Applying Models of User Activity for Dynamic Power Management in Wireless Devices

Caleb Phillips[1], Suresh Singh[2], Douglas Sicker[1], Dirk Grunwald[1]
[1]Department of Computer Science, University of Colorado, Boulder, Colorado, USA
[2]Department of Computer Science, Portland State University, Portland, Oregon, USA
caleb.phillips@colorado.edu, singh@cs.pdx.edu, {sicker,grunwald}@colorado.edu

ABSTRACT

In this paper we use a large dataset of wireless user activity traces to test the various dynamic power management schemes. We also present and test our own empirically-driven dynamic power-saving algorithms, which are based on prior observations of user activity patterns. We believe that this sort of analysis can guide adoption of a user-behavior driven approach to radio and communications power management, and, in networking-centric devices, power management for the entire device. Additionally, understanding the characteristics of user-activity and efficient mechanisms to predict this activity can help inform the design of power-saving schemes for future networking protocols.

Categories and Subject Descriptors

H.1.2 [**Models and Principles**]: User/Machine Systems; D.4.8 [**Operating Systems**]: Performance—*Modeling and prediction*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms, Human Factors, Measurement, Performance

1. INTRODUCTION

Networked mobile devices have become an indispensable part of our lives today. And, as such, there are continual efforts at increasing their capabilities while making them more energy efficient.

Although low-level architectural power-saving is becoming commonplace in mobile systems and huge leaps are being made, dynamic power-saving schemes at a high level have failed to be adopted by industry. Indeed, the current state-of-the-art is a naïve method of powering-down after a long idle period - a system which is completely agnostic to user activity patterns and is greatly inefficient as a result. Moreover, wireless protocols as they are implemented provide only crude power-saving functions, which are similarly agnostic to the actual usage patterns of wireless devices. Although the dynamic power management (DPM) literature has suggested several good algorithms, these have failed to be implemented. In fact, it may be the large number of very different and often complicated algorithms, among which there is no clear winner, that has prevented adoption.

In this paper we aim to use a large and novel dataset of wireless user activity traces to test the various DPM algorithms at making power-saving decisions. We will also present and test our own emperically-driven dynamic power-saving algorithms, which are based on prior observations about user activity patterns [13]. We believe that this sort of analysis can guide adoption of a user-behavior driven approach to radio and communications power management, and, in networking-centric devices, power management for the entire device. Additionally, understanding the characteristics of user-activity and efficient mechanisms to predict this activity can help inform the design of power-saving schemes for future networking protocols.

2. RELATED WORK

For more than fifteen years, predictive power management techniques have been appearing in the literature. Due to this, there is no lack of variety in terms of approach. However, sometimes a wealth of varied approaches can be a burden.

Some of the first work on predictive power consumption was carried out by Wilkes et al. of the Hewlett-Packard Laboratories with the target application of predictive hard-disk spin-down [19]. In [5], Golding et al. present an early taxonomy for power-saving techniques and propose the idea of *idle-tasks* – work which can be accomplished while the device is otherwise idle. They propose several predictors and prediction filters (called "skeptics"), and ultimately conclude that simple window-based predictors and backoff techniques outperform the more complex configurations. In addition to those which concern themselves with hard-disk spin-down, some other early work considered dynamic frequency scaling for processors. In [6], Govil et al. suggest approaches which use local information (a brief history of inactivity periods) and a smoothing function to make predictions.

Another set of approaches explored were more concerned with the general case of system shutdown, or deriving multi-state learning models which could be applied to any device or combination of devices. In [18], Srivastava et al. use regression analysis to derive a best-fit to traces of user-activity in X sessions and attempt to do power-saving on a very fine scale (in terms of tens of milliseconds). Hwang and

Wu propose a simple methodology which uses a weighted average of past idle periods. More complex approaches include [11], where Benini et al. suggest a stochastic optimization methodology for deriving an optimal power policy formalized as a Markov model which is not altogether dissimilar from [14] where the authors suggest a scheme based on a continuous-time Markov decision process. Another very complex solution, also due to Benini et al., suggests the use of an adaptive learning tree [4]. In [8], Irani et al. propose a elegant model which can handle an arbitrary number of power-saving states and learns a probabilistic model from experience. The authors also prove the solution is at least two-competitive.

More recent work has started to approach the problem of dynamic power-saving from the perspective of networking devices and mobile systems – it is this work which most directly addresses the problem space we are interested in. In [17], the authors used a small, contrived, dataset to derive a distributional fit for idle times, and then use this to make predictions. In [10], the authors analyze simple one and two-second timeouts for mobile communication, leaving an adaptive approach to later work. Finally, there are a few papers that address dynamic power-saving specifically from the perspective of infrastructure-mode IEEE 802.11 WLANs. In [9], Krashinsky and Balakrishnan present an elegant bounded backoff approach that outperforms the stock 802.11 power-saving scheme, but is not the least bit empirically derived. In [16], Sheth and Han propose a combination of selective radio activation and adaptive power control using a per-server (from the perspective of client traffic) exponentially weighted moving average. Finally, the work of Chen et al. in [3] serves as a representative of newer work which considers power-saving decisions in an application-specific manner (VoIP in their case).

Despite the rich history of approaches, our observation is that none, or very few, of these models take advantage of the small and stochastically distributed idle periods in real user traces. Additionally, most of them are not thoroughly tested against realistic user traces, nor is there much consensus as to which among them is best for any given domain. Indeed, although some specific architectural components are recently benefiting from power-saving schemes (hard-disks and CPUs are prime examples), general dynamic and user-behavior-driven power-saving has not been adopted by industry [2] – in many ways the state of the art is a simple method of powering down components after a long idle period - a scheme which is agnostic to actual user activity patterns. For this reason, we approach a simple but widely applicable subset of the dynamic power-saving problem with a large dataset of real user traces and no initial hypothesis about which method should prevail (besides hoping it is a simple one).

3. EXPERIMENT

In this section, we will discuss the dataset we used, the subset of the general power-saving problem we have targetted, and the various algorithms we implement and analyze.

3.1 Datum

For our dataset, we use 236 unique, publicly available [1], user-traces which were recorded using passive vicinity sniffing techniques in a variety of locations:

- PDX/VWave2006: A collection of traces collected at public hotspots around Portland, Oregon using a VeriWave sniffer with nanosecond resolution timing. The initial characterization of these traces is in [12].
- UW/SigComm2004: A subset of the traces collected by University of Washington researchers at SigComm 2004. Some characterization of these traces is in [15].
- Microsoft/OSDI2006: A subset of traces collected by Microsoft researchers at OSDI 2006. Specifically we used the traces from sniffers S4 and S5 concerning two APs.

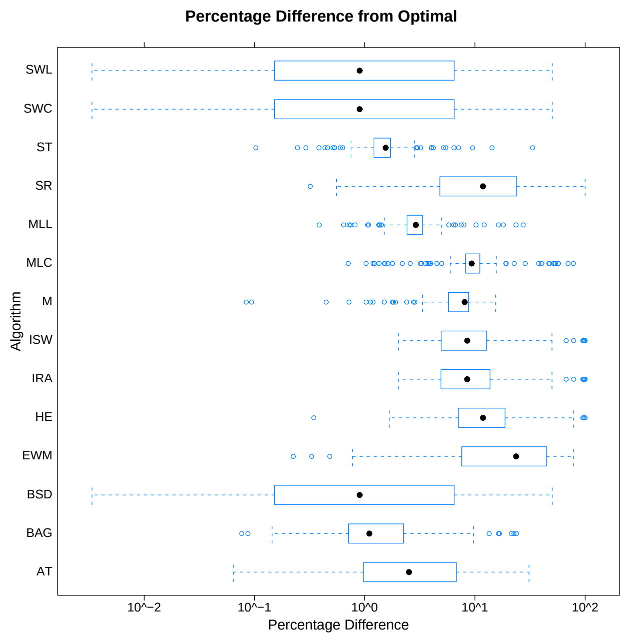
We randomly subdivide these 236 user-traces into a 118 trace training set, which is used to train those dynamic power-saving algorithms which have a training phase, and a 118 trace test set to test all the algorithms. Our test traces range in length from 6 to 34,766 one second buckets, with an average length of 11,057 (3.07 hours). On average, only 8.55% of those buckets are active.

3.2 Algorithms

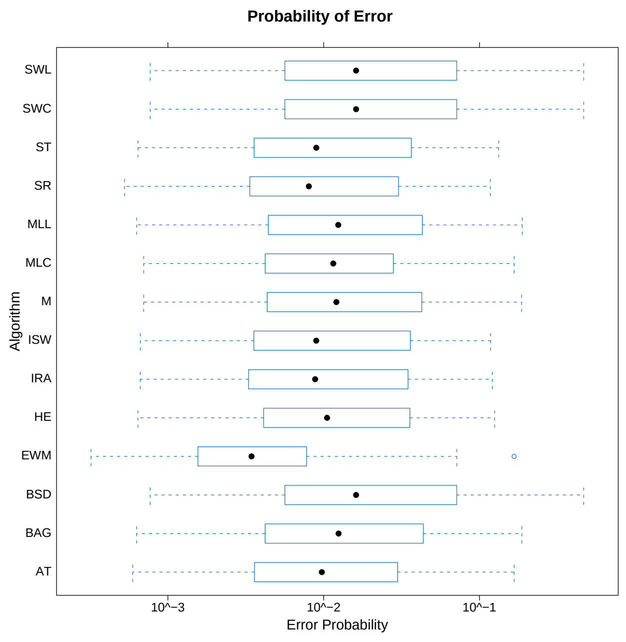
Although one can imagine a power-saving algorithm of almost any complexity, we feel it is useful to use a very common use-case as a benchmark. Specifically, we are interested in a two-state model, where the device can be either *on* or *off*. Additionally, we have discretized our traces into one-second buckets ($t_{bucket} = 1$) and are only interested in predictions on this level of granularity. This bucket size was chosen based on a trade-off between average human motor response times and minimization of effects due to 802.11 DCF contention in the traces [13]. Finally, we make the assumptions that our hypothetical hardware can go to sleep and wake back up in two-seconds ($t_{pd} = 2$) and that one second (i.e., one bucket, $t_{min} = 1$) is the minimum desirable sleep time – these parameters are based on correspondence with industry [2].

We implemented 9 algorithms from the literature, as well as two new algorithms based on our work in [13] (**M**, **MLL**, and **MLC** below). In some cases, the algorithms had to be adapted slightly to fit our problem definition. Following is a brief description of each:

- **M**: An algorithm which predicts the state of the next bucket, using the four-state Markov model presented in [13] and the empirical parameters trained from *sleepless* clients (i.e., only those with power-saving disabled). It makes the prediction by calculating the *worthwhileness* value for a number of seconds we are considering to sleep, x. $w(x)$ is defined as the probability of transitioning to any idle state in both Rx and Tx and for the next $t_{min} + t_{pd} + x$ states also being idle. At each prediction point, we calculate the value of $w(x)$ and choose to sleep if the value is higher than some threshold ($p_{thresh} = 0.582$), which is the optimal value we derived from the training data.
- **BSD**: An algorithm based on the work in [9], but more coarse, with the minimum sleep time set to t_{cost} and the backoff in terms of seconds (not hundreds of milliseconds).
- **MLC** and **MLL**: A modified version of **M** algorithm which adjusts the p_{thresh} in 0.001 increments in response to mistakes. There is a liberal version (**MLL**)



(a) Difference from optimal power-down percentage for each algorithm and each trace.



(b) Error rate for each algorithm and each test trace.

Figure 1: Performance of the Algorithms

and a conservative version (**MLC**). The conservative version starts with the optimal $p_{thresh} = 0.582$. The liberal version starts with a small $p_{thresh} = 0.2$, and grows it to fit the user behavior.

- **SR**: Makes use of the X-session derived regression equation in [18].

- **ST**: A simple timeout approach not dissimilar from those studied in [10]. This one sleeps for 60 seconds after observing a two-second idle period.

- **SWL** and **SWC**: In some sense, this algorithm is the state of the art for system-shutdown. In the liberal case (**SWL**), we sleep permanently (or until woken by the user) after 2 minutes of idle time - this corresponds to the most aggressive setting in the Microsoft Windows Vista operating system. The conservative version (**SWC**), which corresponds to the Windows Vista default, sleeps permanently after 5 minutes of idle time.

- **IRA** and **ISW**: A probabilistic power-saver based on the work in [8]. Our implementation differs from that presented by the authors in that it does not reason about the power-cost of switching states, and simply uses the windowed histogram they suggest to make a prediction about idle period length. We tested the algorithm both with a global memory (**IRA**), and a windowed memory of 100 buckets (**ISW**).

- **AT**: A very complex algorithm from [4]. It is a combination of tree learning, with a lower-envelope power-function, and a branch-prediction-like saturating confidence measure. Their original method is for an arbitrary number of states. Here we have implemented the trivial case of 2 power states - off and on. In this case the behavior of the algorithm is to use the learning tree to decide whether to sleep based on a global history and then to sleep for 1 second, wake up for 2 seconds and then go to sleep forever (presuming there is no active period in there). The learning tree has a branching factor of the number of states and a maximum depth of the global history. Hence, a system with 5 power states and an hour of history might have 5^{3600-1} leaves.

- **BAG**: A simple backoff-based power-saver using the work in [5]. Golding et al. found that the best performance in their tests came from an arithmetic backoff that decreases geometrically on error. Hence, that is what we have implemented. It should be noted that [5] is focussed on hard-disk spin-down, so while the algorithm applies, their dataset (and therefore conclusions) might not fit our problem.

- **EWM**: An algorithm from [16] which makes predictions using an exponentially weighted moving average (and an exponentially weighted moving deviation). If the device is still idle after the predicted idle period (*timeout*), then it sleeps with an exponential back-off. The authors suggest using per-server statistics and taking the maximum timeout, but we are using an aggregate statistic.

- **HE**: An exponential averaging approach proposed by Hwang et al. in response to Srivastava et al. [7].

Each algorithm is analyzed in terms of its effectiveness with respect to two variables: *performance gap*, which is the difference in percentage of buckets powered-down from the optimal algorithm (which always makes the best decision) and *error probability*, the probability that the algorithm sleeps too long and the user must intervene to wake it up. These metrics correspond to Type I and Type II error respectively. An overzealous algorithm will suffer from a large error probability, while an overly conservative algorithm will have a large performance gap. A good algorithm is characterized by both a small performance gap and a small error probability.

4. RESULTS

Figure 1(a) shows the performance gap of each algorithm with respect to the optimal. We can see that many algorithms succeed in power-saving much of the time. Clear winners are the simple timeout algorithm (**ST**), liberal learning version of our algorithm (**MLL**), sleep-until-woken (**SWL** and **SWC**), simple backoff-based algorithm (**BAG**), and adaptive tree algorithm (**AT**). However, this metric must be carefully weighed against the probability of error, which is shown in figure 1(b).

Among those algorithms which performed well in terms of performance gap, the least erroneous are the simple timeout (**ST**), the backoff-based algorithm (**BAG**), our simple learning algorithm (**MLL**), and the adaptive tree algorithm (**AT**). The mean error rates are 0.0232, 0.031, 0.030, and 0.0231 respectively. Recall that in the two-state formulation and with our givens (t_{pd}, t_{min}), the **AT** algorithm uses its predictor to decide to sleep for 1 second, wakes up for two seconds, and then goes to sleep forever. Indeed, for all its additional complexity, the **AT** algorithm is only very slightly better performing than the extremely simple timeout algorithm (**ST**).

5. CONCLUSION

This paper considers the question: can better and simpler device-level power management be achieved by using models of user activity? Previous approaches have either been too simplistic (long, arbitrary timeouts as used in most laptops and other devices today) or too complicated (that maintain significant state in order to make decisions). We have shown that very complex algorithms only show marginal improvement over simpler algorithms. In fact, a simple short timeout performs as well as nearly all the other algorithms. We contend that in many cases, an appropriately sized timeout, will be competitive with even the most complex learning algorithms and will outperform long arbitrary timeouts. Additionally, we observe an understanding of underlying user activity patterns is necessary in order to make appropriate decisions in the design of dynamic power management algorithms.

6. REFERENCES

[1] Crawdad. http://crawdad.org, Feb 2008.

[2] Private communication with intel, hillsboro. 2008.

[3] Y. Chen, N. Smavatkul, and S. Emeott. Power management for voip over ieee 802.11 wlan. *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, 3:1648–1653 Vol.3, 21-25 March 2004.

[4] E.-Y. Chung, L. Benini, and G. De Micheli. Dynamic power management using adaptive learning tree. *Computer-Aided Design, 1999. Digest of Technical Papers. 1999 IEEE/ACM International Conference on*, pages 274–279, 1999.

[5] R. Golding, P. Bosch, C. Staelin, T. Sullivan, and J. Wilkes. Idleness is not sloth. In *TCON'95: Proceedings of the USENIX 1995 Technical Conference*, pages 17–17, Berkeley, CA, USA, 1995. USENIX Association.

[6] K. Govil, E. Chan, and H. Wasserman. Comparing algorithm for dynamic speed-setting of a low-power cpu. In *MobiCom*, pages 13–25, New York, NY, USA, 1995. ACM.

[7] C.-H. Hwang and A. C.-H. Wu. A predictive system shutdown method for energy saving of event-driven computation. *ACM Trans. Des. Autom. Electron. Syst.*, 5(2):226–241, 2000.

[8] S. Irani, S. Shukla, and R. Gupta. Online strategies for dynamic power management in systems with multiple power-saving states. *Trans. on Embedded Computing Sys.*, 2(3):325–346, 2003.

[9] R. Krashinsky and H. Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. *Wirel. Netw.*, 11(1-2):135–148, 2005.

[10] R. Kravets and P. Krishnan. Power management techniques for mobile communication. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 157–168, New York, NY, USA, 1998. ACM.

[11] G. A. Paleologo, L. Benini, A. Bogliolo, and G. D. Micheli. Policy optimization for dynamic power management. In *Design Automation Conference*, pages 182–187, 1998.

[12] C. Phillips and S. Singh. Analysis of wlan traffic in the wild. In *IFIP Networking*, 2007.

[13] C. Phillips and S. Singh. An empirical activity model for wlan users. In *IEEE INFOCOM MiniSymposium*, 2008.

[14] Q. Qiu and M. Pedram. Dynamic power management based on continuous-time markov decision processes. *DAC*, 00:555–561, 1999.

[15] M. Rodrig, C. Reis, R. Mahajan, D. Wetherall, and J. Zahorjan. Measurement-based characterization of 802.11 in a hotspot setting. In *E-WIND*, pages 5–10, New York, NY, USA, 2005. ACM.

[16] A. Sheth and R. Han. Adaptive power control and selective radio activation for low-power infrastructure-mode 802.11 lans. *Distributed Computing Systems Workshops*, pages 812–818, 19-22 May 2003.

[17] T. Simunic, L. Benini, P. W. Glynn, and G. D. Micheli. Dynamic power management for portable systems. In *Mobile Computing and Networking*, pages 11–19, 2000.

[18] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Trans. Very Large Scale Integr. Syst.*, 4(1):42–55, 1996.

[19] J. Wilkes. Predictive power consumption. Technical report, Hewlett-Packard Laboratories, 1992.