Interaction-Aware Energy Management for Wireless Network Cards

Igor Crk, Mingsong Bi, Chris Gniady
Department of Computer Science, University of Arizona
Tucson, AZ 85704
icrk@cs.arizona.edu, mbi@cs.arizona.edu, gniady@cs.arizona.edu

Abstract

Wireless Network Interface Cards (WNICs) are part of every portable device, where efficient energy management plays a significant role in extending the device's battery life. The goal of efficient energy management is to match the performance of the WNIC to the network activity shaped by a running application. In the case of interactive applications on mobile systems, network I/O is largely driven by user interactions. Current solutions either require application modifications or lack a sufficient context of execution that is crucial in making accurate and timely predictions. This paper proposes a range of user-interaction-aware mechanisms that utilize a novel approach of monitoring a user's interaction with applications through the capture and classification of mouse events. This approach yields considerable improvements in energy savings and delay reductions of the WNIC, while significantly improving the accuracy, timeliness, and computational overhead of predictions when compared to existing state-of-the-art solutions.

Categories and Subject Descriptors

D.4.4 [**Operating Systems**]: Communications Management—*Input/Output; Network Communication*

General Terms

Design, Experimentation, Measurement, Performance

1. INTRODUCTION

Mobile devices have become an everyday part of our life. We depend on them for our computation, communication, and entertainment. An ever-increasing demand for performance, functionality, and better user interfaces has resulted in the demand for longer battery life. However, as advances in battery technology continue to lag behind the demands placed upon the battery, power awareness has become an important consideration in the design of mobile systems. The challenge of designing energy efficient systems lies in understanding the role of user interaction in energy consumption and in providing an energy-performance schedule that adequately accommodates user demand. Furthermore, by understanding user interaction we can optimize system performance by tailoring it to a user's patterns of interaction.

Performance and energy consumption are tightly coupled where higher performance is usually achieved at the cost of increased power demand. However, the key observation is that higher power demand does not necessarily translate into an increase in energy consumption. For instance, hardware in a higher performance state may complete a particular task faster than the same hardware operating in a lower performance state. This reduces the time during which the entire system has to be on. On the other hand, a particular device may not be required by all tasks and so may be operated in a low performance state without a significant impact on the performance of the executing application.

The challenge in designing efficient energy management mechanisms is to provide a energy/performance schedule that best matches the task at hand to transparently provide energy savings while satisfying the performance demand. Many energy management techniques have been proposed ranging from hardware optimizations all of the way to application transformation. However, most user interactions are still hidden from the existing approaches, which are unable to capture the context necessary for inferring what a user demands. Monitoring user interaction provides not only the necessary context of execution that was previously unavailable to the predictors, but also enables timely predictions before the need for high performance arrives [2]. The timely transition of a device to a desired performance/energy level is critical to meet performance demand and achieve energy efficiency.

In this paper, we show that the user interactions can be easily monitored and exploited to increase both the timeliness and accuracy of prediction mechanisms. More specifically, we apply user-interaction-based prediction to reduce energy consumption in Wireless Network Interface Cards (WNICs) while maintaining good performance levels. Subsequently, we propose and evaluate a range of prediction mechanisms that balance accuracy, energy consumption, and delay to provide energy efficient management of the WNIC. Each mechanism incorporates high-level contextual information about user's activity to predict network access patterns and provide desired energy/performance levels. The idea is motivated by observing that network traffic (for interactive applications) usually follows a specific interaction with

State	Power (W)
PSM Idle	.05
PSM Receive	.85
PSM Transmit	1.05
CAM Idle	.73
CAM Recive	.91
CAM Transmit	1.16
PSM to CAM Switch	.78
CAM to PSM Switch	.67
Transition	**Time (s)**
PSM to CAM	.28
CAM to PSM	.28

Table 1: WNIC energy consumption specifications.

the application interface. For instance, if the user is chatting with a friend in a webcam-enabled instant messaging application, when the webcam button is clicked it is reasonable to expect additional network traffic. Therefore, the proposed interaction-aware prediction mechanisms monitor user interactions with the application interface and correlate such interactions with resulting network activity to predict future levels of I/O demand. As a result, this paper makes the following set of contributions: (1) proposes a set of low-overhead user-interaction-based predictors, (2) proposes an implementation of transparent monitoring of user and network activity, (3) performs comprehensive evaluations of several proposed prediction mechanisms, (4) shows significant improvement of WNIC energy efficiency, (5) integrates multiple prediction mechanisms to mitigate variability in user interactions.

2. MOTIVATION

The challenge in designing an energy efficient system is to minimize energy consumption without sacrificing performance. Therefore an ideal solution would provide the performance level that closely matches the bandwidth demanded by the application. The IEEE 802.11 standard [12] offers two energy management schedules: Continuous Aware Mode (CAM) and Power Saving Mode (PSM). The default operational mode for wireless network interfaces in portable computers is CAM, where the WNIC is continuously active and responds to a user's requests immediately. CAM provides highest performance both in terms of lowest delay and highest effective bandwidth at a cost of high energy consumption. Alternatively, PSM periodically wakes up the network interface, switching it to a high power state at some time interval during which transmissions can occur, this is known as beaconing. Once transmissions complete, the network interface goes back to sleep. This significantly reduces power consumption [13], but also increases delays and degrades the performance of applications that demand high bandwidth [4]. The delays in network transmissions due to PSM keep the rest of the system operating for longer periods, resulting in higher overall energy consumption in the system presenting a clear need for an adaptive system.

Table 1 illustrates the energy characteristics of the Lucent Orinoco WaveLan Silver WNIC used in our study. We observe that PSM has much lower energy consumption than CAM in the idle state. Therefore, when the application does not transmit or receive any I/O or the bandwidth demand is low the WNIC should remain in the PSM mode.

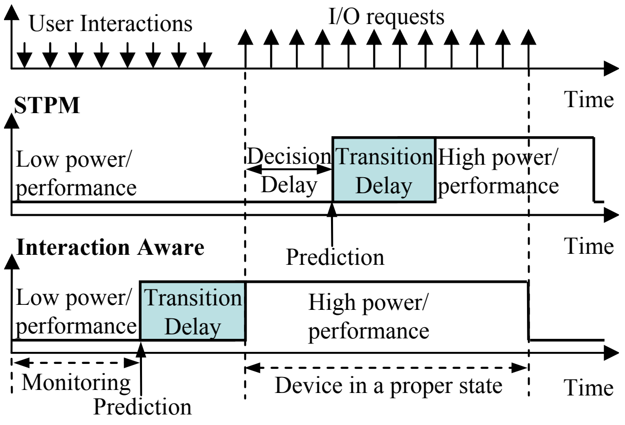

Figure 1: Prediction timeliness example.

Systems with applications showing high bandwidth demand should keep the WNIC in CAM, since transmission power of CAM and PSM are comparable and CAM offers much higher bandwidth. Furthermore, the delays associated with PSM can extend the running time of the application and result in higher energy consumption than CAM. However, selecting the WNIC mode is not simple, since an application may go through multiple states with varying bandwidth demand. Keeping the WNIC in a single state either wastes energy, as is the case with CAM, or increases transmission delays and potentially increases energy consumption, as in the case of PSM.

To accommodate different application behaviors, many cards provide automatic switching between CAM and PSM, based on the amount of traffic observed. Our WaveLan card allows the network interface to remain in the PSM while monitoring for traffic from the access point. If more than one packet is waiting at the network interface, the WNIC is switched to CAM. The WNIC is switched to PSM once the transmission stops. This simple solution provides energy savings for low transmission rates while providing good performance for applications demanding high bandwidth. However, certain cases render this solution ineffective. First, communication patterns that exchange one packet at a time will keep the WNIC in PSM even if the applications require high bandwidth [4]. The pathological case for PSM occurs with NFS directory listings, which can result in a 16-32x slowdown as compared to CAM [4]. Second, small and bursty transmissions may unnecessarily switch the WNIC into CAM. Therefore, more sophisticated mechanisms are needed to better adapt WNIC power mode switching to reflect an application's performance demand.

To match application demand while saving energy, the network card should be transitioned into CAM when high bandwidth is demanded and remain in PSM when the bandwidth demand is low. This observation motivated the development of Self-Tuning Power Management (STPM) [4, 3] to dynamically switch between CAM and PSM. STPM relies on programmers for inserting accurate hints about upcoming bandwidth demand into the application. As a result, STPM can be very accurate, as the programmer knows what part of the applications is executing and perhaps what kind of demand can be expected from the user. However, this introduces power management as another optimization dimension to already difficult programming requirements that target, among other things, performance, reliability, and usability. To provide energy savings for unmodified applications, STPM profiles network traffic to anticipate future network activity [4]. Passively monitoring low-level network access provides little information about user's current context, how the user is interacting with the application. In this case, the context of execution of an application is completely lost and heuristic profiling is not able to fully realize the potential of STPM.

Context of execution is critical to providing accurate and timely switching between power states to match the application demand. Figure 1 shows an example of user interactions with an application and the impact of STPM and user Interaction-Aware Prediction (IAP) mechanisms on transition timeliness. As a result of user interactions the application initiates network I/O activity. STPM evaluates initial I/O requests to verify the need for higher bandwidth and the network interface is switched to the high performance mode when the need for high bandwidth is detected. We observe that both the evaluation period and the mode transition period impact the performance of high bandwidth transfers and may lengthen the time spent serving the I/O requests [4].

We observe that we need hints ahead of time, in order to transition a device to a higher energy state before I/O requests arrive. These hints are difficult to obtain at the operating system level, so the need for a higher context of execution is clear. Since users are responsible for the majority of I/O activity in interactive applications, the natural approach is to observe user interactions and infer from interactions when the increase in performance demand will arrive. Therefore, in this paper, we propose several IAP mechanisms and explore in detail the monitoring and prediction of user activity in order to improve prediction timeliness and accuracy due to the added context of execution. The IAP mechanisms continuously monitor user activity and predict the need to transition the device to a higher power level in time to meet the increased performance demand, as shown in Figure 1.

3. INTERACTION-AWARE PREDICTION

The key assumption here is that there exists a strong correlation between user interaction with the application and resulting network activity. Further, this correlation can be transparently exploited to manage the power states of wireless interfaces. Subsequently, we face several requirements:

- User interactions have to be captured transparently without modification to the application.
- User interaction correlation and classifications should be performed online without any user involvement.
- Capture and prediction have to be efficient to prevent excessive energy consumed by the CPU to train and generate predictions.
- The system has to be able to handle multiple applications in a graphically rich environment.

3.1 The Naïve Predictor

The observation that within interactive applications, network activity is largely caused by the user's interactions leads us to a proposal of a simple mechanisms that switches the WNIC to CAM for every mouse click. This approach was previously explored for managing the power states of a processor [16]. The intuition that motivates the naïve mechanism is that if the user invoked some I/O that followed a mouse click it is probably important and should be served with least amount of the delay. On the other hand, network activity that is not correlated with, or immediately preceded by, mouse clicks is probably less important and can incur some delays without degrading the applications interactive performance. Therefore, our naïve all-click (AC) switching mechanism transitions the WNIC to CAM upon each click and transmits all network I/O that follows the last mouse click within some time interval in CAM, thereby minimizing delays by already being in the high-power mode before I/O arrives and serving data with higher throughput than was measured for PSM. Network I/O that is not preceded by mouse activity is served in PSM, saving energy. Once the network I/O is served, the WNIC is transitioned to PSM to conserve energy during idle periods.

Clearly, this mechanism has the effect of capturing and serving all I/O activity periods preceded by clicks in CAM, with the downside of transitioning the WNIC unnecessarily for clicks that are not followed by network I/O. It is important to note that the total number of clicks may greatly exceed the total number of I/O activity periods, since most of the clicks do not invoke network I/O. In addition, even if there is network activity following a click, it may be small and best served in PSM without the overheads of transition to CAM. The energy consumed in CAM following each switch, the energy required to make each switch, and the energy consumed during unnecessary switches make AC one of the least energy efficient mechanisms. The AC mechanism illustrates the need for more intelligent prediction schemes that decide when the WNIC should stay in PSM or transition to CAM for the upcoming I/O activity period.

3.2 Capturing GUI interactions

To address the shortcomings of the naïve predictor, which does not distinguish between different types of potential interactions with the application, we propose a more sophisticated Interaction-Aware Predictor (IAP) approach that utilizes a detailed context of user interactions to accurately predict WNIC transmission modes. Accurate and detailed monitoring of user activity forms the basis for IAP's design. Virtually all interactions with common interactive applications in a GUI environment can be accomplished through mouse clicks [6]. While the capture of mouse click data, such as absolute screen coordinates of the event or type of click, is relatively trivial, capturing application-specific context using mouse clicks is more problematic. In order to uniquely identify the components of application GUIs that the user is interacting with, we developed a monitoring layer between the X Window Server and applications in Linux that utilizes the GUI window structure to uniquely identify interactive components such as buttons and menu selections with an integer ID. Figure 2 shows the monitoring layer and the kernel structure used to record interaction IDs. This layer provides us with transparent user interaction monitoring, meaning that no modification of applications is necessary. Furthermore, all interaction IDs are obtained while the user is interacting with the applications, allowing for detection, correlation, and prediction to be performed without any offline processing. High prediction accuracy is achieved by use of the hierarchical trees of visible and non-visible windows that fully describe an application's GUI. The structure of the window tree is the same across executions and is used to uniquely identify a particular event.

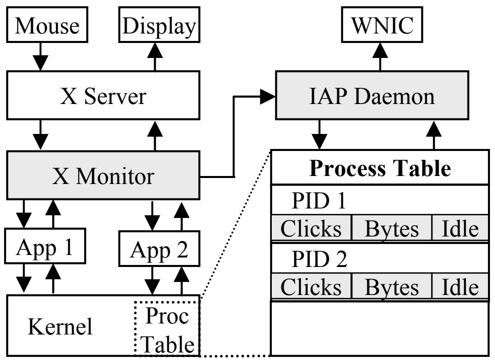

Figure 2: Design architecture.

3.3 Advanced Prediction

Accurate and detailed description of the user interactions allows the IAP to distinguish between different types of user interactions with an application that generate various amounts of network I/O. The central part of the IAP is a prediction table that is organized as a hash table indexed by the unique click IDs. The prediction table stores a 2-bit counter that indicates the most appropriate WNIC mode for a given click ID. The table resides globally in the IAP daemon and is shared among processes to allow table reuse across multiple executions of the same application as well as concurrent executions of the same application. The table can be easily retained in the kernel across multiple executions of the application due to its small size [9, 8], and will reduce training in future invocations of an application.

A prediction table lookup is performed for every click, as shown in the Figure 3. The lookup results in three possible outcomes: (1) the entry is found and the 2-bit saturating counter indicates that the interaction leads to high levels of network activity (best served in CAM), (2) the entry is found and the 2-bit saturating counter indicates that the interaction leads to low or no network activity (best served by PSM), or (3) the entry is not found, in which case a placeholder in the prediction table is created. Based on the lookup outcome the IAP daemon switches the network interface to the appropriate state. The IAP daemon continues to monitor the network activity recording the number of bytes transferred and the time of the I/O activity for each process as shown in Figure 2. Arrival of a new click during the monitoring indicates that there is a new interaction with the application and the prediction table entry for the previous click has to be updated. Detection of a longer idle period indicates that the network activity initiated by the current click ceased, the WNIC can be transitioned from CAM to PSM and the prediction table entry for the given click can be updated.

To update the 2-bit counter in the prediction table we rely on the information about the given activity collected by the IAP daemon and recorded in the process table as shown in Figure 2. The IAP daemon calculates the relative time and energy cost of transmitting the observed network I/O in PSM and CAM relying on equations proposed in [3]. The energy cost of each mode includes: (1) the WNIC energy to transmit the data in a given mode, (2) energy consumed in the idle state waiting for the end of the activity signaled by the arrival of the click or detection of a longer idle period, and (3) the energy consumption of the overall system, since

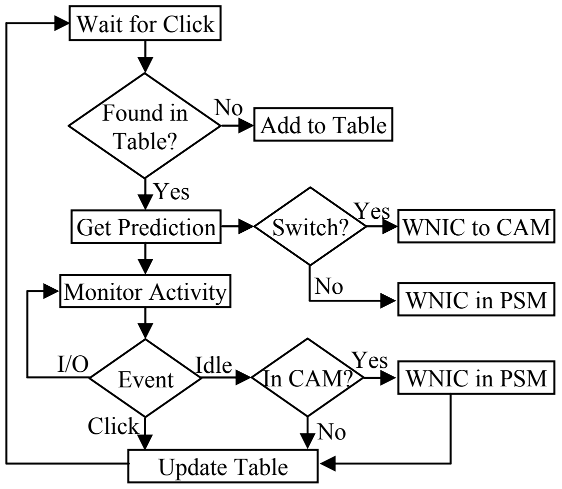

Figure 3: IAP decision flowchart.

it has to remain on when the request is served. The time cost of serving the observed activity in CAM includes the transfer time and the time to transition the WNIC from PSM to CAM. The time cost of serving the observed activity in PSM includes the transfer and the initial latency of receiving data, which is on average one-half of the beacon frequency. Transfer time in PSM is higher than in CAM because throughput is lower in PSM mode. Ignoring the switching latency, a request served in CAM will always complete earlier than if it were served in PSM.

The relative costs of time and energy for every predicted I/O period can be weighted for emphasis on either timely transitions or energy-efficient completion of the requests. In our design, we consider a balanced approach of minimizing delays and energy for completing each activity period to minimize the energy-delay^2 product. The mode with the lowest energy-delay^2 product is selected as the best choice for completing the request and the 2-bit counter in the corresponding prediction table entry is updated. We selected the energy-delay^2 to slightly emphasize the delay reduction in interactive applications.

Finally, we observe that we need energy profiles of both the WNIC and the system. In our case, we obtained the respective profiles prior to our implementation, but we envision that eventually, since energy management continues to increase in importance, the devices themselves will provide energy profiles. In addition, we focused on the PSM to CAM transitions above, leaving the CAM to PSM transition to a simple timeout, making it much easier to directly compare all the proposed mechanisms with traditional and state-of-the-art approaches.

3.4 Mode Switching Heuristics

There are two decisions that IAP has to make upon retrieving the prediction from the table. First, what to do when there is no prediction for a given click. This occurs during training and becomes less important as time passes since there is only a limited number of ways the user can interact with the application. The second decision is when the prediction should be acted upon. IAP can immediately switch to CAM, if predicted to do so, or delay the switch till the network I/O arrives. Both of those questions trade aggressiveness of prediction with higher reduction in delay

	Prediction policy	**Switches ahead of I/O**	**Energy savings**	**Delay reduction**
AC	Switches to CAM for each click	Yes	✓	✓✓✓✓✓
IAPD	Serves all I/O in CAM unless predicted to serve in PSM	Yes	✓✓	✓✓✓✓
IAPE	Serves all I/O in PSM unless predicted to serve in CAM	Yes	✓✓✓	✓✓✓
IAPDW	Same as IAPD	No	✓✓✓✓	✓✓
IAPEW	Same as IAPE	No	✓✓✓✓✓	✓
IAPED	Same as IAPE	Dynamic	✓✓✓✓✓	✓✓

Table 2: Summary of policies and qualitative comparison of efficiency.

for higher accuracy with the potential for higher energy savings. As a result, we evaluate several variations of the IAP approach: IAP with Delay optimization (IAPD), IAPD with deferred switching/Waiting to switch (IAPDW), IAP with Energy optimization (IAPE), IAPE with deferred switching/Waiting to switch (IAPEW), and IAP optimizing the Energy-Delay^2 product (IAPED). The basic mechanisms for context capture and prediction are shared across all heuristics, but the actions taken upon prediction vary significantly from one to the next. An overview of each mechanism is contained in Table 2. Table 2 is a brief summary of switching policies, stating when the switch occurs, and the relative benefit for energy and delay reduction. Checkmarks in energy and delay savings columns compare the relative impact on energy and delay for each of the proposed mechanisms.

3.4.1 IAPD & IAPDW: Minimizing Delays

The goal of IAPD and IAPDW mechanisms is the reduction of transmission delays by serving network I/O in CAM by default and in PSM only when the prediction dictates to do so. IAPD and IAPDW switch the WNIC to CAM for: (1) mouse click IDs that have not been seen before, (2) mouse clicks for which IAPD is still training, (3) mouse clicks that result in CAM prediction, and (4) any activity not preceded by mouse clicks. The main difference between IAPD and IAPDW is that with IAPD transitions from PSM to CAM occur immediately upon prediction from a given mouse click, while IAPDW defers the same transition until network activity arrives following the click. IAPD therefore has the additional benefit of reducing switching delays. The WNIC remains in PSM during idle periods and will remain in PSM after a click when these mechanisms predict that the I/O activity following the click is small and best served in PSM.

In the case of IAPD, delays are reduced at the cost of higher energy consumption, since IAPD switches for all clicks where prediction is uncertain. Both mechanisms incur additional energy consumption due to switching for unpredictable periods, which may be efficiently served in PSM. Figure 4 illustrates the switching differences for IAPD and IAPDW. In the example, IAPD switches the WNIC to CAM when click with ID 1 is seen the first time, but remains in

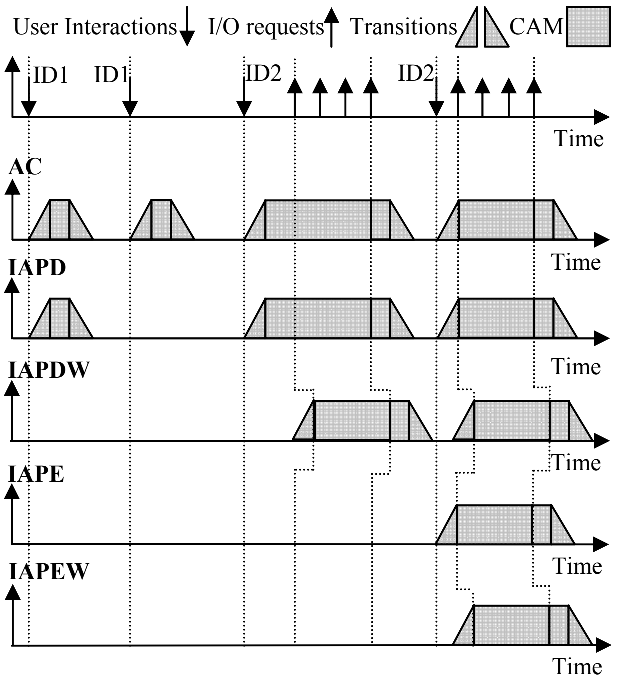

Figure 4: Training and prediction example.

PSM when ID 1 is subsequently seen again, since it was not previously followed by network I/O. When click with ID 2 arrives, the mechanism transitions the WNIC to CAM correctly, subsequently using the prediction to again transition the WNIC to CAM correctly when ID 2 is seen again. IAPDW waits for the network I/O to arrive and does not switch for clicks with ID 1 since they are not followed by the I/O but switches for the network activity following the click with ID 2. Furthermore, Figure 4 shows that IAPDW slightly delays the network transmissions due to switching delay.

3.4.2 IAPE & IAPEW: Maximizing Energy Savings

IAPE and IAPEW mechanisms differ from IAPD and IAPDW in how the predictions are utilized in transitioning the WNIC power modes. Specifically, where IAPD and IAPDW favor CAM for unpredicted or uncertain network I/O, IAPE and IAPEW default to PSM unless the prediction is that the upcoming network I/O is best served in CAM. Therefore, IAPE and IAPEW eliminate unnecessary transitions to CAM during unpredictable periods of low network I/O traffic that can be served efficiently in PSM. The resulting behavior maximizes energy savings at the cost of higher delays. As in IAPD, IAPE transitions the WNIC from PSM to CAM immediately upon prediction following a mouse click in order to reduce switching delays. Similarly to IAPDW and its relationship to IAPD, IAPEW is identical to its counterpart, IAPE, but the predicted transition from PSM to CAM is deferred until I/O activity arrives. IAPEW reduces unnecessary switches due to erroneous predictions and reduces the time WNIC is waiting in CAM for incoming I/O, making it the most energy conscious predictor. However, this is done at the cost of introducing switching delays for each predicted PSM to CAM transitions.

We use Figure 4 to further illustrate the differences between the proposed mechanisms. IAPE and IAPEW default

Application	Trace Length	MB Sent	MB Recv.	I/O Act. Periods	# CAM Periods	# PSM Periods	Total Clicks	Unique Click IDs	Correlated Click IDs
Firefox	37 hrs	43	692	3643	791	2852	3113	128	83
Thunderbird	20 hrs	407	434	632	185	447	2175	32	14
GFTP	12 hrs	339	2140	565	74	491	2992	34	18
Gaim	12 hrs	459	254	1085	137	948	2253	59	31
Pan	15 hrs	64	229	1302	46	1256	7316	50	20
DJGame	30 hrs	281	164	630	214	416	6189	41	13

Table 3: Application execution statistics.

to PSM when a prediction is lacking or uncertain. Therefore, there are no switches for the click with ID 1 and the first I/O period, since click with ID 2 has not been encountered previously. The next time ID 3 is seen, the prediction to switch to CAM is made correctly. Additionally, IAPEW, like IAPDW, does not immediately request that the WNIC transition to CAM, but does so only when the first network I/O is encountered following a prediction.

3.4.3 IAPED: Minimizing Energy-Delay^2

The challenge of designing energy efficient predictors is the balance between reducing delays and reducing energy consumption, which can be accomplished by minimizing the energy-delay^2 product. A detailed evaluation of the mechanisms proposed in the previous sections found that IAPE and IAPEW consistently performed better in this respect. This implies that utilizing the CAM mode during training and transmissions not preceded by mouse clicks is less significant than minimizing delays and wrongful switches. As a result, we propose an additional IAP that minimizes the Energy-Delay^2 (ED) product (IAPED). We selected IAPE and IAPEW, the top performers in Figure 8, to design the IAPED predictor that dynamically selects between the combined predictors in an attempt to minimize the energy-delay^2 product. Training and predictions are done according to the diagram shown in Figure 3. The only additional step needed is deciding if waiting for the I/O or switching ahead of time is more beneficial for the energy-delay^2 product, given the activity period that has just ended. We use energy-delay^2 calculations from each prediction mechanism and if switching ahead of time is more beneficial, an additional 4-bit saturating counter is incremented for each click where this is the case, otherwise it is decremented. Once the prediction to switch to CAM is made, the IAPED consults the 4-bit saturating counter for each click ID and decides if the switch should occur immediately or when I/O activity arrives.

We decided to bias the saturating counter to IAPEW performance during training, since, as it will be shown in subsequent sections, IAPEW outperforms IAPE in terms of the energy-delay^2 product for all applications except *Firefox*. As a result, the initial value of the saturating counter is set to 0 and will require two incorrect choices for each individual mouse click ID before IAPED starts performing like IAPE since this can potentially reduce the energy-delay^2 improvement for applications where IAPEW is the best performer.

4. METHODOLOGY

To accurately compare the various energy management techniques for WNICs we use a trace-driven simulation. We implemented: (1) standard power modes: PSM and CAM; (2) interaction-aware mechanisms: AC, IAPD, IAPDW, IAPE, IAPEW, and IAPED, (3) existing state-of-the art STPM mechanisms, according to the authors' specifications [4], (4) optimal power management mechanism that minimizes the energy-delay^2 product based on future knowledge (OPT), and (5) another reference predictor that switches the WNIC to CAM for each network I/O period (AN). We use the AN mechanism to show the other side of the spectrum where all I/O is served in CAM to minimize transmission delays. This mechanism is similar to the hardware PSM to CAM switching mechanism that switches the card to CAM if there is more than one packet waiting at the network interface. Through experimentation, we have found that an timeout of 4 seconds used to delay the transition of the WNIC from CAM to PSM results in the best energy-delay^2 product for studied mechanisms.

Application traces are composed of two components: the mouse event trace, and the network activity trace. Mouse events are traced using the X monitoring layer. The trace includes the mouse interaction type, a timestamp, and the unique ID that identifies the component with which the user has interacted. Network activity traces are collected using a modified *strace* that captures the system call information from *send* and *recv* interfaces, such as type of request, timestamp, bytes send, and bytes received. The simulator relies on timestamps from each trace to order the incoming events.

We use six applications commonly executed on desktop or portable systems:

- *Firefox* is a widely used web browser and represents the behavior of users surfing the web, reading news articles or downloading files among other activities. Browsing web pages generally results in a low network activity while downloading files will generate high bandwidth demand. Some page browsing activities may invoke external plug-ins to access media content within a page such as flash animations or movies that also may require high bandwidth.
- *Thunderbird* is an example of a mail application where users interact with the application to perform tasks such as sending, receiving, reading and composing email messages. These interactions generate varying amounts of network traffic due to sending or receiving emails with or without attachments. Transferring attachments will usually require much higher bandwidth than transferring a plain text messages.
- *Gaim* is an Internet messaging application. It generates low bandwidth network traffic when users are sending text messages, while occasional file transfers performed by users require high bandwidth.
- *GFTP* is a file transfer client used to upload and download files between the client and a server. The traffic consists of low overhead directory listings and high overhead file transfers.
- *Pan* is a newsreader application, where user interac-

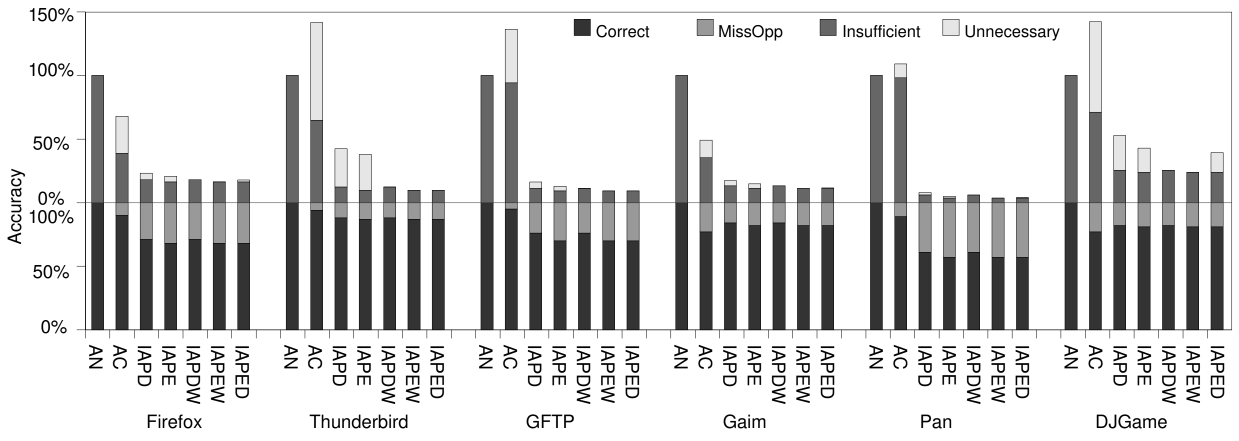

Figure 5: Normalized prediction accuracy.

tions result in low network traffic generated by connecting to a news server, getting lists of groups and message headers, getting messages, and posting messages.

- *DJGame* is an interactive online game that initiates network activity for each action within the game, with some actions leading to the transfer of information regarding user actions and others leading to the larger transfers of game-state related information.

Trace details such as trace duration, the amount of data transferred, and the number of network I/O activity periods representing opportunities for switching the WNIC power mode are included in Table 3. Also included are the numbers of user interactions, represented by mouse clicks. Unique click IDs serve as a measure of GUI complexity for each application. Applications with relatively simple static GUIs, such as *Thunderbird* or *GFTP*, have few unique click IDs, while those with rich interfaces have a somewhat higher number of unique IDs. Correlated click IDs are a subset of unique click IDs that belong to clicks that precede network activity and can be correlation to the network I/O.

5. EVALUATION

We evaluate the efficacy of our proposed mechanisms along several dimensions. First, we consider the prediction performance of the proposed predictors. Second, we compare the delay and energy consumption of all mechanisms to identify the various sources of performance degradation and energy consumption. Third, we consider the relative energy-delay^2 performance of each mechanism, which shows a combined view of predictor efficacy. Finally, we compare the overheads of IAP mechanisms and the state-of-the-art STPM mechanism.

5.1 Accuracy

Figure 5 shows a breakdown of prediction outcomes for the studied mechanisms and combines two related metrics: one for the correct CAM periods and the other for wrong switches, both normalized to AN. The Correct portion of the bar represents the number of correctly predicted CAM periods, while Missed Opportunity represents the number of CAM periods that were missed by the predictor and were subsequently served in PSM. Incorrect switches are divided into Inefficient and Unnecessary switches. The Unnecessary switches occur when the WNIC was switched to CAM upon a prediction but the WNIC did not serve any I/O and the card was transitioned back to PSM. Unnecessary switches waste energy without providing any benefit in reducing delays. Inefficient switches, on the other hand, occur when the WNIC switches to CAM and serves some I/O but the I/O activity is insufficient to justify serving it in CAM. Inefficient switches are less energy efficient but reduce some delays by serving I/O in CAM. Therefore, they are less damaging to the resulting energy-delay^2 product than Unnecessary switches. We use AN as the base for Figure 5 since it shows maximum potential for serving I/O periods in CAM and also the maximum amount of network I/Os that can be served incorrectly in CAM if more sophisticated prediction mechanisms are not employed. We use the term coverage here to indicate the percentage of the correcly predicted CAM periods to the total CAM periods.

As shown in Figure 5, AC covers 87% of CAM periods on average, with the best case being 95% in *GFTP*. CAM period coverage is on average 77% for IAPD and IAPDW; and 74% for IAPE and IAPEW. There is no difference in the converges for the IAPDW and IAPEW as compared to their base cases since the prediction are the same, only the timing of switching is different. The average improvement of wrongful switches for IAP mechanisms over AN is on average 77% for IAPD, 79% for IAPE, 82% for IAPDW, 84% for IAPEW, and 82% for IAPED. The lower wrong switches in IAPDW and IAPEW mechanism are due to delaying switches after prediction. Slightly higher coverages for IAPD mechanisms are due to handling network I/O in CAM during initial training of the predicate when the mouse clicks are observed for the first time.

The naïve AC mechanism switches for all clicks but does not switch for the I/O not preceded by click. Switching for all clicks explains the higher coverage in *Firefox*, *GFTP*, and *Pan*, at a cost of much higher incorrect switches. I/O periods that were not immediately preceded by clicks are served in PSM, since according to the intuition behind AC design they are not latency sensitive. This explains slightly lower

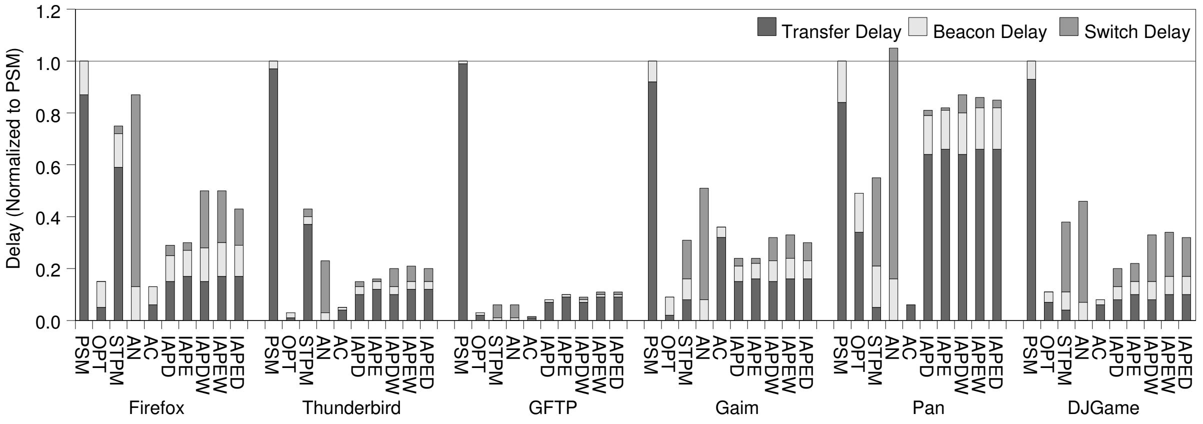

Figure 6: Details of delays incurred by predictors.

coverage in case of *Gaim* and *DJGame* for AC. The periods that are not immediately preceded by clicks are found in *Thunderbird*, *Gaim* and *DJGame* and are due to delays generated by the server providing the data. Those delays can result in the fragmentation of long network I/O periods into several smaller network I/O periods. This delay results in the CAM to PSM transition timeout to expire and AC serves the remaining network I/O in PSM. IAP mechanisms generate persistent predictions that will serve any I/O in the last predicted state until a new prediction is generated at the next mouse click. In this way, fragmented periods are served in a correct WNIC mode, albeit incurring a small amount of switching delay for each switch occurring at the arrival of a new I/O period fragment, if predicted mode of the WNIC is CAM. This policy is responsible for better coverage by IAP mechanisms by an average of 6% in *Gaim* and 5% in *DJGame*.

Lower coverages in *Firefox*, *GFTP* and *Pan* are due to variability in transfers following certain mouse clicks. Those applications are dominated by the PSM periods where most interactions are efficiently handled in PSM. We observe that IAP mechanisms are able to correctly handle PSM periods by having very small fraction of periods that should be handled in PSM but are instead handled in CAM. Occasional interactions that usually result in low bandwidth demand may require higher bandwidth to accomplish, resulting in the predictors missing an occasional CAM switch. Those scenarios are observed in *Firefox* where fetching pages usually requires low bandwidth, while some pages contain rich graphical content with high resolution graphics occasionally requiring higher bandwidth to load them. Similarly, *GFTP* requires high bandwidth for file transfers and usually low bandwidth for retrieving directory listings. However, some large directories will require higher bandwidth to transmit a very long list of file names with corresponding information. Finally, *Pan* retrieves a majority of messages without any attachments, generally requiring low bandwidth that is accurately handled in PSM. Occasional downloads of the message with an attachment will result in lower coverage for the CAM periods.

The goal of the IAP design is to handle I/O periods accurately to minimize energy-delay^2 product. Much higher accuracy is achieved at the cost of reduced CAM coverage, but the ultimate benefit to reduction of energy-delay^2 product is achieved. This is illustrated in case of AN and AC with both mechanisms having higher coverage of CAM periods than IAP mechanisms. However, both AN and AC mechanism are plagued by incorrect handling of PSM periods, where IAP mechanisms on average improve wrongful switches by 80%.

5.2 Delay

Figure 6 breaks down the various delays incurred by each mechanism. The three types of delay shown are Transfer, Beacon, and Switch delay. Since CAM incurs minimum delays, the delay of each mechanism is computed relative to CAM. Transfer delay occurs whenever data is served in PSM and is due to the lower throughput rates of PSM. Beacon delays are caused by the delays that occur due to the WNIC downtime between transmitting beacon frames, where serving data following a beacon is delayed by the time remaining before the next beacon arrives. Switch delay is the delay incurred by transitioning the WNIC power modes and is equal to the time to transition the card from PSM to CAM or vice versa. All results are normalized to the PSM delay, showing the deficiencies of the existing hardware mechanisms.

The delay that results in maximum energy efficiency is show by the energy-delay^2 optimal mechanism, shown as OPT in Figure 6. OPT does not have any switching delay since it transitions the WNIC to CAM prior to the arrival of I/O activity. The delay incurred by OPT is mainly due to serving periods in PSM, that minimize energy-delay^2 product. Similarly to OPT, AC eliminates switching delay by switching upon a mouse click, ahead of network I/O. However, lower than optimal transfer delay indicate that AC sacrifices energy efficiency at the cost of lower delay. AN eliminates transfer delay altogether since it is serving all I/O in CAM. However, it switches from PSM to CAM when I/O arrives, encountering switching delays for the I/O periods that could be served more efficiently in PSM without a switch. Delays, due to the decision-making process in STPM, result in the network interface serving more I/O in PSM, encountering transmission delays even for the periods

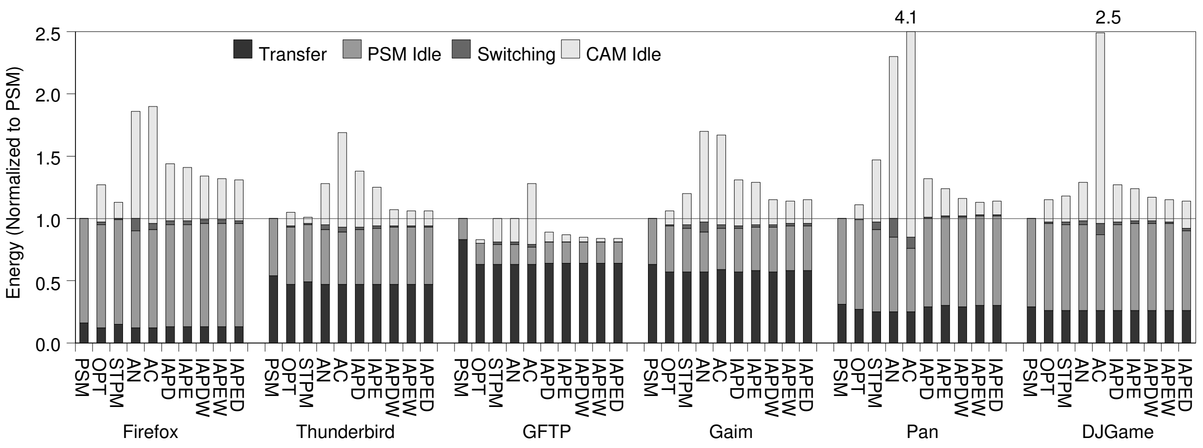

Figure 7: Detailed breakdown of energy consumption results.

that are later switched to be served in CAM. Regardless, STPM improves delay by 59% with respect to PSM. On average IAP mechanisms further improve delay over PSM by 71% for IAPD, 69% for IAPE, 62% for IAPDW, 61% for IAPEW, and 63% for IAPED. It is important to note that the optimization of energy-delay^2 product by each successive variation of the IAP mechanisms results in increasing delay as this is the cost of reducing the energy consumed by incorrect or early switches. IAPED shows the result of the interplay of the two mechanisms that have been combined.

The IAPDW and IAPEW mechanisms were proposed to reduce the energy consumed by unnecessary switches and also reduce the time spent in CAM waiting for the I/O to arrive. Therefore, the only difference for those mechanisms is visible in switching delay, since both IAPDW and IAPEW wait for the I/O to arrive before switching the WNIC mode. *Firefox* shows a significant difference in delay between IAPD and IAPDW, as well as between IAPE and IAPEW, that is driven by switching delay. Switching the WNIC to CAM when the prediction is made, as in the case of IAPD and IAPE, results in a lower switching delay in situations where the transition prior to network activity does not time out before I/O arrives. The switching delay that is present in the case of IAPD and IAPE is due to persistent predictions transitioning the WNIC to CAM once I/O activity arrives after the idle time threshold and the card is switched again from PSM to CAM. For example, the deferred switches comprise about 16% of all switches in *Firefox* for IAPD and IAPE.

Finally, variations in transfer, switching, and beacon delays across the applications are due to the composition of an application's I/O activity. Applications with a large amount of low bandwidth activity will incur higher transmission delays as compared to CAM, since those periods are more efficiently served in PSM. The same applications will also incur beaconing delays due to the PSM mode. The example of this behavior in Figure 6 is *Pan* with 96% of all periods that are best served in PSM. Applications with relatively high number of CAM periods or activity that is hard to predict may incur a higher fraction of switching delay, with *Firefox* serving as an example.

5.3 Energy

Energy consumption in Figure 7 is divided into four components: PSM idle, CAM idle, switching, and transfer energy. PSM idle energy is consumed by the WNIC when it is in PSM and there is no I/O transfer occurring. It includes energy consumed during beaconing as well as the base PSM energy. CAM idle energy is the energy consumed when the WNIC is in CAM but not transferring data. Switching energy is the energy consumed by the WNIC to make the transition from CAM to PSM and vice versa. Finally, transfer energy is the energy consumed while transferring data, either while in CAM or PSM. Similarly to Figure 6, all results are normalized to the PSM energy consumption.

PSM on average consumes the least amount of energy, at the cost of increased delays as shown in Figure 6. Energy consumption below the optimal level again suggests that the mechanism is not efficient when we consider the energy-delay^2 product. The most energy-hungry mechanism is either AN or AC, depending on the application. The behavior of AN and AC, shown in Figure 7, mirrors their behavior in Figure 5, showing that the large number of incorrect switches is responsible for the increased energy consumption due to switching, and subsequently the energy consumed waiting for the timeout to expire before switching back to PSM. We can conclude that accuracy of the prediction is critical to energy consumption. Higher switching accuracy of STPM, which only switches when it encounters network I/O with sufficiently high bandwidth demand, results in much better energy efficiency than the naïve AC and AN mechanisms. Higher accuracy in IAP mechanisms also translates to better energy efficiency. On average, STPM consumes 9% more energy than OPT, IAPD 16%, IAPE 13%, IAPDW 4%, and IAPEW and IAPED only 3% more.

Energy consumption due to data transfers is similar across mechanisms for a given application since all mechanisms have to transfer the same amount of data and transfers in CAM and PSM use approximately the same power, as shown in Table 1. The largest difference is observed in *GFTP*, which can be attributed to large transfers which when performed in PSM take more time and as a result consume more energy. This difference stands out most in *GFTP* since, on

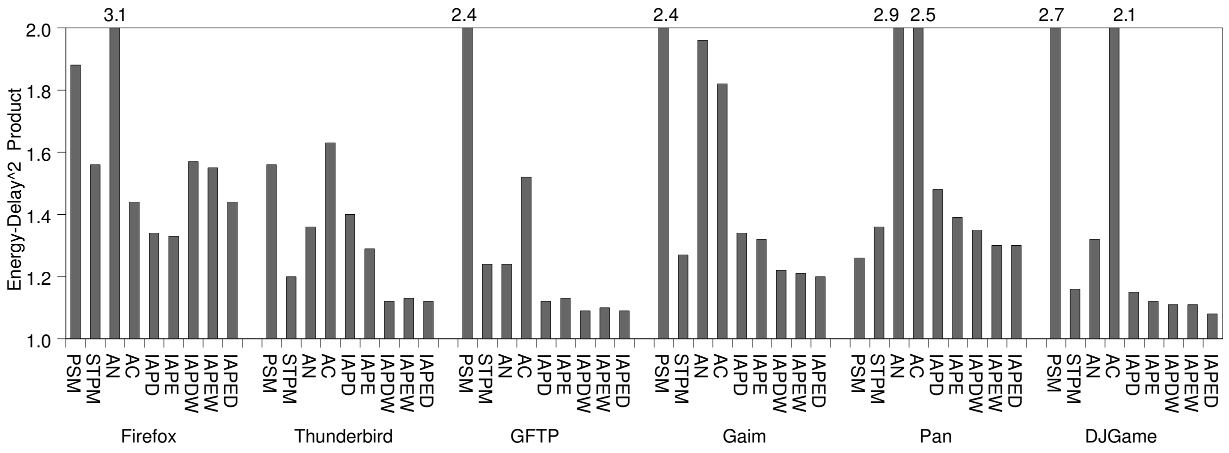

Figure 8: Energy-delay^2 product results.

	OPT		PSM		STPM		IAPED	
	Time (s)	Energy (J)	Time (s)	Energy (J)	Time (s)	Energy (J)	Time (s)	Energy (J)
Firefox	2132	19655	3298	15470	2954	15932	2515	20352
Thunderbird	3068	7296	3930	6955	3428	6995	3226	7413
GFTP	7215	9215	10250	10954	7371	10959	7514	9321
Gaim	2856	5773	4522	5447	3021	6544	3027	6192
Pan	1104	4076	1313	3641	1128	5320	1251	4141
DJGame	1800	8524	3199	7342	1915	8759	1889	8373

Table 4: Energy and delay costs for selected mechanisms.

average 71% of energy consumed in *GFTP* was due to transferring large amounts of data, which is four times more than the average of the remaining applications. The smallest energy consumption during transfers occurred in *Firefox* with an average of 10% of total energy, while PSM idle makes up the bulk of energy consumption at an average of 61%. As a result, those differences in transfer delays are less prominent in other applications.

The largest differences among energy management mechanism are visible in switching energy and CAM idle energy. Every erroneous switch increases switching energy consumption and idle CAM energy consumption while waiting for I/O to arrive and idle timeout to expire. Furthermore, even correct switches can contribute to idle energy consumption in the case when they have to wait for I/O to arrive. IAPE consumes less energy than IAPD, since it has fewer erroneous switches as shown in Figure 5. The energy consumed while waiting for I/O to arrive is further reduced in IAPDW and IAPEW since those mechanisms wait for I/O to arrive before switching to CAM. The only energy consumed in idle CAM state in IAPDW and IAPEW is due to timeout interval after an I/O has been served and before the WNIC was transitioned to PSM.

5.4 Energy-Delay^2

The final metric in evaluating each of the energy management mechanisms is the energy-delay^2 product. The energy-delay^2 product is an established metric that takes into account both performance degradation and energy consumption, with an emphasis on performance degradation, due to the interactive applications' requirements for low-delay operation. Energy-delay^2 provides a single metric and therefore is useful for visualizing the tradeoff between energy consumption and performance. Figure 8 shows the energy-delay^2 product for each mechanism as normalized to OPT. OPT energy-delay^2 product is at 1 and the increases in energy-delay^2 product as compared to OPT are above 1. To compute the energy-delay^2 product, we consider the total WNIC energy consumption for each of the traced applications using each of the described mechanisms and total transmission time, which includes all delays and time to transmit data, but does not include the idling time, which may vary according to user behavior. For additional reference, we include Table 4, which shows the transmission time and energy, in seconds and Joules, respectively, for OPT, PSM, STPM, and IAPED.

We observe that the IAPED mechanism outperforms AC and AN by an average of 75% and 60%, respectively, and is 11% lower than STPM. Of the IAP mechanisms, IAPD and IAPE are generally the weaker performers, with an average energy-delay^2 product that is 31% and 26% higher than OPT, respectively, due to larger number of erroneous switches and resulting higher energy consumption. The exception is *Firefox* where we see that the trend is reversed, where IAPD and IAPE perform better than their energy-optimizing counterparts do. *Firefox* overall exhibits a larger amount of activity than the other applications. In this case, the switching delay incurred by IAPEW and IAPDW is largely eliminated by the early switching of IAPE and IAPD. In *Thunderbird* and *DJGame* the pronounced differences between IAPD and IAPDW, as well as IAPE and IAPEW, are due to long delays between correlated transmissions. While the energy consumed by IAPE is significantly higher, by 17% over IAPEW, for example, the savings in

Apps.	STPM	IAPED
Firefox	22	.08
Thunderbird	8	.05
GFTP	3	.23
Gaim	3	.02
Pan	4	.02
DJGame	5	.03

Table 5: Mechanism overheads in seconds.

delay by IAPE are not significant due to a relatively small number of total switches in these applications. Finally, *Pan* is a PSM dominant application, therefore the performance of IAP mechanisms is comparable to PSM.

5.5 Overheads

A large amount of computational or storage overhead can reduce or eliminate the gains made by implementing energy management for a single system component. For example, high computational overhead results in more energy being used by the CPU. This additional energy that is not normally consumed by the CPU can result in more energy being consumed to run the predictor than is saved by the predictor in the managed component. STPM is a relatively high cost prediction mechanism, due to having to compute probabilities for upcoming activity from histograms describing prior activity. In our simulations, STPM is implemented as described by its authors, including a 10-minute interval between recomputation of probability lookup tables. The update of probability tables used to determine the likelihood of upcoming activity is a high-cost operation which, when performed for each run, results in additional computational overhead without significantly benefiting the resulting mechanism accuracy. The overheads shown in Table 5 can be decreased for STPM at the cost of lower prediction accuracy. We evaluate the computational overhead shown in Table 5 by a timed execution of the simulation of each mechanism as it evaluates the application traces. We find that due to the predictions being made once for each mouse event rather than once for each incoming I/O request, the overheads for the IAPED mechanism are obviously lower. Compared to STPM, the IAPED overhead is between 99% and 98% less.

The computational overhead of all IAP mechanisms comes from the computation of unique IDs used to index into the prediction table. *Firefox* has the deepest tree of 27 levels in the studied applications. We again setup an experiment to measure the average overhead of traversing 27 level of tree hierarchy that is present in *Firefox*. The overhead of calculating the ID in this case is less than .8ms. Currently, each mouse click results in a communication overhead as the application window tree is built through use of X Server requests, so this overhead can be reduced by use of persistent tree representations of the application's GUI. Furthermore, ID calculations and subsequent prediction table lookup are performed once per click, meaning that they need to be performed as infrequently as the user interactions occur.

Finally, the storage overhead is relatively small in IAP mechanisms. The IAP daemon has to maintain prediction entries, which ranged from 32 in *Thunderbird* to 128 in *Firefox*. Each table entry is composed of five fields containing a single unsigned integer each. Therefore, in the worst case *Firefox* requires 2.5KB to store 128 entries. A 6.8KB table would suffice for storing all entries from every one of the six applications we have traced. This relatively small overhead can be further reduced, since only a small number of the entries contain information about clicks that lead to network I/O. As a result, we may consider aging the less useful entries and removing them using a simple replacement mechanism such as LRU.

6. RELATED WORK

To minimize energy consumption without sacrificing performance, the network card should be switched immediately to CAM when a high bandwidth transfer occurs and back to PSM when the transfer is over, since PSM can cause unacceptable 16-32x slowdown for synchronous traffic [4]. The drawbacks of the simple PSM scheme have been addressed through prior work that aimed to modify the PSM protocols in order to further improve the potential for energy savings. One such project produced BSD [14], the bounded slowdown protocol, for PSM. BSD improved on the static PSM protocol by dynamically adapting the length of the beacon period, shortening it during active periods and lengthening, backing off, during periods of inactivity. A further PSM beaconing optimization was introduced with PSM-Dynamic [5], which based the beaconing interval on recently observed RTT of the active connections. The drawback is that dynamic beaconing requires that the hardware be able to support non-static beaconing periods.

As a result, dynamic switching between PSM and CAM have been explored in Self-Tuning Power Management [4] (STPM). STPM explored the use of explicit application-level hints and on-line modeling of application access patterns to set network power management parameters. Application controlled power management has been applied to other devices and system components [7, 10, 17, 19] with a high potential for reducing energy consumption. However, this approach places the burden of inserting power management directives on the programmers and requires modification of existing applications before the energy saving potential can be realized. As a result, mechanisms for automatic application-like hint generation have been proposed [9, 8].

Higher-level context can be leveraged to improve accuracy and timelines of predictors. The context may include many aspects of the user environment, such as location, user activity, and system-user interaction. One of the earliest efforts in this area was the Lumiere Project [11]. Lumiere served as the basis for Microsoft's Office Assistant. The Location Stack project [15] used location based contextual hints to support a range of location-sensitive services. The Location Stack focused on information retrieval and customization. More recently, event-based mechanisms were used for building environments that respond to users activities [1]. Using ubiquitously deployed sensors, perceptual systems in smart environments monitored user behavior and issued hints about user's activities that customized devices. Layered hidden Markov models [18] were used to characterize states of a user's activity based on streams of video, audio and computer (keyboard and mouse) interactions. Finally, statistical models were applied to capture user behavior and correlate it to network activity [2]. This approach suffered from complicated clustering that required offline processing. Furthermore, clusters are only an approximation of user interfaces and result in a high computational overhead due to the retraining that occurs when the displayed window layout or geometry changes.

7. CONCLUSION

This paper proposes a new direction for designing resource management predictors. While current predictors monitor low-level application behavior such as network activity or sometimes reach as high as monitoring the application systems calls, we take the next step by monitoring the key component responsible for the behavior of interactive applications, the user. We have proposed and successfully implemented several user-interaction-aware energy management mechanisms that dynamically learn the context of user interactions with respect to network activity in interactive applications.

Other recent resource management solutions could potentially perform better with hints from modified applications, but for unmodified applications they rely on monitoring low-level activity. Application modifications are impractical due to the additional burden that is placed on programmers, therefore our proposed mechanisms provide a transparent solution that provides high energy efficiency without the need for application modifications. More importantly, the proposed mechanisms are readily implementable in existing systems due to: (1) the simplicity of the proposed mechanism which monitors and correlates user behavior with system activity, (2) quantifiably low computational and storage overheads, (3) online monitoring and prediction that does not require application modification or offline processing for the analysis of user interactions.

8. REFERENCES

[1] F. Albinali, P. Boddupalli, N. Davies, and A. Friday. Correlating sensors and activities in an intelligent environment: A logistic regression approach. *Proceedings of the First European Symposium on Ambient Intelligence*, 2875:318–333, 2003.

[2] F. Albinali and C. Gniady. Cpm: Context-aware power management in wlans. In *Eighteenth Innovative Applications of Artificial Intelligence Conference (IAAI)*, 2006.

[3] M. Anand, E. B. Nightingale, and J. Flinn. Self-tuning wireless network power management. *Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom '03)*, pages 176–189, 2003.

[4] M. Anand, E. B. Nightingale, and J. Flinn. Self-tuning wireless network power management. *Wireless Networks*, 11(4):451–469, July 2005.

[5] Z. Anderson, S. Nath, and S. Seshan. Choosing beacon period for improved response time for wireless http clients. *Proceedings of the Second International Workshop on Mobility Management and Wireless Access Protocols (MobiWac '04)*, 2004.

[6] A. Dix, J. Finley, G. Abowd, and R. Beale. *Human-computer interaction (3rd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[7] C. S. Ellis. The Case for Higher-Level Power Management. In *Workshop on Hot Topics in Operating Systems*, pages 162–167, Rio Rico, AZ, USA, March 1999.

[8] C. Gniady, A. R. Butt, and Y. C. Hu. Program counter based pattern classification in buffer caching. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI)*, December 2004.

[9] C. Gniady, Y. C. Hu, , and Y.-H. Lu. Program counter based techniques for dynamic power management. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, Dec. 2004.

[10] T. Heath, E. Pinheiro, J. Hom, U. Kremer, and R. Bianchini. Application transformations for energy and performance-aware device management. In *Proceedings of the 11th International Conference on Parallel Architectures and Compilation Techniques*, September 2002.

[11] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. *the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998.

[12] IEEE. 802.11 wireless standard. Online Technical Standard Specification, 1999. http://grouper.ieee.org/groups/802/11/.

[13] IEEE. Supplement to ieee standard for information technology: Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Online Technical Standard Specification, 2001. http://grouper.ieee.org/groups/802/11/.

[14] R. Krashinsky and H. Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. *Proceedings of the 8th annual international conference on Mobile computing and networking (MobiCom '02)*, pages 119–130, 2002.

[15] A. LaMarca, J. Hightower, I. Smith, and S. Consolvo. Self-mapping in 802.11 location systems. *Proceedings of the Seventh International Conference on Ubiquitous Computing (Ubicomp 2005)*, pages 87–104, September 2005.

[16] J. R. Lorch and A. J. Smith. Using user interface event information in dynamic voltage scaling algorithms. *Eleventh International Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems*, 00:46, 2003.

[17] Y.-H. Lu, G. D. Micheli, and L. Benini. Requester-aware power reduction. In *Proceedings of the International Symposium on System Synthesis*, pages 18–24, 2000.

[18] N. Oliver, A. Garg, and E. Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, 96(2):163–180, 2004.

[19] A. Weissel, B. Beutel, and F. Bellosa. Cooperative I/O—a novel I/O semantics for energy-aware applications. In *Proceedings of the Fifth Symposium on Operating System Design and Implementation*, December 2002.