Maintaining Performance while Saving Energy on Wireless LANs

Ronny Krashinsky
Computer Networks (6.829) Term Project
MIT Laboratory for Computer Science, Cambridge, MA 02139
ronny@mit.edu

12-17-2001

Abstract

Minimizing the energy consumption of wireless network interfaces is crucial in the design of battery-powered mobile computing devices. The IEEE 802.11 wireless LAN specification describes a power-saving mode which allows the network interface on the mobile device to enter a sleep mode when the link is idle; periodically, the mobile device must wake up to listen to beacons from the access point which indicate if it has any buffered data. This work first presents detailed measurements which show that when TCP is run over existing implementations of the 802.11 power-saving mode, performance suffers because otherwise fast round-trip-times are rounded up to 100 ms. The proposed solution is the Stay-Awake scheme which delays entering sleep mode for a short period of time after the link is active. This work also determines that in typical web browsing traffic scenarios existing 802.11 power-saving mode implementations will spend most of their energy sleeping and listening to beacons during relatively long idle periods. The proposed improvement is ListenInterval-Backoff which allows the network interface to sleep for longer periods of time when the link remains idle. The combination of these proposed power-saving enhancements is shown to reduce the additional per-page delay incurred by the power-saving mode from a mean of around 70% to 3% with negligible change in the overall energy consumption, or to reduce the per-page delay to 10% while reducing the overall energy by 20%.

1 Introduction

The capabilities of mobile computing devices are limited by their battery weight and lifetime. As such, minimizing the energy usage of every component is a crucial goal in designing mobile systems. Wireless network access is a fundamental enabling feature for many portable computers, but if not optimized for power consumption this component can quickly drain a device's batteries. This work seeks to minimize the energy of the wireless network interface for a mobile device generating request/response traffic (e.g. web browsing) without significantly harming network performance.

A wireless network interface consumes a great deal of power when it is sending and receiving data, but it also can have significant power consumption when it is idle with its radio powered up and able to communicate. Since devices typically only access the network infrequently, the network interface can be disabled when it is not in use to save energy. In this sleep mode, the radio is turned off and the device has no way to determine when data is being sent to it over the wireless network link. This clearly breaks the typical transport model that data can arrive from the network at any time and a node should be able to receive it.

The IEEE 802.11 wireless LAN specification [6] describes a power-saving mode (PSM[1]) designed to preserve network traffic while allowing the mobile device (MD[2]) to disable its network interface. In an infrastructure network (as opposed to an ad hoc network), a wireless device communicates with a wired access point (AP). When the power saving mode is enabled, the AP buffers data destined for the MD. Once every BeaconPeriod (typically 100 ms), the AP sends a beacon which contains a traffic indication map (TIM) that indicates whether or not the MD has any buffered data. The MD wakes up to listen to these beacons every ListenInterval (typically it listens to every beacon) and polls the AP to receive any buffered data. Whenever the AP sends data to the MD, it indicates whether or not there is more data outstanding, and the MD only goes to sleep once it has retrieved all pending data from the AP. When the MD itself has data to send, it can wake up to send the data without waiting for a beacon.

The 802.11 power saving mode can have a significant effect on network performance. In particular, round-trip-

[1] Standard 802.11 terminology uses the abbreviation PS.
[2] Standard 802.11 terminology uses the abbreviation STA which stands for 'station'.

times (RTTs) are rounded up to the nearest 100 ms. In Section 2, I present measurements which show that this especially impacts short TCP connections which are the norm in typical web browsing traffic. Once the power saving mode is enabled, the power consumed sleeping and listening for beacons will dominate the total energy consumption if the network is accessed only sporadically. In Section 3, I present analyses of client HTTP traces which suggest that long idle periods are most important in terms of energy usage. In Section 4 I suggest improvements to the 802.11 power saving mode. Stay-Awake delays going to sleep after the mobile device sends data so that short RTTs are not lengthened. ListenInterval-Backoff lets the network interface sleep for longer periods of time when there is no network activity. This reduces the energy consumed listening to beacons, and could potentially allow the network interface to go into a deeper sleep mode. I present a simulation study which evaluates the effectiveness of these PSM enhancements under typical web browsing conditions. Finally, Section 5 discusses some related work, and Section 6 concludes.

2 TCP Performance over 802.11 PSM

TCP is the transport layer of choice for a majority of network applications. Its performance is therefore a critical consideration in the design of any network component. This section evaluates the impact of the 802.11 power-saving mode (PSM) on TCP performance.

During the initial slow-start phase of a TCP connection, round-trip-times (RTTs) dominate the transfer rate. The network bandwidth will only begin to dominate transfer rates when the connection saturates the available bandwidth. For example, if a 5 Mbps 802.11 wireless link is the bottleneck for a connection with a 40 ms RTT, this won't happen until around 25 KB of data is in transit (the bandwidth-delay product). Considering that most HTTP responses are less than a few tens of kilobytes [9, 13] and TCP packets are typically 1500 bytes, RTTs are critical for web-browsing performance.

When TCP is run over an 802.11 link with power saving mode enabled, the effective RTTs will initially be rounded up to the nearest 100 ms. This is because when the MD sends data (i.e., a TCP `syn` or `ack` packet, or a TCP data packet containing a request) to the server at the other end of the TCP connection, it will go to sleep as soon as this data has been transmitted to the AP. When the response data arrives from the server after some delay, it must be buffered at the AP until the next beacon occurs. Figure 1 shows estimated RTT times and slowdowns with PSM enabled; the slowdown depends on how much less the original RTT is than a multiple of 100 ms.

At the beginning of a beacon period the amount of data buffered at the AP is equal to the TCP window size (assuming sufficient bandwidth between the server and the access point). As soon as the MD receives the first TCP packet it will send an acknowledgement, prompting the server to send more data. If the new data arrives from the server before the AP finishes sending the buffered window of data to the client, the 802.11 network interface will stay awake; otherwise it will enter sleep mode until the next beacon time. The new data arrives from the server approximately one RTT after the start of the beacon period, during which time the wireless 802.11 link continually transmits data at maximum bandwidth. For example, with a 5Mbps 802.11 link and a 40 ms RTT, the window size must be around 25 KB in order to keep the wireless link busy until the new window of data begins to arrive from the server; until the window size reaches this point, the transmission of each TCP window will take 100 ms. This problem becomes worse as the bandwidth of the 802.11 link increases — counter-intuitively, TCP transmission times may be shorter over a lower bandwidth 802.11 link because the link will become saturated sooner and prevent the network interface from entering sleep mode.

Figure 2 shows the evolution of a TCP connection both with and without PSM enabled. For this test, the mobile client opened a TCP connection with a server and sent a request for 40 kB of data; the server responded with the data. The client used for this test was a Compaq iPAQ H3600 series hand-held computer running Familiar Linux version 0.4 with an Enterasys Networks RoamAbout 802.11 DS High Rate network interface card (NIC); the card was operating at 11 Mbps, but the maximum effective bandwidth achieved was around 5 Mbps. The server was in the same building as the 802.11 access point; the RTT was around 5 ms, and the bandwidth between the server and AP was at least 10 Mbps. Shown are the times at which data packets were sent from the server, where time zero is the time that the server saw the initial `syn` packet.

With PSM off, the connection quickly saturates the available bandwidth of the network—the maximum is around 5 Mbps, limited by the 802.11 wireless link. However, with PSM on, the initial RTTs are effectively increased to 100 ms. With an actual RTT of around 5 ms, only around 3 KB of buffered data should be required to keep the 5 Mbps link busy for long enough to allow the next window of data to begin arriving from the server and prevent the network interface from going to sleep. As shown in Figure 2, this happens after the TCP window grows to 4 packets (about 6000 bytes). Since the RTT of this connection is so short, it can almost be considered a best-case scenario for the 802.11 PSM.

In another test of the 802.11 PSM, the mobile client

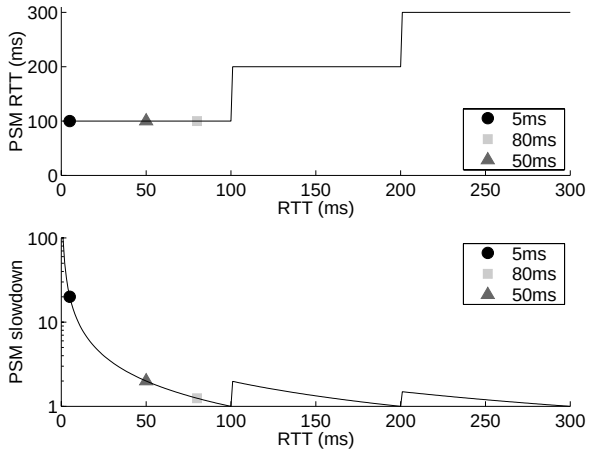

Figure 1: Estimated RTT slowdown due to 802.11 PSM. The upper graph shows the effective RTTs with PSM enabled, and the lower graph shows the slowdown.

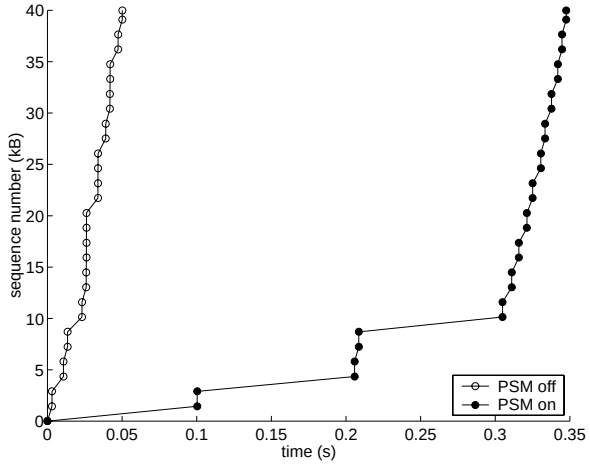

Figure 2: Measured evolution of a TCP connection with and without PSM enabled.

opens a TCP connection to a server and sends a request for some number of bytes; the server responds by sending the requested block of data. By doing this repeatedly for various data block sizes, I determined the relationship between data block size and transfer time. The client used was the same iPAQ device. The server was run on various machines to evaluate different network characteristics. The first server was in the same building as the 802.11 access point; the RTT was 5 ms, and the bandwidth was at least 10 Mbps. The second server was located around 3000 miles away and had a high bandwidth internet connection; the RTT was 80 ms and the bandwidth was at least 10 Mbps. The third server was 3 miles away and behind a DSL internet connection; it had a 50 ms RTT and outgoing bandwidth of 70 Kbps. The performance test measured the transfer time for power-of-two data block sizes between 1 B and 4 MB. It was run ten times alternating between PSM on and PSM off (five tests each). The results showed no significant variations between the runs, and the mean values are presented.

Figure 3 shows the relationship between data block size and total transfer time (including the request and response) for each server both with PSM on and PSM off. Figure 4 presents another view of the same data; it shows the slowdown incurred using PSM. For small data block sizes the entire response fits in one or two TCP data packets, and the total time for the transaction is equal to two RTTs – during the first RTT the client sends a `syn` packet to the server, and the server responds with a `syn+ack` packet; during the second RTT the client sends the request to the server and it responds with up to two data packets. With PSM off, the transfer time is determined by the RTT to each server; however, with PSM on the transfer times are 200 ms regardless of the server. The observed slowdowns match those predicted by Figure 1.

The transfer times for the low-bandwidth (70 Kbps) server become bandwidth-limited even before the transfer requires more than one RTT. For the high-bandwidth servers, the transfer times begin to take multiple RTTs as the data block size increases and eventually become bandwidth-limited; the maximum bandwidth achieved was about 4.9 Mbps. With PSM on, the maximum bandwidth achieved was about 3.4 Mbps. Apparently, the maximum bandwidth is limited by the overhead incurred by the PSM signalling; a close look at Figure 2 reveals that the data packet spacing in steady state is slightly increased with PSM on.

In some cases, the MD sends data to a remote machine rather than vice-versa; for example, this occurs if a user is editing a remote document. In this case, the 802.11 PSM causes the TCP `acks` to be delayed instead of the data packets. I ran the same performance test with the mobile

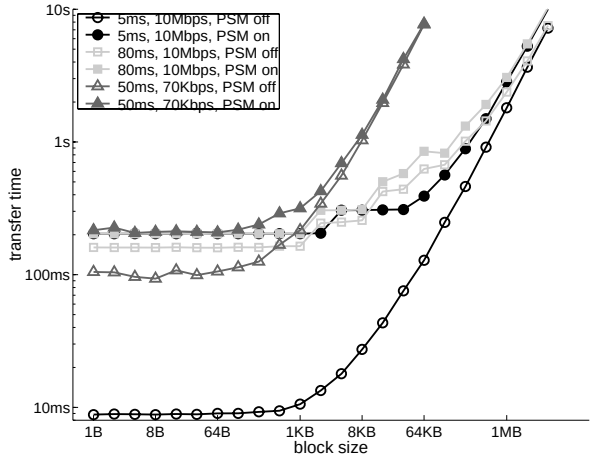

Figure 3: Measured data block size vs. transfer time for request/response transactions over TCP with various servers and PSM on and off.

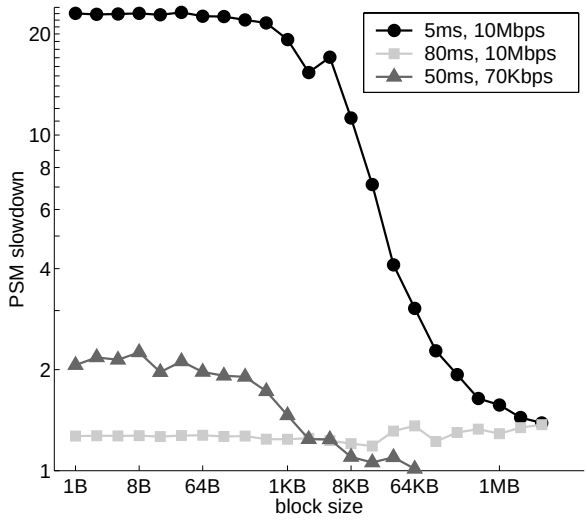

Figure 4: PSM slowdown from Figure 3.

device configured as the server, and a machine on the 5 ms, 10 Mbps network configured as the client. The results were essentially identical to those obtained when the mobile device was the client.

The basic finding from these measurements is that the 100 ms sleep interval used in the 802.11 PSM is *too coarse-grain* to maintain network performance.

3 Client Network Usage Characteristics

In optimizing a network interface to minimize power consumption, it is important to consider how clients use the network. For example, the 802.11 PSM allows the network interface to sleep for periods of 100 ms. Since there is a basic tradeoff between the extent to which power consumption is minimized in sleep mode and how long it takes to wakeup (and also how much energy the transition takes) [1, 14], the sleep period determines how deep the network interface sleep mode can be. Additionally, waking up to listen to beacons is costly in terms of power consumption; the sleep period determines the significance of this overhead when the network interface is continuously sleeping.

To evaluate the characteristics of client network usage, I analyzed client web traffic from the UC Berkeley Home IP dialup service traced for 18 days in November, 1996 [9]. The network activity for these traces is dominated by long transfer times over the slow modem links, but certain aspects can be relevant to general client usage patterns. In particular, the time that clients spend idle or waiting for responses from servers present opportunities for the network interface to enter a sleep mode, and these times are probably not critically dependent on the bandwidth of the client's network link.

For each HTTP transaction, the traces provide the client ID (anonymized IP address), the time at which the client made the request, and the start and end times for the response from the server. To analyze the traces, I first sorted the HTTP transactions by client ID. I then ordered the request, response start, and response end events for each client. Some transactions are never completed and have invalid timestamps; I excluded these from my analysis. I processed the ordered event lists and tracked the client state as one of:

wait: one or more outstanding requests and not receiving any responses

recv: receiving one or more responses

idle: no outstanding requests and not receiving any responses (note that this will include both user 'think time' and browser processing time)

Figure 5 shows the cumulative distribution function (CDF) for the client wait times. The solid line shows the cumulative distribution for the duration of all the wait events (each time a client enters the wait state). The dashed line shows the percentage of the total wait time that these wait events account for. For example, 45% of all wait events take less than 0.1 s, and these events account for 3% of the total wait time; 88% take less than 1 s and account for 19% of the total; 99% take less than 10 s and account for 60% of the total.

Figure 6 shows the CDF for the client idle times. The solid and dashed lines are as in Figure 5. However, in the traces many clients have no activity over a period of several days; if this data is included these idle times completely dominate the total idle time (as shown by the dotted line). Therefore, idle events longer than 1000 s (around 2% of all idle events) were excluded from the total idle time represented by the dashed line. The figure indicates that 6.7% of all idle events take less than 0.1 s, and these events account for 0.01% of the total idle time; 26% take less than 1 s and account for 0.5% of the total; 66% take less than 10 s and account for 7.5% of the total; 93% take less than 100 s and account for 48% of the total. For reference, if only idle events less than 100 s are included, the idle events less than 1 s account for 1.1% of the total; and if only idle events less than 10 s are included, the idle events less than 1 s account for 6.8% of the total.

The important point about these results is that although most wait and idle events are of short duration, most of the total wait and idle times are spent in long latency events. For example, over 80% of the total wait time and virtually all of the total idle time is spent in events longer than 1 s. Since the energy spent in sleep mode is directly proportional to the sleep duration, this means that long wait and idle periods will account for most of the sleep energy. The conclusion is that the 100 ms sleep interval used in the 802.11 PSM is *too fine-grain* to minimize energy effectively.

4 Proposed Improvements to 802.11 PSM

Section 2 demonstrated that the 802.11 PSM is too coarse-grain to maintain performance, and Section 3 demonstrated that it is too fine-grain to minimize energy. This section presents simple alternatives to the basic 802.11 PSM which can maintain performance while minimizing the energy for a wireless network link.

Figure 7 diagrams the behavior of the basic 802.11 PSM and several proposed alternatives. After sending data over the wireless link (e.g. an HTTP request, or a TCP `ack`), a basic PSM implementation goes to sleep and then wakes up to listen to beacons every 100 ms. The main performance problem is that fast response times will be rounded up to 100 ms. A simple solution to this problem is the Stay-Awake scheme which delays going into sleep mode for a short period of time after the link is active (100 ms in the example). The power consumption problem with the basic PSM occurs if the link is idle for a long period of time; the network interface must wake up every 100 ms to listen to beacons. A simple solution to this problem is the ListenInterval-Backoff scheme which allows the network interface to sleep for longer periods of time when there is no activity (twice as long as the previous interval in the example). In this work, the maximum sleep duration is set to 0.9 s to avoid TCP timeouts. Of course the Stay-Awake and ListenInterval-Backoff schemes can be combined; Max-Delay is a PSM scheme that never sleeps for longer than some percentage of the total elapsed time since the last activity on the wireless link (20% in the example).

Table 1 summarizes the effect that the proposed PSM schemes have on the latency and energy consumption of the wireless network link. For example, the Max-Delay scheme exhibits desirable overall properties. Very fast re-

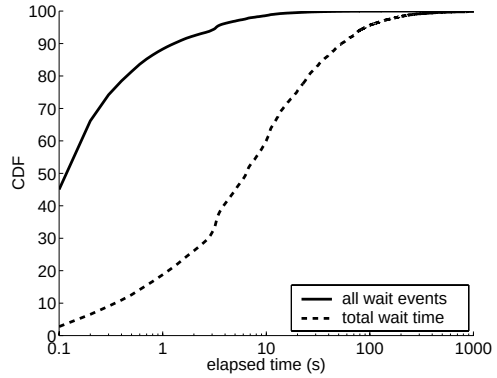

Figure 5: CDF for client wait events and total wait time.

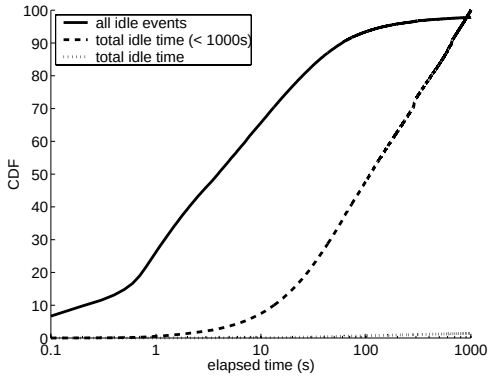

Figure 6: CDF for client idle events and total idle time.

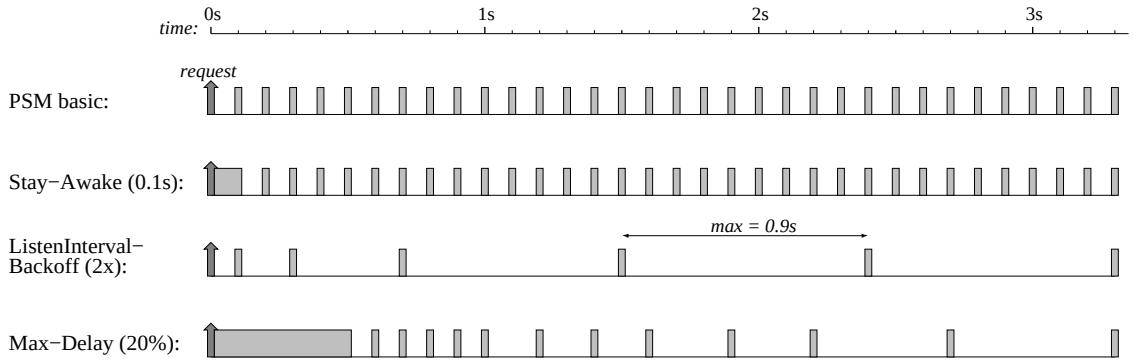

Figure 7: The behavior of various PSM alternatives. The arrow at time 0 indicates activity on the wireless link, and the shaded bars indicate when the network interface wakes up to listen to beacons.

	Latency (vs. no PSM)			Energy (vs. PSM basic)	
	short	medium	long	active (awake)	listening to beacons
PSM basic	Increased by up to 100 ms				
Stay-Awake	Unchanged	Increased by up to 100 ms		Increased	Unchanged
ListenInterval-Backoff (2x)	Increased by up to 100 ms	Increased by up to 2x	Increased by up to 0.9 s	Unchanged	Decreased
Max-Delay (20%)	Unchanged	Increased by up to 20%	Increased by up to 0.9 s	Increased	Decreased

Table 1: Comparison of the latency and energy implications for various PSM alternatives.

sponse times are not delayed, while longer ones are increased by up to a certain percentage with a set maximum limit. The active energy is increased since the transition into sleep mode is delayed with Stay-Awake, but the energy spent listening to beacons is decreased by the ListenInterval-Backoff.

Although Figure 7 shows the PSM schemes starting when request data is sent from the MD, the implementations considered in this work take effect whenever the wireless link is active in either direction. This is simple, and probably the desired behavior if data being sent to the MD indicates a higher probability that there will be more activity in the near future. Additionally, the PSM schemes start over when there is any new activity on the link. It is important to point out that the ListenInterval-Backoff is designed for a MD which initiates request/response network traffic. In general, it is not appropriate for real-time communication, or for a MD which acts as a server and responds to external requests.

Updating the existing 802.11 MAC to support Stay-Awake and ListenInterval-Backoff should be fairly trivial. The access point could be informed of the MD's Stay-Awake time, and could forward data to the MD without delay during this time. Or, it could notify the device as soon as data arrived from the network instead of waiting for the next beacon; in this case the device could retrieve the data if awake. The 802.11 specification already allows for a ListenInterval which is different than the BeaconPeriod; the only enhancement is to enable the ListenInterval to change more dynamically.

4.1 Simulation Methodology

To analyze the performance and energy impact of the basic 802.11 PSM, and to evaluate the proposed PSM enhancements, I simulated a mobile client browsing the Web over a wireless network link.

Using the network simulator `ns-2` [16], I modeled a mobile client communicating with an access point over a wireless link with PSM. Since I was not concerned with many of the complications which the 802.11 protocols accounts for—such as signal strength, channel contention, node movement, and multicast—I chose not to model the detailed MAC protocol, but instead made some simple modifications to the basic link object in `ns-2`. First, I modified the C++ queue element of a link to support a sleep mode in which it does not forward any packets, and in which it alerts its corresponding OTcl object when a new packet arrives. I also modified the C++ delay element of a link to alert its corresponding OTcl object when it goes idle. I modeled the wireless link as an OTcl object which controls AP to MD and MD to AP links. The beaconing is implemented using a timer which expires every 100 ms. During each beacon, I determine whether the MD listens to the beacon based on its ListenInterval-Backoff algorithm. If it does, and there is data buffered in the queue for the AP to MD link, the MD wakes up (the queues are activated). When both links are idle for longer than the Stay-Awake parameter, the MD goes to sleep (the queues are deactivated). Whenever a packet arrives at the queue for the MD to AP link, the MD wakes up immediately. Based on the experiments described in Section 2, I modeled the AP to MD and MD to AP links as 5 Mbps with a latency of 100 μs.

To model a client browsing the web, I used the HTTP traffic generator present in the `ns-2` distribution (in `ns-2.1b8a/tcl/http/`). In the model, the retrieval of a web page begins with a client opening a TCP connection with the server and sending a request. The model uses one-way TCP connections, but I updated it to use FullTcp connections. The server sends a response, and then the client retrieves some number of embedded images. To get these, the client opens up to four parallel TCP connections with the server. After the web page retrieval is complete, the client waits for some amount of think time before retrieving the next page. The various parameters in this model are randomized based on empirical data [13]. As in Section 3, I limited the user think time to 1000 seconds because otherwise think times as long as an entire day would completely dominate the total think time. I also added a server response time which was not present in the original model. I based this on the data in Figure 5, except I subtracted 100 ms from these times since they include the network delay; so, in the model 45% of the time the server responds with no delay.

To test the 802.11 PSM and the proposed enhancements, I modeled a network consisting of a mobile client, an access point, and a server. Admittedly, using a single server with a set bandwidth and RTT is a simplification and significantly effects the performance impact of the PSM. For each of various PSM settings, I simulated a client retrieving 10,000 web pages; these comprised a total of 38,428 HTTP request/response transactions, and around 541,000 seconds of client web browsing time. The random number generator was seeded with the same value for each test; furthermore, since the various PSM modes can in certain cases change the order of events in retrieving a web page, all of the necessary random variables for a page were generated before starting to retrieve it. Figure 8 shows the CDFs for the actual randomized parameters used in the simulations.

To simulate the power consumption of the 802.11 network interface card, I used a very simple model inspired by various reported measurements [3, 5, 8]. I modeled the power usage as 1 W while awake (sending data, receiving data, or idle), and 50 mW while asleep. In reality, 802.11

cards consume somewhat more power while sending and receiving data; but, as demonstrated below, the power consumed while actually transmitting data in the web browsing simulations tends to be insignificant. I modeled the energy consumed in waking up and listening to a beacon as 5 mJ, based on an approximation of 1 W being consumed for 5 ms. I believe that these estimates are reasonable approximations; but, since the values can vary, I present the raw data below in addition to computed energy estimates.

4.2 Performance and Energy Results

Figure 9 shows the mean and median page retrieval times and the mean slowdown per page when PSM is enabled for various AP to server RTTs. The mean slowdown per page values were obtained by dividing the retrieval time for each page without PSM enabled by the retrieval time for the same page with the basic 802.11 PSM enabled, and taking the mean. Thus, each page is given equal weight independent of how long its retrieval time is. As expected, the PSM has the most impact when the connection between the AP and server is very fast.

For the following analyses, I use an AP to server RTT of 40 ms. This is a somewhat arbitrary choice, but since clients will in general communicate with servers that have both shorter and longer RTTs, using a single server with a 40 ms RTT should provide a meaningful data point.

Figure 10 compares the performance of the basic PSM with two alternatives. Each marker on the figure represents the retrieval of a single web page; the x-coordinate is the retrieval time with PSM off, and the y-coordinate is the slowdown when PSM is on. The figure shows that PSM has the greatest negative impact on pages with fast retrieval times. These are slowed down by up to about 2.5 times which is the penalty for extending a 40 ms RTT to 100 ms. On average, page retrieval times are increased by 70%. The Stay-Awake (0.1 s) scheme drastically improves the retrieval times; the mean slowdown is decreased to around 3%. The Max-Delay (20%) scheme also reduces the mean slowdown to around 3%; in this scheme pages with retrieval times less than 1 s are not delayed at all, and pages with retrieval times between 1 s and 10 s tend to be delayed by less than 20%.

Figure 11 compares the mean per-page energy and performance of various PSM alternatives, and Table 2 shows the data used to compute the energy estimates. The first column in the figure shows the basic 802.11 PSM. It uses around 10 times less energy per page than 802.11 with PSM off, but suffers from a slowdown of around 70% additional delay per page. Based on the estimates, the energy spent while awake is negligible since the network interface is in sleep mode for around 1000 times longer than it is awake. Also, the total energy used is equally split be-

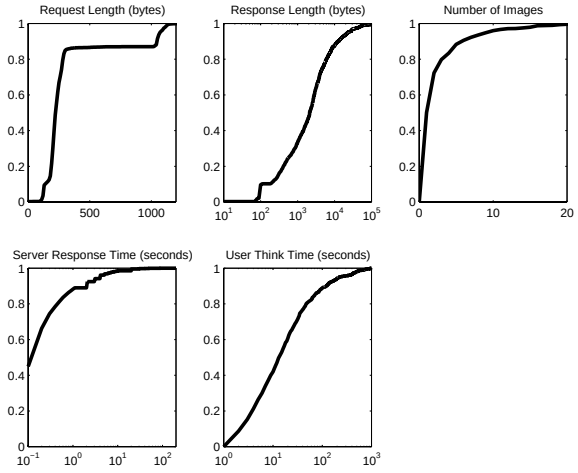

Figure 8: CDFs for randomized parameters used in HTTP traffic simulations.

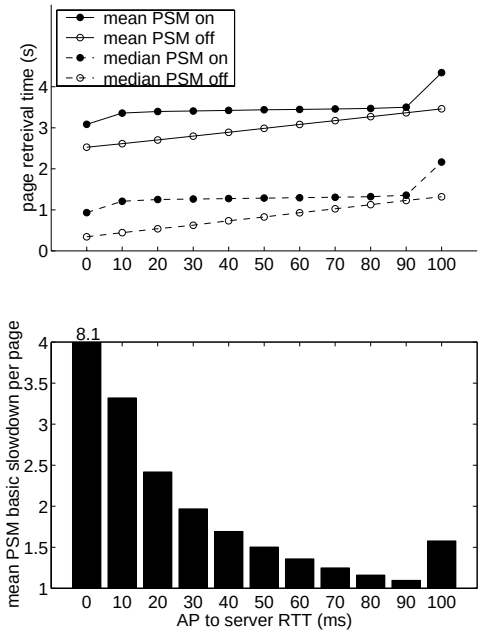

Figure 9: Mean and median page retrieval times and mean slowdown per page with basic 802.11 PSM for various AP to server RTTs.

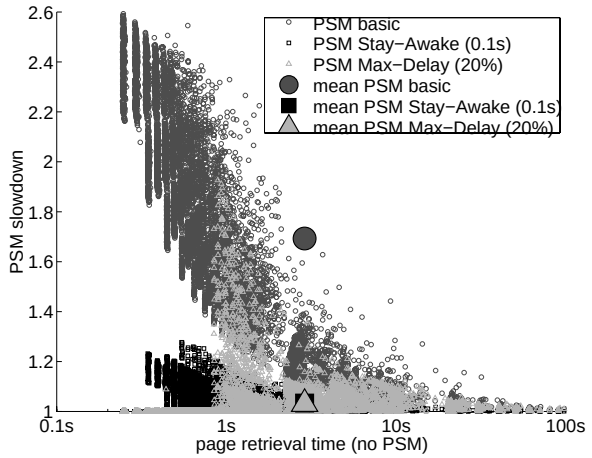

Figure 10: Per-page PSM slowdown for three PSM alternatives. Each marker represents a single web page; the x-coordinate is the retrieval time with PSM off, and the y-coordinate is the slowdown when PSM is on.

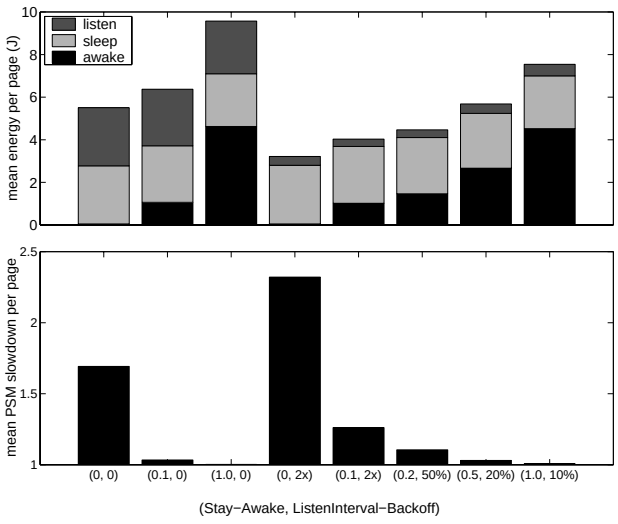

Figure 11: Mean per-page energy and slowdown comparisons for various PSM alternatives. The PSMs are described by a pair consisting of the Stay-Awake time in seconds, and the ListenInterval-Backoff algorithm. The estimated energy per page with PSM off is 54 J.

(Stay-Awake, ListenInterval -Backoff)	awake time (s)	sleep time (s)	beacons listened to
(0, 0)	0.05	54.54	545.85
(0.1, 0)	1.07	53.04	530.84
(1.0, 0)	4.63	49.45	494.86
(0, 2x)	0.05	55.06	82.83
(0.1, 2x)	1.02	53.36	68.80
(0.2, 50%)	1.46	52.81	72.45
(0.5, 20%)	2.66	51.50	88.37
(1.0, 10%)	4.52	49.58	108.38

Table 2: Simulation data used to compute energy values in Figure 11.

tween sleep mode power consumption and waking to listen to beacons. The second two columns show the results when the Stay-Awake scheme is implemented with times of 0.1 s and 1.0 s respectively. These almost completely eliminate the performance slowdown, but have the drawback of successively increasing the awake energy. The next two columns show the results when the ListenInterval-Backoff scheme of doubling the ListenInterval is implemented with Stay-Awake of 0 s and 0.1 s respectively. The ListenInterval-Backoff reduces the energy spent listening to beacons by about 8 times, but without Stay-Awake it increases the PSM slowdown to around 130% per page. With Stay-Awake, it is better than the basic 802.11 PSM in terms of both energy and performance. The final three columns show the Max-Delay scheme with maximum delays of 50%, 20%, and 10% respectively. These schemes successively tradeoff better performance for increased awake energy.

5 Related Work

A survey of energy efficient network protocols for wireless networks is provided in [10]. Although there have been many studies on the performance and energy consumption of ad hoc wireless networks (e.g. [3, 8, 15, 17]), very few have investigated infrastructure networks. Infrastructure networks have fundamentally different requirements than ad hoc networks because the access point is a centralized controller and is not constrained by power consumption. However, the basic concepts behind the Stay-Awake and ListenInterval-Backoff schemes could still potentially improve the performance and energy consumption of these networks.

In [4], Chen *et. al.* evaluate the energy consumption of various access protocols for wireless infrastructure networks. In contrast to this work, the study focuses on the

active energy consumption and the impact of contention for the wireless channel. These factors are certainly important for some environments, but with sporadic network activity the idle energy consumption dominates the active energy.

Kravets and Krishnan investigate the energy and delay impact of a power-aware transport protocol which exposes power management to applications [11]. This approach can be very effective, but it increases overall system complexity. In contrast, the approach taken in this work operates completely at the MAC level and does not require high-level information.

Chandra investigates using history-based strategies for setting the sleep interval to minimize the network power consumption when accessing streaming media [2]. This approach is only applicable for regular access patterns, in contrast to the adaptive algorithms presented in this work.

Another area of related work is power management of hard disks [12, 7]. Like the network interface, hard disks can be disabled to save energy. However, a fundamental difference with the network interface is that the information for determining when to reactivate the component may not be local to the mobile device; a packet can arrive from the external network, and the mobile device must wake up to receive it.

6 Conclusion

The existing 802.11 power-saving mode causes network round-trip-times to be rounded up to the nearest 100 ms. This adversely affects short TCP connections which are limited by the round-trip-time and which are the norm in typical web browsing scenarios. A viable solution is the Stay-Awake scheme in which the network interface stays awake for a short period of time after the link is active. Additionally, with the existing 802.11 power-saving mode implementations, almost all energy consumption is due to sleep power and listening to beacons. ListenInterval-Backoff can reduce the energy spent listening to beacons. Furthermore, it leads to longer sleep intervals which have the potential to enable deeper sleep modes.

References

[1] C. Andren, T. Bozych, B. Rood, and D. Schultz. Prism power management modes. Technical Report AN9665, Intersil Corporation, February 1997.

[2] S. Chandra. Wireless network interface energy consumption implications of popular streaming formats. In *Multimedia Computing and Networking 2002*, January 2002.

[3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks Journal*, 8(5), September 2002.

[4] J.-C. Chen, K. M. Sivalingam, and P. Agrawal. Performance comparison of battery power consumption in wireless multiple access protocols. *ACM/Baltzer Wireless Networks Journal*, 5(6):445–460, 1999.

[5] Cisco Systems, Inc. Quick reference guide cisco aironet 340 series products, January 2001.

[6] IEEE Computer Society LAN MAN Standards Committe. *IEEE, Std 802.11: Wireless LAN Medium Access Control and Physical Layer Specications*, August 1999.

[7] Fred Douglis and P. Krishnan. Adaptive disk spin-down policies for mobile computers. *Computing Systems*, 8(4):381–413, 1995.

[8] L. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *INFOCOM 2001*, Anchorage, Alaska, April 2001.

[9] S. D. Gribble and E. A. Brewer. System design issues for internet middleware services: Deductions from a large client trace. In *1997 USENIX Symposium on Internet Technologies and Systems*, Monterey, California, December 1997.

[10] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. C. Chen. A survey of energy efficient network protocols for wireless networks. *Wireless Networks*, 7(4):343–358, July 2001.

[11] R. Kravets and P. Krishnan. Application-driven power management for mobile communication. In *MOBICOM 1998*, Dallas, Texas, October 1998.

[12] Kester Li, Roger Kumpf, Paul Horton, and Thomas E. Anderson. A quantitative analysis of disk drive power management in portable computers. In *USENIX Winter*, pages 279–291, 1994.

[13] B. A. Mah. An empirical model of http network traffic. In *INFOCOM '97*, Kobe, Japan, April 1997.

[14] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven algorithm and protocol design for energy-efficient wireless sensor networks. In *MOBICOM 2001*, Rome, Italy, July 2001.

[15] S. Singh and C. Raghavendra. PAMAS: Power aware multi-access protocol with signalling for ad hoc networks. *ACM Computer Communication Review*, 28(3):5–26, July 1998.

[16] The VINT Project. *The ns Manual*, November 2001.

[17] H. Woesner, J.-P. Ebert, M. Schlaeger, and A. Wolisz. Power-saving mechanisms in emerging standards for wireless LAN's: The MAC-level perspective. *IEEE Personal Communication Systems*, 5(3):40–48, June 1998.