Choosing Beacon Period for Improved Response Time for Wireless HTTP Clients

Zachary Anderson Suman Nath Srinivasan Seshan

Department of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15232
{zra, sknath, srini}@cs.cmu.edu

Abstract

A significant contributor to the total energy consumed on many battery-powered mobile devices is the wireless network interface card (NIC). Therefore, minimizing the energy consumption of the NIC can result in significant battery life improvement for a mobile device. One way of achieving this is to transition the NIC to a lower-power sleep mode when possible For example, the IEEE 802.11 wireless LAN standard power-saving mode (PSM) makes access points buffer data destined for a client and uses a periodic beacon to inform the mobile client about any buffered data. This allows the NIC to enter sleep mode between beacons.

In this paper, we show that 802.11 PSM performs poorly due to the fact that an access point is unable to adapt to the requirements of each client. Therefore, we propose a novel power saving algorithm, PSM-Dynamic, where the access point uses different beacon periods for different clients. During HTTP downloads, each client carefully chooses a good beacon period for itself, based on the RTT of its current connections, and informs the access point of this beacon period. This technique enables download times for Web pages that are comparable to those without any power-saving and provides energy savings comparable to the standard 802.11 PSM. We show, using real-world easements and results from emulation-based experiments, that it is feasible for both clients and access points to efficiently support such per-client beacon periods, instead of having a common, static beacon for all clients. The solution is simple enough that it can be implemented with just small enhancements to the existing 802.11 specification.

1 Introduction

The wireless network interface card (NIC) is a significant contributor to the total energy consumption on many battery-powered mobile computing devices. As a result, minimizing the energy usage of the NIC can significantly improve the battery life of a mobile device. One way to address this issue is to transition the NIC to a lower-power *sleep* mode when data is not being received or transmitted. For example, the popular IEEE 802.11 wireless LAN standard specifies a power-saving mode (PSM) that periodically turns the NIC off to save energy, and on to communicate [7]. When 802.11 PSM is enabled, the access point buffers data destined for the client[1]. At the end of every

[1]We use the terms *mobile device* and *client* interchangeably.

beacon period, which is typically 100ms but can be set to a multiple of 100ms, the client's NIC wakes up to receive beacon from the access point. A beacon contains information about the data buffered by the access point for the client while it's NIC has been off. If the client finds that the access point has data buffered for it, it pulls that data.

Past work [8] has shown that the static power-saving algorithm used by 802.11 PSM (we call it PSM-STATIC hereafter) is not optimized for browsing the Internet. First, the static beacon period is *too coarse-grained* for most Web accesses, which download small objects through TCP connections. For them, the choice of a 100 ms (or multiples of 100ms) beacon period can cause observed round-trip times (RTTs) to be rounded up to the nearest multiple of the beacon period. This rounding effect can hurt performance dramatically, by increasing the RTT significantly. Second, PSM-STATIC is *too fine-grained* when the NIC is mostly idle, and forces the NIC to wake up frequently and consume energy. Finally, PSM-STATIC exhibits a *performance inversion effect* where, under some conditions, the time to download a file *increases* when the bandwidth of the wireless link increases.

We claim that one major reason for this suboptimal performance of PSM-STATIC is that an access point is unable to adapt to the requirements of each client. For example, consider two clients A and B downloading Web pages from two servers having RTTs[2] of 20ms and 200ms respectively. In this case, a static and common beacon period of 100ms would result in suboptimal download time for client A and a suboptimal energy consumption for client B. Note that even if PSM-STATIC uses a smaller beacon period, like 10ms[3], it may ensure optimal download times for both clients, but it will incur unnecessarily high energy overhead for client B. In an optimal strategy, the access point would use two different carefully chosen beacon periods for these two clients (a smaller one for client A and larger one for client B) that minimizes power and maximizes performance for each client.

In this paper, we propose a novel power saving algorithm, named PSM-DYNAMIC, where the access point uses different beacon periods for different clients. During HTTP downloads, each client carefully chooses a *good* beacon period for itself, based on the RTT of its current HTTP connection, and informs the access point of it. Like PSM-STATIC, the NIC sleeps between the successive beacon periods, and the access point buffers any incoming data that arrives during this period. The access point sends beacons and data to each client separately, with a period set by the client. As we will show later, this allows individual clients to independently optimize their Web page download times without significantly increasing their energy overheads. Note that this approach is significantly different from existing solutions [8, 19] which are mostly client-centric. The Bounded Slowdown algorithm [8] attempts to avoid listening to useless beacons by listening to beacons with decreasing frequency after the mobile host has sent any data. This does not save energy during slow start (the actual download time for most HTTP transfers), and most of the savings come from long think times. The client centric approach described in [19] involves guessing packet arrival times during slow start. However, it does not use the access point to buffer packets arriving when the client's NIC is sleeping. Thus, if RTT variance is reasonably large, then packet arrival will not be predicted correctly, and packets will be dropped, thereby, greatly hurting performance.

The design of PSM-DYNAMIC raises two practicality concerns: (1) Can a client choose a *good* beacon period for itself? and (2) Can an access point efficiently manage multiple beacon periods

[2]We use the term RTT to denote the sum of network level RTT and the time required by the server to generate a packet.

[3]This is a hypothetical value; in 802.11 PSM, the beacon period is always a multiple of 100ms.

for the clients? By analyzing traces collected from several clients and access points in the Carnegie Mellon University campus, and through extensive experimentation, we show that both the questions can be answered affirmatively. First, the RTTs of Web servers remain relatively stable over the short period HTTP download time, and the clients can use their RTTs to choose a reasonably good beacon period.[4]. Second, even with a large user population, the number of clients using an access point simultaneously and the number of concurrent connections is relatively small, and, hence, it is feasible for the access point to use different beacon periods for different clients without high overhead.

In summary, we make the following contributions in this paper.

- We propose PSM-Dynamic, a power saving algorithm for using wireless access points. It allows the clients to choose their own beacon periods which significantly reduces the download times of Web pages, with a comparable energy consumption as PSM-Static. We should note that we never consider the energy consumption of the rest of the device. However, it should be obvious that completing work more quickly would allow the remainder of the device to also enter any appropriate idle mode, thereby, increasing the energy savings of PSM-Dynamic.
- We show, using real workload measurements and results from emulation-based experiments, that it is feasible for both clients and access points to support per-client beacon periods, instead of having a common, static beacon period for all clients. We show that our solution can be implemented with a small enhancement to the existing 802.11 specification.

The rest of the paper is organized as follows. Section 2 discusses the effect of beacon periods on HTTP download. Section 3 describes our proposed algorithm PSM-Dynamic. PSM-Dynamic raises a few practical concerns that we discuss in Section 4. Section 5 presents our evaluation of PSM-Dynamic. We present related work in Section 6 and finally conclude in Section 7.

2 Effect of Beacon Period on HTTP Download

To understand the effect of different beacon periods on HTTP download to mobile devices, we use a combination of simulation and real-world experiments. The metrics we use to quantify performance are the average amount of time used to download an entire Web page and the average amount of energy used for the download. These metrics reflect the desired goals of the user: fast downloads and long battery life.

2.1 Simulation Performance

We first perform simulations in the `ns2` [16] network simulator to get insight into the effect of beacon periods under a controlled environment. We use the implementation of the 802.11 PSM used in [8].

Figure 1 shows the simulation topology consisting of a wireless client, WL, which is connected to the access point, AP, and is downloading HTTP from Web server W_{n+1}. Five additional clients (CC_1 - CC_n) perform concurrent HTTP downloads from the Web servers (W_1 - W_{n+1}) to introduce

[4]Note that even if the RTT of a Web server has high variance, buffering at the access points helps PSM-Dynamic avoid dropping packets when the client is in the sleep mode (unlike the pure client-centric approach in [19] where packets get dropped when the packet arrival time is not predicted correctly).

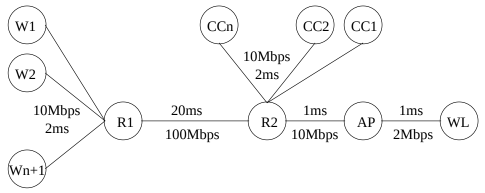


Figure 1: Simulation network topology

Mode of the WNIC	Power Consumption (mW)
Sleep	99
Listen	759
Idle	660
Transmission	1089

Table 1: Power consumption of the NIC in different operating modes. The numbers are provided by the Dell MiniPCI TrueMobile WirelessLAN 1150 NIC documentation [1].

cross traffic in the link between the nodes R_1 and `R2`. The simulation runs for 200s during which time each client WL and CC_i downloads a series of HTTP objects whose size is given by an HTTP traffic generator based on empirical data from [12]. For each configuration, we report the average of 10 experimental runs.

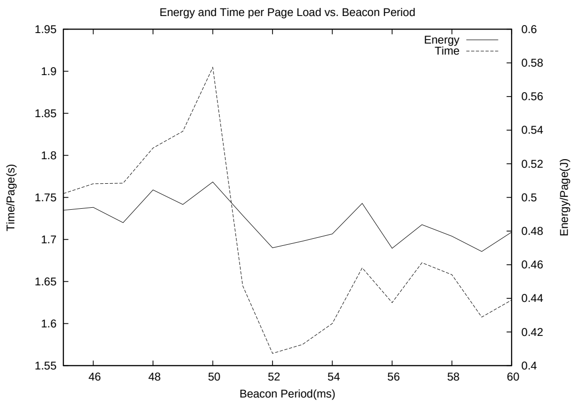


Figure 2: Energy consumption and download time per page vs. beacon period

The download time of an HTTP request is computed as the time difference between when the request is issued and when the download is completed. The energy consumed for a download is computed by analyzing the packet level simulation trace and using the NIC power consumption numbers from Table 1.

Figure 2 shows the effect of varying the beacon period on download time and energy consumption. It shows that both the energy use and time per page load reach a local minimum at a beacon period of 52ms which is a few milliseconds larger than the round trip propagation delay (48ms). Intuitively, it might seem that the optimal beacon period should be equal to the propagation RTT, however, in practice, it should be set slightly higher than the RTT to cope with the variability of

Scheme	Avg. Time/Page Load(s)	Avg. Energy/Page Load(mJ)
No PSM (`NOPSM`)	1.44	860
Optimal (`OPT`)	1.44	580
52ms Beacon Period (`PSM(52ms)`)	1.56	470
100ms Beacon Period (`PSM(100ms)`)	1.87	460

Table 2: Energy and time statistics

the RTT (e.g., due to queueing delay). With this slightly longer beacon period, when the client wakes up again to receive a beacon, packets will either just be arriving from the server, or will, at worst, been buffered for a very short period of time. Very rarely will a client wake up and find that packets from the server have yet to arrive; whereas, such occurrences might be frequent with a slightly smaller beacon period.

Table 2 summarizes simulation results with a number of different schemes. `NOPSM` denotes the scheme 802.11 uses when the power saving mode is off. `PSM(x)` denotes the 802.11 PSM mode, with a beacon period of x. Finally, `OPT` denotes a (hypothetical) optimal scheme that has complete knowledge of future packet arrivals and, hence, wakes the NIC exactly when there are packets to send or receive. Note that `OPT` is not a practical scheme and it is used only for comparison purposes. Figure 2 shows that using a beacon period of 52ms (instead of that of 100ms, as used as default in the 802.11 PSM) provides a near optimal performance in terms of both download time and energy consumption.

2.2 Measured Performance

This section describes the results of experiments using real Internet traffic[5]. We have two objectives for this set of experiments. First, we want to validate the observations we made in the previously described simulation results. Second, we want to see whether a single beacon period performs well for downloads from any Web site. The second issue is important, since an affirmative answer would allow us to use the existing 802.11 PSM, however, possibly with a new beacon period.

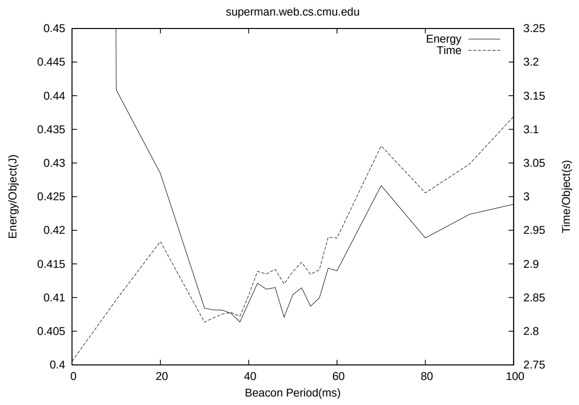


Figure 3: Accessing `superman.web.cs.cmu.edu` with various beacon periods.

[5]A detailed description of the experimental setup will be described in Section 5.

Validation of Simulation Results. In this experiment, we access the Web site `http://superman.web.cs.cmu.edu`, using several different beacon periods between 0 and 100. The average RTT to this Web site from the client is 45ms with a standard deviation of 20ms. The results show a similar trend as the previous simulation results. As shown in Figure 3, the HTTP download is optimized when the beacon period is chosen to be 48ms which is slightly higher than the RTT. Table 3 summarizes the results and shows that a 48ms beacon period provides near optimal performance in terms of both the download time and the energy consumption. Interestingly, `PSM(48ms)` consumes less energy than `PSM(100ms)` which is explained by the fact that the latter has a longer download time and NIC consumes energy even when it is in sleep mode.

Scheme	Avg. Time/Page Load(s)	Avg. Energy/Page Load(mJ)
No PSM (`NOPSM`)	2.75	1930
Optimal (`OPT`)	2.75	390
48ms Beacon Period (`PSM(48ms)`)	2.87	410
100ms Beacon Period (`PSM(100ms)`)	3.12	420

Table 3: Performance of different techniques for accessing `superman.web.cs.cmu.edu`

Variability of the Optimal Beacon Periods. To see how the optimal beacon periods for different Web sites vary, we conduct the above experiment for all the top 100 Web sites given by Alexa [3] (detail experiment setup is in Section 5). For each site, we find the optimal beacon period that minimizes the product of average energy used per object and average time spent per object. Figure 4 shows the cumulative frequency distribution of optimal beacon periods. The data demonstrates that the optimal beacon period varies widely from Web site to Web site; and, hence, a single static beacon period can not provide optimal performance for all clients (who are downloading Web pages from multiple sites).

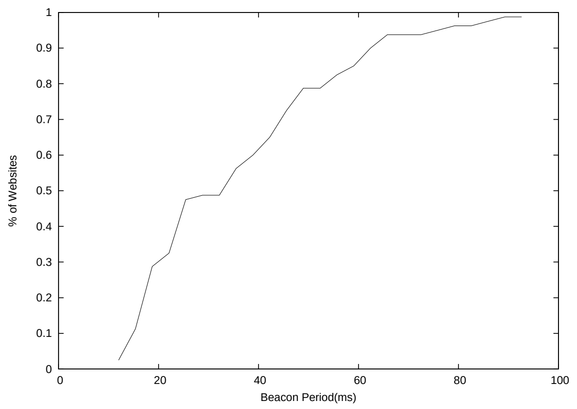


Figure 4: CDF of beacon periods that minimize the product of average energy used per object and average time spent per object for each of the Alexa top 100 Web sites.

One observation we make from these experiments is that the optimal beacon period for each site is slightly larger than its RTT from the client. Even if we set the beacon period to be the estimated RTT to the server, we achieve near optimal performance. Table 4 shows the performance of different schemes for downloading the Alexa top 100 Web sites, relative to a scheme that uses separate beacon periods for each site, set to be the estimated RTT of the site.

Scheme	% Time Difference	% Energy Difference
No PSM (`NOPSM`)	$0\% \pm 4\%$	$79\% \pm 4\%$
Optimal (`OPT`)	$0\% \pm 4\%$	$0\% \pm 10\%$
100ms Beacon Period (`PSM(100ms)`)	$14\% \pm 13\%$	$9\% \pm 10\%$

Table 4: Performance of different techniques for accessing Alexa Top 100 Web Sites. Results are relative to using a beacon period slightly longer than the propagation RTT.

The result of the experiment indicates that both using an RTT-based beacon period or a 100ms beacon period saves about the same amount of energy. However, the method using a RTT-based beacon period has roughly the same response time as `NOPSM` or `OPT`.

2.3 Summary

In summary, our results show the following:

- The default 100ms beacon period for 802.11 PSM is suboptimal. It is possible, by using a different beacon period, to reduce the download time while consuming no more energy than 802.11 PSM.
- The optimal beacon period is different for different sites. A near optimal beacon period can be chosen based on the measured RTT of the server from where the page is being downloaded.

In the rest of the paper, we describe and evaluate a power saving scheme, named PSM-Dynamic, that is motivated by observations above. PSM-Dynamic allows each mobile host to choose its own beacon period to achieve near optimal performance.

3 The Dynamic Power Saving Algorithm

The dynamic power saving algorithm (PSM-Dynamic) differs from the standard 802.11 PSM (PSM-Static) in that the access point may maintain different beacon periods for different clients currently sending or receiving data. For simplicity, we now assume that the beacon periods can be arbitrary. Later, we will relax the assumption and show how PSM-Dynamic can be incorporated into the 802.11 specification with only a minor modification.

Figure 5 shows the core of the PSM-Dynamic algorithm run by a mobile device. After initiating a HTTP connection, the mobile device computes the RTT of the server, using a technique we describe later. It then determines a good beacon period for itself based on the RTT value and informs the access point (AP) of this beacon period. Thus, the APs have different beacon periods, BP_i, for different active (i.e., currently sending or receiving data) clients, i, at any point of time. At the end of a beacon period BP_i, the AP sends a beacon containing a traffic indication for the client i, followed by the data destined for the client. Each active client i wakes up at every beacon period BP_i and polls the AP to receive any buffered data. Note that the beacon period BP_i is essentially a prediction of when the next packet will arrive for the client i. Even if the prediction is not perfect (e.g., due to RTT variability), the AP will buffer the packets until the client wakes

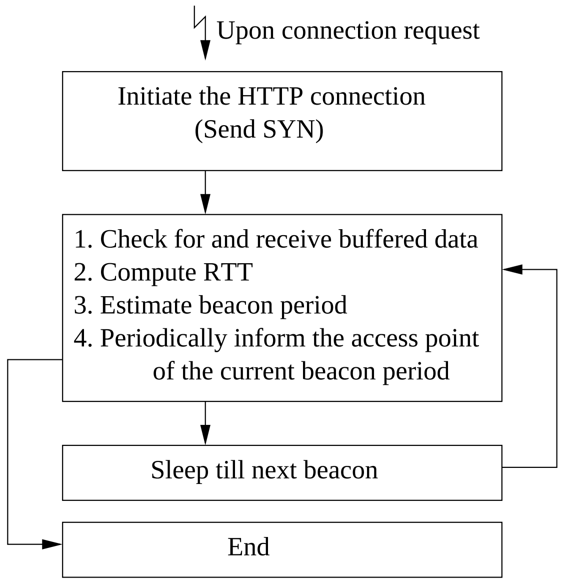

Upon connection request
Initiate the HTTP connection (Send SYN)
1. Check for and receive buffered data
2. Compute RTT
3. Estimate beacon period
4. Periodically inform the access point of the current beacon period
Sleep till next beacon
End

Figure 5: The dynamic power saving algorithm.

up. When mobile client i does not have any ongoing HTTP connection, it sets BP_i to be a large value (3s in our evaluation).

Ideally, a client will be awake as soon as the AP has received packets destined for it. However, this requires a perfect prediction about when the next packets will arrive, and setting the beacon period accordingly. Unfortunately, such accurate predictions are impossible due to the high variance of the RTT, resulting from congestion, loss, etc. The effectiveness of PSM-Dynamic largely depends on the ability to choose a good beacon period. In the next section, we address how PSM-Dynamic achieves this goal.

3.1 RTT Estimation

It is difficult to estimate RTTs at the receiver side because it is hard to associate incoming packets with the outgoing acknowledgement that triggers them [11]. Fortunately, TCP provides a timestamp option[6] that can be used to measure RTT from either side when both the sender and receiver agree to use the option (note that [19] uses the same technique). Once enabled, up-to-date timestamps are always sent and echoed in the TCP header of each packet. Upon receiving a packet, either endpoint can calculate a new RTT sample as the time difference between the current timestamp value and the echoed value.

However, it is not possible to get an accurate estimate of the RTTs when packets get buffered at the AP before being delivered to the client, since an unknown amount of buffering time at the AP is added to the RTT. To overcome this issue, we require that the AP tags each packet delivered to the client with the amount of time it has been buffered at the AP. Adding this functionality is relatively easy. On receiving a packet from the server, AP tags it with the current timestamp t_1; and before delivering the packet to the client at time t_2, AP tags each packet as $(t_2 - t_1)$. Thus,

[6]TCP uses these accurate RTT samples to improve the quality of the TCP timeout value (RTO), which, in turn, improves TCP performance. As a result, the TCP timestamp option is used in most TCP implementations [17].

the client can subtract the buffering times from the sample RTTs.

The RTT estimate is maintained in the same way that many implementations of TCP maintain it: as an exponential weighted moving average ($\texttt{ERTT} = \frac{7}{8}\texttt{ERTT} + \frac{1}{8}\texttt{SRTT}$, where `ERTT` is the RTT estimate, and `SRTT` is an RTT sample). The client may periodically inform the access point of the RTT estimate so that a good beacon period can be used for each connection.

Furthermore, the client may keep a cache of RTT estimates so as to avoid calculating an estimate on the fly. In the absence of an entry in the cache for a particular remote host, the client may leave its NIC in the higher power state so that an RTT estimate can be calculated–based on the time between sending a SYN packet and receiving an ACK packet – before informing the access point of the desired beacon period.

3.2 Beacon Period Estimation

As shown by the experimental results presented in Section 2, the optimal beacon period for a client is very close to the RTT of the Web site it is downloading the Web page from. In practice, we use beacon period = $\alpha\times$ RTT, where α is a constant slightly larger than 1. This allows PSM-Dynamic to cope with small variability of the estimated RTTs. We will evaluate the effect of the values of α in Section 5.

3.3 Supporting Concurrent Connections

The techniques described above assume that a client has at most one ongoing HTTP connection at any point of time. However, Web browsers generally issue multiple concurrent connections to possibly multiple sites to retrieve embedded objects. PSM-Dynamic uses the following two techniques to support multiple concurrent connections. First, each client independently measures the RTT and uses it to estimate the next packet arrival time for each connection. Second, it dynamically sets its beacon period to the difference between the current time and the *soonest* estimated packet arrival time among all its connections. This essentially emulates the behavior of multiple independent beacon periods.

3.4 Granularity of the Beacon Period

So far in the discussion, we have assumed that each client can have the beacon periods for its connections at an arbitrary granularity. However, it may impose overhead on both the clients and the access points who may need to track many concurrent connections. One optimization we use in PSM-Dynamic to reduce this overhead is to have beacon periods chosen at a coarser granularity – i.e., the beacon period is given as the next closest multiple of the granularity. For example, if the beacon period granularity is 30ms, then two connections having RTTs of 55ms and 48ms respectively will both have a single beacon period of 60ms (thus, a granularity of 100ms emulates the behavior of PSM-Static). It reduces the number of distinct beacon periods the clients and the access points need to keep track of. We will evaluate the effect of the granularity of beacon period in Section 5.

3.5 Enhancing the 802.11 Protocol to Support PSM-Dynamic

PSM-Dynamic requires a small modification to the standard 802.11 protocol. The 802.11 specification already allows for a *ListenInterval* which defines the number of beacons a client can skip.

Each client can be configured to have its own ListenInterval. Thus, if a client has ListenInterval=2, it wakes up to listen to every third beacon and the AP makes sure that buffered packets destined for this client are delivered only on every third beacon. With this feature, updating the existing 802.11 to support PSM-Dynamic should be straightforward. It only requires that the AP should have a fixed but smaller (e.g., 10 ms) beacon period (the current specification suggests it to be a multiple of 100ms). With this change, a client can choose a good beacon period (rounded to the nearest multiple of AP's static beacon period) for itself by dynamically configuring it with a suitable value of ListenInterval. Note that the static beacon period of the AP defines the granularity at which each client can decide when it wakes up to receive packets, but as we will show in our evaluation, even a granularity of 20 ms provides a significantly better performance than PSM-Static.

Despite the fact that this is a conceptually simple modification to the standard 802.11 PSM, there are a couple of complications. Questions still remain about the variance of RTT, and the effects of increased communication and computation load on access points. These concerns will be addressed in the next section.

3.6 Supporting Non-TCP connections with PSM-Dynamic

In our description of PSM-Dynamic, we have concentrated on using the TCP RTT to guide the setting of the beacon period. However, the core requirements for PSM-Dynamic are: 1) support for fine grain beacons and 2) predictable traffic patterns. In TCP, we use the fact that groups of packet arrivals are typically an RTT apart to predict the next time that the NIC needs to be awake to receive packets. For other traffic, such as constant-bit-rate (CBR) audio/video, we can create similar predictions for packet arrival times. We envision that a PSM-Dynamic implementation would include a set of standard APIs for transport protocols and applications to register their predictions for their next packet arrival. PSM-Dynamic could then merge the predictions and determine the appropriate beacon period.

4 Practical Considerations

This section addresses two concerns that must be responded to in order for PSM-Dynamic to be practical. First, it is not clear whether a client can predict when data will arrive accurately enough to set the beacon period. Second, the PSM-Dynamic may incur high overheads if there are a large number of concurrent transfers by different clients of the same access point. We show that both these concerns can be addressed satisfactorily.

4.1 RTT measurements

The effectiveness of PSM-Dynamic relies on the accuracy of the prediction of when more data will be available at the access point, which in turn relies on the accuracy of the client's estimate of the current RTT. To determine how the RTT of a typical HTTP transfer varies over the duration of the transfer, we download the top 100 Alexa Web pages from a laptop behind a 3Mbps line. The RTTs were measured over the course of several `HTTP` connections while loading the front page of the Web sites and all top level objects. A typical cumulative frequency distribution (specifically, for the site `http://www.nature.com`) is shown in Figure 6. The average RTT was 57ms and the standard deviation was 37ms for this site. Note that 80% of the RTTs were less than 70ms. This indicates that using a 70ms beacon period would work well for this site – the client would rarely wake up to find no packets waiting for it.

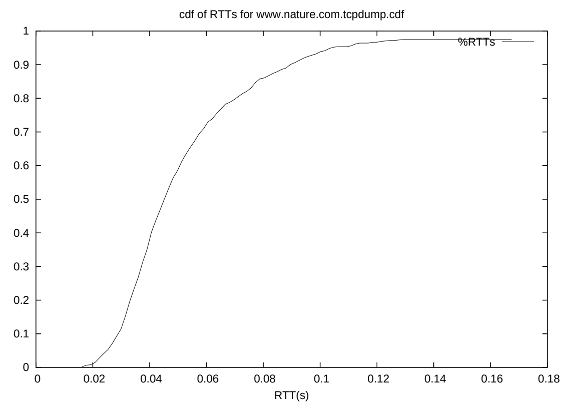


Figure 6: The CDF of RTTs to www.nature.com.

Our experimental results show that the variance of RTTs for most of the tested Web sites are small enough ($< 30ms$) that a beacon period slighly longer than the RTT would work well for PSM-Dynamic. Note, that the variance is certainly large enough that occasional errors are likely (i.e., the client could wake up to find that it has missed the packet transmission). Note that even if the prediction of the beacon period is not accurate, the access point buffers the packets while the destination client's NIC is sleeping and, therefore, there is no packet drop. This is one of the important differences between PSM-Dynamic and the client centered approach in [19]. The client centric approach does not use the access point for buffering and, hence, even slight errors in predicting packet arrivals results in packets being lost since they arrive when the client's NIC is sleeping.

4.2 Access Point Population

PSM-Dynamic requires that an access point maintain separate beacon periods for each of the clients that have outstanding connections. Maintaining separate beacon periods for each client imposes additional overhead on the access point, since it must maintain states associated with each client. Here, we show, by analysis of a trace collected from a busy collection of access points, that, in practice, each access point generally has a small number of clients downloading at the same time[7].

We have collected access point population data from 18 access points over the course of a work week from the Graduate School of Industrial Administration building on the Carnegie Mellon University campus. The data shows the number of registered users for each access point at different points of time in that work week. Figure 7 shows a CDF of the access point populations. Over half the time the access point was not populated at all or the population was only one. In addition, 90 percent of the time an access point was populated with fewer than 10 clients. It should be noted that population provides an upper bound on load on the access point; the actual number of concurrent connections will be less than or equal to the population.

[7]Note that PSM-Dynamic does not care about the total population registered to an access point at any time; it only cares about how many clients are downloading at the same time.

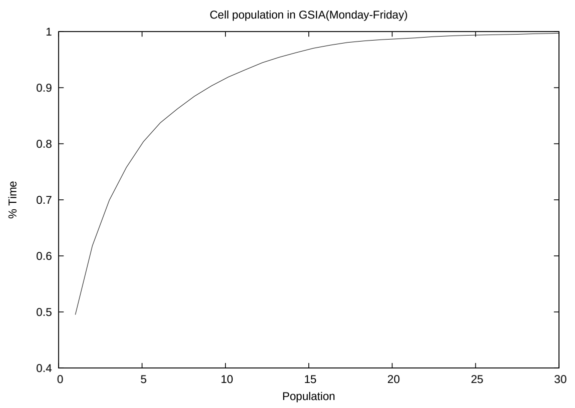


Figure 7: The CDF of average access point populations.

The increased traffic caused by additional beacons is acceptable because the access point population is typically small, and the per connection beacons may consist of only a few bytes. Also, in the rare case that the access point population is large, the access point could be configured to revert back to using the same beacon period for all the clients (similar to PSM-STATIC). This beacon period need not be the standard 100ms, but could be a value based on the beacon periods being requested by the clients. Also, access points could enforce a minimum beacon period in order to keep the wireless channel from being flooded with beacons.

5 Evaluation

In this section, we compare PSM-DYNAMIC with a number of existing algorithms in terms of the download time and energy consumed. First, we explore the impact of various tuning parameters on PSM-DYNAMIC's performance. We use this evaluation to determine parameters to use in the rest of the evaluation. Second, we consider the performance of a variety of schemes in accessing actual Internet Web sites. Finally, we explore the sensitivity of the algorithms to various factors in a laboratory setting.

5.1 Evaluation Methodology

This section describes our evaluation methodology. We discuss the algorithms we compare with PSM-DYNAMIC, the metrics we use to quantify their performance, and the setup of our experiments, in turn.

Algorithms Compared. We compare PSM-DYNAMIC (also called `DBP`, for Dynamic Beacon Period, from now on) with the following five algorithms.

1. `BSD` (bounded slowdown protocol, described in [8]). `BSD` attempts to avoid listening to useless beacons by listening to beacons with decreasing frequency after the client has sent any data. We use a default beacon period of 100ms, and a slowdown parameter of 1/2 in our evaluation.

2. `CC` (The client-centric approach, described in [19]). `CC` guesses the future packet arrival times and awakens the NIC only at that time. Data is not buffered at the access point and, hence, any packets that arrive while the client's NIC is in sleep mode get dropped.

3. `NOPSM`: The standard 802.11 protocol, with the power saving mode off.

4. `SBP`: The standard 802.11 protocol, with the power saving mode on and with a static beacon period of 100ms.

5. `OPT`: An oracle, that knows all future packet arrivals, and lets the NIC wake up only when the packet arrives. It should provide the same download time as, but should be more energy efficient than `NOPSM`. Note that because of the requirement of knowing the future, this algorithm is not practical. We present it only for comparison with the other algorithm.

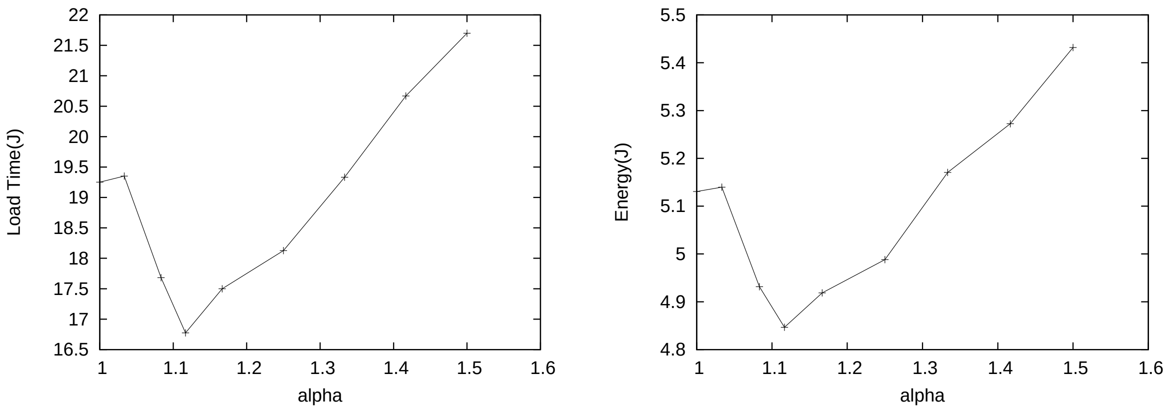
We have implemented all the algorithms (except `OPT`) in Linux kernel module. The implementations use Netfilter [2], a generalized framework of hooks in the network stack of Linux. Our code implements queues and timers to emulate the buffering and beaconing at the access point (for `SBP`, `DBP`, and `BSD`). It also drops packets for individual clients to emulate `CC`'s behavior when packets arrive at the client's NIC while it is in sleep mode. Finally, we use the KURT real-time patch [9] to provide the high resolution timing necessary for accurate reproduction of real behavior. For `OPT`, we use `tcpdump` to log all the packets sent or received by the client and later analyze the logs to infer the optimal performance.

Metrics. We use two metrics in our evaluation: *download time* and *energy consumed*. The download time metric for a Web page is defined as the time elapsed between the transmission of a HTTP GET request for the HTML page and completion of the download of the page HTML page and all its embedded objects to the client. The *energy consumed* metric is defined as the energy used by the client's NIC to send and receive all the packets relevant to the downloaded page. To measure this energy, we log all packets sent and received by the client using the `tcpdump` tool. We compute the total energy used by post-processing the log using the power consumption numbers given in Table 1.

Experimental Setup. We use two experimental setups in our evaulation: a real-world setup and laboratory emulation setup. All the experiments are done from the Carnegie Mellon University (CMU) network. We use a 1.8 GHz Pentium IV laptop with 512 MB RAM and running Linux 2.4.18 as the client accessing the Web pages. The bandwidth of the access link to the test machine was 10Mbps.

Real-world Experiment Setup. In experiments with real Internet traffic, we download the top 100 Web pages given by Alexa [3]. Each of the Web pages was downloaded 30 times using the Mozilla *remote* command line feature. The Web pages were accessed from the network of CMU between the hours of 5pm and 9am EST on weekdays. Note that measurements at this off-peak period shows a smaller variance in the RTTs, which favors some of the algorithms (especially, `CC`) we compare PSM-Dynamic with. Since PSM-Dynamic uses the access point to buffer data when the estimation is not very accurate, it is robust against the variation in RTTs, as we will show in Section 5.4.2. Linux by default puts the jiffie count (10ms resolution) into the timestamp field. In order to have more accurate timing, we use a modified kernel that uses a millisecond resolution timestamp. This allows us to analyze traces from our real Internet experiments and infer RTT with high accuracy.

Laboratory Emulation Setup. We use a more controlled laboratory environment to study the sensitivity of the algorithms to different parameters of the environment. In this setup, we configured an Apache Web server to run on a machine (with the same configuration as the client machine mentioned above) in the same local area network as the client. The server serves a Web page



(a) Average download times (b) Energy consumed

Figure 8: Effect of α, where beacon period = $\alpha\times$ RTT

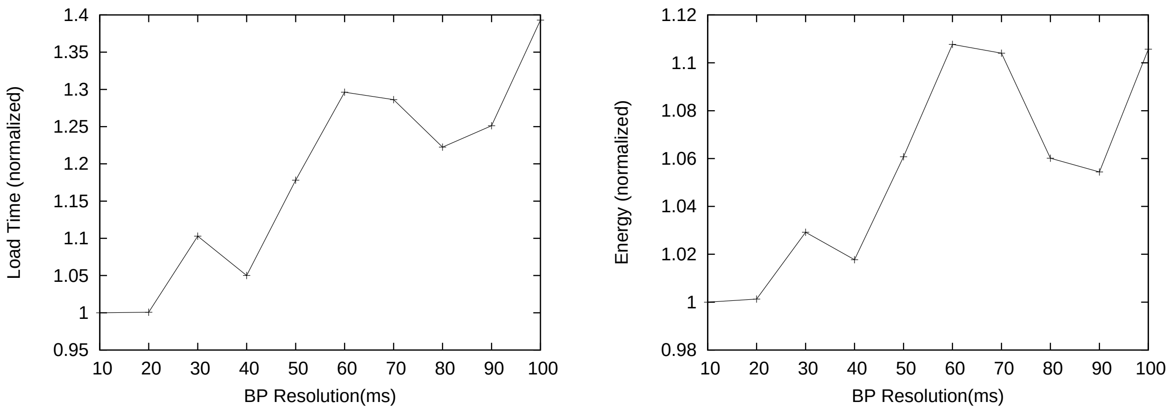
identical to `http://www.microsoft.com` and all its associated embedded objects (total size is around 168 KB). The bandwidth between the client and the server was regulated by the rshaper kernel module [13] running on the server machine. The RTT of the client and the server is normally distributed with the mean of 60ms and a variance of 5ms. In the experiments with varying RTT, we use the Netfilter module to impose delays on packet arrival time.

5.2 Choosing the Values of PSM-Dynamic Parameters

As mentioned in Section 3, PSM-Dynamic has two parameters: the value of α which gives the beacon period of a client as *beacon period* $= \alpha \times RTT$, and the granularity of the individual beacon periods. Here, we describe experiments (done in the emulated environment) designed to help choose good values for these parameters.

5.2.1 Choosing the value of α

As mentioned in Section 3.2, each client chooses its best beacon period based on the RTT estimate from its ongoing connections as follows: *beacon period* $= \alpha\times$ RTT. To understand the impact of α on performance, we use experiments on our controlled testbed with the RTT set to 60ms and the RTT variance set to 5ms. Clearly, $\alpha \geq 1$, otherwise no packets are likely to arrive between two successive beacons. Figure 8 shows the effect of different values of α on the download time and energy consumption. Figure 8(a) shows that a value of α slightly greater than 1.1 provides the optimal download time for this setup. A value of $\alpha >> 1.1$ makes the NIC to wake up less frequently and, thus, packets are buffered and delayed at the access point – significantly reducing performance. A value of $\alpha < 1.1$ is also suboptimal since the NIC often wakes up before the packet arrives. As a result, the NIC often goes back to sleep and the packet arrives just a few moments later. This results in almost an entire RTT of extra delay for the packet, also significantly reducing performance. The α parameter has a less significant impact on energy consumption. However, a similar value of $\alpha = 1.1$ is optimal for energy consumption too, as shown in Figure 8(b). The reason values of $\alpha > 1.1$ or $\alpha < 1.1$ consume more energy is that a longer download time consumes more energy since the NIC consumes some energy even when it is in the sleep mode.



(a) Average download times (b) Energy consumed

Figure 9: Effect of granularity of beacon periods of the PSM-Dynamic.

We should note that the exact optimal value of α depends greatly on the relation between RTT and RTT variance in the environment. While these experiments were done in a controlled environment, we plan to explore the tuning of α more carefully in our real-world testbed. In all the experiments in rest of the section, we use a value of $\alpha = 1.13$.

5.2.2 Effect of Granularity of Beacon Periods

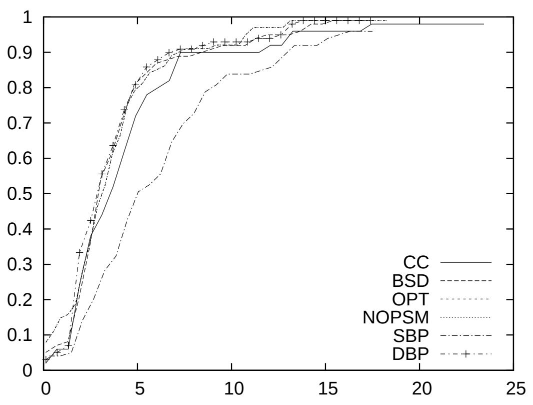
We evaluate the impact of different beacon period granularities on the performance of the PSM-Dynamic algorithm using experiments on our emulation setup. We take the RTT distribution of the connections required to download the top 100 Alexa Web pages, and take every 20% percentile of that distribution. For each of those five RTTs, we set up an emulation configuration with an RTT of RTT_i and download our test page 30 times.

Figure 9 reports the average download time and energy consumption of all the downloads over all the buckets. Both the download time and the energy consumption are normalized to the corresponding values found with 10ms granularity. As the figure shows, a small granularity reduces download time and energy consumption. This is because a smaller value allows a client to set its beacon period closer to the desired value, which reduces the time that packets remain buffered at the access point. This significantly reduces the total download time and also reduces energy consumption but to a lesser extent. Note, a small granularity imposes greater overhead since different connections are more likely to choose different wake up times. Unfortunately, this overhead is not accurately reproduced in our measurements. We choose to use a 20ms granularity for beacons since smaller granularities seem to provide minimal performance gains and possibly incur larger overheads.

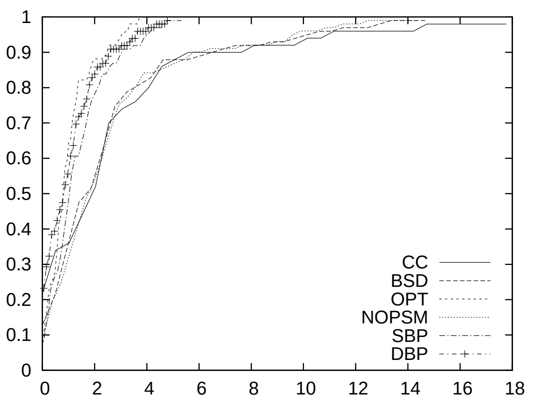
5.3 Real-World Experiments

In this section, we report the performance of PSM-Dynamic and other algorithms under our real-world experimental setup.

Figure 10 shows the CDF of average download time and average energy consumed to download the top 100 Web sites given by Alexa. In addition, we show the 80th percentile of these metrics



(a) Average download times



(b) Energy consumed

Figure 10: Average time and energy consumed to download 100 top Web sites with different algorithms.

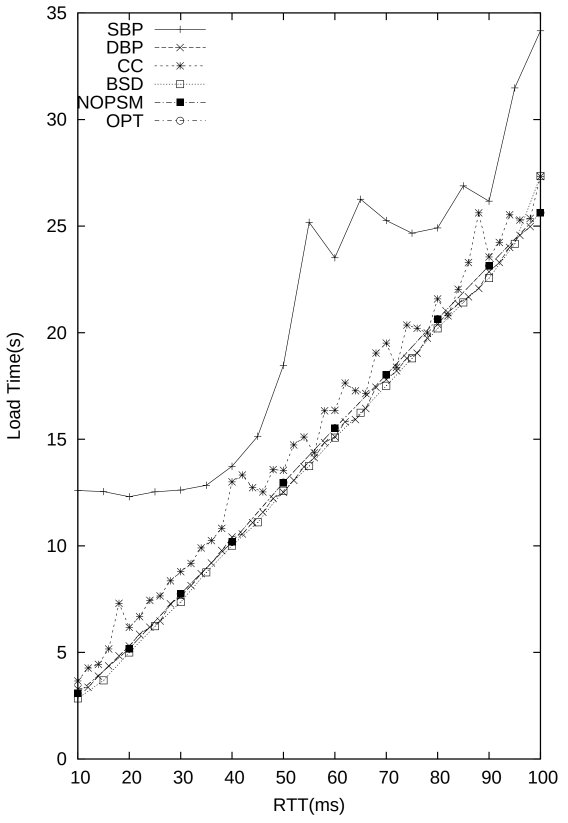
Algorithm	Avg. download time (sec)	Avg. energy consumed (J)	Time × energy
Optimal (`OPT`)	4.85	1.38	6.7
802.11 PSM-Static (`SBP`)	9.19	2.17	19.96
802.11 No PSM (`NOPSM`)	4.85	3.57	17.31
PSM-Dynamic (`DBP`)	4.89	1.8	8.8
Bounded slowdown (`BSD`)	4.82	3.7	17.84
Client-Centric (`CC`)	6.09	4.07	24.77

Table 5: 80 percentile of the average download time and average energy consumed by different protocols.

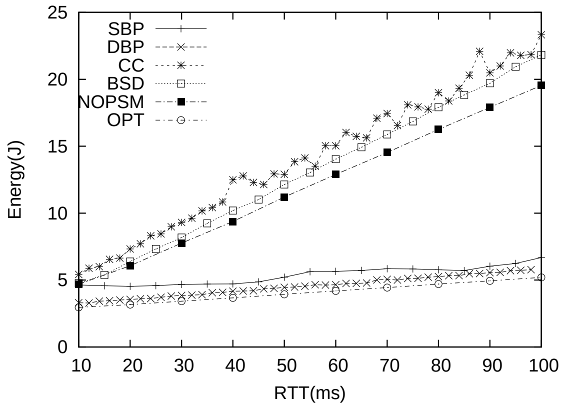
in Table 5. First, note that these results show that PSM-Dynamic provides a near optimal average download time. The only other practical (`OPT` is not practical) schemes that provide such near optimal download times are `NOPSM` and `BSD`. `NOPSM` provides the optimal download time since, by definition, it keeps the NIC awake all the time and, thus, never delays packets to save power. `BSD` gives a near optimal download time since it keeps the NIC awake during the slow start period, and most HTTP transfers finish during this period[8]. Second, note that PSM-Dynamic also provides near optimal energy consumption. Both the `NOPSM` and `BSD` schemes, which provide similar performance, do not allow the NIC to sleep during small HTTP transfers. As a result, both schemes consume a large amount energy, as shown in the second column of Figure 5.

It is also interesting to note that the client-centric approach performs worse than other algorithms in both the metrics. This occurs for two reasons. First, the variance of the RTT is so high that the algorithm fails to make a right guess of the sleep time. Second, most Web pages contain embedded objects and most browsers (like Mozilla) use concurrent connections to download them. In the presence of concurrent connections, the client-centric approach does not get enough chance to keep the NIC in sleep mode.

[8]The BSD paper [8] primarily focuses on providing good performance for long HTTP transfers.



(a) Average download times



(b) Energy consumed

Figure 11: Effect of varying RTTs of the Web sites.

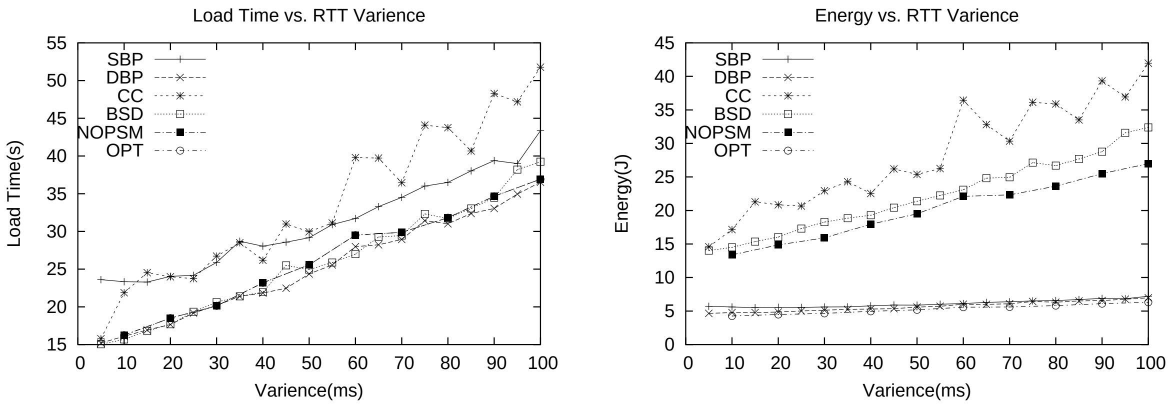
In summary, these results show that PSM-Dynamic provides the unique combination short download times, close to that given by `NOPSM`, and low energy consumption, close to that given by `SBP`. To compare the schemes with a single metric, we use the value of *download time* × *energy consumed*. As the third column of Figure 5 shows, PSM-Dynamic performs very close to the optimal and outperforms all the other practical schemes.

5.4 Emulation Results

In this section, we use our emulated setup to study how sensitive different algorithms are to the different parameters of the environment.

5.4.1 Effect of varying RTTs

Figure 11 shows performance of different algorithms as the RTT of the server from the client changes. The RTTs have a variance of 5ms. Figure 11(a) shows that, as expected, download time increases as the RTT increases. PSM-Static pays a high penalty in download times, showing that its 100ms sleep time is too coarse grained for typical HTTP transfers. PSM-Dynamic performs very close to the `OPT`, `NOPSM`, and `BSD` under the entire range of RTT values. Note that `CC` per-



(a) Average download times (b) Energy consumed

Figure 12: Effect of variance of the RTTs of the Web sites.

forms very similar to other algorithms (in contrast to our real-world experiments in 5.3), since our emulated variance in RTT is much smaller than what we found in the real world.

Figure 11(b) shows that the energy consumed by PSM-Dynamic is independent of the RTT, a very attractive property shown by the PSM-Static as well. This is due to the fact that even though the increase of RTT increases the download time, both these algorithms can sleep effectively and wake up only to receive the packets buffered in the access point. Both `NOPSM` and `BSD` keep the NIC awake during the entire download period, which increases with RTT. Therefore, the energy consumed by these two algorithms increases linearly with RTT. The same behavior is shown by `CC` as well since it places the NIC into sleep mode infrequently when multiple concurrent connections are in progress. In addition, wrong guesses for the sleep time make `CC` losses packets sent by the access point.

5.4.2 Effect of variance of RTTs

Figure 12 shows the effect of the RTT variance on the performance of different algorithms. For each value of the variance, the RTTs are generated as follows: a random RTT is picked from the normal distribution given by the mean RTT of 60 ms; if the random value is less than 10ms, a value of 10ms is used. Note that this 10ms lower bound means that increasing the variance effectively increases the mean RTT. Therefore, the general trend of this graph is very similar to that in the RTT graphs in Figure 11. However, as variance increases, `CC` makes more mistakes in its guess of RTT, and hence pays more penalty as shown by the graphs. Although PSM-Dynamic also guesses RTT to determine the sleep time, it does not suffer as significant a penalty for incorrect guesses since packets arriving while the client's NIC is asleep are buffered by the access point. This cooperation from the access point makes sure that packets are not dropped in PSM-Dynamic and, thus, the download time and energy consumed remains almost unaffected due to the variance in the RTT.

5.5 Summary

In summary, our real-world and emulation experiments show the following results.

- The 802.11 PSM-Static scheme provides reasonable power savings but extremely poor page download times. With the default beacon period of 100ms, it has an average download time almost twice the optimal.
- Bounded slowdown algorithm is not well optimized for small HTTP transfers. It provides good download times but does not save much energy.
- The client-centric approach does not perform well in practice, mainly because of the high variance of the RTTs and the access point not buffering the packets arriving when the client NIC is sleep.
- PSM-Dynamic provides the best combination of download performance and energy savings of the schemes considered. Its download times were extremely close to optimal and its energy consumption was marginally higher than optimal. Note that we did not consider the energy consumption of the rest of the device (CPU, display, etc.). However, it should be obvious that completing work more quickly would allow the remainder of the device to also enter any appropriate idle mode, thereby, increasing the energy savings of PSM-Dynamic over schemes with power download performance (PSM-Static and the client-centric approach).

These results show the effectiveness of the two design principles we use in PSM-Dynamic: (1) different clients should guess and use their own beacon periods based on the RTTs of their current connections, and (2) the access point should buffer the packets arriving while the destination client is asleep.

6 Related Work

Past research suggests that, due to the lack of comparable advancement in battery technology, efforts should be directed at reducing the power used by the NIC [10]. As a result, a great deal of work has been done to improve the energy efficiency of all parts of the wireless network stack. For example, at the physical layer, many energy saving methods are considered in [6]. Here, we discuss work primarily at the higher layers.

A number of studies have explored Media Access Control(MAC) protocols (not necessarily 802.11-based) that minimize energy consumption. The EC-MAC protocol [15] avoids collisions during reservation and data packet transmission. The basic premise is that minimizing wasted transmissions saves power. Another approach in [15] proposes that the base station publish a transmission schedule so that wireless devices can avoid monitoring the channel at all times. While not monitoring the channel, wireless devices can drop back into a low-power standby mode. The Power Aware Multi-Access(PAMAS) protocol [14] suggests the use of separate channels for Request to Send(RTS)/Clear to Send(CTS) control packets and data packets. While data is being transmitted over the data channel, a "busy signal" is transmitted over the control channel. Devices can enter a low power state if they detect that they are unable to send or receive packets.

At the network layer, most work has concentrated on energy-aware ad hoc routing protocols. Most ad hoc routing systems use shortest-hop, shortest-delay, or locality stability based metrics. However, these metrics can result in the overuse of the energy resources of a small set of mobile devices. Recent work [18] has developed routing algorithms that consider metrics such as energy consumed per packet, and variance in power levels across mobile devices.

Little work directly addresses redesigning TCP to improve its energy consumption. Recent work [20] has analyzed the energy properties of different versions of TCP. One obvious way to reduce power is to minimize retransmissions and reduce transfer duration. There is a large body of work [5, 4] with this goal.

Our work is most similar to efforts that try to exploit the low power states of NICs [19, 8]. As we have shown, our approach has significant performance advantages over either the client-centric approach [19] or the bounded slowdown approach [8]. We should note that the approach presented here is orthogonal to the bounded slowdown approach.. The two approaches could be used together as bounded slowdown is primarily concerned with avoiding being awake to listen to beacons when no connections are active. Our approach is primarily concerned only with open connections. The work presented here is independent of energy saving work at other layers of the wireless network stack.

7 Conclusion

In this paper, we have shown that the standard 100ms beacon period provides poor Web browsing performance. Our approach, called PSM-Dynamic, allows each client to choose their own beacon period at a much finer granularity. Using a combination of simulation, laboratory and wide-area experiments, we have shown that using a beacon period closely related to a connection's RTT provides a combination of transfer performance and power savings that is unmatched by either the standard 802.11 power saving mode or other more recently proposed techniques [19, 8]. In addition, we have shown that PSM-Dynamic can support typical access point demand with little overhead. In addition, PSM-Dynamic is able to handle both RTT variation present in today's environment as well as much larger variations that might be present in some situations.

We believe that the availability of finer grain beacon periods can enable clever optimization beyond those presented in this paper. Promising future directions for this work include: 1) evaluating better techniques for handling concurrent connections without increasing the number of beacons and 2) exploring interaction with power savings techniques at other layers.

References

[1] Dell latitude truemobile. http://www.dell.com/downloads/emea/products/latit/Truemobile_uk.pdf.

[2] Netfilter/iptables project homepage. http://www.netfilter.org.

[3] Alexa web search - top 500. http://www.alexa.com/site/ds/top_500, 2004.

[4] Balakrishnan, H., Seshan, S., and Katz, R. Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks 1*, 4 (1995).

[5] Caceres, R., and Iftode, L. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications 13*, 5 (1995), 850–857.

[6] Chandrakasan, A., and Brodersen, R. *Low Power CMOS Design.* Kluwer Academic Press, Norwell, MA, 1995.

[7] IEEE Computer Society. *IEEE standard 802.11: Wireless LAN medium Access Control and Physical Layer Specifications*, August 1999.

[8] Krashinsky, R., and Balakrishnan, H. Minimizing energy for wireless web access with bounded slowdown. In *ACM Mobicom* (2002), pp. 119–130.

[9] Kansas university real-time kernel. http://www.ittc.ku.edu/kurt, 2003.

[10] Lettieri, P., and Srivastava, M. Advances in wireless terminals. *IEEE Personal Communications 6*, 1 (1999).

[11] Lu, G., and Li, X. On correspondency between TCP acknowledgement packet and data packet. In *ACM Internet Measurement Conference* (2003).

[12] Mah, B. An empirical model of http network traffic. In *IEEE Infocom* (1997).

[13] Rubini, A. rshaper. http://freshmeat.net/projects/rshaper, 1999.

[14] Singh, S., and Raghavendra, C. Pamas: Power aware multi-access protocol with signalling for ad hoc networks. *Computer Communication Review 28*, 3 (1998).

[15] Sivalingam, K., Chen, J.-C., Agarwal, P., and Srivastava, M. Design and analysis of low-power access protocols for wireless and moble atm networks. *ACM/Baltzer Wireless Networks 6*, 1 (2000), 73–87.

[16] The VINT Project. *The NS Manual*, 2001.

[17] Wendland, R. How prevalent is timestamp options and paws. Web survey result published in end-to-end interest list, 2003.

[18] Woo, M., Singh, S., and Raghavendra, C. Power aware routing in mobile ad-hoc networks. In *ACM Mobicom* (1998).

[19] Yan, H., Krishnan, R., Watterson, S. A., and Lowenthanl, D. J. Client-centered energy savings for concurrent http connections. In *NOSSDAV* (2004).

[20] Zorzi, M., and Rao, R. Energy efficiency of tcp in a local wireless environment. *ACM/Balter Mobile Networks and Applications* (2000).