 percent.

We use RDRAM for our study, but other shutdown-enabled memory technologies have different retention powers. Fig. 13 illustrates the sensitivity of $S^*$, $w^*$, $Q^*$, and energy savings to changes in retention energy. We focus our explanation on the microdrive. In this figure, we vary $p_{b,mb}$ by one order of magnitude above and below the RDRAM value. This figure assumes single-byte resolution, so $p_{b,mb}$ is expressed in W/MB, rather than W/bank, as in Fig. 1. As $p_{b,mb}$ decreases, the values of $Q^*$ and $S^*$ increase by a proportional factor of $\frac{1}{\sqrt{p_{b,mb}}}$. This relation is intuitive from (8) and (12). The value of $w^*$ is unchanged by varying $w^*$ because it is strictly a function of $\lambda$ and the video properties. Energy savings asymptotically approach 7.88 percent.

## 6 Conclusion

This paper presents a method for reducing energy consumption by adding a data buffer between a producer and a consumer. The method calculates optimal buffer sizes for probabilistic rates of data consumption and application-specific quality-of-service. Energy savings are compared to a reference system and heuristically determined buffer sizes for four different hardware configurations. The buffer sizes depend on hardware and data type and average 16 MB for a microdrive, a local-area network, and a wide-area network. An IDE disk requires a buffer size between 16 MB and 176 MB with an average of 80 MB. Our analytical solution is superior to heuristic-based methods for many reasons. The most significant advantage to our analytical solution is the ability to generate optimal buffer sizes without running costly simulations, allowing for rapid exploration of the design space and runtime modifications based upon data consumption rates. Since no user interaction is required other than the specification of QoS, the allocation of buffers can be performed automatically by the application or the operating system as workloads vary. Finally, we demonstrate evidence our statistically optimal DPM policy will continue to be an effective means of reducing energy consumption as technology improves.

## Acknowledgments

The authors would like to thank the reviewers for their constructive suggestions and the `mplayer-users` newsgroup for their aid in understanding MPlayer's internals. Nathaniel Pettis is supported by the US Department of Education GAANN Fellowship. Le Cai is supported by the Purdue Research Foundation. This work has been supported by US National Science Foundation Career CNS 0347466. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

## References

[1] L. Benini, A. Bogliolo, and G.D. Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management," *IEEE Trans. Very Large Scale Integration Systems,* vol. 8, no. 3, pp. 299-316, June 2000.

[2] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," *Proc. Int'l Symp. Low Power Electronics and Design,* pp. 197-202, 1998.

[3] J. Luo and N. Jha, "Static and Dynamic Variable Voltage Scheduling Algorithms for Real-Time Heterogeneous Distributed Embedded Systems," *Proc. Asia and South Pacific Conf. VLSI Design,* pp. 719-726, 2002.

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

[5] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G.D. Micheli, "Dynamic Power Management for Nonstationary Service Requests," *IEEE Trans. Computers,* vol. 51, no. 11, pp. 1345-1361, Nov. 2002.

[6] Q. Qiu and M. Pedram, "Dynamic Power Management Based on Continuous-Time Markov Decision Processes," *Proc. Design Automation Conf.,* pp. 555-561, 1999.

[7] D. Bertozzi, L. Benini, and B. Ricco, "Power Aware Network Interface Management for Streaming Multimedia," *Proc. Wireless Comm. and Networking Conf.,* pp. 926-930, 2002.

[8] L. Cai and Y.-H. Lu, "Dynamic Power Management Using Data Buffers," *Proc. Design Automation and Test in Europe Conf.,* pp. 526-531, 2004.

[9] L. Cai and Y.-H. Lu, "Energy Management Using Buffer Memory for Streaming Data," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, no. 2, pp. 141-152, Feb. 2005.

[10] N. Pettis, L. Cai, and Y.-H. Lu, "Dynamic Power Management for Streaming Data," *Proc. Int'l Symp. Low Power Electronics and Design,* pp. 62-65, 2004.

[11] C. Im, H. Kim, and S. Ha, "Dynamic Voltage Scheduling Technique for Low-Power Multimedia Applications Using Buffers," *Proc. Int'l Symp. Low Power Electronics and Design,* pp. 34-39, 2001.

[12] C.-H. Hwang and A.C.-H. Wu, "A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation," *ACM Trans. Design Automation of Electronic Systems,* vol. 5, no. 2, pp. 226-241, Apr. 2000.

[13] D. Ramanathan and R. Gupta, "System Level Online Power Management Algorithms," *Proc. Design, Automation and Test in Europe Conf.,* pp. 606-611, 2000.

[14] T. Simunic, L. Benini, P. Glynn, and G.D. Micheli, "Dynamic Power Management for Portable Systems," *Proc. Int'l Conf. Mobile Computing and Networking,* pp. 11-19, 2000.

[15] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Magazine,* vol. 40, no. 8, pp. 102-114, Aug. 2002.

[16] M. Bhardwaj, T. Garnett, and A.P. Chandrakasan, "Upper Bounds on the Lifetime of Sensor Networks," *Proc. Int'l Conf. Comm.,* pp. 785-790, 2001.

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

[18] P. Krishnan, P. Long, and J. Vitter, "Adaptive Disk Spindown via Optimal Rent-to-Buy in Probabilistic Environments," *Algorithmica,* vol. 23, no. 1, pp. 31-56, Jan. 1999.
[19] Q. Qiu, Q. Wu, and M. Pedram, "Dynamic Power Management of Complex Systems Using Generalized Stochastic Petri Nets," *Proc. Design Automation Conf.,* pp. 352-356, 2000.
[20] T. Simunic, L. Benini, P. Glynn, and G.D. Micheli, "Event-Driven Power Management," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 20, no. 7, pp. 840-857, July 2001.
[21] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "Reducing Disk Power Consumption in Servers with DRPM," *Computer,* vol. 36, no. 12, pp. 59-66. Dec. 2003.
[22] A. Ramachandran and M.F. Jacome, "Xtream-Fit: An Energy-Delay Efficient Data Memory Subsystem for Embedded Media Processing," *Proc. Design Automation Conf.,* pp. 137-142, 2003.
[23] Rambus Inc., "Rambus 128/144-Mbit Direct RDRAM Data Sheet," June 2000, http://www.rambus.com.
[24] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M.J. Irwin, "Hardware and Software Techniques for Controlling DRAM Power Modes," *IEEE Trans. Computers,* vol. 50, no. 11, pp. 1154-1173, Nov. 2001.
[25] A.R. Lebeck, X. Fan, H. Zeng, and C. Ellis, "Power Aware Page Allocation," *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems,* pp. 105-116, 2000.
[26] J. Pisharath and A. Choudhary, "An Integrated Approach to Reducing Power Dissipation in Memory Hierarchies," *Proc. Int'l Conf. Compilers, Architectures, and Synthesis for Embedded Systems,* pp. 89-97, 2002.
[27] Z. Hu, S. Kaxiras, and M. Martonosi, "Let Caches Decay: Reducing Leakage Energy via Exploitation of Cache Generational Behavior," *ACM Trans. Computer Systems,* vol. 20, no. 2, pp. 161-190, May 2002.
[28] G. Chen, R. Shetty, M. Kandemir, N. Vijaykrishnan, M.J. Irwin, and M. Wolczko, "Tuning Garbage Collection for Reducing Memory System Energy in an Embedded Java Environment," *ACM Trans. Embedded Computing Systems,* pp. 27-55, Nov. 2002.
[29] Q. Qiu, Q. Wu, and M. Pedram, "Dynamic Power Management in a Mobile Multimedia System with Guaranteed Quality-of-Service," *Proc. Design Automation Conf.,* pp. 834-839, 2001.
[30] J.S. Chase, D.C. Anderson, P.N. Thakar, A.M. Vahdat, and R.P. Doyle, "Managing Energy and Server Resources in Hosting Centers," *Proc. ACM Symp. Operating Systems Principles,* pp. 103-116, 2001.
[31] J. Flinn and M. Satyanarayanan, "Energy-Aware Adaptation for Mobile Applications," *Proc. ACM Symp. Operating Systems Principles,* pp. 48-63, 1999.
[32] F.S. Hillier and G.J. Lieberman, *Introduction to Operations Research.* McGraw-Hill, 1990.
[33] J. Watkinson, *The MPEG Handbook: MPEG-1, MPEG-2, MPEG-4.* Focal Press, 2001.
[34] Y.-H. Lu, L. Benini, and G.D. Micheli, "Dynamic Frequency Scaling with Buffer Insertion for Mixed Workloads," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 21, no. 11, pp. 1284-1305, Nov. 2002.
[35] The MPlayer Project, "MPlayer—The Movie Player," 2000-2004, http://www.mplayerhq.hu.
[36] R.C. Dorf and R.H. Bishop, *Modern Control Systems,* ninth ed. Prentice Hall, 2001.

**Nathaniel Pettis** (S'99) received the BSEE degree from the University of Arkansas, Fayetteville, in 2002. He is currently pursuing the PhD degree in electrical and computer engineering at Purdue University, West Lafayette, Indiana. His research interests include operating systems-directed power management and embedded systems. He is a student member of the IEEE.

**Le Cai** (S'03) received the BSEE and MSEE degrees from Tsinghua University, Beijing, China, in 1999 and 2001, respectively. He is currently pursuing the PhD degree in electrical and computer engineering at Purdue University, West Lafayette, Indiana. His research interests include low-power system design. He is a student member of the IEEE.

**Yung-Hsiang Lu** (S'90-M'03) received the PhD degree in electrical engineering from Stanford University, California, in 2002. He is an assistant professor in the School of Electrical and Computer Engineering and, by courtesy, the Department of Computer Science at Purdue University, West Lafayette, Indiana. In 2004, he received A US National Science Foundation Career Award for studying energy management by operating systems. His research focuses on energy management for computer systems, mobile robots, sensor networks, and image processing. He is a member of the IEEE.