Choosing Beacon Periods to Improve Response Times for Wireless HTTP Clients*

Suman Nath Zachary Anderson Srinivasan Seshan
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
{sknath,zra,srini}@cs.cmu.edu

ABSTRACT

The IEEE 802.11 wireless LAN standard power-saving mode (PSM) allows the network interface card (NIC) to periodically sleep between receiving data. 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, named *Dynamic Beacon Period*, 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 measurements and 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.

Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Wireless communication*

General Terms

Algorithms, Measurement, Performance

Keywords

Wireless access, beacon periods.

*This research was supported in part by NSF Grant No. ANI-0092678 and funding from Intel Research.

MobiWac'04, October 1, 2004, Philadelphia, Pennsylvania, USA.

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/mobile device. At the end of every *beacon period*, which is typically 100ms (but can be set to a multiple of 100ms), the client's NIC wakes up to receive a beacon from the access point. The 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 `SBP` for static beacon period 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, `SBP` is *too fine-grained* when the NIC is mostly idle, and forces the NIC to wake up frequently and consume energy.

We claim that one major reason for this suboptimal performance of `SBP` 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[1] of 20ms and 200ms respectively. In this case, a static and common beacon period of 100ms would result in a suboptimal download time for client A and a suboptimal energy consumption for client B. Note that even if `SBP` uses a smaller beacon period, like 10ms[2], 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

[1]We use the term RTT to denote the sum of network level RTT and the application level processing time.
[2]This is a hypothetical value; in `SBP`, the beacon period is always a multiple of 100ms.

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 *Dynamic Beacon Period* (DBP), 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 SBP, 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.

The design of DBP raises two concerns: (1) Can a client choose a *good* beacon period for itself? and (2) Can an access point efficiently manage multiple beacon periods 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 questions can be answered affirmatively. First, the RTTs of Web servers remain relatively stable over the short duration of HTTP downloads, and the clients can use their RTTs to choose a reasonably good beacon period.[3] 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.

- We propose DBP, 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 SBP. 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 DBP.

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

[3]Note that even if the RTT of a Web server has high variance, buffering at the access points helps DBP avoid dropping packets when the client is in the sleep mode (unlike the pure client-centric approach in [17].)

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

Table 1: Power consumption of the NIC.

2. RELATED WORK

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 MAC protocols (not necessarily 802.11-based) that minimize energy consumption. The EC-MAC protocol [14] minimizes power related to wasted transmissions by avoiding collisions during reservation and data packet transmission. Other MAC approaches [13, 14] allow the NIC to easily identify when it is unable to send or receive packets, thereby, allowing the NIC to enter a low-power standby mode. At the network layer, most work has concentrated on ad hoc routing protocols, which currently optimize for shortest-hop, shortest-delay, or stable routes. Recent work [16] has developed routing algorithms that consider metrics such as energy consumed per packet, and variance in power levels across mobile devices. Finally, at the TCP layer, recent work [18] has analyzed the energy properties of different versions of TCP. One obvious way to reduce power is to minimize retransmissions and reduce transfer duration by using existing techniques [4, 5].

Our work is most similar to efforts that try to exploit the low power states of NICs [8, 17]. Note that our approach is significantly different from these existing solutions 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. Thus the approach is orthogonal to DBP, and they both can be used together. The client centric approach described in [17] 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.

3. EFFECT OF BEACON PERIOD ON HTTP DOWNLOADS

To understand the effect of different beacon periods on HTTP downloads to mobile devices, we use a number of 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 (computed by analyzing the packet level trace and using the NIC power consumption numbers from [1], as shown in Table 1). These metrics reflect the desired goals of the user: fast downloads and long battery life.

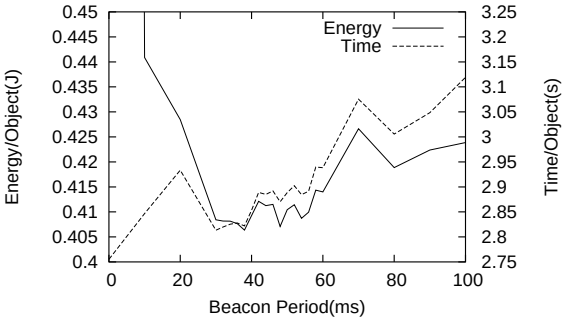


Figure 1: Impact of various beacon periods on accesses to `superman.web.cs.cmu.edu`.

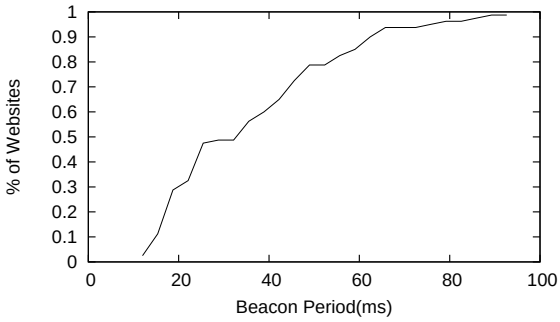


Figure 2: CDF of beacon periods that minimize the delay-energy product for Alexa top 100 Web sites.

Optimal Beacon Period. To understand how the default beacon period (100ms) for `SBP` compares with other possible beacon periods, we download the Web site `http://superman.web.cs.cmu.edu`, using several different beacon periods between 0 and 100. The propagation RTT to this Web site from the client is about 35ms. As shown in Figure 1, the HTTP download is optimized when the beacon period is chosen to be 38ms which is equal to the RTT. In practice, we find that a beacon period slightly higher than the RTT works best since it accomdates variability of the RTT (e.g., due to queueing delay). With this slightly longer beacon period, when the client wakes up 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 the 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, 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 practical and it is used only for comparison purposes. Figure 2 shows that using a beacon period of 38ms provides a near optimal performance in terms of both download time and energy consumption. Interestingly, `PSM(38ms)` 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	Time(sec)	Energy(mJ)
`NOPSM`	2.75	1930
`OPT`	2.75	390
`PSM(38ms)`	2.82	410
`PSM(100ms)`	3.12	420

Table 2: Average download time and energy required for accessing `superman.web.cs.cmu.edu` with different schemes.

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] (experiment setup details are in Section 6). For each site, we find the beacon period that minimizes the product of average energy used per object and average time spent per object. Figure 2 shows the CDF of choosen beacon periods. The data demonstrates that the desired 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).

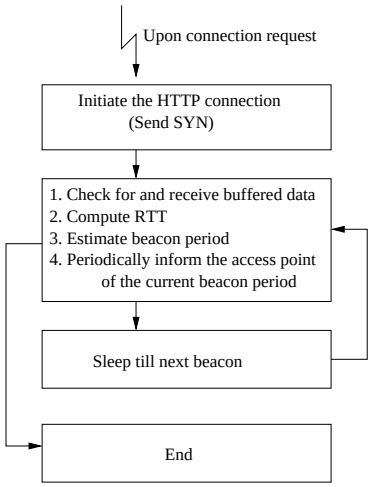


Figure 3: The dynamic power saving algorithm.

Summary. The default 100ms beacon period for `SBP` is suboptimal and the optimal beacon period is different for different sites. Additional simulation results validating these observations are omitted here for lack of space and can be found in [11].

4. THE DBP ALGORITHM

The dynamic power saving algorithm (`DBP`) differs from the standard 802.11 PSM (`SBP`) 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 `DBP` can be incorporated into the 802.11 specification with only a minor modification.

Figure 3 shows the core of the `DBP` 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, an APs has a different beacon periods, BP_i, for each active (i.e., currently sending

or receiving data) client, C_i. At the end of a beacon period BP_i, the AP sends a beacon containing a traffic indication for the client C_i, followed by the data destined for the client. Each active client C_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 up. When mobile client C_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 awake just as the AP receives 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 DBP largely depends on the ability to choose a good beacon period. In the next sections, we address how DBP achieves this goal.

4.1 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 [10]. Fortunately, TCP provides a timestamp option[4] that can be used to measure RTT from either side when both the sender and receiver agree to use the option (note that [17] 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, 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.

[4]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 [15].

4.2 Beacon Period Selection

As shown by the results presented in Section 3, the optimal beacon period for a client is very close to the RTT of the Web site it is downloading the Web page from. However, a few other factors determine the actual chosen beacon period.

First, we use beacon period = $\alpha \times$RTT, where α is a constant slightly larger than 1. This allows DBP to cope with variability in the actual RTTs.

Second, DBP scheme may limit the possible choices of beacon period to reduce the overhead on both the clients and the access points. One optimization we use in DBP 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 SBP). 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 6.

4.3 Enhancing 802.11 to Support DBP

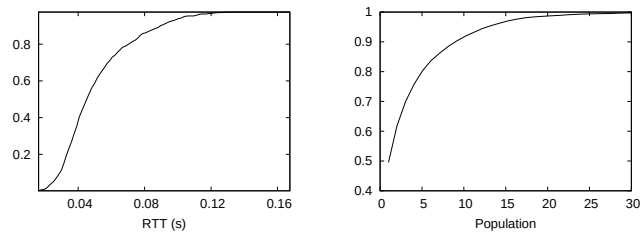
DBP requires a small modification to the standard 802.11 protocol. The 802.11 specification already allows for a *ListenInterval* which defines the number of beacons a client can skip. Each client can be configured to have its own ListenInterval. Thus, if a client has ListenInterval=2, it wakes up to 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 DBP 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 SBP.

4.4 Supporting Other Traffic Patterns

The techniques described above assume that a client has at single HTTP connection at any point of time. However, a power savings scheme must support more complex traffic patterns such as multiple connections and non-TCP traffic.

DBP 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.

DBP can also support non-TCP traffic. The core requirement for DBP is 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



(a) RTTs to www.nature.com. (b) Access point population

Figure 4: The CDF of RTTs to the site `www.nature.com` and the CDF of average access point populations .

constant-bit-rate (CBR) audio/video, we can make similar predictions for packet arrival times. We envision that a DBP implementation would include a set of standard APIs for transport protocols and applications to register their predictions for their next packet arrival. DBP could then merge the predictions and determine the appropriate beacon period.

5. PRACTICAL CONSIDERATIONS

This section addresses two concerns with the practicality of DBP. First, it is not clear whether a client can predict when data will arrive accurately enough to set the beacon period. Second, DBP may incur high overheads if there are a large number of concurrent transfers by different clients of the same access point.

5.1 RTT measurements

The effectiveness of DBP 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 CDF of RTT (specifically, for the site `http://www.nature.com`) is shown in Figure 4(a). 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.

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 DBP. 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 DBP and the client centered approach in [17]. 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.

5.2 Access Point Population

DBP 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 state 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[5].

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 4(b) 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.

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 SBP). 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.

6. EVALUATION

In this section, we compare DBP 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 DBP'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.

6.1 Evaluation Methodology

This section describes our evaluation methodology. We discuss the algorithms we compare with DBP, the metrics we use to quantify performance, and the setup of our experiments.

Algorithms Compared. We compare DBP with the following five algorithms.

1. BSD (bounded slowdown protocol, described in [8]). 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 [17]). CC guesses the future packet arrival times and awakens

[5]Note that DBP 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.

the NIC only at that time. 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 these algorithms (except OPT) in a 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]. We download each of the Web pages 30 times using the Mozilla *remote* command line feature. The Web pages are accessed from the network of CMU between the hours of 5pm and 9am EST on weekdays. Note that measurements at this off-peak period show a smaller variance in the RTTs, which favors some of the algorithms (especially, CC). Since DBP uses the access point to buffer data when the estimation is not very accurate, RTT variation has little impact on its performance, as we will show in Section 6.4.2. 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.

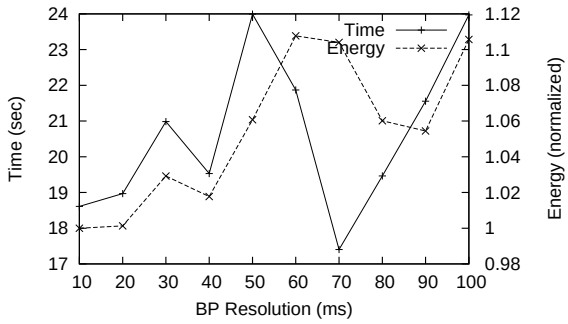


Figure 5: Effect of granularity of beacon periods of the DBP.

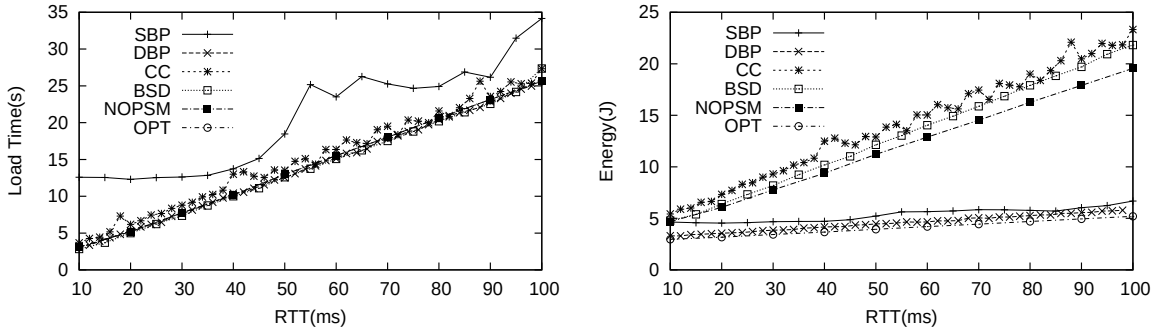
Laboratory Emulation Setup. We use a more controlled laboratory environment to study the sensitivity of the algorithms to different parameters of the environment. In this setup, we configure an Apache Web server to run on a machine (with the same configuration as the client machine mentioned above) in the same local area network as the client. The server serves a Web page identical to http://www.microsoft.com and all its associated embedded objects (total size is around 168 KB). The rshaper kernel module [12] running on the server machine regulates the bandwidth between the client and the server. 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.

6.2 Choosing Parameter Values for DBP

As mentioned in Section 4, DBP 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. We should note that the exact optimal value of α depends greatly on the relation between RTT and RTT variance in the environment. Our experience with DBP indicates that a value of α around 1.1 works well (details in [11]). In all the experiments in rest of the section, we use a value of $\alpha = 1.13$. In the rest of the section, we describe experiments (done in the emulated environment) designed to help choose a good values for the beacon granularity

We evaluate the impact of different beacon period granularities on the performance of the DBP 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 5 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



(a) Average download times (b) Energy consumed

Figure 6: Effect of varying RTTs of the Web sites.

Scheme	Time(S)	Energy(J)	Time × energy
OPT	4.85	1.38	6.7
SBP	9.19	2.17	19.96
NOPSM	4.85	3.57	17.31
DBP	4.89	1.8	8.8
BSD	4.82	3.7	17.84
CC	6.09	4.07	24.77

Table 3: 80 percentile of the average download time and average energy consumed by different protocols.

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.

6.3 Real-World Experiments

Table 3 shows the 80th percentile of average download time and average energy consumed to download the top 100 Web sites given by Alexa (the CDF of these measurements can be found in [11]). First, note that these results show that DBP 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[6]. Second, note that DBP 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 3.

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.

In summary, these results show that DBP 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 3 shows, DBP performs very close to the optimal and outperforms all the other practical schemes.

6.4 Emulation Results

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

6.4.1 Effect of RTT

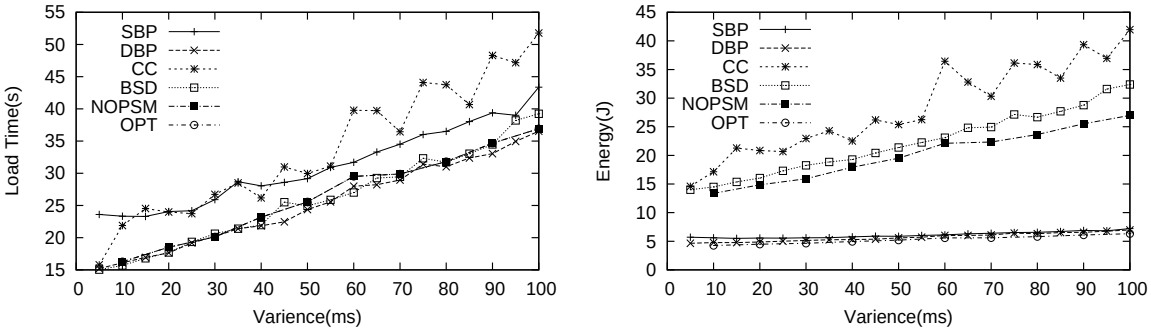
Figure 6 shows performance of different algorithms as the RTT of the server from the client changes. The RTTs have a variance of 5ms. Figure 6(a) shows that, as expected, download time increases as the RTT increases. SBP pays a high penalty in download times, showing that its 100ms sleep time is too coarse grained for typical HTTP transfers. DBP performs very close to the OPT, NOPSM, and BSD under the entire range of RTT values. Note that CC performs very similar to other algorithms (in contrast to our real-world experiments in 6.3), since our emulated variance in RTT is much smaller than what we found in the real world.

Figure 6(b) shows that the energy consumed by DBP is independent of the RTT, a very attractive property shown by the SBP 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.

6.4.2 Effect of RTT variance

Figure 7 shows the effect of the RTT variance on the performance of different algorithms. For each value of the vari-

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



(a) Average download times (b) Energy consumed

Figure 7: Effect of variance of the RTTs of the Web sites.

ance, 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 6. However, as variance increases, `CC` makes more mistakes in its guess of RTT, and hence pays more penalty as shown by the graphs. Although `DBP` 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 `DBP` and, thus, the download time and energy consumed remains almost unaffected due to the variance in the RTT.

7. CONCLUSION

In this paper, we have shown that the standard 100ms beacon period of 802.11 PSM provides poor Web browsing performance. We have also shown that other existing solutions are not suitable for HTTP download: BSD fails to save energy for small HTTP transfers, client-centric approach fails to perform well in practice due to high variance of RTTs. Our approach, called `DBP`, 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 [17, 8]. In addition, we have shown that `DBP` can support typical access point demand with little overhead. In addition, `DBP` 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 optimizations 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.

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

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

[6] A. Chandrakasan and R. Brodersen. *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] R. Krashinsky and H. Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. In *ACM Mobicom*, pages 119–130, 2002.

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

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

[11] S. Nath, Z. Anderson, and S. Seshan. Choosing beacon period for improved response time for wireless http clients. CMU Technical Report, 2004.

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

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

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

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

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

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

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