Know Your Neighborhood: A Strategy for Energy-efficient Communication

Farhana Ashraf, Riccardo Crepaldi and Robin H. Kravets
University of Illinois at Urbana-Champaign
Email: {fashraf2, rcrepal2, rhk}@illinois.edu

Abstract— **Wireless sensor networks typically conserve energy by following a periodic wakeup-sleep schedule: nodes minimize idle time and spend most of their time in a low power sleep state. In order to communicate and exchange data in such a network, the current duty-cycling MAC protocols either require tight synchronization between the neighbor wakeup schedules or spend a significant amount of energy in signaling the sleeping nodes. In contrast, this paper presents Neighborhood-based Power Management (NPM), an energy efficient asynchronous MAC protocol that minimizes signaling overhead through opportunistically gained knowledge about neighbor wakeup schedules. Unlike the synchronization-based MAC protocols, NPM does not require a priori knowledge of the wakeup schedules. Using only a minimal exchange of schedule information, NPM reduces the signaling overhead by shortening the wakeup signal. Furthermore, NPM uses its wakeup signal to awaken all receivers in the neighborhood of the sender, enabling all sender-receiver pairs in that neighborhood to communicate. Our extensive evaluations show that NPM outperforms popular B-MAC [1], X-MAC [2] and SCP [3] protocols under all network conditions. NPM reduces the signaling overhead, with up to 74% savings in energy and 80% reduction in delay.**

I. Introduction

Duty cycling is commonly used in sensor networks to reduce the energy consumed during idle period in communication. By following a periodic wakeup schedule, nodes spend less time listening to an idle channel and more time in a low-power sleep state [4]. Since both the sender and the receiver must be awake to communicate, the challenge lies in coordinating their awake schedules. In an ideal scenario, a sender knows when a receiver is awake through a priori knowledge of or synchronized wakeup schedules. However, this approach requires tight clock synchronization, which is difficult and expensive to maintain in large sensor deployments [5]. While orthogonal control protocols can be added for maintaining synchronization, these protocols impose high control overhead and so increase energy consumption [3], [6], [7]. In some networks, this increase may even dominate the energy saved by following the synchronized periodic wakeup schedule [6], [7]. In response, many new protocols have been proposed [1]–[3], [8]–[13] that eliminate the need for such synchronization overhead by allowing nodes to have asynchronous wakeup schedules.

In networks with asynchronous wakeup schedules, nodes periodically wake up and listen to see if any neighbor has data for them. The challenge now stems from the fact that a sender has no knowledge of a receiver's schedule and so does not know when to send a signal to the the receiver. To enable such communication without out of band synchronization mechanisms, there must be some guaranteed overlap between a sender's signal and a receiver's wakeup period. While some protocols propose extending the receiver wakeup periods [8]–[13], recent work shows that extending a sender's signal instead greatly improves energy efficiency [1], [2]. However, the extended signal now results in increased delay and reduced energy savings in networks with moderate traffic load [4].

The main challenge behind designing a duty-cycling MAC protocol comes from the need to balance synchronization costs with signaling costs. Current MAC protocols represent two extremes. Protocols such as IEEE 802.11 PSM [14], S-MAC, T-MAC [7] and DW-MAC [15] have very low signaling overhead but high synchronization overhead. On the other hand, asynchronous protocols such as B-MAC [1] and X-MAC [2] mostly incur signaling overhead and have little or no synchronization overhead. While the synchronization-based approaches are more appropriate for networks with high traffic rates, the signaling-based approaches are more suitable for networks with very low traffic rates. Since sensor networks are expected to manage dynamic load demands, neither approach provides a complete and effective solution. Our goal is to find a balance between signaling and synchronization and use these insights to design a new energy-efficient MAC protocol.

Our research is based on two observations. First, high synchronization overhead stems from the periodic exchange of schedule information, even when there is no data to transmit. While knowledge of sleep schedules can greatly reduce energy consumption, such knowledge can be obtained during active communication through piggy-backed timing information. Use of such *opportunistic scheduling* can achieve much of the benefits of full knowledge of sleep schedules. Second, the main reason for high signaling costs is the inefficient use of the signal. Although a signal wakes up all nodes that can hear it, only one sender-receiver node pair are allowed to exchange data. All other nodes go back to sleep, even if they have data to transmit. By allowing the signal to wake up and enable *opportunistic sending* for all nodes that hear the signal, signaling costs can be amortized over multiple transmissions from multiple nodes, while incurring only minor additional synchronization costs.

Our main contribution is the design of *Neighborhood-based Power Management (NPM)*, an energy-efficient MAC protocol that integrates opportunistic sending and opportunistic

scheduling to balance synchronization and signaling overhead. The novelty of NPM lies in its use of the wakeup signals. Unlike other signal-based approaches, NPM uses signals as neighborhood wakeup tones, enabling communication for all nodes in the neighborhood of the signaling node and amortizing the cost of signaling over multiple transmissions. Moreover, NPM uses opportunistically gained knowledge about the sleep schedules to adapt and ultimately reduce the signal length (i.e., preamble) based on one or a set of destinations. Although we show that the adaptive preambling provides the highest energy savings, the combination of the two mechanisms results in an energy-efficient protocol that supports efficient use of the communication channel with improved communication delay. Our evaluations show that by integrating neighborhood wakeup and adaptive preambling, NPM reduces the signal cost, resulting in higher energy-efficiency than signal-based protocols, B-MAC, X-MAC, and the hybrid protocol SCP. Our targeted approach to adaptive preambling, which only aims to wake up the destination and any other nodes that have similar wakeup times, outperforms SCP at high traffic loads, and B-MAC and X-MAC at all traffic loads, achieving the highest energy-efficiency.

The remainder of the paper is organized as follows. Section II presents the challenges of designing an energy-efficient MAC protocol, discusses how other protocols address these challenges and present the motivation for the design of NPM. Section III describes NPM and presents three policies to optimize signal length and reduce signaling overhead. In section IV, we model the energy consumption of NPM and analyze its energy-efficiency with respect to other protocols. In Section V, we evaluate NPM in comparison to other related approaches. Finally, Section VI concludes the paper and presents future research directions.

II. Energy-efficient Communication

Given the dynamic communication load expected in sensor networks, the most common target for energy conservation is the wasted idle time during communication. In an ideal network, senders and receivers would wake up at the same time, transmit their data, then go back to sleep. In real networks with clock drift and limited energy reserves, nodes can compensate by using *synchronization* to ensure that both the sender and the receiver know when to wake up and *signaling* to know when to communicate.

Synchronization-based coordination requires the exchange of schedule information between neighbors so senders can transmit exactly when the receiver is going to be awake. However, clock skew quickly results in a loss of synchronization, resulting in the need to periodically exchange schedule information [3], [6], [7], [15]. Synchronization-based protocols can either use dedicated synchronization messages or piggyback synchronization information onto data messages. For example, IEEE 802.11 PSM [14] uses a dedicated beacon for synchronization. However, since the frequency of PSM's beacon message exchange is not tied to the traffic generation rate or the clock skew, nodes send beacons even when there is no data to send, causing unnecessarily high synchronization costs at low rates and small clock drift. In response, S-MAC [6] and T-MAC [7] are able to choose the length of their synchronization period based on clock drift. Additionally, synchronization information can be piggybacked onto data messages whenever possible. While this works well at high traffic rates, when the traffic rate is not high enough to handle the clock drift, nodes must still send dedicated synchronization messages, resulting in high synchronization overhead.

Signaling-based coordination can be used in networks where nodes are completely unaware of the wakeup schedules of their neighbors. The sender's signal lets the receiver know that there is a transmission ready. The most straightforward approach is to send signals (i.e., preambles) at least as long as a channel polling interval (e.g., B-MAC), enabling receivers to periodically wake up and poll the channel. When receivers detect radio activity, they remain awake waiting for the real transmissions. In general, signaling costs can be controlled by varying the channel polling interval. However, while more frequent channel polling can reduce signal length, it results in increased listening overhead for the receivers. B-MAC provides optimized channel polling intervals using knowledge of network size and traffic patterns. However, without a priori knowledge or in networks with dynamic traffic patterns, it is difficult to find optimal parameters, potentially resulting in ineffective energy management, reduced network throughput and increased packet delay. Alternatively, sender nodes in X-MAC [2] and SpeckMAC [16] send short strobed signals that include the address of the intended receiver. A receiver acknowledges the signal to stop the signal transmission and to initiate the actual data transmission, shortening wakeup signals, while still using effective polling periods. However, this reduction is achieved at the cost of high idle listening costs at low traffic rates. Since a sender switches between transmit and listen mode during the entire preambling period, a receiver needs to poll the channel longer to detect the signal. X-MAC further reduces the signaling overhead by allowing any node to send all queued packets to the same receiver using a single signal. However, to apply this optimization, the traffic must be bursty. For periodic traffic unless the rate is very high, nodes using X-MAC still transmit one wakeup signal per data message. Thus, despite shortened preambles and support for traffic bursts, the signaling overhead is still very high at all rates for periodic traffic.

Hybrid protocols, like SCP [3] and AS-MAC [17], combine both synchronization and signaling mechanisms to coordinate the senders and the receivers. By assuming a loose synchronization between the wakeup schedules of the neighboring nodes, these hybrid protocols can use a larger synchronization period to reduce their synchronization overhead. Although the loose synchronization necessitates the use of signaling, the knowledge obtained from the synchronization complements the signaling mechanism by allowing the protocols to have shorter signals (i.e. preambles). Even with the loose synchronization requirement, both SCP and AS-MAC will require a short synchronization period in networks with any

significant clock drift. Additionally, at low traffic rates, these protocols incur very high synchronization overhead, since there is little opportunity for piggy-backing synchronization messages. Moreover, SCP uses the synchronization information to synchronize the wakeup schedules of its neighbors, which results in more contention in the network and increased delay. AS-MAC, on the other hand, maintains an asynchronous wakeup schedule and uses the synchronization information to directly reduce the length of the preambles. However for AS-MAC, additional synchronization overhead is incurred since each node now performs periodic neighbor discovery to allow the new nodes to join the network and obtain wakeup schedule information about its neighbors. Besides the synchronization overhead, the signaling costs for SCP and AS-MAC are also significant at high data rates, since signaling is done on a per data message basis.

In response to these limitations, we present Neighborhood-based Power Management (NPM). NPM tackles the problem of huge signaling overheads with the help of a data-driven low overhead synchronization mechanism. NPM nodes piggyback information about their schedules when they send data and control packets. Using this information, NPM dynamically adapts the length of the wakeup signal. The signaling overhead is further reduced since NPM uses its wakeup signal as a neighborhood wakeup signal. Once awakened by the preamble, all nodes can try to send the messages that are in their queues. By introducing a small control window, the overhearing cost due to listening to a signal are not wasted and all nodes are allowed to send packets without needing to repeat the wakeup procedure. The cost of sending a preamble is then shared among multiple nodes and across multiple packets.

III. Neighborhood-based Power Management

Neighborhood-based Power Management (NPM) is a hybrid protocol that utilizes both signaling and synchronization mechanisms to coordinate nodes before a data exchange. Since nodes in NPM exchange schedule information only when there is data, synchronization cost is low. However, synchronization can drift when the network has low traffic or nodes have high clock drift. To account for such loose synchronization, nodes send wakeup signals (i.e., preambles) before sending their data messages to make sure that the receivers are awake. However, unlike other signal-based protocols, NPM's preamble enables all nodes in the neighborhood of the signaling node to coordinate and exchange data, amortizing the signaling cost over multiple data transmissions. Nodes further reduce signaling costs by dynamically adapting preamble length using schedule information obtained from the low-overhead loose synchronization mechanism. In this section, we describe the two signaling mechanisms adopted by NPM: neighborhood wakeup and adaptive preambling. We also describe the details of NPM's beacon-based data-driven synchronization.

A. Signaling Mechanisms

All nodes in NPM wake up periodically and poll the channel for activity to receive incoming data messages. Due to the imperfect (out-of-date) synchronization information available to the nodes, NPM must use preambles before the actual data messages, to signal the receivers that they must stay awake until they receive the data messages. Since the control overhead due to signaling can be high in a network with unsynchronized wakeup-sleep schedules, NPM integrates two mechanisms to reduce the cost of signaling. While neighborhood wakeup reduces the total number of preambles required to send the data messages by allowing multiple data transmissions per preamble, adaptive preambling reduces the length of each preamble by utilizing the available synchronization information about the neighborhood.

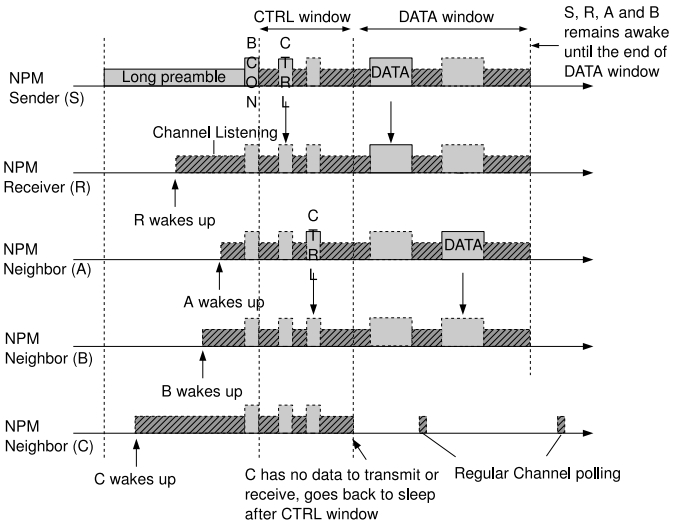

Fig. 1. Neighborhood wakeup with full preamble

1) Neighborhood Wakeup: When nodes in NPM detect a preamble during their periodic wakeup, they remain awake until the end of the preamble to receive and transmit data, if they have any. In this way, a single preamble could serve for multiple communications within the same neighborhood.

NPM can achieve lower signaling overhead since with one preamble it can send multiple packets to the same receiver or to different ones. It is also possible that the preamble wakes up both the sender and the receiver for a different flow, which can then initiate communication without the preambling phase.

NPM divides communication time into three components, the signal (i.e., preamble), the control window and the data window, which all together form a *transmission phase* (see Figure 1). After detecting a preamble, all nodes in the neighborhood of the preambling node coordinate to complete their data transmissions. Therefore, senders exchange *signal messages* with their intended receivers during the control window (similar to the ATIM window of IEEE 802.11 PSM). Nodes that have no data to send or receive, or cannot reach any nodes they have packets for, quickly go back to sleep after the control window.

All source-destination pairs complete their data transmissions using CSMA/CA in the data window. To support short term bursts of traffic, the nodes remain awake through the

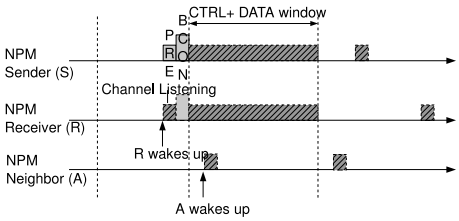

Fig. 2. NPM-Targeted Approach

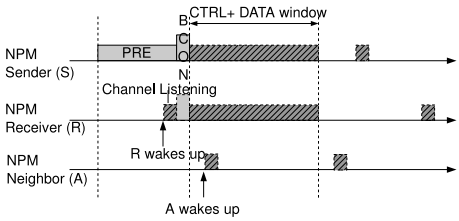

Fig. 3. NPM-Finish Early Approach

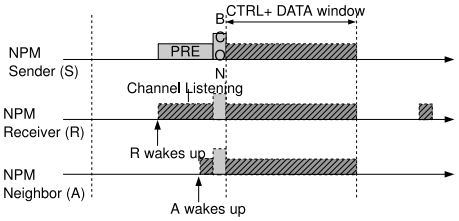

Fig. 4. NPM-Coverage Approach

entire data window enabling NPM to opportunistically send newly generated data within the same data window without additional signaling. Transmissions that fail during opportunistic sending are rescheduled in the next transmission phase.

2) Adaptive Preambling: Nodes in NPM adopting the basic neighborhood wakeup mechanism always send full length preambles (see Figure 1). We refer to this protocol as NPM *full*. The *adaptive preambling* component of NPM utilizes the available loose synchronization information (described in section III-B) to dynamically adapt the length of the preambles.

To account for clock drift, nodes add a guard time to both ends of their preambles. Nodes only reduce the length of their preambles when they have recent synchronization information about their receivers. However, when the available synchronization information is out-of-date, nodes must transmit full length preambles. The adaptive preamble mechanism is also never applied to broadcast messages since there is no guarantee to wake up all nodes in the neighborhood if the preamble length is shorter than the nodes' sampling interval.

NPM supports three different approaches for adapting the length of the preambles:

- NPM *targeted* aims to wake up only one receiver. Since nodes transmit preambles only during the expected wakeup times of their receivers, NPM *targeted* achieves the shortest preambles (see Figure 2).
- NPM *finish early* aims to improve delay performance while utilizing the neighborhood wakeup mechanism. Thus, nodes stop transmitting preambles as soon as their receivers wake up, similar to X-MAC (see Figure 3).
- NPM *coverage* aims to fully utilize the neighborhood wakeup mechanism, while still reducing the length of the preambles. Preambles are long enough to wake up all known nodes in the neighborhood (see Figure 4).

By reducing the length of the preambles, adaptive preambling decreases the potential use of the neighborhood wakeup and opportunistic sending components of NPM. Essentially, the shortened preambles in NPM *targeted* wake up fewer nodes, increasing the total number of preambles required to complete all data transmissions. On the other hand, NPM *coverage*, which benefits the most from the neighborhood wakeup, has the longest preambles. The choice of the approach that most suits the user's needs is the results of a trade off between energy efficiency and latency, and will be thoroughly explored in Section V.

B. Synchronization Mechanisms

NPM uses beacon-based synchronization to obtain the wakeup-sleep schedule information. Instead of exchanging schedule information periodically (like S-MAC, SCP and IEEE 802.11 PSM) and spending a fixed overhead for synchronization, nodes in NPM exchange their schedule information only when there is data to send. This data-driven synchronization approach saves NPM from wasting energy on synchronization when there is no traffic in the network.

Schedule information (i.e., synchronization) is exchanged via two types of control messages: beacons and signal messages. Instead of sending a beacon at the beginning of each wakeup period and thus incurring a fixed synchronization overhead like IEEE 802.11 PSM, NPM nodes send one beacon message per transmission phase. The preambling node broadcasts the beacon message just after sending the preamble (see Figure 1) and thus provides its schedule information to all awakened neighbors. Thanks to neighborhood wakeup, the synchronization overhead due to beaconing is amortized over multiple data transmissions. Nodes in NPM also piggyback their schedule information when they reply to the signal messages during the control window. Although the synchronization overhead due to signal acks increases as more data messages are transmitted, the size of the overhead is very small (8 bytes for the schedule information).

Synchronization information is sent as the time difference between the local time of the sender and the time when the sender will wakeup for the next cycle according to its periodic wakeup schedule. Thus, the schedule information is not affected by the clock skews of the different nodes in the neighborhood. Each node keeps track of the synchronization information of each of its neighbors by maintaining a *schedule table*. Each entry in the schedule table contains two pieces of information: the time difference between the neighbor's schedule and the local node's own wakeup schedule, and the age of the entry. Since synchronization information is not exchanged periodically, this information can become stale. The age information is used to purge stale data.

C. Fairness and Load Balancing Issues

When multiple nodes in the neighborhood have data to send, any one of them can act as the *preambling* node. To avoid preamble collisions, one of the nodes is chosen randomly as the *preambling node*. However, a random choice may cause some flows to starve or may drain more energy from some of

the nodes in the network. NPM adopts a priority-based random backoff mechanism to enforce fairness and load balancing while achieving higher throughput. All potential senders back off for a random time period, and the node with the lowest backoff wins the contention and becomes the *preambling* node for the entire neighborhood. NPM uses the following parameters to choose the random backoff period in each node independently:

- number of packets in the queue;
- age of the oldest packet in the queue;
- number of preambles sent.

NPM achieves load balancing by distributing the responsibility of preambling among all potential senders within the neighborhood. NPM increases the probability of a node being the *preambling* node if its queue is heavily loaded. This is achieved by reducing the length of its contention window. With this mechanism, NPM aims to achieve higher throughput. Since biasing the choice of preambling node based solely on queue length can lead to starvation for nodes with lower traffic, NPM also improves the probability for those nodes that have packets that waited a long time inside the queues.

Algorithm 1 shows the algorithm that we used for choosing the backoff time (the details are described in section V).

Algorithm 1 Preambling node selection

```
IF packets in the queue >= Queue len threshold
    backoff = RANDOM(1, queue len slot)
ELSE IF age of the oldest packet >= Age threshold
    backoff = RANDOM(1, age slot)
ELSE IF preambles sent >= Preambling threshold
    backoff = RANDOM(1, preamble send slot)
ELSE
    backoff = RANDOM(1, default slot)
ENDIF
```

IV. Analytic Model with Periodic Traffic

In this section, we describe the energy model of NPM and four other protocols: B-MAC, X-MAC, IEEE 802.11 PSM and SCP. Our analysis shows how parameters such as network density, data generation rate and duty cycle influence the performance of each protocol. We model a single hop network with $N+1$ nodes, which provides insights into the expected protocol behavior in larger networks. We model CBR traffic, where each of the $N+1$ nodes generate r packets per second. We analyze energy efficiency by comparing the energy consumed by each protocol for sending M data messages.

Total energy, E_{tot}, depends on the time that the radio transceiver spends in each of the four different power states, *transmitting*, *receiving*, *idling* and *sleeping*. We denote the power in both *transmitting* and *receiving* state as P_a. and the idle power as P_i, with $P_a >> P_i$. We ignore the (very small) power required to keep the transceiver in *sleep*. Thus, E_{tot} is a combination of active energy, $E_a = P_a \cdot t_a$, and idle energy, $E_i = P_i \cdot t_i$, where t_i and t_a are the time spent in idle and active states.

To minimize E_{tot}, a protocol has to minimize both t_a and t_i. Idle energy E_i can be minimized with low duty-cycles. However, minimizing active energy, E_a, requires minimizing its two components: data transmission energy, E_d, and control overhead, E_c. Since E_d must be incurred to transmit data, we focus on minimizing E_{tot} by minimizing E_c.

Overall, E_c has three major components: E_{pre} for preambling, E_{sync} for synchronization and E_{sig} for exchanging control signals before sending the actual data messages. The two overhead are determined by: the rate at which each protocol exchanges control messages and the number of awake nodes during the exchange.

Preambling overhead, E_{pre}, which is used by B-MAC, X-MAC, SCP and NPM, depends on the length of each preamble, t_{pre}, and the number of preambles, n_{pre}, used for sending M data messages. Since any node that wakes up during a preamble continues to receive the rest of the preamble, preamble length, t_{pre}, determines the number of awake nodes, A. The portion k of the entire preamble that the receiver actually receives is, however, determined by the synchronization between the wakeup schedules.

$$E_{pre} = n_{pre} \cdot (1 + k \cdot A) \cdot t_{pre} \cdot P_a$$

B-MAC wakes up all N nodes in the network by using a long preamble ($t_{pre} = T_{wup}$). However, SCP can wake up all nodes using short preambles due to its loosely synchronized network. SCP's synchronization also causes each node to receive the entire preamble ($k = 1$). Whereas, in an unsynchronized network (such as for B-MAC and NPM), each awake node receives on average only half of the preamble ($k = 0.5$). The value of A in NPM depends on the node's duty cycle and the length of the preamble t_{pre} used as will be explained later in the section. For periodic traffic, B-MAC and SCP uses one preamble for each of its M data messages. NPM however amortizes the E_{pre} over multiple transmissions by sending one preamble per transmission phase. Assuming NPM sends R_d data messages per transmission phase, $n_{pre} = \frac{M}{R_d}$ preambles are needed to transmit M messages.

$$E_{pre}^{scp} = M \cdot (1 + N) \cdot t_{pre} \cdot P_a$$
$$E_{pre}^{bmac} = M \cdot (1 + \frac{N}{2}) \cdot T_{wup} \cdot P_a$$
$$E_{pre}^{npm} = \frac{M}{R_d} \cdot (1 + \frac{A}{2}) \cdot t_{pre}^{npm} \cdot P_a$$

The strobed preambles in X-MAC result in an average preamble length of half of the maximum preambling time, $T_{wup}/2$. Half of this time is spent in sending preambles, while the other half is spent listening for an acknowledgment.

$$E_{pre}^{xmac} = M \cdot \left(\frac{T_{wup}}{4} \cdot (P_a + P_i) + (\frac{N}{2} + 2) \cdot t_{pre} \cdot P_a \right)$$

For all protocols, E_{pre} increases as the network density N increases. NPM consumes the least E_{pre} since it amortizes the cost of preambling over multiple data transmissions. BMAC uses the highest energy due to the long preamble used. The comparison of preambling energy between SCP and NPM will

depend on the length of the preamble used and the value of A. For NPM *full*, $A = N$ and $t_{pre} = T_{wup}$, increasing energy consumption. However, the value of R_d increases which in turn reduces E_{pre}. If the rate r is relatively high that $R_d > 1$, NPM *targeted* uses much less energy than SCP.

To determine the value of A, the number of nodes awakened by a preamble of length t_{pre}, assume a uniform distribution for the wakeup schedules across a wakeup period T_{wup}. If each node remains awake for T_l and asleep for T_s within each duty cycling period, the probability p that a preamble of length t_{pre} wakes up a particular node is defined as follows: $p = \min\left(\frac{T_l+t_{pre}}{T_l+T_s}, 1\right)$. Thus the average number of nodes awakened by such a preamble is defined as: $A = 1+p\cdot(N-1)$. For NPM *full*, the $T_s$long preamble wakes up all nodes in its neighborhood. The remaining NPM protocols wake up fewer nodes using their shorter preambles, and thus reduce the energy spent due to preambling.

The next major component of E_c is synchronization, which is used by IEEE 802.11 PSM, SCP and NPM. Nodes synchronize by exchanging beacon messages of length t_b. Assuming that n_b beacon messages are required and A_b nodes are awake during the transmission of a beacon, energy overhead for synchronization E_{sync} is:

$$E_{sync} = n_b \cdot (A_b + 1) \cdot t_b \cdot P_a.$$

In SCP and NPM, the number of nodes that receive the preamble and the beacon are the same, thus $A_b = A$. In PSM, all N nodes wake up at the beginning of the control window and receive the beacon. PSM sends one beacon per wakeup period, thus n_b does not depend on the data generation rate r, and it wastes energy even in the absence of data traffic. However, the high clock drift in wireless sensor networks causes the beaconing of PSM to be insufficient. Depending on the traffic rate r, SCP piggybacks synchronization information on each of its data messages, causing minimal overhead. However, when the traffic rate r is low, SCP needs to send separate synchronization messages. In that case, n_b for SCP depends on the synchronization period T_{sync}. NPM sends one beacon per transmission phase, requiring a total of $\frac{M}{R_d}$ beacons.

$$\begin{aligned}
E_{sync}^{psm} &= n_b^{psm} \cdot (N+1) \cdot t_b \cdot P_a \\
E_{sync}^{scp} &= n_b^{scp} \cdot (N+1) \cdot (t_{pre} + t_b) \cdot P_a \\
E_{sync}^{npm} &= \frac{M}{R_d} \cdot (A+1) \cdot t_b \cdot P_a
\end{aligned}$$

Thus, we observe that unlike other protocols, NPM's synchronization overhead is traffic-driven and is amortized over multiple data transmission similar to the preamble energy.

E_{sig} is used in protocols (such as IEEE 802.11PSM and NPM) that allow transmissions from multiple nodes within a single transmission phase. Nodes coordinate by exchanging control signals of length t_{sig} during a control window, T_a. According to our traffic model, each of the $A + 1$ awake nodes send control signals to their receivers. However, only the flows that have both of their sender-receiver pair awake receive signal acknowledgements back from their receivers. The number of such flows, F, depends on the previously defined probability p: $F = 1 + p + p^2 \cdot (N-1) = 1 + A \cdot p$.

After transmitting the control signals, nodes stay idle during the remaining control window T_a. The total idle energy consumed during the control signaling depends on the total number of transmission phases used to send M messages. The number of transmission phases is determined by the total number of messages R_d sent during one transmission phase.

$$\begin{aligned}
E_c^{npm} &= \frac{M}{R_d}\left((1+A)\left((1+A)t_{sig} + Ft_{sig}^{ack}\right)(P_a - P_i) + T_a P_i\right) \\
E_c^{psm} &= M \cdot (1+N) \cdot (t_{sig} + t_{sig}^{ack}) \cdot (P_a - P_i) + \frac{M}{R_d} \cdot T_a \cdot P_i
\end{aligned}$$

R_d, the total number of data messages transmitted per data window, depends on F, the number of nodes that remain awake during the data window of length T_d. Each node transmits all data messages that they have accumulated since their last active data window. With a total of $N + 1$ nodes in the network and F active nodes within each data window, each node has to wait $((N+1)/F) - 1$ transmission phases before its active data window arrives. Thus, R_d is the total number of data messages accumulated over a time period of $(N+1)/F$ transmission windows by each of the F nodes.

$$\begin{aligned}
R_d^{npm} &= \texttt{max}(r \cdot (N+1) \cdot (t_{pre} + t_b + T_a + T_d), 1) \\
R_d^{psm} &= r \cdot (N+1) \cdot (T_a + T_d)
\end{aligned}$$

NPM avoids the unnecessary idling during the control window when the data generation rate r is low. Also, NPM reduces the control overhead by sending fewer control signals. Since fewer nodes are awake in NPM, the receiving cost is also less.

E_d for IEEE 802.11 PSM and NPM is calculated similar to E_c. NPM consumes similar energy as PSM when exchanging data messages in the data window T_d. The higher value of R_d relative to PSM, allows NPM to use fewer transmission windows to transmit M messages, and thus consume less energy idle listening during its data window.

$$\begin{aligned}
E_d^{psm} &= M \cdot (N+1) \cdot (t_d + t_{ack}) \cdot (P_a - P_i) + \frac{M}{R_d} \cdot T_d \cdot P_i \\
E_d^{npm} &= M \cdot F \cdot (t_d + t_{ack}) \cdot (P_a - P_i) + \frac{M}{R_d} \cdot T_d \cdot P_i
\end{aligned}$$

For protocols that do not allow multiple transmissions from different nodes within a transmission window (such as X-MAC, SCP, B-MAC), E_d depends on F, the number of nodes awake during data transmission.

$$\begin{aligned}
E_d^{xmac} &= M \cdot 2 \cdot (t_{data} + t_{ack}) \cdot P_a \\
E_d^{bmac,scp} &= M \cdot (N+1) \cdot (t_{data} + t_{ack}) \cdot P_a
\end{aligned}$$

To demonstrate the expected energy consumption as derived in our model, we plotted the total energy per packet (i.e., the sum of each contribution) spent by each of the protocols in a single-hop network of 10 nodes in Figure 5. B-MAC is omitted

from this graph to make it more readable, since B-MAC's values are several times higher than the competitors. As expected, the energy consumption for SCP and X-MAC is constant since both protocols incur a fixed overhead per message. However, SCP requires more energy from overhearing the synchronized wakeup. In comparison, PSM and all of the NPM approaches decrease linearly with increasing message rate. Although NPM is more efficient with high traffic rates, the NPM *full* and coverage policies suffer from an excessive preambling in low traffic conditions. Although PSM looks very promising even from this model, it is important to note how this model does not account for clock drift that would cause PSM to require additional synchronization overhead. At very low data rates, it is clear from this graph that finish early and targeted are more expensive. However, we will show in the next section that in realistic networks, the crossover points are at very low data rates and that for bursty traffic, the crossover points disappear. In the next section, we present our simulation results to verify the expected energy consumption shown in our model and to evaluate the energy efficiency and performance of all of the protocols in larger multihop networks.

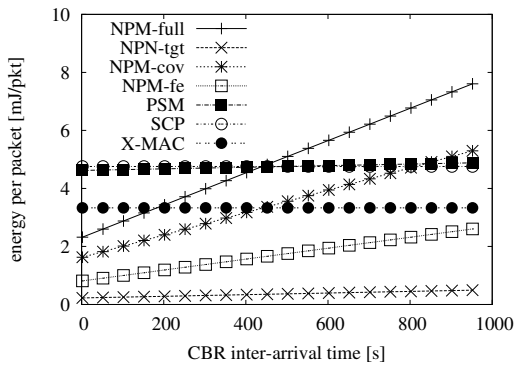

Fig. 5. Energy per packet described by our model for single-hop network

V. Evaluation

The goal of our evaluation is to show that by maintaining loose data-driven synchronization, NPM can achieve energy-efficient communication without affecting performance (i.e., delay, throughput). To demonstrate these benefits, we analyze the contributions of the two signaling mechanisms of NPM, neighborhood wakeup and adaptive preambling, in reducing the energy waste due to coordination. As a baseline for our evaluation, we compare NPM to signaling-based protocols: B-MAC and X-MAC, a hybrid protocol: SCP , and a synchronization-based protocol: IEEE 802.11 PSM. Although IEEE 802.11 PSM is designed for an ideal network with perfectly synchronized wakeup schedules, we compare NPM to PSM to understand how NPM performs in a realistic unsynchronized network compared to PSM working in a perfectly synchronized network. To verify the effectiveness of the neighborhood wakeup, we compare the NPM *full* approach to its dual B-MAC and the NPM *finish early* approach to its dual X-MAC. The comparison between NPM *finish early* and X-MAC is particularly interesting, since the burst support in X-MAC is similar to the neighborhood wakeup of NPM, but has a more constrained criterion compared to NPM. The combined effect of neighborhood wakeup and adaptive preambling is revealed when we compare NPM *targeted* to the hybrid protocol SCP.

To compare protocol efficiency, we consider three common metrics: throughput, delay per hop and energy. We characterize the energy efficiency of the protocols by analyzing the contribution of signaling and synchronization to total energy consumption. For all protocols, the length of the signals and the number of signals required to send all data messages together determine the total signaling overhead. For NPM, these two metrics also indicate the effectiveness of adaptive preambling and neighborhood wakeup respectively.

While the underlying protocols determine the time a radio spends in each radio state (transmit, receive, idle and sleep), the power consumed at each state is determined by the radio characteristics. For our evaluation, we used the power profile of MICAz radio [18] (current drawn in transmit mode = 17.4mA, receive mode = 18.8mA, idle mode = 20uA, and sleep mode = 0.1uA) with an external power source of 3.0V.

We evaluated protocol behavior across different traffic patterns, and in networks with different neighborhood sizes using *ns-2*. The target network contained 100 nodes in a 10 X 10 grid network. To analyze the effect of neighborhood size on performance, we varied the spacing among the nodes to 200m, 140m and 75m. With a radio coverage range of 250m, these node distances created a neighborhood of size 4, 8 and 18 nodes respectively. Since we obtained similar trends for the different node densities, only results for 140m are presented.

All protocols except IEEE 802.11 PSM and SCP were evaluated in randomly synchronized networks (i.e., random and unsynchronized wakeup times). SCP and IEEE 802.11 PSM were always run with synchronized wakeup schedules. Moreover, the results for IEEE 802.11 PSM in this section, do not capture any control overhead due to its costly [19] out-of-band synchronization mechanism and thus acts as a baseline in an ideal scenario. The only control overhead that we consider for PSM is the synchronization overhead due to beaconing.

Nodes in NPM, B-MAC and SCP duty cycle using a 1ms awake duration in every 100 ms period. Nodes in X-MAC remained awake for 2ms in every 100ms. The higher duty cycle for X-MAC was to support receiving the strobed preambles. Both NPM and PSM used a 100ms ATIM window and a 600ms data window for its nodes (Experiments were performed with various window sizes, and showed similar trends). To make the simulations more realistic, all protocols (except IEEE 802.11 PSM) added random clock drifts (maximum drift 100 ppm = 100s drift per 1 million s) to each node. To handle the drift, in our simulations, NPM and SCP added a 1ms guard time to its preambles and refreshed its neighbor tables every 60s. For NPM, the thresholds that we used for selecting the preambling nodes are: queue len = 20 packets, age of the oldest packet = 4s, and the number of preambles sent = 20 preambles. These parameter values for thresholds

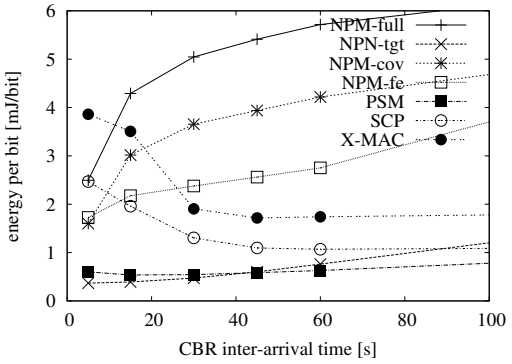

Fig. 6. Energy per bit: CBR

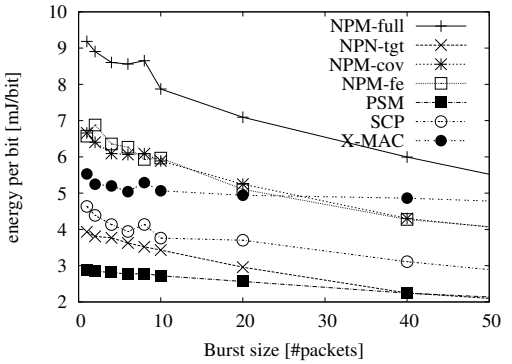

Fig. 7. Energy per bit: Bursty traffic

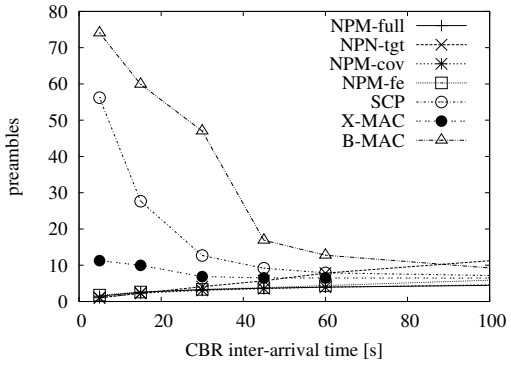

Fig. 8. Preamble count per data: CBR traffic

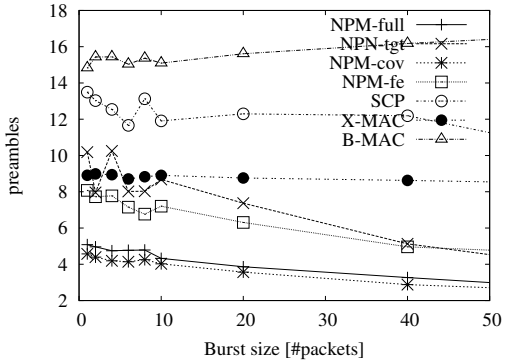

Fig. 9. Preamble count per data: Bursty traffic

were obtained empirically via extensive simulations.

To analyze protocol performance in different traffic scenarios, we evaluated the protocols with two traffic patterns: CBR (common for habitat monitoring applications) and bursty (common for event detection applications). Since traffic bursts increase the opportunity to utilize neighborhood wakeup, it lays out a favorable scenario for NPM and X-MAC. The effectiveness of neighborhood wakeup in CBR traffic scenario depends on the traffic generation rate. To make the CBR traffic more realistic, we chose the inter-arrival time between packets for each node from an exponential distribution. To create the different load conditions in this pseudo-CBR scenario, we varied the exponential average from a 600sec packet inter-arrival time for low load to a 5sec inter-arrival time for high load. For the bursty traffic, we simulated the protocols in a network, where 9 nodes at one corner of the 100 node grid generated bursts every 60s. We increased the number of packets in each burst to overload the network. In both cases, all data packets (size 80 bytes) were destined to a sink (transmit data rate of 250 Kbps [18]), which was placed in the distant corner of the grid. All protocols used shortest hop routing.

All results are obtained from an average of 10 simulation runs. For each run, all protocols were simulated for 1 hour.

A. Energy

To compare the energy profiles of the protocols at different rates, we use *energy per bit*. This metric captures the contributions of both data transmission and control overhead to the

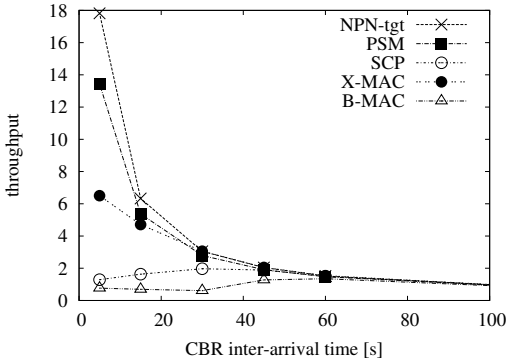

Fig. 10. Throughput: CBR traffic

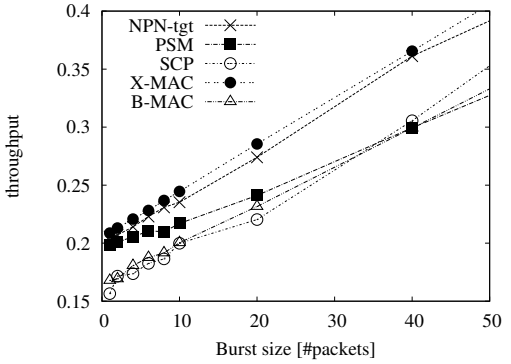

Fig. 11. Throughput: Bursty traffic

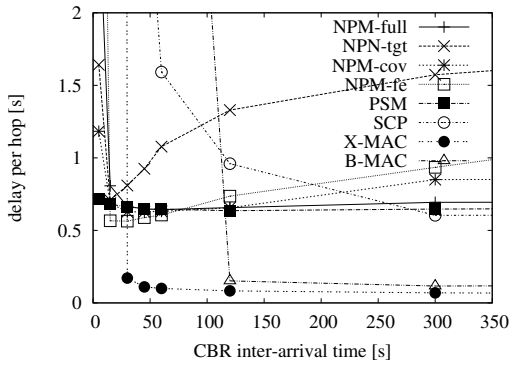

Fig. 12. Average Delay: CBR traffic

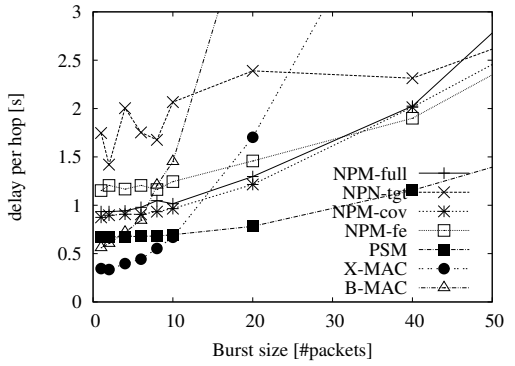

Fig. 13. Average Delay: Bursty traffic

total energy consumption. Thus, any protocol that incurs a low control overhead, will have a low energy per bit value.

As the traffic rate increases, protocols follow two interesting trends with respect to the total energy per bit (see figures 6 and 7). For both CBR and bursty traffic, energy for the protocols that allow neighborhood wakeup, such as PSM and NPM, either remain stable or have a decreasing trend with higher rates. For the rest of the protocols, B-MAC, X-MAC and SCP, energy consumption increases as the rate increases. To understand these trends, we analyzed the signaling and the synchronization overhead incurred by each of these protocols (we omit the graphs for signaling and synchronization overhead due to space constraints).

Signaling overhead is the only component of B-MAC and X-MAC's control overhead. At the low and moderate traffic rates, due to the one preamble per packet policy, both of these protocols consume fixed signaling overhead per bit at low rates. However, as the network gets overloaded, more packets get dropped due to collision and contention, increasing the energy consumed per received data bit (we omit B-MAC from the total energy per bit graphs due to its very high and skewed values relative to the other MAC protocols). The strobed preambles of X-MAC allow it to avoid overhearing and to have shorter preambles, resulting in much lower signaling overhead than B-MAC. For the same reason, the increasing trend of X-MAC starts at a much higher traffic rate than it does for B-MAC.

Although NPM uses a combination of both signaling and synchronization mechanisms, it is the signaling overhead that shapes the total energy per bit of NPM. Since NPM amortizes the cost of both signaling and synchronization over multiple transmissions (see figures 8 and 9 for the number of preambles required per data message), its total energy follows a decreasing trend as the traffic rate increases. The high energy efficiency of NPM *full* compared to its dual B-MAC for all traffic conditions, shows the effectiveness of neighborhood wakeup in reducing total energy consumption. However, the efficiency of NPM *finish early* with respect to its dual X-MAC is dependent on both the type and the rate of traffic. NPM *finish early* consumes less energy than X-MAC only when the effectiveness of NPM's neighborhood wakeup outperforms the benefit of X-MAC's very low signaling overhead. Achieving this goal becomes even more difficult at extremely low rates. Thus, despite the neighborhood wakeup, NPM *finish early* consumes higher energy per bit for CBR traffic until the rate becomes sufficiently high. NPM *targeted*, however, by using very low overhead for signaling, consumes much less energy than X-MAC for almost all traffic rates except for extremely low rates. At extremely low rates, NPM *targeted* has very few opportunities to use its neighborhood wakeup and to use the targeted preambles, causing targeted to consume higher energy than X-MAC. The situation is completely different with bursty traffic. Despite the traffic burst support of X-MAC, this protocol uses its preambles less efficiently than NPM *finish early*. Due to NPM's broader traffic support, NPM *finish early* uses fewer preambles than X-MAC (see figure 8 for CBR and figure 9 for bursty traffic). This allows NPM *finish early* to have a steeper decreasing trend for signaling overhead per bit than X-MAC. Thus, NPM *finish early* outperforms X-MAC even at very low burst rates.

SCP's low total energy per bit stems from its very low signaling overhead. SCP achieves this by spending more energy in synchronization. At low rates, SCP does better than the NPM *targeted*. But, as the rate increases, NPM's neighborhood wakeup amortizes the cost of both of its signaling and synchronization over multiple transmissions, causing NPM *targeted* to consume less energy than SCP. However, for the bursty traffic, NPM *targeted* always perform better than NPM *targeted* due to the effective neighborhood wakeup.

For NPM, at high rates, all approaches can effectively utilize the entire DATA window. So, the benefits of opportunistic sending becomes less important. And reducing the average length becomes the main source of energy-efficiency. During the simulation, for full, coverage, finish early and targeted, NPM average preamble length is 100ms, 80ms, 50ms and 5ms. Thus, NPM *finish early* and NPM *coverage* incur lower signaling overhead than NPM *full*. NPM *targeted*, having extremely short preambles, incurs the lowest signaling overhead among all NPM approaches at all traffic rates for both CBR and bursty traffic.

NPM *targeted*, which is the best in terms of energy among all NPM approaches, reaches very close to PSM in a perfectly synchronized network (we consider no control overhead for

PSM due to explicit synchronization). For higher rates, NPM *targeted* performs even better than PSM.

B. Delay per hop and Throughput

Both the neighborhood wakeup and the adaptive preambling mechanisms play important roles in shaping the delay characteristic of the NPM approaches. With low traffic, NPM and PSM, the two protocols that allow neighborhood wakeup, incur higher delay than B-MAC and X-MAC, due to the introduction of the control window and the fixed length data window. NPM delays the packets even more because of the preambles. SCP, on the other hand, in spite of using very short preambles, incurs higher delay than PSM and NPM *full*, due to the high contention from synchronized wakeups. The delay becomes worse for SCP for bursty traffic (we omit SCP from figure 13 because of its very high and skewed values relative to the other MAC protocols). Thus, at low traffic, X-MAC incurs the lowest delay due to its relatively shorter preambles than B-MAC.

Although the neighborhood wakeup mechanism induces higher delay in NPM and PSM at low traffic, it allows these protocols to maintain a relatively steady delay (more than 90% reduced delay of NPM at high CBR rate) and a linear throughput (see Figure 10, 11. we include only NPM *targeted*, since throughput for the different NPM approaches were similar) at higher traffic. B-MAC and X-MAC's inability to handle higher traffic causes the network to overload at a relatively low rate, CBR rate of 1 packet per 120sec for B-MAC, and CBR rate of 30sec for X-MAC (see Figures 12, 13). For NPM, the neighborhood wakeup determines the delay performance at high traffic; however, once the network gets saturated, delay is entirely determined by the effectiveness of the adaptive preambling component, causing NPM *targeted* to have the lowest delay at relatively higher traffic.

At high burst rates, NPM drops slightly more packets than X-MAC, as there is more contention within the transmission phase. However, NPM throughput is always within 5% of X-MAC's throughput (even at the highest rate).

When looking at all the results, it appears that NPM *targeted* is extremely efficient in terms of energy but suffers higher delays for bursty or low rate CBR traffic. In these scenarios, when delay is the main concern, NPM *finish early* provides good performance at a reasonable energy cost.

VI. Conclusion

This paper presents a new energy-efficient asynchronous MAC protocol for wireless sensor networks, called Neighborhood-based Power Management (NPM). By combining synchronization and signaling effectively, NPM reduces energy consumption across a large variety of network conditions. The data-driven synchronization mechanism in NPM allows nodes to opportunistically gain schedule information about their neighbors. Using this information, NPM reduces the signaling cost by applying adaptive preambling. Moreover, by strategically using the wakeup signal as a neighborhood wakeup tone, and thus allowing all awake neighbors to complete their transmissions using that signal, NPM further reduces signaling costs by amortizing it over multiple nodes and multiple transmissions. The detailed evaluation of the protocol shows its effectiveness in achieving low energy overhead and lower delay. We are currently implementing NPM in real testbed to verify its feasibility with the constrained resources of wireless sensors, and to validate the results from our simulations.

As for many protocols of the same family, the energy-efficiency achieved by NPM can be compromised by the presence of malicious nodes in the network. We intend to study the applicability of the existing solutions to our protocol. However this is an orthogonal problem and adding security extensions will not require changes to the protocol described in this paper.

References

[1] J. Polastre, J. L. Hill, and D. E. Culler, "Versatile low power media access for wireless sensor networks," in *SenSys 2004,*. ACM, 2004.

[2] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks," in *SenSys*, 2006.

[3] W. Ye, F. Silva, and J. S. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," in *SenSys*, 2006.

[4] C. Sengul, A. Harris, and R. Kravets, "Reconsidering power management," in *Broadband Communications, Networks and Systems, 2007. BROADNETS 2007. Fourth International Conference on*, 2007.

[5] K. Romer, P. Blum, and L. Meier, "Time synchronization and calibration in wireless sensor networks," *Handbook of Sensor Networks: Algorithms and Architectures*, 2005.

[6] Y. Wei, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks." IEEE, 2002.

[7] T. van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *SenSys03*, 2003.

[8] Y. Wong, L. Ngoh, and W. Wong, "An adaptive wakeup scheme to support fast routing in sensor networks," in *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, 2005.

[9] A. Keshavarzian, H. Lee, and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *MobiHoc*, 2006.

[10] R. Zheng, J. C. Hou, and L. Sha, "Asynchronous wakeup for ad hoc networks," in *MobiHoc*, 2003.

[11] J. Jiang, Y. Tseng, C. Hsu, and T.-H. Lai, "Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks," in *International Conference on Parallel Processing (32th ICPP'03)*, 2003.

[12] Y. Tseng, C. Hsu, and T. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," *Computer Networks*, vol. 43, no. 3, pp. 317–337, 2003.

[13] L. M. Feeney, "A QoS aware power save protocol for wireless ad hoc networks," 2000.

[14] B. O'Hara and A. Petrick, *The IEEE 802.11 handbook: A designer's companion*. IEEE Computer Society Press, 2004.

[15] Y. Sun, S. Du, O. Gurewitz, and D. Johnson, "DW-MAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, 2008.

[16] K.-J. Wong and D. K. Arvind, "SpeckMAC: Low-power decentralised MAC protocols for low data rate transmissions in specknets," in *REALMAN '06: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, New York, NY, USA, 2006.

[17] B. Jang, J. Lim, and M. Sichitiu, "AS-MAC: An asynchronous scheduled MAC protocol for wireless sensor networks," in *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008*, 2008.

[18] M. Datasheet, "Crossbow Technology Inc," *San Jose, California*, 2006.

[19] C. Hu, R. Zheng, J. Hou, and L. Sha, "A microscopic study of power management in IEEE 802.11 wireless networks," *International Journal of Wireless and Mobile Computing*, 2006.