LLAMA: An Adaptive Strategy for Utilizing Excess Energy to Perform Background Tasks on Mobile Devices

Denitsa Tilkidjieva Nilanjan Banerjee† Maria Kazandjieva
Sami Rollins‡ Mark D. Corner†

Mount Holyoke College, {dntilkid, makazand}@mtholyoke.edu
†University of Massachusetts, {nilanb, mcorner}@cs.umass.edu
‡University of San Francisco, {srollins}@cs.usfca.edu

Abstract

To support pervasive applications mobile computing devices often perform background tasks, such as prefetching RSS newsfeeds, that improve the user's experience but require no interactivity. In an effort to conserve energy, background tasks are typically scheduled infrequently and only when a device is already in-use. The underlying assumption is that the user will need every last joule in the battery. However, the results of our four-month user study suggest that mobile devices often begin to recharge with 30% or more of their energy remaining. The goal of this work is to utilize that excess energy to increase the frequency of background tasks. We present an online algorithm that adaptively predicts how much energy a device will have remaining at recharge and schedules self-initiated wake-up to utilize the energy for background tasks. We use the data collected by our user study to evaluate our algorithm. Our results indicate that that we can perform tasks as frequently as every 12 minutes while remaining 95% confident that the user will recharge her device before the battery dies.

1 Introduction

A fundamental goal of pervasive computing is *invisibility*: masking variations in context by proactively adapting the state of a user's devices to current conditions. To achieve this goal, devices perform *background tasks* that improve the user's experience but require no interactivity. For example, operations such as prefetching, hoarding, replication, and synchronization promise to smooth variations in reliability, connectivity, and availability by transparently transferring data that may become useful in the future.

Although background tasks are intended to improve the user's experience, they compete with interactive tasks for device resources such as bandwidth, CPU cycles, and energy. In many cases, partitioning resources is straightforward. For example, if a network link is idle it can be used for a background task with no further consequences. However, sharing energy on battery-powered mobile devices is a zero-sum game; energy allocated to a background task is energy that cannot be spent to perform an interactive task. If background tasks consume too much energy, the device may run out of battery before the user recharges and the user-perceived lifetime of the device will be affected.

Fortunately, background tasks such as web-prefetching and file replication are speculative. Thus, their frequency and fidelity can be altered to minimize their impact on user-perceived lifetime. However, most current strategies for performing background tasks are excessively conservative, performing tasks only when the user is active, hence consuming minimal additional energy. Any energy remaining in the device when it is recharged is effectively *wasted* and, in hindsight, could have been used to benefit the user.

Though wasted energy can be used to increase the frequency and fidelity of background tasks, determining how much energy a device will have remaining at recharge is a challenge. Energy usage varies significantly from device to device, from user to user, and also temporally. One user may fully charge her music player, take it on a bike ride, and drain the battery completely in one use, while another charges her digital camera, uses it intermittently over a weekend, and only drains it partway. Though in the first scenario a conservative energy management strategy is warranted, in the second scenario the device could spend the remainder of its battery transparently uploading images to a web server or sharing them with friends. Other devices might fetch email or RSS feeds, replicate data to backup servers, automatically adjust screen brightness, or perform virus scans. However, the key challenge is to identify how much energy can be used for background tasks without affecting the user-perceived lifetime of the device.

The goal of this work is to *predict* how much energy a device will have remaining at recharge and use that energy to perform background tasks. Because of the significant variations in usage patterns of devices, no single strategy is likely to work well for all users. However, a single device is often associated with a particular user and a specific purpose. An effective strategy should be able to learn the habits of the user and *adapt* energy usage based on the patterns observed.

This paper presents the design and evaluation of Llama, a system designed to adaptively manage excess energy on mobile computers. First, we use battery traces collected from 13 laptops during a four-month study to demonstrate that users exhibit predictable behavior and typically have energy to spare. We then present an online algorithm that determines, based on previous battery usage, how much energy a device can devote to supporting background tasks, such as fetching RSS feeds, without affecting user-perceived lifetime. Finally, we use the data collected during our user study to evaluate how much energy our algorithm is able to use and how often it impacts the user-perceived lifetime of the device. Our results indicate that we can perform tasks as frequently as every 12 minutes while remaining 95% confident that the user will recharge the device before the battery dies.

2 A Mobile User Study

Our work is predicated on two hypotheses. First, mobile users often leave a substantial portion of energy in their batteries. Second, devices exhibit predictable energy usage patterns. To validate these hypotheses, we conducted a small-scale user study on the energy habits of laptop users. The outcomes of this study are threefold. First, it demonstrates that users often leave 30% or more of their battery unused, and many of the users leave 60% or more of the capacity in the battery. Second, user behavior, including our own, can be classified into three distinct categories that closely correspond to our perceptions of energy usage. Third, our collected data set provides us with the foundation for a simulated evaluation of our system.

We deployed a Java-based measurement tool that periodically collects the following measurements: the percentage of battery remaining, the charging status of the device, the current CPU usage, the available disk space, a coarse measurement of available network bandwidth, and the duration of time since the last keyboard or mouse input (i.e., idle time). It records this information every 1 to 10 minutes, based on a user-configurable parameter, and reports data collected to a central server. In addition, it only records measurements when the device is in an active or idle state and does not attempt to override its power settings—it does not wake the device from suspension in order to collect measurements.

In total, the tool was deployed on 24 laptops, both Windows and Macintosh. It was distributed in a purely word-of-mouth manner and most of the devices belong to computer science faculty, students, and their friends and families. Though we suspect that the distribution of the user population is highly biased, we believe that these results demonstrate the feasibility of running such a system on laptops and other pervasive computing devices, such as portable music players, PDAs, mobile phones, and tablets.

Our analysis and experiments utilize data collected by 13 of the devices between February 16th

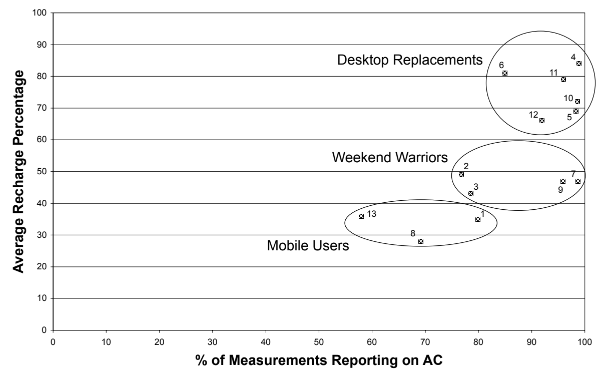

Figure 1: This figure shows the results of using a clustering algorithm that reveals three categories of mobile users.

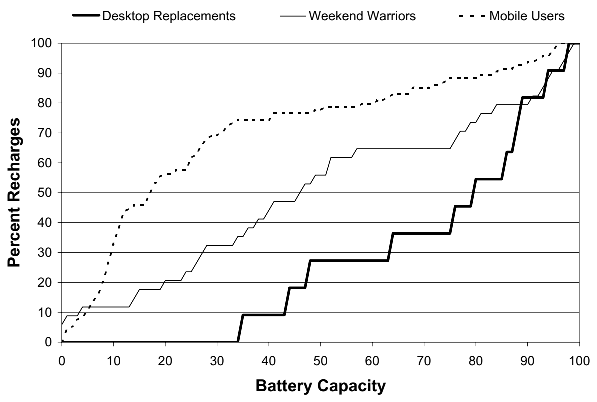

Figure 2: This figure illustrates the cumulative distribution of the remaining battery at recharge for three sample devices.

and June 19th, 2006—roughly a period of four months. We chose to use data from the 13 devices that consistently reported data during this period. Some devices did not consistently report data because the users uninstalled the software, or stopped using their laptops altogether.

In this work, we primarily consider the battery usage characteristics of the participating devices, including the percentage of battery remaining at recharge and the charging status of the device. We have discovered several interesting anomalies in the data collected. For example, on March 18, 2006, the battery of one device reported 11% capacity at 8:25PM and 87% at 8:29PM—several instances of this behavior can be found in the traces. As we are unaware of any laptops that can charge this quickly, we believe that two batteries were used on these devices. Another interesting occurrence is that some devices consistently report 95-96% charged, even while on AC for extended periods. We have tracked this to the fact that OS X avoids repeatedly charging the battery to full capacity to ensure the longevity of the battery.

To classify the battery usage patterns of the participating devices we considered the two characteristics that we believe are the most telling of a user's habits: the percentage of measurements reported while plugged in and the average battery capacity remaining when recharging began. We used a complete linkage clustering algorithm with a threshold of 25 to categorize the 13 devices based on these attributes [3].

Figure 1 shows the three distinct categories that emerge from our data set. Unsurprisingly, nearly half of the devices fall into the category we refer to as *desktop replacements*. These devices spend most of their time on AC power and, on average, recharge their batteries when they are very full. The evidence suggests that desktop replacement users purchase laptops for their form-factor, and not for their mobility. We also identified three *mobile users* that often allow their batteries to drain completely, though still record 76% of their measurements while on AC power. Though this may seem counterintuitive, a closer look at the data reveals that these devices often report significant drops in battery during periods when the device is not collecting data. This suggests that the devices run on battery more than 76% of the time, but likely expend much of their energy while suspended. Finally, the *weekend warriors* fall somewhere in between the other two classes. The data also reveal that most of the weekend warriors actually act as desktop replacements much of the time, but occasionally exhibit bursts of mobile user-like behavior.

The average percentage of battery remaining at recharge is an useful attribute for classification of users. However, it does not provide us with a sufficiently fine-grained look at the amount of energy that goes unused by the devices. Figure 2 shows the cumulative distribution of the percentage of the battery remaining when a device begins to recharge. It illustrates three example devices, one from each class. Other devices from the same class are substantially similar.

The graph shows that all three of the devices have energy to spare most of the time. For example, the weekend warrior can spare over 50% of its battery half of the time and the desktop replacement never uses the last 30% of its battery. Even the mobile user recharges at above 30% roughly 30% of the time. We observe that any strategy that attempts to utilize this excess energy must adapt to varied patterns of device usage. No one policy will work well for all devices, and no one policy will work for any device all of the time.

3 Monitoring and Prediction

Based on the results of our user-study, we have designed a system, named Llama, to manage excess energy on pervasive computing devices. Llama tracks how much of the battery is typically used on a device, predicts when recharges will occur, and devotes excess energy to background applications. Though Llama is capable of supporting many kinds of background applications, we have chosen to focus on applications designed to run periodically for short periods of time. An application, such as RSS newsfeed prefetching, that spends a few seconds downloading updates every few minutes falls into this category.

There are three key features of the Llama system design. First, Llama uses a *probabilistic* policy, ensuring that background applications impact the user-perceived lifetime of the device only a small percentage of the time. Second, Llama is *adaptive*; it responds to short-term and long-term changes in user behavior. Third, Llama enables a new kind of functionality not typically used in laptops, PDAs, and portable music players: *self-initiated wake-up.* Generally, once a device places itself in a low-power state, such as suspension, hibernation, or off, it requires user intervention to start again. However, Llama can schedule wake-sleep cycling to proactively run background tasks while the user is not actively using the device. Note that at a hardware level this is already well-supported.

The main component of Llama is a *predictor* that estimates, using a measured histogram of previous battery usage and the current battery capacity, how much energy will be left in the battery when the device begins its next recharge cycle. The histogram is measured using the same technique used in the user-study—if the device is awake, it periodically records the remaining charge in the battery and tracks when a recharge begins. Using the predicted energy remaining, the predictor schedules self-initiated wake-up to utilize the remaining energy to performing background tasks. This process is necessarily adaptive; it recalculates the energy to devote to background tasks each time a background task is executed. Furthermore, the predictor utilizes a tunable parameter that specifies the confidence with which the device will not run out of battery before the next expected recharge.

As an example of how Llama operates, suppose a user wants to assure with a confidence of 95% that her laptop battery will not run out before the next expected recharge. The algorithm determines the current battery capacity to be 30%. It consults the histogram of recharges and determines that 95% of the time that the battery drains below 30% the user recharges at or above 10%. The algorithm uses this to estimate that the user will use 90% of the battery. It then conservatively allows Llama to use 10% of the remaining 30%, or 3%.

We formally define the optimization problem using the variables defined in Table 1. E_b and E_f are the energy consumed by background and foreground tasks in between two consecutive recharges. The algorithm tries to maximize E_b—subject to $Pr[E_b + E_f > C] \leq p$. Our current system then uses E_b, to calculate the frequency with which the device should transition into an active state to perform a background task.

When Llama executes, it first determines if the device is plugged in or is already in-use. In both cases, the device is already in an active state and can perform its operation as frequently as necessary. We currently assume that the applications we support do not incur any additional energy usage when run in the background during times that the user is actively utilizing the device. This is consistent with applications, such as RSS prefetching, that are short and consume only a small amount of energy. Though, this parameter is user-configurable and can be easily changed.

p	confidence of not exceeding battery capacity
E_b	energy for background tasks before next recharge
E_f	energy for foreground tasks before next recharge
C	total capacity of the battery
C_p	present capacity of the battery
H	histogram for CDF of recharges
R	maximum rate at which battery discharges
E_t	energy to transition to active state
e_b	energy for one background task

Table 1: The table shows the list of variables.

If the device is running on battery or not in-use, the algorithm consults the histogram H, which tracks the probability distribution of when the user recharges the device given a remaining battery capacity, C_p. Using H, the scheme finds x such that $H(x) \leq (1-p) \leq H(x+\Delta H)$, where ΔH is the size of the histogram bins. With probability at least p, the energy used by foreground processes will not exceed $(100-x) \cdot C_p$. The amount of excess energy that can be used for background processes is consequently calculated as $E_b = C_p - E_f$. The system then uses E_b to calculate how often the device can transition into an active state to perform an operation and evenly distributes the transitions across the time until the next expected recharge as shown below.

The algorithm conservatively estimates the time before the next recharge (T_d) as $T_d = \frac{C_p}{R}$. Thus, the interval after which a background operation should occur is calculated as $\frac{T_d}{E_b} \cdot (e_b + E_t)$, where e_b is the energy for one background task, and E_t is the cost to transition the device to the active state to complete the task. However, E_t will vary depending on the beginning state of the device; it costs more to transition a device from suspended to active than from idle to active. Since, we do not know what state the device will be in when we execute a task we predict it will be in the same state as the last time a task was executed. If the the last time Llama initiated a background operation the device was active, then $E_t = 0$. Otherwise we use the energy needed to resume and then suspend the device, which can be measured online using information provided by the operating system.

4 Trace-Based Device Simulator

We have developed a trace-based simulator that uses the traces from our four-month user study to demonstrate Llama's performance. Unfortunately, the data collected during the user study do not contain all of the information needed to provide a completely accurate trace. For example, the trace collection mechanism cannot distinguish between suspended and off states, and it does not record transitions between charging and discharging that occur when the device is suspended or off. However, using measurements taken before and after an off or suspended state, we can infer a probable course of events that led to the trace. We have found through experimentation that our results are largely insensitive to the trace reconstruction method.

4.1 Trace Reconstruction

The core of the simulator is a state machine that can be in one of the following states:

- *Active* – The device is on and there is keyboard or mouse input.
- *Idle* – The device is on, but there is no keyboard or mouse input.
- *Suspended* – The device is suspended.
- *Off* – The device is off.

The above power states are derived from energy management paradigms used on laptops. For example, a laptop is assumed to be in idle state if there is no user activity for a given period of time. In each of the above states, the device can also be either *charging* or *discharging*—plugged in or running on battery.

We derive the transitions from one state to the next from the data collected during our user study. For each device, we analyzed the measurements recorded and generated the following transitions:

Charging/Discharging: If a measurement records the device is on AC power and the previous measurement recorded the device as not on AC power, we transition to *charging* status. Conversely, if a measurement indicates the device is not on AC and the previous measurement recorded as on AC we transition the device to *discharging*.

Active/Idle: If the amount of time since the last keyboard or mouse input is less than 2 seconds and the device was previously in an idle state, we transition to *active* and the charging/discharging status remains unchanged. If the amount of idle time is greater than 2 seconds and the device was previously in an active state, we transition to *idle* and the charging/discharging status remains unchanged.

Suspension/Power-off: If a significant amount of time has elapsed between measurements it is likely that the device has suspended itself or been turned off. We infer what has occurred based on the percentage of the battery reported at each measurement. We calculate the average rate at which the device consumed or gained energy over the interval and use it to determine the state,

Active charging	27 W	Suspended charging	42 W
Active discharging	-31 W	Suspended discharging	-0.4 W
Idle charging	28 W	Off discharging	0W
Idle discharging	-10 W	Total energy to suspend and wake	178 J
Task charging	28 W		
Task discharging	-15 W	Full battery	205488 J

Table 2: The power and energy characteristics of the simulated device.

or combination of states, the device was in over the interval. For example, if the average rate is larger than the suspended power consumption rate but smaller than the active consumption rate, we determine that the device was active for part of the interval and off for the remaining part of the interval.

4.2 Simulated Llama

We use the trace as input to a simulated Llama system. The simulator introduces a new state, *task*, in which the device is on and performing a task, but there is no keyboard or mouse input. When Llama determines that a background task should be performed, it transitions the device into the *task* state, and the charging/discharging status remains unchanged. If the device was previously *active* or *off* this transition will have no effect. If active, the device will continue to be active and perform the task in the background. If off, the device will be unable to perform the task. When the task is complete, Llama returns the device to the appropriate state, keeping in mind that another transition may have occurred while the device was performing the task.

5 Evaluation

Our evaluation of Llama answers the following questions: *1. Does Llama efficiently utilize excess energy? 2. Does use of Llama impact the user-perceived lifetime? 3. Is an adaptive strategy really necessary?*

To answer these questions, we conducted a series of simulated experiments. We used the raw data collected from the 13 devices discussed in Section 2 to generate our usage models. Recall that the data collection period lasted from February 16th to June 19th, 2006.

Table 2 outlines the power and energy characteristics of our simulated device. These numbers were measured using the BatteryMon tool [2] on a Dell Latitude 100L laptop. We measured the energy consumed in each state for a full discharge/charge cycle and took an average of the measurements. The characteristics of each state were derived from common laptop power management

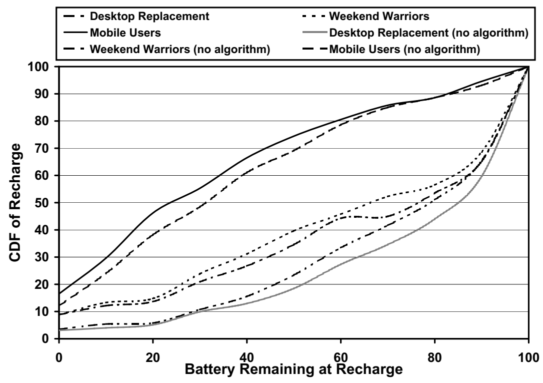

Figure 3: This figure illustrates the percentage of the battery remaining when a device transitions from discharging to charging for p=95%. Llama is able to use excess energy thus reducing the battery remaining at recharge

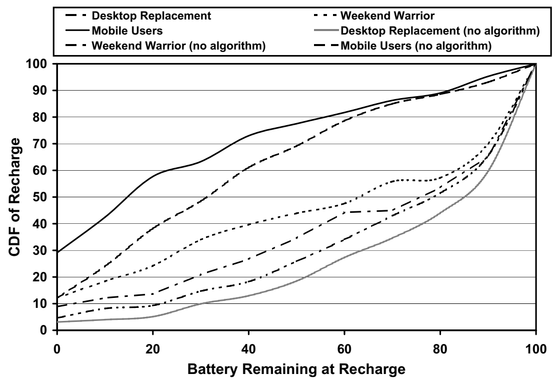

Figure 4: This figure illustrates that using a lower confidence with which Llama will not exceed the battery capacity Llama is more aggressive and utilizes more excess energy.

strategies that commonly shut off unnecessary components, such as the network interface, to conserve power. The active measurement was taken with the device and screen on, running several applications and using the network interface, the idle measurement was taken with the screen and network interface off, and the task measurement was taken with the network interface on, but the screen off.

For all experiments, we use 15 seconds as the length of a single background task—the approximate time to download 25 RSS feeds. To condense our results, each experiment reports an average of all of the measurements taken for the devices in each class. Where appropriate, we also report the minimum and maximum measurements. Generally, all devices in a particular class perform similarly.

Figures 3 and 4 show the cumulative distribution of the battery remaining when a device transitions from discharging to charging. For this experiment, we vary p—the confidence with which Llama should not exceed the battery capacity. Figure 3 reports the results of p=95% and Figure 4 reports the results of p=75%. For each class, we also show the result of using *no algorithm*: the simulator applies the transitions generated from our battery traces but does not schedule self-initiated wake-up or perform background tasks.

We observe that Llama reduces the amount of energy remaining when a mobile device begins to recharge and uses roughly 7% more of the battery for p=95%. In the case of p=75%, Llama is considerably more aggressive using up to 18% more battery for the mobile users and weekend warriors. Interestingly, the desktop replacements are less affected by the more aggressive strategy

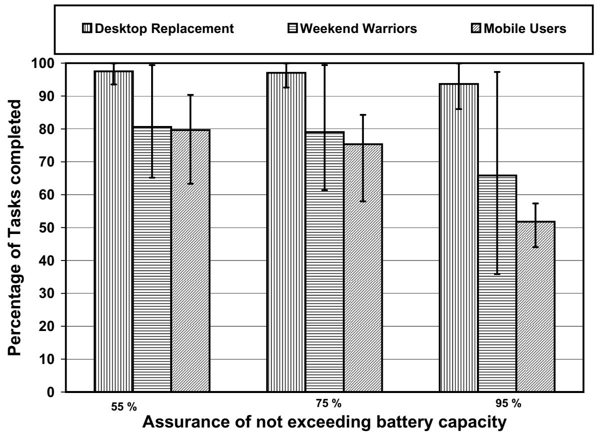

Figure 5: This figure illustrates the percentage of tasks Llama completes for three values of p. Llama is able to complete a significant portion of tasks, becoming more conservative as p increases.

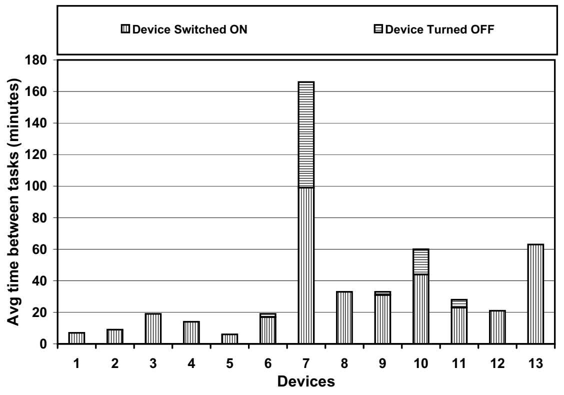

Figure 6: This figure shows the average number of minutes between tasks. Even the mobile users are able to complete a task every 35 minutes on average.

because they spend much of their time plugged in, reducing the opportunity for Llama to use excess energy. Even if Llama continuously executed background tasks during times when the devices were running on battery, it still would not be able to drain the batteries. Another interesting note is that Llama does reduce the amount of battery remaining at recharge, potentially impacting the amount of time the device must spend plugged in. However, the data indicate that 85% of the time users actually leave their devices plugged in long after they are fully charged. Moreover, with the advent of faster rechargable batteries [15], we argue that the amount of time used to recharge should not remain an issue.

In general we expected Llama to be more aggressive, ideally using all energy such that 100% of recharges occur at 0%. Though, even for very low values of p this does not happen. The reason is that Llama operates primarily during periods when the device is suspended or idle. A key issue is that we currently assume that Llama does not use additional energy when run in the background during times that the user is actively utilizing the device. By taking advantage of the opportunity to increase the power consumption of an already-active device, for example by increasing the screen brightness or processing speed, we anticipate even better performance for Llama.

Figure 5 shows the percentage of tasks completed by Llama in comparison to the ideal number of tasks the device could have completed if it were able to perform a task every 5 minutes without using additional energy. In other words, the ideal assumes that the energy cost of a task is nothing. We report results for three values for p. The error bars indicate the maximum and minimum values across the devices in each class.

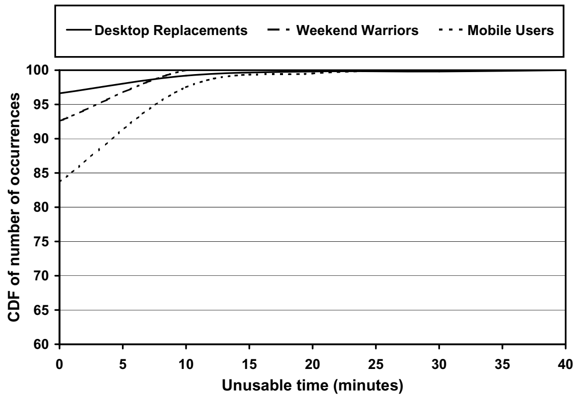

Figure 7: This figure shows the percentage of time that Llama affects user-perceived lifetime. In most cases, Llama introduces less than 10 minutes of unusable time.

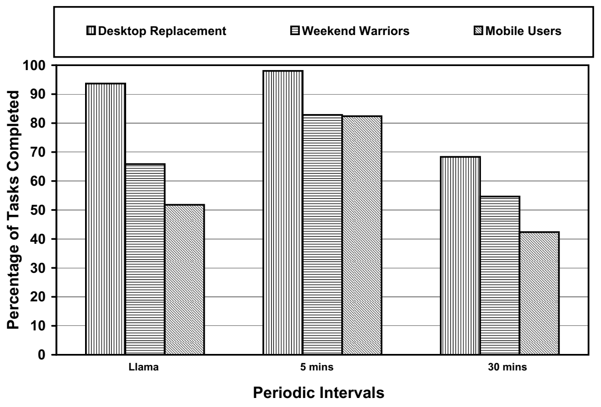

Figure 8: This figure shows the percentage of tasks completed by Llama and two schemes that use fixed inter-task intervals. Though the 5 minute scheme completes a large percentage of tasks, this figure does not show the associated cost—a significant amount of unusable time for mobile users.

These results illustrate that Llama provides a significant benefit, completing a substantial portion of tasks. Even in the most conservative case, desktop replacements complete well over 90% of the tasks and mobile users well over 50%. As expected, the mobile users and weekend warriors become considerably more conservative as p increases, though the desktop replacements are less affected by p because they spend much of their time plugged in. Another interesting observation is that the weekend warrior class shows considerably more variation than the others. Though the clustering algorithm grouped them into the same class, the data for each of the four weekend warriors differs considerably; one device looks much like a mobile user while another is rarely even used.

To gain a better understanding of the impact Llama has for the end-user, Figure 6 shows the average number of minutes between the completion of tasks. The first six devices belong to the desktop replacement class, the next four are weekend warriors, and the last three are mobile users. The portion of each bar labeled *Devices Switched ON* excludes from the average the time between two tasks if the device was in an off state at any point between the tasks. Because the desktop replacements have more energy to spare, they are able to perform a tasks, on average, every 12 minutes. Though the mobile users have less energy to spare, they are still able to perform a task every 35 minutes on average. Device 7 is clearly an outlier; it performs tasks every 160 minutes because it spends a significant amount of time off.

The previous figures clearly show the benefit of using Llama. The associated cost is the potential

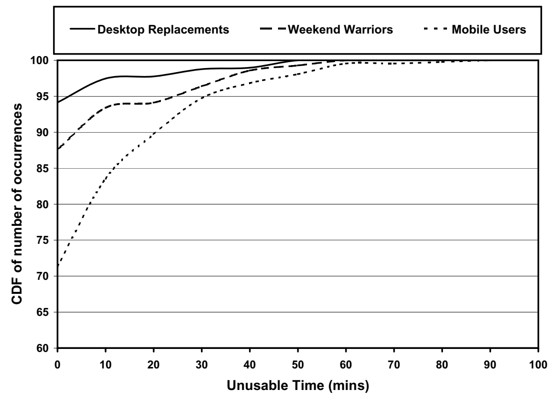

Figure 9: This figure illustrates the unusable time metric for the 5 minute scheme. This scheme is appropriate for desktop replacements, but debilitating for mobile users.

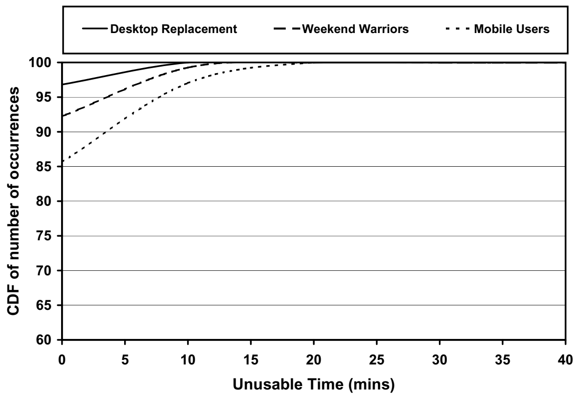

Figure 10: This figures illustrates the unusable time metric for the 30 minute scheme. This is is appropriate for the mobile users, but is much too conservative for the desktop replacements.

for Llama to impact the user-perceived lifetime of the device by using too much energy. Figure 7 plots the cumulative distribution of the *unusable time* of each class for p=95%. The unusable time is defined as the amount of time a device spends with its battery below 3% when running Llama minus the amount of time it spends with its battery below 3% without Llama. In other words, this measurement tells us the amount of time the user wished to use the device, but was unable.

Intuitively, for p=95% we expected to see 95% of occurrences introducing 0 minutes of unusable time. This is true for the desktop replacements, but for the mobile users over 15% of the recharges introduce some unusable time. Fortunately, all but 5% of these occurrences introduce unusable time of 10 minutes or less—a very small cost. Additionally, closer analysis of the data indicates that Llama is more aggressive in scheduling tasks when users are active and users often suspend their devices while a task is in-progress. In these cases, Llama uses a small amount of energy to complete the task before actually suspending the device. An alternate strategy, which would reduce unusable time, would be to preempt in-progress tasks when the device suspends.

We have also looked at the unusable time and the percentage of tasks completed for varied task length and lower values of p. In summary, longer tasks use more energy, in particular when devices often suspend while completing a task. However, the percentage of tasks completed remains roughly the same for longer tasks. For lower values of p, the unusable time increases, sometimes reaching 40 minutes but the percentage of recharges that occur at 0% is much closer to p. For example, with p=75%, mobile users recharge above 0%, 73% of the time.

Our final experiments consider whether an adaptive strategy is necessary. Figures 8, 9, and

10 compare Llama against a periodic scheme that always uses a fixed inter-task interval of either 5 minutes or 30 minutes. Note that the 5 minute scheme does not complete 100% of the tasks because the ideal scheme assumes that a task incurs no additional energy cost.

We find that while the 5 minute scheme performs significantly more tasks, over 30% more in the case of the mobile users, it also introduces a significant amount of unusable time. The mobile users run out of battery almost 30% of the time and in some instances are unusable for more than 60 minutes. The more conservative 30 minute scheme is much better for the mobile users, but is too conservative for the desktop replacements, reducing the percentage of tasks completed by almost 25%. Our adaptive scheme provides a necessary device-specific balance between the frequency of background tasks and user-perceived lifetime.

6 Related Work

A significant amount of research has focused on power management techniques for mobile devices. Several projects, including Odyssey [4], ECOSystem [18], and work by Simunic, et al. [12], attempt to balance performance and system-wide energy usage. Other projects have more specifically addressed individual resources such as processing [9], memory [7], and the network interface [6, 14]. This work is certainly complementary to ours. During periods of activity, we can leverage any combination of other strategies for reducing the amount of energy consumed. The smaller the amount of energy consumed, the more often we can execute transparent applications.

Wake-on-Wireless (WoW) [11] and Turducken [13] take a slightly different approach to power management. The goal of both projects is to enable low-power operating modes in mobile devices by combining several computing platforms that operate at varying power points. Low-power platforms can perform basic tasks, such as wireless network discovery, on behalf of high-power platforms. Though we could envision integrating our algorithms into such a platform, our contribution is an adaptive strategy for determining how to spend excess energy on a mobile device.

Also related to our work is the collection of work on caching and prefetching. Traditionally, caching and prefetching have applied specifically to the hard disk [8]. Central to this work are algorithms for determining when and what to prefetch. There have also been a number of projects that have focused on caching and prefetching for a mobile environment [16, 1, 5, 10]. This work leverages techniques such as extending the idle time of the mobile device and reducing the amount of data prefetched. However, this work does not attempt to make adaptive decisions based on energy availability.

Yin and Cao [17] propose an adaptive scheme. However, their scheme focuses solely on determining the number of items to be prefetched whereas we propose a general algorithm that can be used for prefetching or other tasks. Further, they require either static knowledge of the target lifetime of the device or a static set of thresholds for determining when the device should modify its prefetching strategy. In contrast, we track energy usage on the device and adapt our algorithm based on prior usage patterns.

7 Conclusion

Llama is a system designed to adaptively and transparently utilize excess energy to perform background tasks on mobile devices. We validate the hypotheses that mobile users have excess energy to spare and exhibit predictable behavior through an extensive four-month user study of 13 laptops. We present the design of an adaptive, online algorithm that learns user behavior and predicts how much energy a device can devote to background tasks without affecting user-perceived lifetime. Using the data collected from the user study we show that Llama can help the device spend a large amount of excess energy while remaining highly confident that the user-perceived lifetime will not be affected.

References

[1] D. Barbara and Imielinksi. Sleepers and workaholics: Caching strategies for mobile environments. In *International Conference on Management of Data*, Minneapolis, Minnesota, USA, May 1994.

[2] BatteryMon tool. Battery monitor. `http://www.programurl.com/batterymon.htm`.

[3] Johnson S. C. Hierarchical clustering schemes. In *Psychometrika*, 1967.

[4] J. Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *SOSP 99*, pages 48–63, Kiawah Island, SC, USA, December 1999.

[5] S. Gitzenis and N. Bambos. Power-controlled data prefetching/caching in wireless packet networks. In *INFOCOM*, New York, NY, USA, June 2002.

[6] R. Kravets and P. Krishnan. Power management techniques for mobile communication. In *MobiCOM 98*, pages 157–168, Denver, Colorado, USA, September 1998.

[7] A. Lebeck, X. Fan, H. Zeng, and C.S. Ellis. Power aware page allocation. In *ASPLOS*, Cambridge, MA, USA, November 2000.

[8] A. Papathanasiou and M. Scott. Energy efficient prefetching and caching. In *USENIX*, Boston, MA, USA, June 2004.

[9] A. Rudenko, P. Reiher, G. Popek, and G. Kuenning. Saving portable computer battery power through remote process execution. *Mobile Computing and Communications Review*, 2(1):19–26, 1998.

[10] H. Shen, M. Kumar, S. Das, and Z. Wang. Energy-efficient caching and prefetching with data consistency in mobile distributed systems. In *IPDPS*, Santa Fe, NM, USA, April 2004.

[11] E. Shih, P. Bahl, and J. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *International Conference on Mobile Computing and Networking*, Atlanta, GA, USA, September 2002.

[12] T. Simunic, L. Benini, P. Glynn, and G. DeMicheli. Dynamic power management for portable systems. In *MobiCom*, Boston, MA, USA, August 2000.

[13] J. Sorber, N. Banerjee, M. Corner, and S. Rollins. Turducken: Hierarchical power management for mobile devices. In *MobiSys*, Seattle, WA, USA, June 2005.

[14] M. Stemm and R. Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Communications*, August 1997.

[15] The Register. Toshiba super charge battery, March 2005. `http://www.theregister.co.uk/2005/03/29/toshiba_li-ion_battery/`.

[16] J. Xu, W. Lee, Q. Hu, and D. Lee. An optimal cache replacement policy for wireless data dissemination under cache consistency. In *International Conference on Parallel Processing*, Valencia, Spain, September 2001.

[17] L. Yin and G. Cao. Adaptive power-aware prefetch in wireless networks. *IEEE Transactions on Wireless Communication*, 3(5):1648–1658, 2004.

[18] H. Zeng, C.S. Ellis, A. Lebeck, and A. Vahdat. ECOSystem: managing energy as a first class operating system resource. In *ASPLOS*, San Jose, CA, USA, October 2002.