Improving Power Save Protocols Using Carrier Sensing and Busy-Tones for Dynamic Advertisement Windows[*]

Technical Report

December 2004

Matthew J. Miller
Dept. of Computer Science,
and Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
mjmille2@uiuc.edu

Nitin H. Vaidya
Dept. of Electrical and Computer Engineering,
and Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
nhv@uiuc.edu

ABSTRACT

Energy efficient protocols are important in ad hoc networks since battery life for wireless devices is limited. The IEEE 802.11 protocol specifies a simple power save mechanism (PSM) to conserve energy. However, the protocol needlessly wastes energy when traffic is light in a network, and significantly increases packet latency.

In this paper, we consider three techniques to improve the 802.11 PSM protocol. The first technique, Carrier Sense ATIM, uses carrier sensing to drastically reduce the energy wasted listening to the channel in 802.11 PSM. The second technique, Dynamic ATIM, dynamically extends the listening period in 802.11 PSM based on packet advertisements that are overheard. In this manner, we are able to advertise data packets just as effectively as 802.11 PSM while significantly reducing wasteful listening. The final technique, Per-Link Beacon Intervals, allows sender and receiver pairs to schedule their wake-up times independent from all the other nodes in the network based on past packet arrival times. Our results show that this technique is able to significantly reduce the average packet latency causing no or relatively little increase in energy. All of the protocols are extensively tested using the *ns-2* simulator.

1. INTRODUCTION

As the use of wireless devices continues to increase, it is evident that energy consumption is a major concern. The batteries in devices such as laptops do not allow users to stay untethered for longer than a few hours. In sensor networks, nodes may be expected to operate for weeks with a limited power supply due to the difficulty of replacing batteries for a large number of sensors in possibly difficult to access areas. Reducing energy consumption requires work at every layer of the network stack.

In this paper, we address energy saving techniques for the wireless radio. It has been shown that the wireless radio uses a significant amount of energy on wireless devices [1, 2]. The energy consumption of the wireless interface can be reduced in many ways, such as power control, using multiple channels, and routing (see [3, 4] and references therein for discussion of these techniques).

In this paper, we focus on MAC layer power save protocols. Fundamentally, power save protocols seek to answer the question: *when should the radio be put to sleep and for how long?* The motivation for power save is that sleep mode typically consumes much less power than listening to the channel [5,6]. Thus, allowing a radio to sleep as much as possible can significantly reduce its energy consumption. However, the trade-off is that a node cannot communicate with other nodes when its radio is sleeping. Therefore, packet latency usually increases as more energy is saved.

Our work proposes techniques to improve the IBSS *Power Save Mode* (PSM) in IEEE 802.11 [7]. IBSS (Independent Basic Service Set) is the protocol set for ad hoc networks. While the techniques we propose are tested with 802.11 PSM, in Section 3 we discuss how they can augment other power save protocols. Our results show that the proposed improvements to 802.11 PSM can greatly reduce energy consumption with little or no increase in the average packet latency. The techniques we use to improve 802.11 PSM are:

- *CS-ATIM* (Section 3.1) uses carrier sensing to determine if it is necessary to listen for traffic advertisements. This allows us to avoid listening for long periods when no packets will be advertised.
- *D-ATIM* (Section 3.2) dynamically re-sizes the ATIM window based on the number of advertisements to be sent. This technique leads to smaller ATIM windows in lightly loaded networks. Consequently, less energy is wasted listening during the ATIM window.
- *PLBI* (Section 3.3) schedules Per-Link Beacon Intervals based on past traffic patterns. This allows sender-receiver pairs to communicate their packets while decreasing the amount of traffic advertisement overhead.

In Section 2, we describe related work in the area of power save protocols, including a description of 802.11 PSM. In Section 3, we present techniques to improve 802.11 PSM. Section 4 compares the performance of the new protocols with 802.11 PSM using *ns-2*. Section 5 concludes the paper and discusses avenues for future work.

2. RELATED WORK

[*]This work is partially funded by NSF grant ANI-0125859 and a NSF fellowship.

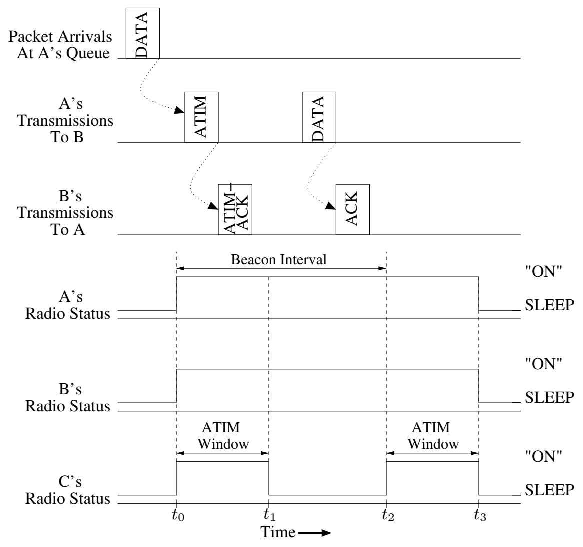

Figure 1: IEEE 802.11 IBSS power save mode [7].

Power save protocols take a variety of forms. Our primary focus is on the IEEE 802.11 IBSS PSM protocol. We start by describing the 802.11 PSM protocol [7]. Nodes are assumed to be synchronized[1] and awake at the beginning of each *beacon interval*. After waking up, each node stays on for a period of time called the *Ad hoc Traffic Indication Message (ATIM) window*. During the ATIM window, since all nodes are guaranteed to be listening, packets are advertised that have been queued since the previous beacon interval. These advertisements take the form of ATIM packets. More formally, when a node has a packet to advertise, it sends an ATIM packet to the intended receiver during the ATIM window (following IEEE 802.11's CSMA/CA rules). In response to receiving an ATIM packet, the destination will respond with an ATIM-ACK packet (unless the ATIM specified a broadcast or multicast destination address). When this ATIM handshake has occurred, both nodes will remain on after the ATIM window and attempt to send their advertised data packets before the next beacon interval, subject to CSMA/CA rules. If a node remains on after the ATIM window, it must keep its radio on until the next beacon interval. If a node does not send or receive an ATIM, it will enter sleep mode at the end of the ATIM window until the next beacon interval. This process is illustrated in Figure 1. The dotted arrows indicate events that cause other events to occur. Node **A** sends a data packet to **B**, while **C**, not receiving any ATIM packets, returns to sleep for the rest of the beacon interval.

In [8], it is shown that a static ATIM window does not work well for all traffic loads. Intuitively, higher traffic loads need larger ATIM windows. Based on this observation, DPSM [9] attempts to dynamically adjust the ATIM window size in single-hop networks (i.e., WLANs) based on indications such as the listening time at the end of the ATIM, the number of packets pending for a node, and the number of packets that could not be advertised in the previous beacon interval. Unlike our work, these protocols adjust the current ATIM window based on traffic in past beacon intervals. By contrast, our protocol adjusts the current ATIM window based on the traffic in the current beacon interval. IPSM [10] is similar to D-ATIM in that the ATIM window ends when the channel is idle for a specified amount of time. In Section 3.2, we discuss the differences between D-ATIM and IPSM.

[1]We will discuss synchronization later in the paper.

The technique we use to dynamically adjust the ATIM window is similar to previous work in priority scheduling in ad hoc networks [11,12]. In this body of work, high priority packets use a smaller backoff and/or defer interval than low priority traffic. In our work, ATIMs are essentially high priority packets. Unlike previous priority scheduling work, in which low priority traffic occasionally preempts high priority traffic, our dynamic ATIM window protocol is designed such that data packets never preempt ATIM packets.

In TIPS [13], the ATIM window is divided into two slots. If a beacon packet is received during the first slot, it indicates that nodes should stay on to receive ATIMs later in the ATIM window. If the first beacon packet is not received until the second slot, then the node can return to sleep since no more advertisements will follow. In our work, carrier sensing is used as an indication that nodes should remain on longer. The time it takes to carrier sense is usually much shorter than the time it takes to access the channel and send an entire packet. The idea of preamble sampling has been used with B-MAC [14] and Aloha [15]. The basic idea of preamble sampling is that the packet preamble is long enough to be detected by all nodes that are periodically sampling the channel in between sleep periods (i.e., the preamble must be slightly longer than the sleep time between sampling periods). When sleeping nodes sample the channel and detect the preamble, they remain on to receive the entire packet. The advantage of this technique is it works in completely unsynchronized environments. However, unlike our proposed protocol, the long preambles make packets more vulnerable to collisions (e.g., hidden terminals) and retransmissions are more costly. Also, all nodes detecting the preamble must remain on to receive the data packet, even if they are not the intended receiver. Our carrier sensing mechanism works despite collisions and is used with 802.11 PSM to avoid having all one-hop nodes receive each data packet.

One of our techniques is to dynamically adjust a Per-Link Beacon Interval based on past traffic patterns. In [16], a coarse-grained power save adjustment is made based on traffic. When a node using on-demand routing realizes that it is on an active route, it stops doing any kind of power save. When a node is not on an active route, it enters 802.11 PSM. The T-MAC [17] protocol extends a power save protocol for sensors, S-MAC [18], by adjusting the time nodes are awake based on past traffic patterns. In LISP [19], nodes attempt to predictively remain on in beacon intervals based on the correlation of overhearing ATIM-ACKs in one ATIM window and an ATIM in the next ATIM window.

Another common power save strategy is for nodes to remain awake based on their local topology and/or traffic [5, 20, 21]. Other protocols use an out-of-band channel (e.g., a second, non-interfering radio) to wake up sleeping neighbors [22, 23]. Other work focuses on nodes communicating despite uncoordinated sleep schedules [24–26]. Finally, other protocols have used TDMA approaches [27, 28] where communicating nodes attempt to schedule non-interfering time slots to wake up and transmit or receive data packets.

3. PROTOCOL DESCRIPTIONS

From the description of 802.11 PSM in Section 2, we can see that the ATIM window can waste a significant amount of data when the traffic load is low. For example, in previous work [9, 19] some typical values for the ATIM window and beacon interval have been 20 ms and 100 ms, respectively. Thus, even when *no* traffic is being sent, nodes listen to the channel for 20% of the time. It is obvious that more energy could be conserved by reducing the size of the ATIM window when traffic is sparse. However, if the ATIM window becomes too small, then nodes will not be able to advertise their data since the window ends before they are able to access the channel and send an ATIM. Thus, the techniques in Section 3.1 and Section 3.2 reduce the overhead of the ATIM window when traffic is sparse and provide larger ATIM windows when there is more data to advertise.

The protocols in Section 3.1 and Section 3.2 allow the ATIM window overhead to remain small when few ATIMs need to be sent. In Section 3.3, we present an idea to reduce the number of ATIMs that are sent to further decrease the dynamic ATIM window overhead. Each of these three protocols can either be implemented individually or they can be combined with each other.

For the protocols in Section 3.2 and Section 3.3, the node's clocks must be synchronized. We assume that synchronization is done via an out-of-band mechanism. For example, time synchronization can be achieved by adding a chip-scale atomic clock to a wireless device. The chip-scale atomic clocks currently being built use less than 75 mW of power, with a goal of using 10-30 mW eventually [29]. In comparison, WLAN cards use about 130 mW in the sleep state [5]. Thus, it may be feasible to add chip-scale atomic clocks to devices with enough power to support WLAN cards. For the protocol in Section 3.1 and 802.11 PSM, we relax this assumption by only requiring that each node in the network is synchronized within Δ seconds of every other node.

3.1 Beacon Interval Carrier Sensing

From the description of 802.11 PSM in Section 2, we observe that it is possible that most beacon intervals have no packets to be advertised. In this case, the ATIM window needlessly wastes energy. However, when there is traffic at the beginning of a beacon interval, nodes need a mechanism to advertise their packets. Thus, the ATIM window concept cannot be completely removed. What is needed is a energy-efficient boolean signal so that a node can let neighbors know when it has traffic to advertise and, hence, an ATIM window is needed for that beacon interval.

For this purpose, we propose *Carrier Sense ATIM* (CS-ATIM) which adds a short carrier sensing period at the beginning of each beacon interval as shown in Figure 2. The basic idea is that the time it takes to carrier sense the channel busy or idle, T_{cs}, is significantly smaller than the ATIM window, T_{aw}. Rather than every node waking up for T_{aw} at the beginning of every beacon interval, the nodes will only wake up for T_{cs} at the beginning of every interval when there are no packets to be advertised. When there are packets to be advertised, the nodes will wake up for an entire ATIM window after the carrier sensing period.

Using Figure 2, we will explain how CS-ATIM works. The shaded regions in Figure 2 indicate that a node is transmitting a packet. At time t_0, there are no packets to be advertised so all nodes wake up for T_{cs} time and return to

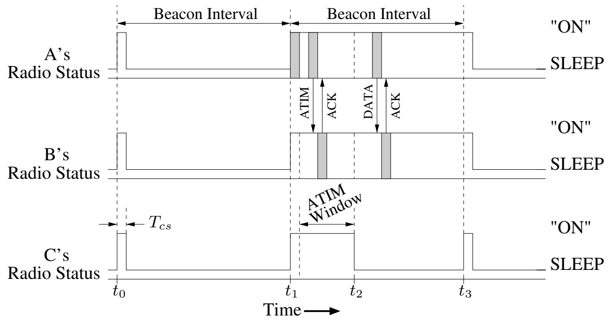

Figure 2: CS-ATIM protocol.

sleep when the channel is detected idle. At time t_1, the nodes wake up for the start of the next beacon interval. This time, node **A** has a packet to advertise, so it transmits a "dummy" packet to make the channel busy. When nodes **B** and **C** finish carrier sensing the channel at time $t_1 + T_{cs}$, the channel is detected busy because of **A**'s packet transmission. Thus, all nodes who carrier sensed the channel busy or transmitted a "dummy" packet will remain on for an ATIM window of length T_{aw} after the carrier sensing period. During the ATIM window, **A** sends an ATIM to **B** and **B** replies to **A** with an ATIM-ACK. Because of this exchange, **A** and **B** will remain on for the rest of the beacon interval. Because **C** did not send or receive an ATIM during the ATIM window, it returns to sleep at the end of the ATIM window at time t_2. After the ATIM window, **A** and **B** exchange the data packet and corresponding ACK. At time t_3, a new beacon interval begins and all of the nodes return to sleep after carrier sensing the channel as idle.

The value of T_{cs} is chosen to be long enough to carrier sense the channel as idle or busy with a desired level of reliability. According to the 802.11 specification [7], the clear channel assessment (CCA) for compliant hardware must be less than 15 μs. In our experiments, we use a much larger value for T_{cs} to mitigate the effects of short-term fading. The dummy packet transmitted by a node with packets to advertise does not contain any information that needs to be decoded; its only purpose is to cause other nodes to detect the channel as busy. The advantage of not having information in the dummy packet is that multiple nodes can transmit simultaneously, causing collisions at the receivers, without hindering the protocol. If a collision occurs at the receiver, it can still detect the channel as busy and remain on for the ATIM window. In the ATIM window, nodes use the standard 802.11 CSMA/CA protocol to send their ATIMs and ATIM-ACKs without causing collisions. A node that transmits a dummy packet cannot carrier sense dummy packets being sent by other nodes at the beginning of the beacon interval. However, this does not affect the protocol since a node stays on for the ATIM interval whenever it transmits a dummy packet *or* carrier senses the channel busy.

From this description of CS-ATIM, it is clear that nodes can use significantly less energy than 802.11 PSM listening at the beginning of each beacon interval when there are no packets to be advertised. When there are packets to be advertised, CS-ATIM uses slightly more energy than 802.11 PSM because of the short carrier sensing period. In terms of packet latency, 802.11 PSM does slightly better than CS-ATIM. One reason is that data packets that arrive after the carrier sensing period but before the end of the ATIM window may be sent in the current beacon interval in 802.11 PSM. In CS-ATIM, such packets may have to wait until the next beacon interval. Also, there is a slightly larger delay in CS-ATIM since the ATIM window does not end until $T_{cs} + T_{aw}$, whereas the 802.11 PSM ATIM window ends T_{aw} after the beginning of the beacon interval.

With CS-ATIM, we note that carrier sensing for energy on the channel, as opposed to actually decoding a packet, creates the risk that nodes may erroneously carrier sense energy that is due to interference in the frequency band rather than the transmission of a dummy packet. In this case, a node remains on for the ATIM window even though none of its neighbors sent a dummy packet. We refer to this as a *false positive*. In Section 4, the effects of false positives on CS-ATIM are tested.

As mentioned, CS-ATIM can be adapted to operate in networks without perfect synchronization. We assume that the node's clocks are always within Δ seconds of each other. Thus, Δ represents the maximum error between the clocks of *any* two nodes in the network. To handle synchronization errors, the following changes are made to CS-ATIM:

- A dummy packet is transmitted for $2\Delta + T_{cs}$ time instead of T_{cs} time.
- Nodes that do not have packets to advertise begin their carrier sensing period Δ time after the beginning of the beacon interval (according to their local clock). Originally, these nodes would begin carrier sensing immediately at the beginning of a beacon interval.
- ATIM windows last for $2\Delta + T_{aw}$ time instead of T_{aw} time. A node is not allowed to transmit any ATIMs for the first Δ time of the ATIM window and the last Δ time of the ATIM window. However, a node *may* send ACKs, ATIM-ACKs, and receive packets during the entire $2\Delta + T_{aw}$ duration of the ATIM window.

To preserve the flow of the paper, we move the discussion of the correctness of these modifications to Appendix A.

The basic idea from CS-ATIM can be adapted to other power save protocols besides 802.11 PSM. Whenever a node is scheduled to listen in a power save protocol, it can do carrier sensing at the start of its scheduled wake-up time to determine if it can return to sleep because there are no nodes with data to send. For example, in a TDMA protocol, nodes can carrier sense at the beginning of their scheduled slot and return to sleep if there is no data to be sent.

3.2 Dynamic ATIM Window Adjustment

The CS-ATIM protocol is more energy efficient than 802.11 PSM when there are a large number of beacon intervals in which no nodes have packets to advertise. However, if there is a small number of packets to be advertised each beacon interval, then requiring nodes to listen for the entire ATIM window wastes energy. Ideally, the ATIM window should be long enough for all the ATIMs which need to be transmitted and then the ATIM window should end right after the last ATIM-ACK is received[2]. Essentially, this is what the work in [9] attempts to do by using heuristics based on feedback about the size of a node's ATIM window in the previous beacon interval. We take a different approach by dynamically extending, up to an upper bound, the ATIM window as long as ATIMs are being sent in the current beacon interval. When the channel is idle for a sufficiently long period of time, which is discussed below, the nodes assume that all ATIMs have been sent and commence sending data packets until the beginning of the next beacon interval.

Our protocol, Dynamic ATIM (D-ATIM), uses a similar idea to what has been proposed for priority traffic scheduling [11, 12]. Basically, ATIM packets have a shorter defer time before accessing the channel than the first data packet a node tries to send after it is finished sending ATIMs. We will start by describing how the protocol operates in a WLAN environment where all nodes are within a single hop of each other. In Section 3.2.1, we describe how the protocol can be modified to work in a multi-hop setting.

First, we give ATIM packets a different maximum contention window size (CW_{aw}) than data packets (CW_{data}). In the IEEE 802.11 specification [7] for direct-sequence spread spectrum (DSSS), the default CW_{data} is 1023 slots and the default slot time, T_{slot}, is 20 μs. Using such a large contention window for ATIMs is unnecessary when the entire ATIM window is typically on the order of tens of milliseconds. Also, only one ATIM is sent per sender-receiver pair whereas multiple data packets may then be sent over that link after the ATIM window. Thus, the number of ATIM packets sent in the ATIM window should be less than or equal to the number of data packets sent following the ATIM window. This means there should be less nodes contending for access during the ATIM window since each sender-receiver link contends for the channel only once during the ATIM phase, but potentially multiple times during the data phase. Therefore, it is not unreasonable to make $CW_{aw} < CW_{data}$ in most scenarios. Thus, nodes that have ATIMs to send during the ATIM phase use the same protocol as 802.11 CSMA/CA, but use CW_{aw} as the maximum contention window size rather than the default CW_{data}.

At the beginning of a beacon interval, every node listens to the channel and sets a timer to expire after:

$$T_{idle} = DIFS + T_{slot} \cdot CW_{aw} + T_{retry} \tag{1}$$

time, where $DIFS$ is the DCF Interframe Space as specified by IEEE 802.11 [7] and T_{retry} is the time a node waits before attempting to retransmit an ATIM when an ATIM-ACK is not received. In our implementation [30]:

$$T_{retry} = prop_{max} + SIFS + T_{ack} + prop_{max} \tag{2}$$

where $SIFS$ is the Short Interframe Space as specified by IEEE 802.11 [7], $prop_{max}$ is the maximum propagation delay in the network, and T_{ack} is the time it takes to send an ATIM-ACK packet[3].

If a node sends or receives a packet before the timer expires, the timer is reset to end T_{idle} time after the packet is sent or overheard. To avoid starving data packets, an upper limit is set on the size the dynamic ATIM window can reach. Currently, this upper bound is equal to the default, static ATIM window size, T_{aw}, used for unmodified 802.11 PSM.

Whenever a node does not send or overhear a data packet for T_{idle} time or the upper bound on the dynamic ATIM

[2]This statement assumes traffic is not so heavy that the ATIM window grows large enough that data packets can never be sent.

[3]T_{ack} is a constant since all ATIM-ACK packets have a fixed, specified size.

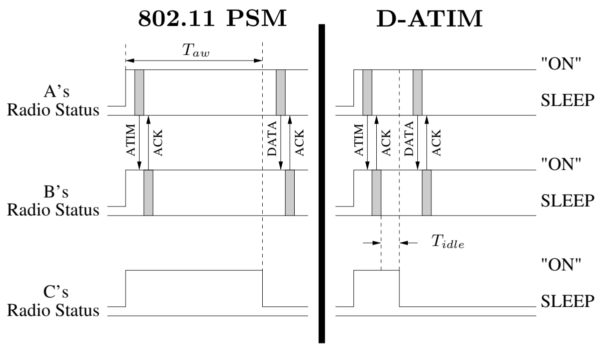

Figure 3: 802.11 PSM vs. D-ATIM.

window is reached, the node will end the ATIM phase and enter into the data phase. As in 802.11 PSM, if a node sent or received an ATIM during the ATIM window, it remains on for data communications. Otherwise, the node returns to sleep until the beginning of the next beacon interval.

An example of D-ATIM compared to 802.11 PSM is given in Figure 3. Here, **A** sends an ATIM to **B**. With 802.11 PSM, **A** and **B** must wait until after T_{aw} to send the data packet and **C** must remain on for the entire T_{aw} time of the static ATIM window. However, with D-ATIM, **A** and **B** start sending the data packet T_{idle} time after the ATIM-ACK packet is received. Also, **C** returns to sleep T_{idle} time after receiving the ATIM-ACK rather than waiting for the entire T_{aw} duration of the ATIM window as in 802.11 PSM.

From this description, we see that D-ATIM never uses more energy in the ATIM window than 802.11 PSM and may use much less energy when a small number of ATIMs are sent. In terms of latency, D-ATIM may perform worse than 802.11 PSM if a data packet arrives at the node towards the end of 802.11 PSM's static ATIM window. In this case, 802.11 PSM can advertise the packet and send the data in the current beacon interval. By contrast, if D-ATIM's dynamic ATIM window has already ended, the node may have to wait until the next beacon interval to advertise the packet. However, D-ATIM can improve packet latency over 802.11 PSM. If, for example, only one ATIM is sent, then D-ATIM transmits the first data packet after the dynamic ATIM window ends (i.e., about $T_{atim} + T_{ack} + T_{idle}$, where T_{atim} is the time to send an ATIM). By contrast, 802.11 PSM must wait until the end of the ATIM window and cannot transmit the first data packet until about T_{aw}.

We note some differences between D-ATIM and a similar protocol, IPSM [10]. First, D-ATIM is adapted for multi-hop environments in Section 3.2.1. IPSM is designed and tested exclusively in single-hop scenarios.

Second, the timeout, T_{idle}, for IPSM is essentially equal to $T_{slot} \cdot CW_{aw}$, whereas D-ATIM adds the $DIFS$ and T_{retry} terms to T_{idle} (see Equation 1) for correctness. To see why D-ATIM uses Equation 1, consider the following situation. During the ATIM window, a sending nodes transmits an ATIM which is lost. Thus, T_{retry} time after sending the ATIM, the sending node begins contending to transmit another ATIM (up to a specified retransmission threshold). This contention time for the channel has an upper limit of $DIFS + T_{slot} \cdot CW_{aw}$ if the channel remains idle during this period. If the channel does not remain idle due to another node transmitting a packet, both the sender and receiver detect this channel and reset their timer for T_{idle} time. Thus, the the longest the channel can remain idle when at least one node has a packet to send is $DIFS + T_{slot} \cdot CW_{aw}$. Therefore, Equation 1 represents the longest time the channel can remain idle after a packet transmission when a node has a packet to retransmit.

The final difference in IPSM and D-ATIM is that IPSM uses more coarse-grained timeouts which can result in unnecessary energy consumption. IPSM divides the ATIM window into discrete decision checkpoints to determine if the ATIM window should end. These checkpoints are equally spaced $ATIMinc$ apart. At each checkpoint, a node ends its ATIM window if the channel has been idle for at least T_{idle} time. Let a checkpoint occur at time t_0 and the channel has not been idle for T_{idle} time. Then, assume that at time $t_0 + \epsilon$ (where $\epsilon < ATIMinc$), the channel has been idle for exactly T_{idle} time. In this scenario, the IPSM node does not end its ATIM window until the next checkpoint at time $t_0 + ATIMinc$. Essentially, the node could have ended its ATIM window $ATIMinc - \epsilon$ earlier. D-ATIM, in contrast, ends the ATIM window *exactly* T_{idle} time after the channel was last busy.

3.2.1 Modifications for Multi-Hop Environments

As described above, D-ATIM only works correctly in WLAN, single-hop environments. However, if the use of a busy-tone channel [31, 32] is available, then the protocol can easily be adapted to multi-hop environments. A busy-tone provides a separate control channel with which nodes can transmit one bit of information to their one-hop neighbors via the absence or presence of the busy-tone signal on the channel.

To see why D-ATIM does not work correctly in multi-hop topologies, consider the simple topology **A**↔**B**↔**C**↔**D**. Assume that **B** has a packet to advertise to **A**, and that **C** has a packet to advertise to **D**. Let **B** send its ATIM to **A** first, at time t_0. Note that **C** will overhear this ATIM and refresh its dynamic ATIM window timer to expire at time $t_0 + T_{atim} + T_{idle}$. However, **D** will not overhear this ATIM and its timer will still expire at time $t_0 + T_{idle}$. Thus, since **C** and **D** have different end times for their dynamic ATIM window, it is possible that **C** (with the longer ATIM window) tries to send an ATIM to **D** after **D** has already ended its dynamic ATIM window and returned to sleep.

To combat this problem, we propose D-ATIM-BT, which adds a busy-tone to D-ATIM. The original D-ATIM protocol is modified as follows. Whenever a node overhears a data channel transmission and it has a packet to advertise, it immediately begins transmitting a busy-tone to its one-hop neighbors for the duration of the overheard transmission. In addition to resetting the dynamic ATIM window timer as specified by D-ATIM, nodes also reset their timer when they finish receiving a busy-tone (and remain in the dynamic ATIM window as long as the busy-tone is active).

Now, we can see that D-ATIM-BT correctly handles the situation described above since **C** would transmit a busy-tone when **B** is sending the ATIM. **D** would overhear this busy-tone from **C** and reset its dynamic ATIM window timer to $t_0 + T_{atim} + T_{idle}$ when the busy-tone (and hence ATIM transmission) are finished. Thus, **C** and **D** will have the same ending time for their dynamic ATIM windows.

Busy-tones collisions at the receiver do not affect the correctness of D-ATIM-BT. The reception of multiple, concurrent busy-tones causes a node to detect energy on the busy-tone channel the same as if only one busy-tone was sent.

The basic concept of D-ATIM-BT can also be used in other power save protocols. When nodes are scheduled to listen for packets, they can return to sleep if no activity is detected on the data or busy tone channels for a short time. Senders that cannot yet access the channel, but know that their intended receiver is on, can transmit busy tones to keep their receiver awake. For example, in S-MAC [18], a sender could dynamically extend the time that its receiver is listening by transmitting busy tones when the sender is unable to win access to the channel.

3.3 Per-Link Beacon Intervals

In this section, we present another power saving technique, largely to compliment the protocols in Section 3.1 and Section 3.2. We observe that nodes with little or no traffic to send or receive may still suffer a significant amount of listening with CS-ATIM and D-ATIM if a node in their vicinity is frequently sending ATIMs. In this case, the nodes use about the same amount of energy as 802.11 PSM.

ATIM windows are necessary, for example, to contact a neighbor with which a node has not previously communicated. However, once nodes have initially communicated with each other, they could schedule Per-Link Beacon Intervals (PLBIs) with that particular neighbor. By PLBIs, we mean that each pair of communicating nodes agree on a future time to wake up and send more data independent of the default 802.11 beacon interval. By using this technique, we can reduce the number of ATIMs sent, which can significantly decrease the listening in the CS-ATIM and D-ATIM protocols. If the first packet of a flow is advertised using the ATIM procedure, then all subsequent packets can be sent using PLBIs. In this scenario, only one ATIM needs to be sent per flow rather than one ATIM per beacon interval in which there is a data packet in the queue.

For example, consider a sender that generates a packet for a given destination every two beacon intervals. Normally, this would require the sender to also transmit an ATIM for the destination once every two intervals. However, if the sender and receiver independently agree on a time to wake up once every two intervals, then the data packet can be sent without an ATIM advertisement.

As mentioned in the Section 2, traffic indications have been used to control power save protocols [16, 17, 19, 33]. The inherent difficulty in such schemes is predicting future events accurately based on correlation with past events. With PLBIs, this translates to the sender predicting the next packet arrival time for a destination based on past arrivals.

As a proof-of-concept, we tested PLBIs with a simple traffic prediction protocol. We note that more complex algorithms in statistics or signal processing may make the traffic prediction more robust to outliers and traffic variations, but our purpose is merely to show that even simple prediction methods can complement CS-ATIM and D-ATIM.

To predict when the next beacon interval for a specific receiver will occur, a sender keeps track of the interarrival of data packets. Using this data, the sender updates the predicted average beacon interval, BI_{avg}, for the destination using an exponentially weighted moving average (EWMA):

$$BI_{avg} = \alpha \cdot BI_{avg} + (1-\alpha) \cdot t_{diff} \tag{3}$$

where α is a knob to control the relative weight of the most recent sample and t_{diff} is the difference in MAC layer arrival times of the two most recent packets for that destination. We also compute the exponentially weighted moving standard deviation (EWMSD) of the packets [34] with β as the knob for the relative weight of the most recent sample:

$$BI_{dev} = \sqrt{\beta \cdot BI_{dev}^2 + (1-\beta) \cdot (t_{diff} - BI_{avg})^2} \tag{4}$$

Based on these measurements, the sender piggybacks the next time the receiver should wake up, BI_{cur}, on every data packet. This value is computed as:

$$BI_{cur} = (t_{last} + BI_{avg} - k \cdot BI_{dev}) - t_{now} \tag{5}$$

where t_{last} is the last time a packet arrived for the destination at the sender's MAC layer, t_{now} is the current clock time at the sender, and k is a specified number to control how many standard deviations early the nodes will wake up. If BI_{cur} is a negative number (e.g., $k \cdot BI_{dev} > BI_{avg}$), then BI_{cur} is set to zero and the nodes enter an "always on" state where they do not return to sleep. If an 802.11 PSM beacon interval begins and a node has a packet for a destination with a PLBI scheduled, it will go ahead and advertise the packet at the beginning of the 802.11 PSM beacon interval rather than wait for the PLBI to occur. This helps reduce the worst case latency of data packets.

Whenever the nodes turn on for their current PLBI, they set a timer for a specified time, BI_{idle}. When this timer expires, the nodes can return to sleep until the next scheduled PLBI or start of an 802.11 PSM beacon interval. Similar to the dynamic ATIM window timer discussed in Section 3.2, this beacon interval idle threshold timer gets reset for BI_{idle} more time whenever a packet is sent or received. If a node's PLBI extends into the beginning of the next 802.11 PSM beacon interval, then the PLBI is rescheduled to occur after the ATIM procedure is complete. Similarly, if a node's PLBI is scheduled to begin during the ATIM procedure, then the PLBI is rescheduled after the ATIM procedure finishes.

From Equation 5, we see that a trade-off exists in energy and latency. With a smaller k value, nodes wake up less frequently, but also may have a larger latency (although, the latency will not be more than when PLBIs are not used). Also, the more variance the packet interarrival time shows, the more frequently the nodes have to wake up due to the unpredictability of the traffic.

The basic idea of PLBIs can be applied to other power save protocols. See [33] for an example of how a similar technique has been adapted to a power save protocol which uses an out-of-band channel.

4. SIMULATION RESULTS

To test our protocols, we simulated them by modifying the MAC and physical layers of *ns-2* [30]. We use the notation P_{tx}, P_{rx}, P_{listen}, and P_{sleep} to refer to the power a node consumes to transmit, receive, listen, and sleep, respectively. We test the following protocols:

- **ALWAYS ON [7]:** This is the IEEE 802.11 protocol with no power save. It is the default, unmodified MAC protocol in *ns-2*. Because nodes never sleep, ALWAYS ON uses the most energy, but has the lowest latency.

- **802.11 PSM ON [7]:** This is the standard IEEE 802.11 protocol with power save enabled. 802.11 PSM is described in Section 2.

- **CS-ATIM:** This is 802.11 PSM with the carrier sensing modification described in Section 3.1.
- **D-ATIM-BT:** This is 802.11 PSM with the dynamic ATIM modification for multi-hop networks described in Section 3.2.
- **PLBIs:** We test all three power save protocols (802.11 PSM, CS-ATIM, D-ATIM-BT) with the Per-Link Beacon Interval addition discussed in Section 3.3. When this technique is added to the protocol, it is denoted by adding a "+" symbol after the protocol name.
- **802.11 OPT:** This protocol needs more explanation because we are unaware of any other work which uses it. 802.11 OPT represents the optimal latency and energy consumption possible *for the IEEE 802.11 protocol*. We do not claim, nor believe, that it is optimal across the entire range of possible MAC protocols. However, it provides a useful baseline to measure other protocols against, since energy and latency are two competing metrics and the desired trade-off between these metrics is application-dependent. The latency for 802.11 OPT is simply equal to the latency for ALWAYS ON. Generally, ALWAYS ON is better than any power save protocol in terms of latency since a node can immediately begin contending for medium access rather than waiting for the next scheduled wake-up time for the sender and receiver. To calculate 802.11 OPT's energy, a node consumes P_{tx} power while sending a packet, P_{rx} power while receiving a packet for which it is the intended receiver, P_{listen} power while deferring and backing off as required by IEEE 802.11, and P_{sleep} power at all other times. Essentially, for a given scenario, 802.11 OPT represents the lowest possible energy achievable for nodes using IEEE 802.11 if they slept as aggressively as possible. Obviously, such a protocol is not possible since it requires the receiver to have perfect, advance knowledge of when a sender will attempt to begin contending for the channel to send a packet and wake up at that time (even if the two nodes had never communicated previously).

In this section, we only present results from multi-hop scenarios for brevity. We did extensive testing of the protocols in a WLAN, single-hop environment as well and the trends are similar. We use 2 Mbps radios that have a 250 m range. For each scenario, we place 50 nodes uniformly at random in a 1000 m $\times$ 1000 m area and only consider scenarios in which every node has a route to every other node in the network. To avoid second-order effects from routing protocols (e.g., the long delay for RREQs to traverse a power save network), we use Floyd-Warshall's All-Pairs Shortest Path algorithm [35] to precompute routes for all the nodes. Each data point is averaged over 20 tests.

We vary different parameters for each test, but the following values are used when the parameter is not being varied. The beacon interval length is 100 ms and T_{aw} is 20 ms. There are five flows sending 512 byte data packets at a rate of 1 kbps per flow (i.e., each flow uses about 0.05% of the channel bitrate *per hop*). We use a relatively low rate because at high rates, power save protocols become ineffective since nodes essentially transition to the ALWAYS ON state.

The sender and receiver of each flow are chosen uniformly at random and the traffic is constant bitrate (CBR) unless otherwise noted. With CS-ATIM, the carrier sensing time, T_{cs}, is set to 1 ms, which is about 66 times larger than the 15 μs required by 802.11 compliant hardware. We set T_{cw} to be large to mitigate the effects of short-term fading. In D-ATIM-BT, the maximum backoff interval size, CW_{aw}, is set to be 127 slots. For the power characteristics of the radio [5,16], we use: $P_{tx} = 1.4$ W, $P_{rx} = 1.0$ W, $P_{listen} = 0.83$ W, and $P_{sleep} = 0.13$ W.

In tests where we use PLBIs, we set both the EWMA and EWMSD weights, α and β, to be 0.9. The number of standard deviations, k is set to 1. The beacon interval idle timeout threshold, BI_{idle}, is set to 5 ms. Refer to Equation 5 and Section 3.3 to recall the usage of these parameters.

As mentioned in Section 3.1, CS-ATIM is vulnerable to false positives when it erroneously carrier senses the presence of a busy-tone signal. Thus, in some of our tests we evaluate the effect of false positives on CS-ATIM by specifying a percentage that represents the probability that a node remains on for the ATIM window even when none of its neighbors transmitted a dummy packet. For example, a protocol denoted as "CS-ATIM, 10%" means that with probability 0.1, a node running the CS-ATIM protocol remains on for the ATIM window even though there were no dummy packets being transmitted.

In this paper, we present tests that measure energy and latency by varying the following parameters:

- **Beacon Interval Time:** We vary the length of the beacon interval to increase the amount of sleep time between beacon epochs.
- **Number of Flows:** We increase the number of flows to see how the protocols handle the additional load.
- **False Positives:** For CS-ATIM, we show how false positives (i.e., erroneously detecting the busy-tone channel as busy) affect the energy consumption.
- **Synchronization Errors:** For 802.11 PSM and CS-ATIM, we show the effect that errors in the time synchronization of local clocks has on the protocols.
- **Per-Link Beacon Intervals:** We show how the addition of PLBIs affects the power save protocols with both CBR (low and higher rates) and Poisson traffic.

In our tests, energy is measured in units of Joules/bit. This is calculated by dividing the total energy consumed by all nodes in a scenario by the total number of data bits that are received by their final destination. The latency is calculated as the average end-to-end latency over all packets received by their final destination in a given scenario.

4.1 Evaluating CS-ATIM and D-ATIM

First we tested the power consumption and latency of CS-ATIM and D-ATIM. For these tests, we varied the length of the beacon interval from 40 ms to 150 ms. As shown in Figure 4, all of the power save protocols show a decrease in energy as the beacon interval is increased since this allows nodes more time to sleep between ATIM windows. We can see that CS-ATIM (with no false positives) and D-ATIM-BT both perform significantly better than 802.11 PSM. CS-ATIM and D-ATIM-BT use about the same amount of energy and consume anywhere from 30 to 60% less energy than 802.11 PSM for the parameters tested. All of the protocols

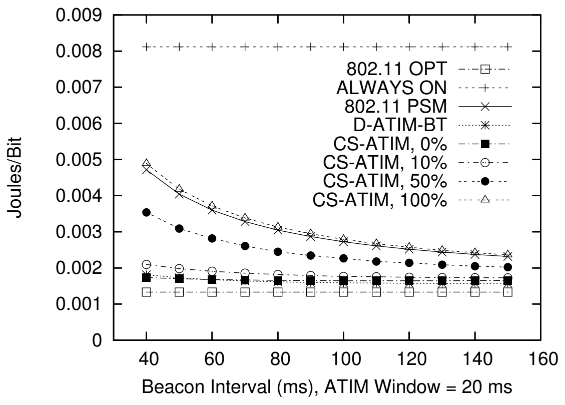

Figure 4: Energy vs. beacon interval.

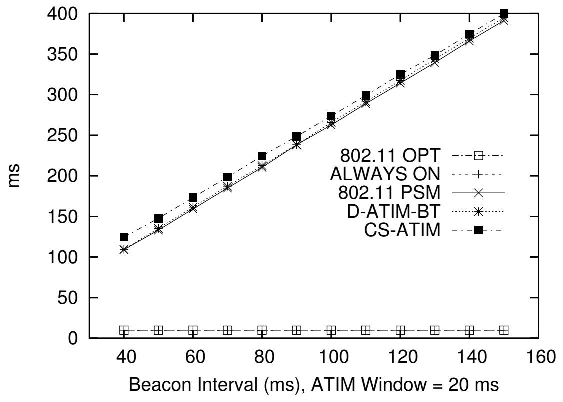

Figure 5: Latency vs. beacon interval.

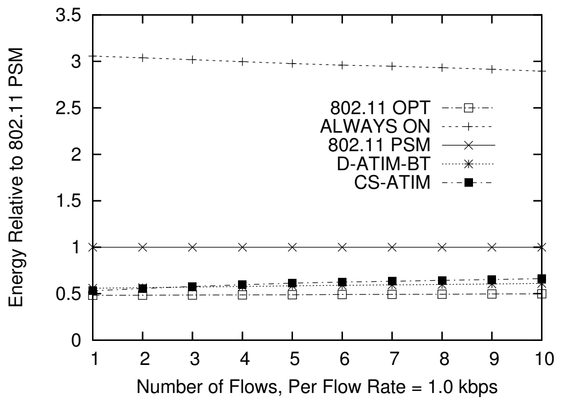

Figure 6: Energy vs. number of flows.

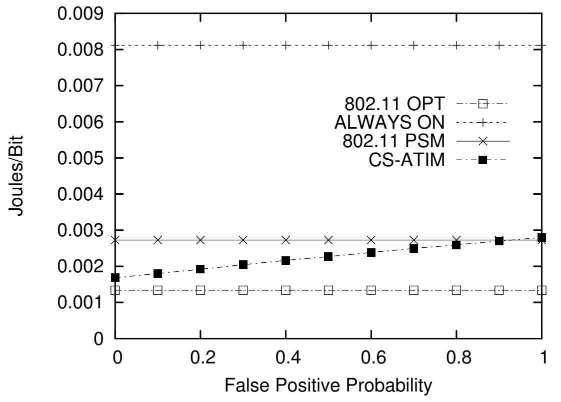

Figure 7: Energy vs. false positive probability.

do significantly better than ALWAYS ON; even 802.11 PSM consumes anywhere from 40 to 70% less energy than ALWAYS ON. When compared to 802.11 OPT, CS-ATIM and D-ATIM-BT use only about 18 to 30% more energy.

From Figure 4, we also see how false positives affect CS-ATIM. Even with false positives as high as 50%, CS-ATIM does significantly better than 802.11 PSM. With 100% false positives, CS-ATIM converges to about the same protocol as 802.11 PSM. Because CS-ATIM still has the overhead of the small carrier sensing time with 100% false positives, it uses slightly more energy than 802.11 PSM. However, 100% false positives is the worst case scenario, and, in practice, the false positive rate is expected to be much smaller.

The disadvantage of using power save protocols is evident in Figure 5, which shows the latency of the protocols. Just as an increasing beacon interval decreases the energy consumption, it increases the latency since there is a greater probability packets arrive outside the ATIM window and the time that these packets have to wait to be advertised increases. ALWAYS ON, and hence 802.11 OPT by definition, always do significantly better than the power save protocols.

For the power save protocols, D-ATIM-BT and 802.11 PSM always have about the same latency. This indicates that the decrease in latency that 802.11 gets from packets arriving during its longer ATIM window is equalized by the fact that D-ATIM-BT is able to start sending its data packets sooner by ending its ATIM window earlier. CS-ATIM, however, tends have a slightly higher latency. The difference between CS-ATIM's latency and the latency of D-ATIM-BT and 802.11 PSM is relatively constant in the range of 8 ms to 15 ms. This small increase in CS-ATIM latency comes from the fact that there is a greater probability that packets may arrive during 802.11's longer ATIM window.

In Figure 6, we see how a traffic increase affects the protocols' energy consumptions. Note that the y-axis for the graph is energy consumed (in Joules/bit) relative to 802.11 PSM. We see that the benefits of CS-ATIM and D-ATIM slightly decrease as the traffic in the network increases. However, even when there are 10 flows in the network, CS-ATIM uses about 35% less energy than 802.11 PSM and D-ATIM-BT uses about 40% less energy than 802.11 PSM.

4.1.1 *False Positives in CS-ATIM*

As mentioned earlier, CS-ATIM is susceptible to false positives when nodes carrier sense the channel as busy even when no dummy packet was sent. In this case, nodes waste energy by staying up for an ATIM window when there are no packets to be advertised. In Figure 7, we see that CS-ATIM shows a linear increase in energy as the false positive probability increases. In the worst case, when the false positive probability is equal to 1, the energy consumption of CS-ATIM converges to slightly more than that of 802.11 PSM since CS-ATIM still has the overhead of carrier sensing.

4.1.2 *Synchronization Errors in CS-ATIM*

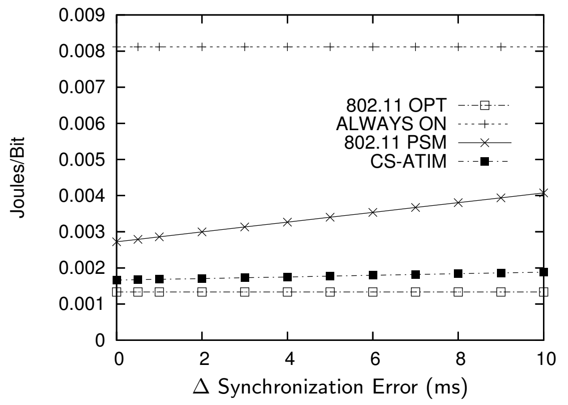

Figure 8: Energy vs. synchronization error (Δ).

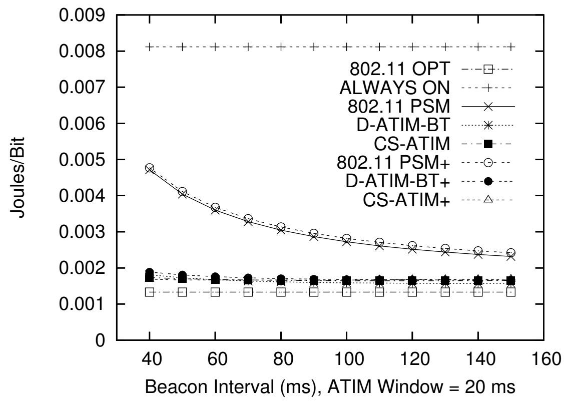

Figure 9: Energy vs. beacon interval using PLBIs.

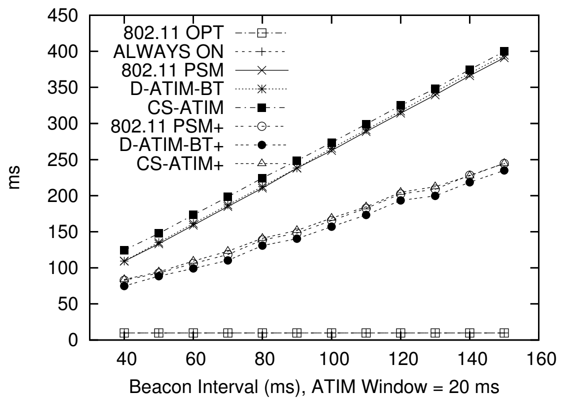

Figure 10: Latency vs. beacon interval using PLBIs.

As mentioned in Section 3.1, CS-ATIM can be adapted to handle synchronization errors caused by the local clocks being different for various nodes. In Figure 8, we show how the magnitude of the synchronization error can affect the energy of the power save protocols. Recall from Section 3.1 that Δ represents the maximum difference between *any* two local clocks in the network per beacon interval.

To test 802.11 PSM in this setting, we made a simple modification: at the beginning of each beacon interval (according to a node's local clock), a node will remain on for $2\Delta + T_{aw}$ time rather than the default T_{aw} time. For the first and last Δ of this interval, a node cannot transmit any packets, but can receive packets. This is similar to the protocol described in Section 3.1, but without the carrier sensing part.

We see that synchronization errors have a much smaller effect on CS-ATIM than on 802.11 PSM. The reason for this is that 802.11 PSM is affected by an increased ATIM window size *every* beacon interval. CS-ATIM, on the other hand, is only affected in beacon intervals where a node transmits a dummy packet. In this case, the CS-ATIM nodes have to transmit the dummy packet for a longer time and remain on longer for the ATIM that follows. This shows that CS-ATIM is more immune to the effects of synchronization errors than 802.11 PSM for lower traffic rates.

4.2 Evaluating Per-Link Beacon Intervals

In this section, we evaluate the addition of PLBIs to the power save protocols, as described in Section 3.3. From Figures 9 and 10, we see that PLBIs have little effect on the energy consumption of the protocols (i.e., the PLBI protocols nearly overlap with their base protocols), although PLBIs significantly improve latency.

We identify three reasons why PLBIs do not result in a significant decrease in energy consumption. First, the chosen data rate is so low that, many times, there are no packets to advertise at the beginning of an 802.11 PSM beacon interval. In this case, PLBIs save no extra energy over the base power save protocols. Therefore, the few times that PLBIs do save energy are less significant. Later in this section, we show that a higher sending rate improves the performance of PLBIs. Second, when PLBIs overlap with an 802.11 PSM beacon interval, they get postponed until the end of the ATIM window (see Section 3.3). If a node had a packet to send in a PLBI that was pushed back, then it may be advertised during the current ATIM window. In this case, PLBIs do not add any benefit. Finally, in CS-ATIM and D-ATIM-BT, the protocols are already so close to 802.11 OPT that it is difficult to save much more energy. For 802.11 PSM, PLBIs do not alleviate the ATIM window listening overhead and, therefore, have little effect on the protocol. However, in Figure 10, we see that PLBIs can make a significant improvement in latency. Without PLBIs, a packet that arrives while a sender or receiver is asleep must always wait until the next beacon interval to be sent. With PLBIs, the sender-receiver pair can be scheduled to wake up during their sleeping phase to send a packet in the current beacon interval. Thus, an advantage of PLBIs is that protocols have a lower latency while maintaining about the same energy consumption.

Since PLBIs do not show much improvement in terms of energy consumption for low traffic rates, we evaluated the effects of using a higher rate. We tested one 20 kbps flow being sent in the network (i.e., the flow uses 1% of the channel bitrate *per hop*). In Figure 11, we see that PLBIs lead to less energy consumption for CS-ATIM and D-ATIM-BT. CS-ATIM shows the biggest improvement when PLBIs are added with about a 28% decrease in energy consumption. For D-ATIM-BT, the gains are more modest, but PLBIs still lead to a 5 to 10% reduction in energy consumption. From Figure 12, we see that PLBIs show an even larger improvement in latency at the higher rate.

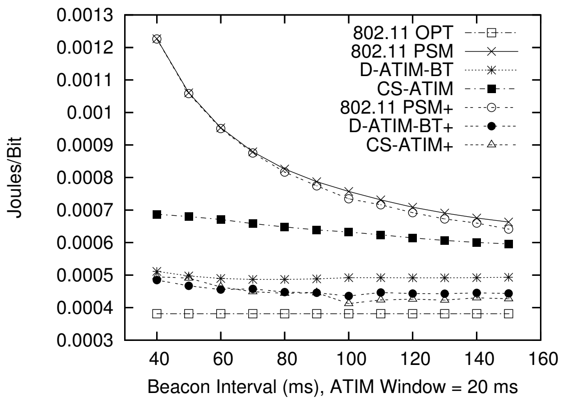

Figure 11: Energy vs. beacon interval using PLBIs with one 20 kbps flow.

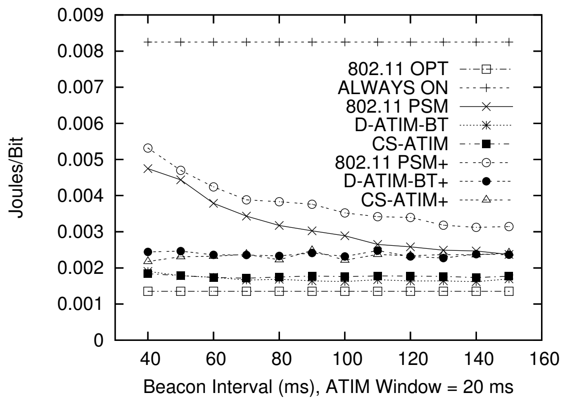

Figure 13: Energy vs. beacon interval using PLBIs with Poisson traffic.

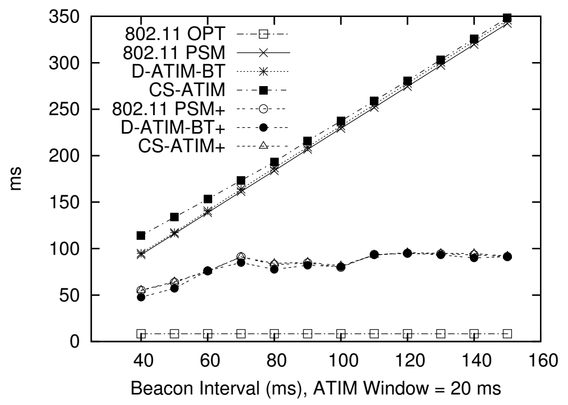

Figure 12: Latency vs. beacon interval using PLBIs with one 20 kbps flow.

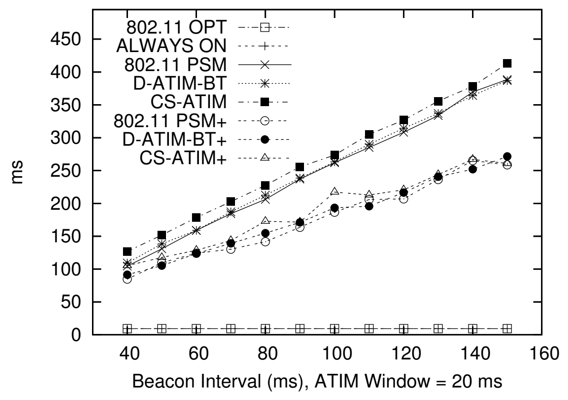

Figure 14: Latency vs. beacon interval using PLBIs with Poisson traffic.

4.2.1 Per-Link Beacon Intervals with Poisson Traffic

The above results for PLBIs with CBR traffic are the best case scenario since there is zero variance and, hence, the protocol can make very accurate predictions about future packet arrival times. In most other types of traffic, there is some variance in the interarrival times of packets. This causes the calculation in Equation 5 to be less accurate. Thus, nodes use more energy because they wake up more frequently due to the non-zero standard deviation term.

To test the effects of variance in the packet interarrival time, we simulated PLBIs with Poisson traffic (i.e., an exponential distribution for packet interarrival times). For an exponential distribution, the average packet interarrival time and the standard deviation of the packet interarrival time are equal [36]. Thus, according to Equation 5, BI_{cur} will occasionally be less than zero, putting a node pair in the ALWAYS ON state. In this state, the nodes will use more energy but have a lower delay.

In Figure 13, we see that PLBI scheduling uses more energy that the baseline protocol for 802.11 PSM, CS-ATIM, and D-ATIM-BT. This is because nodes are occasionally entering the ALWAYS ON state and waking up more frequently due to the variance of the packet interarrival time. Even longer beacon intervals do not help the PLBIs as much since nodes may still wake up frequently even with the longer time between 802.11 PSM beacon intervals.

However, in Figure 14, we see that PLBIs retain their positive effect on latency. As the 802.11 PSM beacon intervals get longer, the PLBI protocols show a slower increase in average packet latency because the nodes are waking up more frequently. This indicates that PLBIs are most useful with CBR traffic because they do not adversely affect energy consumption and improve the latency. CBR traffic may be used, for example, in an ad hoc network where the nodes periodically report sampled data back to a base station. However, when the traffic is more variant, PLBIs offer an energy-latency trade-off for the designer to control.

5. CONCLUSION AND FUTURE WORK

In this work, we have studied three techniques that can be used to improve power save protocols and focus on the 802.11 PSM as an example. Such work is important because wireless devices need more energy efficient protocols to improve battery life and to allow the devices to be untethered as long as possible. The major disadvantage of 802.11 PSM is the use of a static ATIM window which leads to a “one size fits all” approach regardless of traffic patterns. In practice, this approach is inefficient. If traffic is light, a large ATIM window wastes energy listening to the channel. If traffic is heavy, then a small ATIM window does not allow enough time to advertise all the packets that need to be sent.

To this end, we suggest two methods that allow the ATIM window length to be dynamic and waste less energy when traffic is light. In the first technique, CS-ATIM, nodes use a short carrier sensing period at the beginning of each beacon interval as a boolean indication of whether or not there are any packets to be advertised (and hence whether an ATIM window is necessary). When there are no packets to be advertised, CS-ATIM uses much less energy listening to the channel than 802.11 PSM.

In the second technique, D-ATIM, nodes dynamically extend their ATIM window as long as ATIMs and ATIM-ACKs continue to be sent. To avoid excessive ATIM window lengths when traffic is heavy, an upper bound is imposed on how long the ATIM window can be extended (e.g., not longer than the ATIM window of 802.11 PSM). When no ATIMs or ATIM-ACKs have been overheard for a sufficiently long time, a node can either return to sleep or begin sending/receiving data, depending on whether it sent or received any ATIMs. D-ATIM improves 802.11 PSM by maintaining small ATIM windows even when there are few or no packets to send, while still allowing larger ATIM windows when traffic is heavy. Thus, D-ATIM does not use more energy than 802.11 PSM and usually consumes significantly less. Additionally, D-ATIM can improve the packet latency, in some cases, over 802.11 PSM by starting to send data packets earlier than 802.11 PSM.

The third technique we introduce is intended to augment CS-ATIM and D-ATIM. By using Per-Link Beacon Intervals (PLBIs), a sender and receiver can schedule their wake-up times separate from other nodes in the network based on their past packet arrival history. Simulations show that adding such a technique to the power save protocols can be used to improve the average per packet latency while maintaining low energy consumption.

Future work will explore how these protocols can be integrated with energy efficient routing and transport protocols. Also, we will investigate if the protocols can be more efficient by dynamically adapting more parameters. For example, the BI_{idle} time in PLBIs can be dynamic based on the traffic variance and the average time channel access time for a sender. Additionally, we plan to study the feasibility of adopting D-ATIM and PLBIs to environments with less stringent synchronization requirements. Finally, we will consider if D-ATIM can be modified for multi-hop networks without using an extra busy-tone channel.

6. REFERENCES

[1] B. Wang and S. Singh, "Computational Energy Cost of TCP," in *IEEE Infocom 2004*, March 2004.

[2] V. Raghunathan, *et al.*, "Energy-Aware Wireless Microsensor Networks," *IEEE Signal Processing Magazine*, March 2002.

[3] A. J. Goldsmith and S. B. Wicker, "Design Challenges for Energy-Constrained Ad Hoc Wireless Networks," *IEEE Wireless Communications*, pp. 8–27, August 2002.

[4] A. Ephremides, "Energy Concerns in Wireless Networks," *IEEE Wireless Communications*, August 2002.

[5] B. Chen, *et al.*, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in *ACM MobiCom 2001*, July 2001.

[6] MICA2 Mote Datasheet, `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-05_A_MICA2.pdf`.

[7] IEEE 802.11, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.

[8] H. Woesner, J.-P. Ebert, M. Schläger, and A. Wolisz, "Power-Saving Mechanisms in Emerging Standards for Wireless LANs: The MAC Level Perspective," *IEEE Personal Communications*, pp. 40–48, June 1998.

[9] E.-S. Jung and N. H. Vaidya, "An Energy Efficient MAC Protocol for Wireless LANs," in *IEEE Infocom 2002*, June 2002.

[10] ——, "Improving IEEE 802.11 Power Saving Mechanism," 2004, in submission.

[11] I. Aad and C. Castelluccia, "Differentiation mechanisms for IEEE 802.11," in *IEEE Infocom 2001*, April 2001.

[12] M. Barry, A. T. Campbell, and A. Veres, "Distributed Control Algorithms for Service Differentiation in Wireless Packet Networks," in *IEEE Infocom 2001*, April 2001.

[13] J.-M. Choi, Y.-B. Ko, and J.-H. Kim, "Enhanced Power Saving Scheme for IEEE 802.11 DCF based Wireless Networks," in *IFIP Personal Wireless Communication (PWC) 2003*, September 2003.

[14] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *ACM SenSys 2004*, November 2004.

[15] A. El-Hoiydi, "Aloha with Preamble Sampling for Sporadic Traffic in Ad How Wireless Sensor Networks," in *IEEE International Conference on Communications (ICC) 2002*, April 2002.

[16] R. Zheng and R. Kravets, "On-demand Power Management for Ad Hoc Networks," in *IEEE Infocom 2003*, April 2003.

[17] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *ACM SenSys 2003*, November 2003.

[18] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *IEEE Infocom 2002*, June 2002.

[19] C. Hu and J. Hou, "LISP: A Link-Indexed Statistical Traffic Prediction Approach to Improving IEEE 802.11 PSM," in *IEEE International Conference on Distributed Systems (ICDCS) 2004*, March 2004.

[20] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed Energy Conservation for Ad Hoc Routing," in *ACM MobiCom 2001*, July 2001.

[21] A. Cerpa and D. Estrin, "ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies," in *IEEE Infocom 2002*, June 2002.

[22] C. Schurgers, *et al.*, "Optimizing Sensor Networks in the Energy-Latency-Density Design Space," *IEEE Transactions on Mobile Computing*, vol. 1, no. 1, pp. 70–80, January-March 2002.

[23] C. Guo, L. C. Zhong, and J. M. Rabaey, "Low Power Distributed MAC for Ad Hoc Sensor Radio Networks," in *IEEE GlobeCom 2001*, November 2001.

[24] R. Zheng, J. C. Hou, and L. Sha, “Asynchronous Wakeup for Ad Hoc Networks,” in *ACM MobiHoc 2003*, June 2003.

[25] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, “Power-Saving Protocols for IEEE 802.11-Based Multi-Hop Ad Hoc Networks,” in *IEEE Infocom 2002*, June 2002.

[26] O. Dousse, P. Mannersalo, and P. Thiran, “Latency of Wireless Sensor Networks with Uncoordinated Power Saving Mechanisms,” in *ACM MobiHoc 2004*, May 2004.

[27] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, “Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks,” in *ACM SenSys 2003*, November 2003.

[28] M. L. Sichitiu, “Cross-Layer Scheduling for Power Efficiency in Wireless Sensor Networks,” in *IEEE Infocom 2004*, March 2004.

[29] Chip-Scale Atomic Clocks, `http://www.boulder.nist.gov/timefreq/ofm/smallclock/`.

[30] `ns-2` – The Network Simulator, `http://www.isi.edu/nsnam/ns`.

[31] X. Yang and N. H. Vaidya, “Priority Scheduling in Wireless Ad Hoc Networks,” in *ACM MobiHoc 2002*, June 2002.

[32] J. Deng and Z. J. Haas, “Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks,” in *IEEE ICUPC 1998*, October 1998.

[33] M. J. Miller and N. H. Vaidya, “Minimizing Energy Consumption in Sensor Networks Using a Wakeup Radio,” in *IEEE WCNC 2004*, March 2004.

[34] J. F. MacGregor and T. J. Harris, “The Exponentially Weighted Moving Variance,” *Journal of Quality Technology*, vol. 25, no. 2, pp. 106–118, April 1993.

[35] T. H. Cormen, C. E. Leiserson, , R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[36] E. W. Weisstein, “MathWorld,” `http://mathworld.wolfram.com`.

APPENDIX

A. CS-ATIM MODIFICATIONS FOR SYNCHRONIZATION ERRORS

In this section, we show the correctness of the modifications to CS-ATIM discussed in Section 3.1. As mentioned previously, we assume that the node's clocks are always within Δ seconds of each other. Thus, Δ represents the maximum error between the clocks of *any* two nodes in the network. The modifications, shown in Figure 15, are as follows. Without loss of generality, we assume that a node with the fastest clock in the network begins the current beacon interval at time T_{f0}. Thus, the latest a node's beacon interval can begin is:

$$T_{s0} = T_{f0} + \Delta \tag{6}$$

To account for Δ, nodes which have no packets to advertise must wake up Δ seconds after the beacon interval is scheduled according to their local clock. Thus, a node with

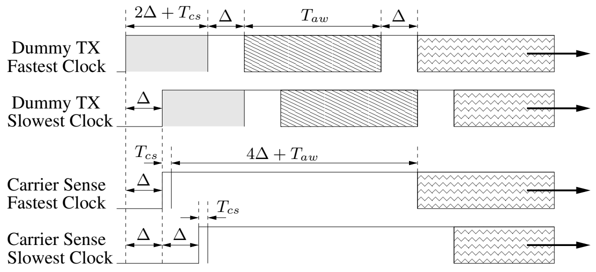

Figure 15: CS-ATIM time synchronization. The shaded area denotes a dummy packet being sent. The slanted lines represent the ATIM window when ATIM packets can be transmitted. The wavy lines denote when data packets can be transmitted.

the fastest clock carrier senses the channel from time:

$$T_{f1} = T_{f0} + \Delta \tag{7}$$

until:

$$\begin{aligned} T_{f2} &= T_{f1} + T_{cs} \\ &= T_{f0} + \Delta + T_{cs} \end{aligned} \tag{8}$$

A node with the slowest clock carrier senses the channel from time:

$$\begin{aligned} T_{s1} &= T_{s0} + \Delta \\ &= T_{f0} + 2\Delta \end{aligned} \tag{9}$$

until

$$\begin{aligned} T_{s2} &= T_{s1} + T_{cs} \\ &= T_{f0} + 2\Delta + T_{cs} \end{aligned} \tag{10}$$

For a node that has packets to advertise, it begins transmitting the dummy packet when the beacon interval begins according to its local clock and transmits the dummy packet for $2\Delta + T_{cs}$ time. Thus, a node with the fastest clock transmits its dummy packet from time:

$$T_{f3} = T_{f0} \tag{11}$$

until:

$$T_{f4} = T_{f0} + 2\Delta + T_{cs} \tag{12}$$

Since $T_{f3} < T_{f1} < T_{f2} < T_{f4}$ and $T_{f3} < T_{s1} < T_{s2} = T_{f4}$, nodes with the fastest clock and nodes with the slowest clock are both guaranteed to carrier sense this dummy packet for the specified T_{cs} length of time. A node with packets to advertise with the slowest clock will transmit its dummy packet from time:

$$\begin{aligned} T_{s3} &= T_{s0} \\ &= T_{f0} + \Delta \end{aligned} \tag{13}$$

until:

$$\begin{aligned} T_{s4} &= T_{s0} + 2\Delta + T_{cs} \\ &= T_{f0} + 3\Delta + T_{cs} \end{aligned} \tag{14}$$

Since $T_{s3} = T_{f1} < T_{f2} < T_{s4}$ and $T_{s3} < T_{s1} < T_{s2} < T_{s4}$, nodes with both the fastest and slowest clocks will carrier sense this dummy packet for the specified time, T_{cs}.

If a node without packets to advertise detects the channel idle at the end of the T_{cs} time, it will return to sleep. However, if the node detects the channel as busy, it will remain on for an additional $4\Delta + T_{aw}$ time, as show in Figure 15, for reasons explained below. For a node that *does* have a packet to advertise, it will begin sending ATIM packets Δ seconds after it finishes transmitting the dummy packet. During this Δ time gap between the end of the dummy packet and the beginning of the ATIM window, the node *may* receive packets and reply with ACKs, however, it *may not* send any ATIMs or data packets during this time. At the end of the ATIM window T_{aw} seconds later, the node waits another Δ seconds before it starts sending data packets. Again, during the Δ time gap, the node may receive and reply with ACKs, but may not send ATIMs or data packets. For a node with the fastest clock, it is guaranteed to be on from time T_{f4} (the time it finished transmitting the dummy packet) until:

$$\begin{aligned} T_{f5} &= T_{f4} + 2\Delta + T_{aw} \\ &= T_{f0} + 4\Delta + T_{cs} + T_{aw} \end{aligned} \tag{15}$$

A node with the fastest clock is allowed to transmit during its ATIM window which starts at time:

$$\begin{aligned} T_{f6} &= T_{f4} + \Delta \\ &= T_{f0} + 3\Delta + T_{cs} \end{aligned} \tag{16}$$

and ends at time:

$$\begin{aligned} T_{f7} &= T_{f6} + T_{aw} \\ &= T_{f0} + 3\Delta + T_{cs} + T_{aw} \end{aligned} \tag{17}$$

For a node with the slowest clock, it is guaranteed to be on from time T_{s4} until:

$$\begin{aligned} T_{s5} &= T_{s4} + 2\Delta + T_{aw} \\ &= T_{f0} + 5\Delta + T_{cs} + T_{aw} \end{aligned} \tag{18}$$

A node with the slowest clock is allowed to transmit during its ATIM window which starts at time:

$$\begin{aligned} T_{s6} &= T_{s4} + \Delta \\ &= T_{f0} + 4\Delta + T_{cs} \end{aligned} \tag{19}$$

and ends at time:

$$\begin{aligned} T_{s7} &= T_{s6} + T_{aw} \\ &= T_{f0} + 4\Delta + T_{cs} + T_{aw} \end{aligned} \tag{20}$$

Because $T_{f4} < T_{s6} < T_{s7} = T_{f5}$, a node with the fastest clock is guaranteed to be listening during the entire ATIM window of a node with the slowest clock. Similarly, because $T_{s4} = T_{f6} < T_{f7} < T_{s5}$, a node with the slowest clock is guaranteed to be listening after its dummy packet transmission during the entire ATIM window of a node with the fastest clock.