Improving Power Save Protocols Using Carrier Sensing for Dynamic Advertisement Windows

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

Nitin H. Vaidya
Department 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. Packets are advertised for a fixed length of time, known as an advertisement window, at epochs known as beacon intervals. However, the protocol needlessly wastes energy when traffic is relatively light in a network. In this paper, we address this problem by proposing the use of carrier sensing to dynamically adjust the size of the advertisement windows. The adjustment is based on the amount of traffic that needs to be advertised in the current window as opposed to the static window size used by 802.11 PSM. Carrier sensing is used for two different aspects of our protocol. First, carrier sensing is used as an energy efficient method to provide a binary signal which lets neighbors know if a node intends to advertise any packets in the upcoming window. Second, carrier sensing is used as a mechanism for nodes to keep track of whether their neighbors have already stopped listening for advertisements and possibly returned to sleep. Using the *ns-2* simulator we show that our techniques can significantly reduce the energy consumption of 802.11 PSM while only slightly increasing latency.**

I. 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 (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.

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

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 III 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 increase in the average packet latency.

The major contributions of this work are:

- We show how carrier sensing can be used 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.
- We dynamically re-size the ATIM window based on the number of advertisements to be sent in the current window. While we have explored dynamic adjustment of the ATIM window previously [8], [9], this is the first work of which we are aware that achieves this in a multi-hop environment using a single channel.

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

II. Related Work

Power save protocols take a variety of forms. Our primary focus is on IEEE 802.11 IBSS PSM. We start by describing 802.11 PSM [7]. Nodes are assumed to be synchronized 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 that have been queued since the previous beacon interval are advertised. These advertisements take the form of ATIM packets. More formally, when a node has a packet to advertise,

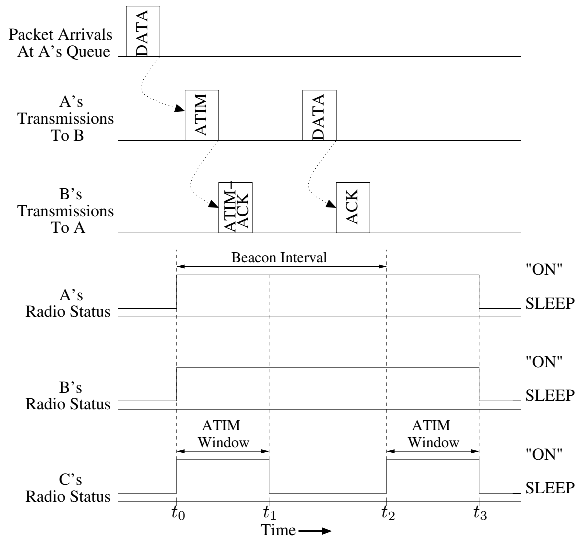

Fig. 1. IEEE 802.11 IBSS power save mode [7].

it sends an ATIM packet to the intended receiver during the ATIM window (following 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 [7]. 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 [10], 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 [8] 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, this protocol adjusts 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 [11] is similar to our work in that the ATIM window ends when the channel is idle for a specified amount of time. However, IPSM only works in single-hop networks since it relies on a node and all its neighbors having a consistent view of channel activity.

In TIPS [12], 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. Additionally, TIPS only uses static ATIM window sizes whereas our work allows dynamic adjustment of the window.

The idea of preamble sampling has been used with B-MAC [13]. 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. WiseMAC [14] improves on B-MAC by having nodes store the next sampling time of a node with which it is sending packets. Thus, after accounting for the maximum clock drift since the last packet was sent, a node can usually transmit a much shorter preamble than is required by B-MAC and, therefore, greatly improves energy consumption. The advantage of these techniques is that they work in completely unsynchronized environments. However, when there are several senders transmitting to a receiver (e.g., in sensor networks when aggregation is performed or in ad hoc routing when a node's route replies attract multiple senders), then WiseMAC significantly increases latency beyond 802.11 PSM and our proposed protocols. Also, when broadcast packets are sent, WiseMAC must resort to B-MAC and, hence, significantly increases energy consumption.

Other power save techniques include predictively listening in response to previously received data or ATIM packets [15]–[17]. Another common strategy is for nodes to remain awake based on their local topology and/or traffic [5], [18], [19]. Other protocols use an out-of-band channel (e.g., a second, non-interfering radio) to wake up sleeping neighbors [20], [21]. Other work focuses on nodes communicating despite uncoordinated sleep schedules [22]–[24]. Finally, some protocols have used TDMA approaches [25], [26] where communicating nodes attempt to schedule non-interfering time slots to wake up and transmit or receive data packets.

III. Protocol Description

From the description of 802.11 PSM in Section II, we can see that the ATIM window wastes a significant amount of energy when the traffic load is low. For example, in previous work [8], [17] some typical values for the ATIM window and beacon interval are 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, our techniques reduce the overhead of the ATIM window when traffic is sparse and provide larger ATIM windows when there is more data to advertise.

In Section III-A, we use a short carrier sensing period preceding the ATIM window where nodes can indicate whether or not they intend to advertise any data. Thus, when none of a node's neighbors are going to advertise any data, the node can return to sleep without remaining on for the ATIM window. In Section III-B, we further improve the energy consumption of the protocol by allowing nodes that participate in the ATIM window to dynamically adjust the size of their ATIM window. By using this technique, nodes that do not receive any ATIMs can usually return to sleep sooner than if a static ATIM window size is used.

We make the assumption that the nodes in the network are time synchronized by some out-of-band means. For example, the nodes may be GPS-equipped. In a tech report [9], we discuss modifications to the protocols to handle some synchronization error. For brevity, we omit the discussion here.

As a result of our assumption, the timing synchronization function (TSF) of 802.11 is disabled and beacons are never sent. For consistency with the terminology in related work, we will still refer to the time between ATIM windows as a "beacon interval" even though no beacons are sent.

A. Carrier Sensing Preceding the ATIM Window

From the description of 802.11 PSM in Section II, 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 binary 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 in their neighborhood. 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 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.

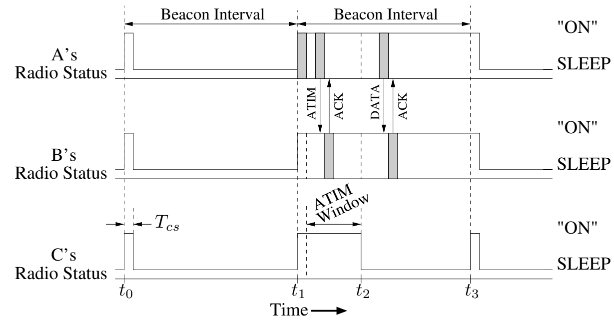

Fig. 2. CS-ATIM protocol.

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 while avoiding 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 only 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, runs the risk that nodes may erroneously carrier sense energy that is due to interference in the frequency band rather than the dummy packet transmission. 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 IV, we test the effects of false positives on CS-ATIM.

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.

B. 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 in a 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.[1] This is what past work attempts to achieve either through heuristics [8] or dynamically extending the window when packets are received [9], [11]. Unlike the previous work that dynamically extends the ATIM window based on packet reception, our goal is to have a protocol that works in multi-hop environments and does not use a second channel (e.g., a busy-tone channel). We refer to this extension of CS-ATIM as *Dynamic* CS-ATIM (DCS-ATIM).

First, we distinguish between two types of packet reception in IEEE 802.11. When a packet is received at a power level above the RX_THRESHOLD, we say that the receiver is within the *transmission range* of the sender. When a packet is received at a power level below the RX_THRESHOLD, but above the CS_THRESHOLD (carrier sense threshold), the receiver is said to be within the *carrier sensing range* of the sender. Packets received by nodes in the carrier sensing range cannot be decoded, but do cause the node's clear channel assessment to classify the channel as busy. We assume that, most of the time, a node's carrier sensing range is at least twice as big as its transmission range [27]. Thus, when S sends a packet and R is within the transmission range of S, the nodes within the transmission range of R are likely to be within the carrier sensing range of S.

We note that there are several cases in which a node may receive a packet above the RX_THRESHOLD, but its neighbors do not receive the packet above the CS_THRESHOLD. This may occur due to short-term fading or obstructions in the line-of-sight of a node pair. While DCS-ATIM can recover from such occurrences, we assume such events are rare.[2] In the worst case, when a node detects little or no correlation between its packet receptions and a neighbor's carrier sensing of these packets, then the node can fall back to CS-ATIM to advertise packets to that neighbor.

In DCS-ATIM, there are *two* carrier sensing periods that follow the beginning of the beacon interval:

- **CS**$_1$: As in CS-ATIM, DCS-ATIM begins with a carrier sensing period of length T_{cs} during which time nodes use the protocol from Section III-A to indicate whether they have packets to advertise. We refer to this carrier sensing interval as **CS**$_1$.
- **CS**$_2$: DCS-ATIM adds a second carrier sensing period, **CS**$_2$, (of duration T_{cs}) that immediately follows **CS**$_1$. If a node wants its neighbors to use a static ATIM window, as in CS-ATIM, then it transmits a dummy packet during **CS**$_2$. Otherwise, its neighbors use the dynamic window scheme described below. For example, a node may use a static ATIM window if it has not been able to advertise a packet for the past k intervals. This is a fail-safe mechanism when a packet is unable to be advertised after attempting for several dynamic windows.

We now describe the protocol after the above two carrier sensing periods when nodes have decided to use dynamic ATIM windows. First, we give ATIM packets a different maximum contention window size (CW_{aw}) than data packets (CW_{max}). In the IEEE 802.11 specification [7] for direct-sequence spread spectrum (DSSS), the default CW_{max} 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 $CW_{aw} < CW_{max}$ 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_{max}.

At the start of the dynamic ATIM window, every node listens to the channel and sets a timer to expire after:

$$\begin{aligned} T_{idle} = {} & DIFS + T_{slot} \cdot CW_{aw} + prop_{max} \\ & + T_{atim} + SIFS + prop_{max} + T_{ack} \\ & + DIFS + T_{slot} \cdot CW_{aw} + prop_{max} \end{aligned} \tag{1}$$

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

[2]Currently, we do not test these situations in our simulations.

where $DIFS$ and $SIFS$ are the DCF and Short Interframe Space as specified by IEEE 802.11 [7], respectively. The values T_{atim} and T_{ack} are the time durations required to send an ATIM and ATIM-ACK, respectively.[3] The maximum propagation delay between two nodes is denoted as $prop_{max}$. T_{idle} is designed to be long enough to give a node the chance to access the channel after it was in the carrier sensing, but not transmission range, of an ATIM/ATIM-ACK handshake.

If a node sends or carrier senses a packet before the timer expires, the timer is reset to end T_{idle} time after the packet is sent or carrier sensed. To avoid starvation, an upper limit is set on the size that 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.

A node may transmit ATIM packets as long as it has sent a packet or received a packet above the RX_THRESHOLD within the past T_{idle} time. When one of these two conditions is met, it implies that the node's neighbors have either received or carrier sensed a packet within the past T_{idle} interval and, hence, refreshed their timers to continue listening for ATIM packets. If a node has carrier sensed a packet within the past T_{idle} time, but not sent or received a packet during that time, then it must continue to listen for ATIMs until its timer expires, but it cannot send anymore ATIMs until the next beacon interval. If a node is unable to send an ATIM for k consecutive intervals, it uses $\mathbf{CS}_1$ to let its neighbors know to resort to a static ATIM window size.

Whenever a node does not send or carrier sense a data packet for T_{idle} time or the upper bound on the dynamic ATIM window is reached, the node ends the ATIM phase and waits for the data phase to begin. 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. The data phase begins T_{aw} after the start of the ATIM window. It is postponed until this time to avoid sending potentially long data packets while other neighbors are trying to transmit ATIMs.

An example of DCS-ATIM compared to 802.11 PSM is given in Figure 3. First, there is the additional carrier sensing at the beginning of DCS-ATIM. Because **A** has a packet to advertise, it sends a dummy packet at the start of the beacon interval. In this example, **A** desires a dynamic ATIM window, so no dummy packet is sent during the second carrier sensing period. After both carrier sensing periods have ended, **A** sends an ATIM to **B**. In this example, **C** does not carrier sense anymore transmissions after **B**'s ATIM-ACK. Thus, with DCS-ATIM, **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. With 802.11 PSM, **C** must remain on for the entire T_{aw} time of the static ATIM window.

From this description, we see that, in the worst case, the ATIM window for DCS-ATIM only uses slightly more energy in the ATIM window than 802.11 PSM (for the carrier sensing periods) and may use much less energy when a small number of ATIMs are sent. In terms of latency, DCS-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 DCS-ATIM's dynamic ATIM window has already ended, the node may wait until the next beacon interval to advertise the packet. Additionally, DCS-ATIM may not be able to advertise as many packets as 802.11 PSM if a node with a packet to advertise does not send or receive any packets above the RX_THRESHOLD as discussed above. In this case, the node will wait until the next ATIM window to advertise the packet.

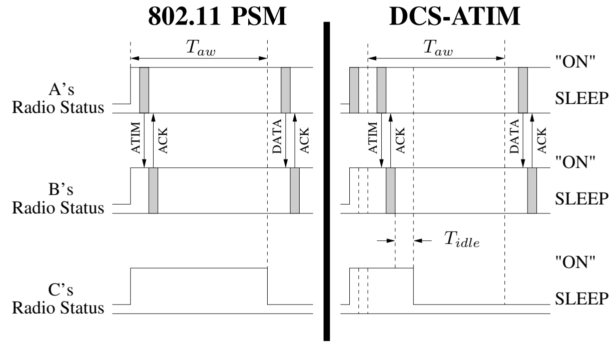

Fig. 3. 802.11 PSM vs. DCS-ATIM.

IV. Simulation Results

To test our protocols, we simulated them by modifying the MAC and physical layers of *ns-2* [28]. 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 II.
- **CS-ATIM:** This is 802.11 PSM with the carrier sensing modification described in Section III-A.
- **DCS-ATIM:** This is 802.11 PSM with the dynamic ATIM modification for multi-hop networks described in Section III-B.
- **802.11 MIN:** This protocol needs more explanation because we are unaware of any other work which uses it. 802.11 MIN represents the minimum 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 MIN is

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

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 MIN's energy, a node consumes P_{tx} power while sending a packet, P_{rx} power while overhearing a packet, 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 MIN represents the lowest possible energy achievable for nodes *using IEEE 802.11* if they slept as aggressively as possible (i.e., a node sleeps whenever they are not sending a packet, overhearing a packet from a transmitting neighbor, or attempting to access the channel). 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).

We use 2 Mbps radios that have a 250 m range. Each data point is averaged over 30 tests. In Section IV-A, we only present results from multi-hop scenarios for brevity. We tested the protocols in a WLAN, single-hop environment as well and the trends are similar.

The choice of a different channel bitrate would have the following effects on the protocols. For CS-ATIM, the carrier sensing period is rate-independent [29]. Thus, if T_{aw} is normalized to the packet transmission time, the carrier sensing time will be a higher overhead at a high bitrate and a lower overhead at a low bitrate. For DCS-ATIM, T_{idle} from Equation 1 will be larger for a lower bitrate and smaller for a higher bitrate since it is a function of how long it takes to transmit ATIM and ATIM-ACK packets.

For each multi-hop 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 [30] to precompute routes for all the nodes.

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 test the effects of increasing the per-flow rate. 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 DCS-ATIM, the maximum backoff interval size, CW_{aw}, is set to be 63 slots. For the parameters we use, T_{idle} is set to 3.19 ms according to Equation 1. For the power characteristics of the radio [5], [15], we use: $P_{tx} = 1.4$ W, $P_{rx} = 1.0$ W, $P_{listen} = 0.83$ W, and $P_{sleep} = 0.13$ W.

As mentioned earlier, CS-ATIM and DCS-ATIM are vulnerable to false positives when they erroneously carrier sense the presence of a signal. Thus, in some of our tests we evaluate the effect of false positives on the protocols 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 10% chance of false probabilities means that with probability 0.1, a node running the protocols remains on for the ATIM window even though there were no dummy packets 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.
- **Per-Flow Rate:** We increase the rate at which each of the flows in the network is sending packets.
- **False Positives:** For our protocols, we show how false positives (i.e., erroneously detecting the channel as busy) affect the energy consumption.

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.

A. Evaluating CS-ATIM and DCS-ATIM

First we tested the power consumption and latency of CS-ATIM and DCS-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 sleep time between ATIM windows. We see that CS-ATIM and DCS-ATIM both perform significantly better than 802.11 PSM. CS-ATIM and DCS-ATIM 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 protocols 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 MIN, CS-ATIM and DCS-ATIM use only about 18 to 30% more energy.

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 MIN by definition, always do significantly better than the power save protocols.

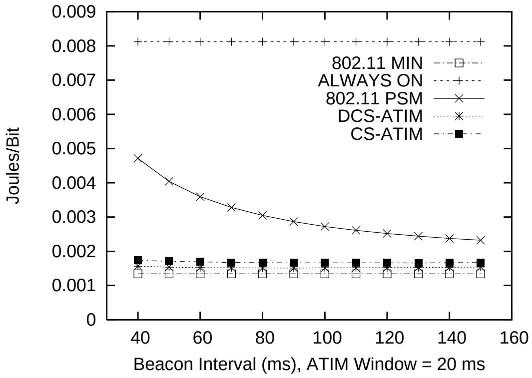

Fig. 4. Energy vs. beacon interval.

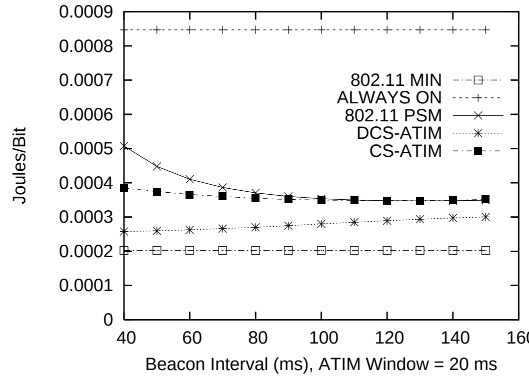

Fig. 6. Energy vs. beacon interval with10 kbps flows.

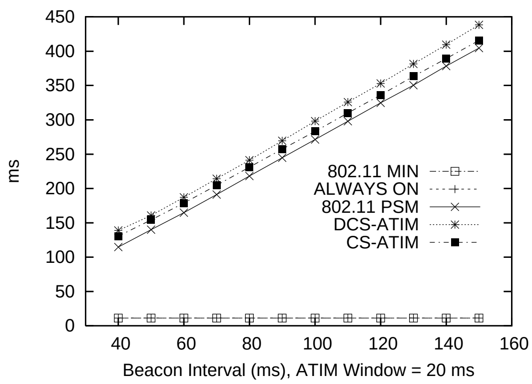

Fig. 5. Latency vs. beacon interval.

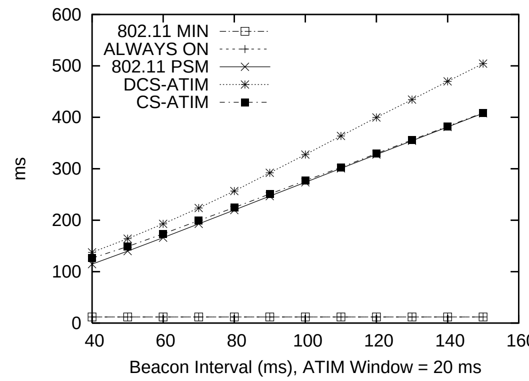

Fig. 7. Latency vs. beacon interval with10 kbps flows.

For the power save protocols, 802.11 PSM always has the lowest latency of the power save protocols. CS-ATIM, however, tends have a slightly higher latency. The difference between CS-ATIM's latency and the latency of 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. DCS-ATIM has a slightly larger latency than CS-ATIM because of the extra carrier sensing period as well as the fact that sender's may occasionally have to postpone their advertisement until a later ATIM window.

In Figure 6 and Figure 7 we show how an increased sending rate affects the protocols. In these tests, the sending rate of each of the five flows is increased from 1 kbps to 10 kbps (i.e., each flow uses about 0.5% of the channel bitrate *per hop*). We see that DCS-ATIM does even better relative to CS-ATIM in this setting since a larger fraction of the ATIM windows have at least one advertisement to be sent. In this case, CS-ATIM has the same energy consumption as 802.11 PSM. However, DCS-ATIM can do better by allowing nodes to return to sleep earlier when only one advertisement is sent.

However, as Figure 7 shows, this improved relative energy consumption comes at the cost of increased latency. As the beacon intervals get longer with the higher sending rate, there is more contention during the ATIM window and, hence, a greater chance that a node with an ATIM to send will have to delay the transmission until a later ATIM window, which significantly increases the delay of that packet.

This increased contention and advertisement delay also explains the gradual increase in energy for DCS-ATIM seen in Figure 6. We set DCS-ATIM to resort to CS-ATIM when a packet cannot be advertised for three consecutive ATIM windows. Thus, as DCS-ATIM uses more static ATIM windows, its energy consumption approaches that of CS-ATIM.

1) False Positives: As mentioned earlier, our protocols are susceptible to false positives when nodes carrier sense the channel as busy even though 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 8, we see that CS-ATIM and DCS-ATIM show 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 our protocols converges to slightly more than that of 802.11 PSM since they still have the overhead of carrier sensing.

V. Conclusion and Future Work

In this work, we have studied 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.

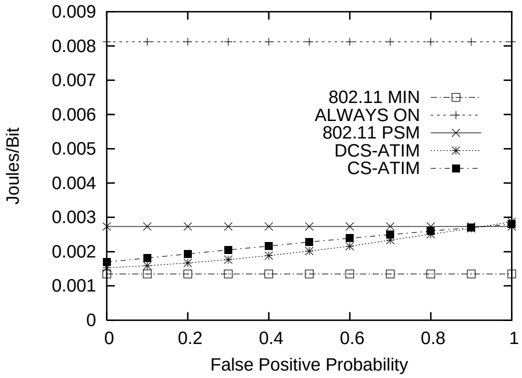

Fig. 8. Energy vs. false positiveprobability.

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 use carrier sensing to 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 binary 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, DCS-ATIM, which extends CS-ATIM, nodes dynamically extend their ATIM window as long as ATIMs and ATIM-ACKs continue to be sent and their neighbors remain on. 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 packets have been carrier sensed for a sufficiently long time, a node can either return to sleep or wait for the data phase to begin. DCS-ATIM improves 802.11 PSM and CS-ATIM 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, DCS-ATIM uses about the same amount of energy as 802.11 PSM in the worst case and usually consumes significantly less.

Future work will explore how these protocols can be integrated with energy efficient routing and transport protocols. Also, we will investigate the feasibility of adapting the protocols to have less stringent synchronization requirements.

References

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

[2] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "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, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in *ACM MobiCom 2001*, July 2001.

[6] Crossbow Technology Inc., `http://www.xbow.com`.

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

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

[9] M. J. Miller and N. H. Vaidya, "Improving Power Save Protocols Using Carrier Sensing and Busy-Tones for Dynamic Advertisement Windows," University of Illinois at Urbana-Champaign, Tech. Rep., 2004.

[10] 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.

[11] E.-S. Jung and N. H. Vaidya, "Improving IEEE 802.11 Power Saving Mechanism," 2004, in submission.

[12] 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.

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

[14] A. El-Hoiydi and J.-D. Decotignie, "WiseMAC: An Ultra Low Power MAC Protocol for Multi-hop Wireless Sensor Networks," in *Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS) 2004*, July 2004.

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

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

[17] 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 Computing Systems (ICDCS) 2004*, March 2004.

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

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

[20] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "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.

[21] 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.

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

[23] 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.

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

[25] 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.

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

[27] E.-S. Jung and N. H. Vaidya, "A Power Saving MAC Protocol for Wireless Networks," University of Illinois at Urbana-Champaign, Tech. Rep., 2002.

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

[29] X. Yang and N. Vaidya, "On Physical Carrier Sensing in Wireless Ad Hoc Networks," in *IEEE Infocom 2005*, March 2005.

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