Extending the Lifetime of Wireless Sensor Networks through Adaptive Sleep

Giuseppe Anastasi, Marco Conti, Mario Di Francesco

Abstract - In recent years, the use of wireless sensor networks for industrial applications has rapidly increased. However, energy consumption still remains one of the main limitations of this technology. As communication typically accounts for the major power consumption, the activity of the transceiver should be minimized, in order to prolong the network lifetime. To this end, this paper proposes an Adaptive Staggered sLEEp Protocol (ASLEEP) for efficient power management in wireless sensor networks targeted to periodic data acquisition. This protocol dynamically adjusts the sleep schedules of nodes to match the network demands, even in time-varying operating conditions. In addition, it does not require any a-priori knowledge of the network topology or traffic pattern. ASLEEP has been extensively studied with simulation. The results obtained show that, under stationary conditions, the protocol effectively reduces the energy consumption of sensor nodes (by dynamically adjusting their duty-cycle to current needs) thus increasing significantly the network lifetime. With respect to similar non-adaptive solutions, it also reduces the average message latency and may increase the delivery ratio. Under time-varying conditions the protocol is able to adapt the duty-cycle of single nodes to the new operating conditions while keeping a consistent sleep schedule among sensor nodes. The results presented here are also confirmed by an experimental evaluation in a real testbed.

Keywords – Wireless Sensor Networks, Energy Conservation, Power Management, Sleep/Wakeup Scheduling, Performance Evaluation

I. INTRODUCTION

A wireless sensor network (WSN) consists of a number of tiny sensor nodes deployed over a geographical area. Each node is a low-power device that integrates computing, wireless communication, and sensing capabilities. Sensor nodes are thus able to sense physical environmental information (e.g., temperature, pressure, vibrations) and process the acquired data locally, or send them to one or more collection points, usually referred to as sinks or base stations [1]. In recent years, the number of sensor network deployments for real-life applications, including industrial applications, has rapidly increased. In the industrial field, WSNs are currently used for factory automation [2], distributed and process control [3],[4],[5], and real-time monitoring [6]. An important class of applications is remote monitoring of industrial machinery and equipments. By continuously measuring temperature, pressure, vibrations and other parameters, it is possible to monitor the health of machinery, and prevent possible failures or malfunctions. WSNs are also used for timely detection of

Giuseppe Anastasi is with Dept. of Information Engineering, Univ. of Pisa, Italy, (e-mail: giuseppe.anastasi@iet.unipi.it)

Marco Conti is with IIT-CNR, Pisa, Italy (e-mail: marco.conti@iit.cnr.it)

Mario Di Francesco is with Dept. of Information Engineering, Univ. of Pisa, Italy, (e-mail: mario.difrancesco@iet.unipi.it)

liquid/gas leakage, radiation check, and other environment monitoring applications [6]. In addition, the ease of deployment and the ability to self organize and perform unattended operations, make WSNs particularly suitable to scenarios where human presence is impossible or unsafe (e.g., in a chemically contaminated field).

Based on recent studies [7],[8], the employment of WSNs for industrial applications is expected to increase more and more in the next years, especially in the fields of logistics, automation and control. However, energy consumption still remains one of the main obstacles to the spreading of this technology, especially when a long network lifetime is required. In fact, sensor nodes are generally powered by batteries which provide a limited lifetime (no more than a week, if nodes are always active) and, often, cannot be replaced nor recharged, due to environmental or cost constraints. In some cases it may be possible to scavenge energy from the external environment to recharge a secondary battery [9]. In any case, energy is a limited resource and must be used judiciously. Hence, efficient energy management strategies must be devised at sensor nodes to prolong the network lifetime as much as possible [10],[11].

If we break down the energy expenditure of a sensor node we can see that the radio subsystem typically consumes much more than the sensing and processing components. In addition, while being idle, the radio transceiver consumes approximately the same power as in the transmit or receive modes [12]. On the other hand, it consumes significantly less power when it is put in the sleep (low power) mode. Thus, the most effective approach to energy conservation is duty-cycling, which consists in putting the radio in sleep mode during idle periods. Sensor nodes alternate between sleep and wakeup periods, and they have to coordinate their sleep schedule in order to make communication feasible and efficient [13].

Unfortunately, designing efficient duty-cycling schemes is not straightforward. First, duty-cycling introduces additional delays in the message delivery process. Moreover, latency requirements are highly dependent on the application. For example, the detection of a flammable gas leakage requires a quick response, so high latencies may not be tolerated. Designing energy efficient solutions which, at the same time, provide low latency in message delivery is thus a challenging task. Second, most duty-cycling schemes use fixed parameters. i.e., the wakeup and sleep periods are defined before the deployment and cannot be changed during the operational phase. Fixed duty-cycling schemes require rather simple coordination mechanisms but, typically, have non-optimal performance. Adaptive schemes are thus required to adjust the sleep/wakeup periods, depending on the observed operating conditions.

In this paper we present an Adaptive Staggered sLEEp Protocol (ASLEEP) which automatically adjusts the activity of sensor nodes, achieving both low power consumption and low message latency. This protocol is targeted to data collection applications (e.g. monitoring applications), in which sensor nodes have to periodically report to a sink node. With respect to other similar approaches, our scheme provides two main advantages. First, it is not tied to any particular MAC (Medium Access Control) protocol, so that it can be used with different sensor platforms. Second, it is able to quickly adapt the sleep/wakeup periods of each single node to the actual operating conditions (e.g., traffic demand, network congestion, link quality, node density etc.), resulting in a better utilization of the energy resources and, hence, in a longer network lifetime.

ASLEEP was originally presented in [14], where a preliminary simulation analysis was also performed. In this paper the protocol is extended with several mechanisms which enhance its robustness against message losses. In addition, a detailed simulation analysis is carried out to investigate its performance in more general scenarios and under different operating conditions. The obtained results show that, thanks to its flexibility, ASLEEP largely outperforms commonly used fixed duty-cycling schemes in terms of energy efficiency, message latency, and delivery ratio. Hence, ASLEEP turns out to be a significant improvement in the context of monitoring, making thus possible a long-term deployment of WSNs.

The remainder of this paper is organized as follows. Section II surveys the related work. Section III describes the ASLEEP protocol. Section IV presents the simulation setup and discusses the obtained results. Finally, Section V concludes the paper.

II. RELATED WORK

In recent years, a very large number of energy conservation schemes for WSNs have been proposed. The reader can refer to [15] for a detailed survey on the most relevant proposals. In the following we will focus on *duty-cycling*, i.e., techniques that switch off the radio subsystem during idle times.

According to [13], duty-cycling can be achieved through two different (and complementary) approaches: *Topology Control* and *Power Management*. Topology Control (TC) protocols exploit node redundancy and adaptively activate the minimum subset of nodes which allow network connectivity. Nodes that are not currently needed for connectivity can switch off their radio and save energy. This increases the network lifetime by a factor (typically in the order of 2-3) that depends on the degree of redundancy. The reader can refer to [16] and [17] for detailed surveys on TC protocols. However, even nodes selected by the TC protocol do not need to remain active all the time. Instead, they can switch off

their radio when there is no network activity, thus alternating between sleep and wakeup periods. Hence Power Management (PM) protocols are aimed at coordinating the sleep periods of neighboring nodes so as to allow communication even in presence of a very low duty-cycle. As the activity of sensor nodes is typically very limited, PM protocols can reduce the energy consumption to some percent (with respect to the case without PM), thus increasing significantly the network lifetime.

In the following we will focus on PM protocols. Power Management can be implemented either at the MAC layer – by integrating a duty-cycling scheme within the MAC protocol – or as an independent sleep/wakeup protocol on top of the MAC layer (e.g., at the network or application layer). Duty-cycled MAC protocols (e.g., [18],[19],[20],[21],[22],[23],[24],[25],[26],[27]) allow the designer to optimize the channel access from an energy conservation perspective. However, they lack flexibility as a specific MAC protocol could not be always used in any actual sensor platform. In addition, it might be unable to exploit information made available by the application. On the other hand, independent sleep/wakeup protocols allow a greater flexibility as they can be tailored to the application needs, and, in principle, can be used with any MAC protocol. The solution proposed in this paper belongs to the class of independent sleep/wakeup protocols. We detail this class below.

We can broadly classify (independent) sleep/wakeup schemes into three main categories: *on demand*, *asynchronous* and *scheduled rendezvous*. *On-demand* schemes assume that destination nodes can be awakened somehow just before receiving data. To this end, two different radios are typically used [28], [29]. The first radio (*data radio*) is used during the regular data exchange, while the second one (*wakeup radio*) is a very low-power radio which is used to awake a target node when needed. These schemes can achieve a very high energy efficiency and a very low latency. However, they cannot be always used in practice because commonly available sensor platforms only have one radio. In addition, the wakeup radio has typically a transmission range significantly shorter than the data radio. A different option is using an *asynchronous* scheme [30],[31],[32]. In this case a node can just wakeup whenever it wants and it can still communicate with its neighbors. Although being robust and easy to implement, asynchronous schemes generally present high latency in message forwarding and have issues with broadcast traffic. The last category of independent sleep/wakeup schemes is represented by *scheduled rendezvous* schemes, which require that nodes are synchronized and neighboring nodes wake up at the same time. Our ASLEEP protocol belongs to this last category.

By focusing on scheduled rendezvous schemes, a possible approach consists in establishing a coarse-grained TDMA-like schedule defined at the application layer, and exploiting an underlying MAC protocol for actual data transfer. This approach is used by *Flexible Power Scheduling* (FPS) [33],[34],

which includes an on-demand reservation mechanism able to dynamically adapt to traffic demands. Since slots are relatively large, strict synchronization among nodes is not required. However, FPS borrows some drawbacks [35] from TDMA schemes (such as [18],[20],[23],[27]), i.e., it lacks flexibility in adapting to traffic and/or topology changes, and has a limited scalability.

Most solutions based on a scheduled rendez-vous approach consist in plain duty-cycling techniques. For instance, the well-known TinyDB query processing system [36] includes a sleep/wakeup scheme based on a fixed duty-cycle. All sensor nodes in the network wake up at the same instant and remain active for a fixed time interval. An improvement over this simple approach is the staggered scheme included in TAG (Tiny AGgregation) [37], which relies on a routing tree rooted at the sink node. In this scheme the active times of sensor nodes are staggered according to their position along the routing tree. Nodes located at different levels of the routing tree wake up at different, progressive times, like in a pipeline. This scheme has been considered and/or analyzed in many subsequent papers (including [26], [38],[39],[40],[41],[42]).

Although it provides a basic form of adaptation (wakeup times are staggered to the network topology), this scheme is not able to react to varying operating conditions as active times are fixed and equal for all nodes in the network. This constraint simplifies the coordination among nodes, but results in low energy efficiency and high message latency. Like TAG, our proposal leverages a staggered approach. However, in our proposal the active periods of nodes are dynamically adapted to the observed network conditions and are tailored to the actual needs. By minimizing the active period of each single node, our adaptive protocol (significantly) increases network lifetime and reduces message latency.

III. PROTOCOL DESCRIPTION

In this section we present the ASLEEP protocol. After a general overview, we will describe the core components of the protocol, i.e., the *sleep prediction algorithm* (used by each node to dynamically estimate the length of its expected active period), and the *sleep coordination algorithm* (used to enforce the new sleep schedule throughout the network). Finally, we will introduce two mechanisms to improve the robustness of the protocol. A pseudo-code description of the protocol can be found in [43].

A. Protocol Overview

In the following we will refer to a data collection scenario where data typically flow from source nodes to the sink, while data from the sink to the sources are much less frequent. We will assume that nodes are organized to form a logical *routing tree* (or *data gathering* tree) rooted at the sink, and use an underlying CSMA (Carrier Sense Multiple Access) MAC protocol for communication. These

assumptions are quite realistic, as most MAC protocols commonly used in WSNs are CSMA-based (e.g., [19],[21],[22],[24],[25]), and many popular routing protocols for WSNs rely on a routing tree (e.g. [26],[36],[37]). In a real deployment the routing tree is re-computed periodically to cope with possible topology changes and better share the energy consumption among nodes. However, as nodes are supposed to be static, we can assume that the routing tree remains stable for a reasonable amount of time. We also assume that sensor nodes are synchronized through some synchronization protocol (e.g., the one described in [44] and used in our experimental testbed [45]).

The communication between a parent and its children occurs in *communication periods* which repeat periodically. Each communication period includes an *active interval* during which nodes communicate by using the underlying MAC protocol, and a *silence interval* during which nodes turn their radio off to save energy. As shown in Figure 1, active intervals are staggered so that nodes at the lower levels in the routing tree wake up earlier than their ancestors. The active interval of each (intermediate) sensor node consists of two adjacent *talk intervals* (TI), the first one with its children and the other one with its parent[2]. Throughout, we will refer to the talk interval shared by a generic node j and all its children, during the *m-th* communication period γ^m, as τ_j^m.

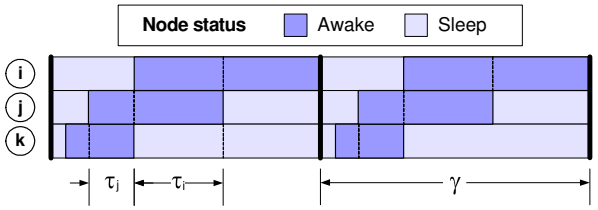


Figure 1. Parameters of the sleep scheduling protocol.

During each communication period every parent node estimates the duration of the talk interval to share with its children in the next communication period by means of the algorithm described in Section III-B. Although parent nodes can independently set their talk interval, a collective effort is needed for the schedule of the whole network to remain consistent and energy efficient. Hence, as a result of a change in the talk interval of a single parent node, the network-wide schedule must be rearranged. This is accomplished by appropriately shifting active intervals of a number of nodes, so as to ensure that (**i**) the active intervals of *all* nodes are properly staggered, and (**ii**) the two talk intervals of each node are contiguous (see Section III-C).

Two special messages, *direct beacons* and *reverse beacons,* are used for propagating schedule parameters to downstream and upstream nodes, respectively. Direct beacons are broadcast by every

[2] Obviously, the sink has only the talk interval with its children, while leaf nodes have only the talk interval with their parent.

parent node to all its children during each communication period. Instead, reverse beacons are sent in the opposite direction, i.e., from a child to its parent. As it will be shown below, direct beacons are critical for the correctness of the protocol. Hence, ASLEEP also includes mechanisms for (**i**) increasing the probability of successful delivery of direct beacons, and (**ii**) enforcing a correct (even if non-optimal) behavior of nodes in case they miss a direct beacon. In particular, to increase the probability of successful delivery, direct beacons are transmitted at the end of each talk interval, in a reserved time period (Beacon Period).

B. Talk Interval Prediction

In the ASLEEP protocol a sleep schedule is basically defined by the communication period and the talk interval of each individual parent node. The length of the communication period is closely related to the specific application and, thus, it is a global parameter specified by the sink when distributing the query. A variation in the communication period corresponds to a modification of the query, i.e. the new interval for the periodic data acquisition.

Choosing an appropriate talk interval is somewhat more involved. Ideally, each parent node should set the talk interval with its children to the minimum time needed to successfully receive *all* messages from *all* children. However, this time depends on a number of factors such as the underlying MAC protocol, channel conditions, degree of contention, number of messages to be received, and so on. In its turn, the number of messages to be received by a sensor node depends on the number of its children, the message generation rate at source nodes, and the network topology. Therefore, computing the ideal talk interval would require the global knowledge of the network. Moreover, this value should be continuously updated as the operating conditions change over time. Since such an approach is not practical, we propose here an adaptive technique that approximates this ideal scheme by letting every parent node to choose autonomously its own talk interval with its children. The decision involves only local information and, thus, it does not require to know the network topology.

In principle, any algorithm can be used to estimate the expected talk interval in the next communication period. We used the simple algorithm discussed below. Each parent node measures and stores the following quantities.

- *Message inter-reception time* (Δ). This is the difference between the time instants at which two consecutive messages are correctly received.
- *Number of received messages* (n_{pkt}). The total number of messages correctly received in a single communication period.

The time expected to get all messages sent by children in the next communication period is then estimated as $\overline{\Delta} \cdot n_{pkt}^{max}$, where $\overline{\Delta}$ and n_{pkt}^{max} are the average inter-reception time and the maximum number of received messages over the last L communication periods (observation window), respectively. Using n_{pkt}^{max} is a conservative choice to cope with possible message losses.

The former time interval should be appropriately increased to allow the parent node to send a direct beacon at the end of talk interval. Finally, to reduce the number of possible values, the expected talk interval is discretized into a number of time slots, whose duration is denoted by q. Hence, the expected talk interval for the $(m+1)$-*th* communication period can be expressed as $\tau_{est}^{m+1} = ceil(\overline{\Delta} \cdot n_{pkt}^{max} + \beta)/q) \cdot q$, where β denotes the length of the *Beacon Period*, i.e., the time interval reserved for beacon transmission only (see Section III-D). The duration q of the time slot should be chosen as a trade-off between efficiency and stability. From one hand, a low value of q allows a fine granularity in setting the talk interval duration, but may introduce frequent changes in the sleep schedule. On the other hand, a large value of q makes the schedule more stable, but may lead to talk intervals larger than necessary, thus wasting energy. It may be worthwhile noting that the expected talk interval cannot be lower than one slot. This guarantees that any child has always a chance to send messages to its parent, even after a phase during which it had no traffic to send.

Advertising τ_{est}^{m+1} to children as the next talk interval might lead to frequent variations in the schedule parameters of sensor nodes. Hence, the talk interval for the next communication period, τ^{m+1} is determined as follows. If $\tau_{est}^{m+1} - \tau^{m} > 0$, then τ_{est}^{m+1} is chosen as the estimated value and advertised to children (i.e., $\tau^{m+1} = \tau_{est}^{m+1}$). If the predicted talk interval is below the current value and the difference is greater than, or equal to a guard threshold g_{down} (i.e., $\tau^{m} - \tau_{est}^{m+1} \geq g_{down}$, with $g_{down} \geq 2q$), then the talk interval is decreased by just *one* time slot ($\tau^{m+1} = \tau^{m} - q$). Finally, if $0 < \tau^{m} - \tau_{est}^{m+1} < g_{down}$, the talk interval is not immediately decreased. However, if the same condition persists for a number L_{down} of consecutive communication periods, then the talk interval is reduced anyway. According to the rules discussed above, an increase in the talk interval is managed less conservatively than a decrease. This is because a more aggressive increase tends to minimize the probability that a node can miss messages from its children.

C. Sleep Coordination

As anticipated, the sleep coordination algorithm is based on two special messages (*direct* and *reverse*

beacons), and the sleep schedule re-arrangement after a talk interval variation is accomplished by appropriately shifting the talk intervals of nodes – so as to ensure that the network-wide schedule remains consistent.

Direct beacons are broadcast at *each* communication period by *every* parent node during the Beacon Period, i.e. at the end of the talk interval with its children. They include the schedule parameters for the next communication period. Specifically, the direct beacon sent by a node j in the m-th communication period contains:

- the length of the next communication period γ^{m+1};
- its next wakeup time $t_{parent,j}^{m+1}$;
- the length of the next talk interval to be shared with its children τ_j^{m+1} .

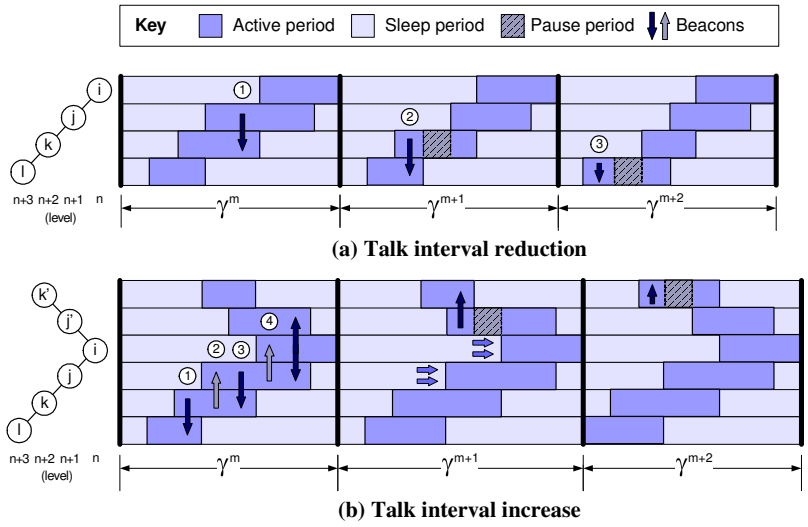


Figure 2. Examples of talk interval adaptation.

Conversely, *reverse beacons* are sent by child nodes. They may be sent at any time during the talk interval, and only include the amount of time the talk interval of the parent node has to be shifted. As schedules are local, nodes only have to coordinate with their parent, i.e. they have to know the wakeup time of their parent, and use it as a basis for establishing schedules with their children.

Let's discuss now the operations performed by ASLEEP when the operating conditions change. The interested reader can refer to [43] for a in-depth description of the algorithm and its pseudo-code. In the following, we will describe the main operations performed when the talk interval increases or decreases. In both cases, ASLEEP enters a transient phase which is required to propagate the new scheduling parameters and ensure that the new network-wide schedule is consistent and energy-efficient. Recall that a generic node has first the talk interval with its children, and then the talk interval

with its parent. As a consequence, a node advertises schedule parameters to its children before receiving the updated information coming from its parent. Hence, there might be the case in which the two talk intervals (the one with the children and the other with the parent) are not adjacent. In such a situation, the node inserts a *pause period*, in which it cannot communicate with any other node. This ensures that the new scheduling parameters are consistently handled during the transient propagation of schedules. Note that nodes can go to sleep during the pause period, if it is long enough to make it convenient to switch off and on again. In addition, pause periods are only used during the transient phase, so that they are not used in steady state conditions. Further details are given in the following discussion.

When a node shortens the talk interval with its children, it also defers its activation by a period corresponding to the difference between the previous talk interval and the new one. To better understand, let's consider the case illustrated in Figure 2-a. Let's suppose that during the *m*-th communication period a node *j* at the (*n+1*)-th level has decided to reduce its forthcoming talk interval with its children, i.e., nodes at the (*n+2*)-th level (including node *k*) . In the *m-th* communication period node *j* announces the next talk-interval duration to its children by means of the direct beacon (step 1). Its children receive the direct beacon and wait for the (*m+1*)-th communication period to inform their children – i.e., nodes at the (*n+3*)-th level – of the new scheduling parameters (step 2). Because of this, nodes at the (*n+2*)-th level introduce a pause period between their talk interval with their parent and the other with their children. This behavior ensures that nodes at the (*n+3*)-th level (e.g., node *l*) do not lose coordination with their parent, because they have already sent the information about their wakeup times. The above actions are repeated by nodes at the (*n+3*)-th level and their descendants (if any) in the next communication periods (step 3). Therefore, the pause period shifts to lower levels one communication period at a time. Hence, a new steady state schedule is reached after a number of communication periods equal to the depth of the subtree rooted at the node originating the new parameters. Note that only the descendants of the node which reduces the talk interval are affected by the transient phase through the pause period. The other nodes just operate with their own parameters as usual.

A similar approach is employed also when a node increases the talk interval with its children. In this case, the node has to force its ancestors to defer their talk intervals, in order to accommodate the additional time required for communications. To this end, the node makes use of a reverse beacon, which is sent to its parent and forwarded up to the tree until the sink node is reached. Note that this step is required to ensure the correctness of the protocol, i.e. that the talk intervals of intermediate nodes do

not overlap. As above, the example depicted in Figure 2-b will help understanding. Suppose that node *k* at the (*n+2*)-th level of the tree decides to increase the talk interval with its children, i.e., nodes at the (*n+3*)-th level (including node *l*). First, node *k* advertises the new talk interval to its children through the direct beacon (step 1). Second, in the same communication period, the node sends the reverse beacon (step 2) to its parent *j* at the (*n+1*)-th level, to force a talk interval shift ahead in time. Node *j* receives the reverse beacon, adjusts the parameters for the next communication period and advertises them, via the direct beacon (step 3), to its children. Because all ancestors have to shift their talk interval, node *j* also propagates the reverse beacon (step 4) up to its parent *i* at the *n*-th level. Note that in this case the schedule propagation impacts all nodes in the network. In fact, aside from the ancestors of the node which increases its talk interval, also other nodes can be involved in a transient phase which may require the introduction of a pause period. For instance, consider a node *j'* at the (*n+1*)-th level of the tree (illustrated in the second row of the scheme in the figure) which is not a direct ancestor of the node originating the new schedule. Its parent (i.e. node *i*) will shift ahead and advertise the new talk interval information during the *m*-th communication period. For reasons similar to the talk interval reduction, a pause period is introduced in the subtree rooted at nodes *i*, i.e. the nodes below the *n*-th level of the tree which are not direct ancestors of node *k* (which originated the new schedule).

Assuming that (**i**) clocks of nodes are properly synchronized, and (**ii**) direct and reverse beacons never get lost, it can be shown that the following properties hold (the corresponding proofs are in [43]):

Property 1 (Schedule agreement). *Child nodes wake up at the instant, and for the duration, enforced by their parent, even when talk intervals change*.

Property 2 (Non overlapping schedules). *For any two nodes i and j such that j is a child of i, the talk intervals* τ_i *and* τ_j *are not overlapped.*

Property 3 (Adjacent schedules). *In steady state conditions, the talk intervals shared by any node with its children and its parent, respectively, are contiguous.*

The above properties guarantee that, after a change has occurred in one or more talk intervals, the global sensor network is able to reach a new coordinated and energy-efficient schedule. In particular, Property 1 guarantees that activity times of a parent and its children are coordinated even after a schedule variation. Property 2 ensures that talk intervals of different parent nodes remain properly staggered. Finally, Property 3 guarantees that, in steady state conditions, each sensor node wakes up and goes to sleep just once per communication period.

These properties hold under assumptions (**i**) and (**ii**). Clock synchronization is beyond the scope of this paper and can be achieved through any available clock synchronization protocol (e.g. the protocol described in [44]). Note that, ASLEEP operates on top of the MAC layer and uses coarse-grained time parameters, so that a tight synchronization among nodes is not required. Assumption (**ii**) above is rather strong and unlikely to hold in practice, since beacons can get lost due to transmission errors and/or collisions. To overcome this problem we introduced a *Beacon Protection* mechanism to increase the probability of successful beacon reception at sensor nodes, and a *Beacon Loss Compensation* mechanism to offset the negative effects of direct beacon losses. These mechanisms are discussed in the next section.

D. Schedule robustness

Beacon Protection

Beacon messages are critical for correctness of the protocol. When a node misses a direct beacon containing the new parameters, it cannot schedule its activity for the next communication period. In addition, the node cannot send direct beacons to its children until it re-acquires the correct schedule information. As a consequence, the loss of coordination propagates along the routing tree to its descendants. Direct beacons may get lost, for example, due to communication errors or collisions with other beacons or regular messages transmitted by interfering nodes. As direct beacons are sent through broadcast frames, they cannot be re-transmitted by the underlying MAC protocol. Instead, reverse beacons are unicast messages and, thus, they are retransmitted by the MAC protocol if not received correctly.

To add robustness to the direct beacon transmission and prevent collisions, the last part of the talk interval – referred to as *Beacon Period* – is reserved for the direct beacon transmission only. Child nodes must refrain from initiating regular message transmissions during the Beacon Period. In addition, the transmission of the direct beacon is initiated with a random backoff delay. Finally, two back-to-back copies of the direct beacon are transmitted.

Beacon Loss Compensation

The Beacon Protection mechanism increases the probability that a direct beacon is successfully received by child nodes, but it does not solve the problem of direct-beacon losses. Therefore, we also devised the following mechanism to compensate the negative effects that derive from missing a direct beacon. Since talk intervals typically remain constant for a number of communication periods, when a

node misses a direct beacon, it uses the current schedule parameters also in the next communication period. However, if the node misses the direct beacon even in the subsequent communication period, it remains awake until it re-acquires a new direct beacon.

Obviously, this heuristic produces a correct schedule if the parent node has not changed the talk interval in the meantime, which is true in almost all cases. Otherwise, it produces a non-optimal behavior of the node (and its descendants as well) for a limited number of communication periods. The actual effect of a wrong prediction is different, depending on whether the talk interval has been increased or decreased. If a parent has reduced its talk interval, the corresponding child node wakes up earlier than the correct instant and we can now have overlapping schedules (i.e., Property 2 is lost). This results in energy wasting and useless message transmission that can potentially interfere with transmissions from other nodes. However, the child node remains awake until the end of the communication period and, very likely, receives a fresh direct beacon. On the other hand, if the talk interval has been increased, according to old schedule parameters, the child node wakes up at the right time but would go to sleep earlier than the correct instant. However, since it missed the direct beacon in the previous communication period, it doesn't go to sleep until it receives a fresh direct beacon. Thus, it is very likely that it receives the new direct beacon almost immediately. If this is not the case, it will remain active until a new direct beacon is received. Despite its simplicity, this compensation mechanism is able to ensure a correct schedule, even in the presence of direct beacon losses, at the cost of an increased energy consumption and message loss. In scenarios where the message loss probability is high, the number of consecutive direct beacons to be missed before remaining always on could be larger than two.

Incorrect schedules may also be originated by losses of reverse beacons. In this case a child node may wake up after the talk interval with its parent has elapsed. Hence, it is forced to remain awake until a new direct beacon is received. Fortunately, such extreme situations occur rarely (as reverse beacons are unicast messages, they are typically retransmitted by the underlying MAC protocol up to a maximum number of times). Nevertheless, ASLEEP is able to recover from this situation as well, at the cost of higher energy expenditure and increased message loss.

IV. SIMULATION ANALYSIS

To evaluate the performance of ASLEEP, we implemented[3] it by using the ns2 simulation tool [46].

[3] Since ASLEEP relies on a routing tree, we also implemented a simple routing tree formation algorithm in our simulator (see [43] for details).

We split the analysis in two parts. In the first part we investigated how the protocol reacts to changes in the operating conditions. In the second part we analyzed the performance of ASLEEP in steady state conditions.

A. Simulation setup

In both parts of our analysis we referred to a network scenario consisting of 30-50 nodes randomly deployed over a 50x50 m^2 area, with the sink placed at the center of the sensing area. Each node in the network generates a fixed number of messages per communication period, independent of its position on the routing tree. This scenario corresponds to a random deployment of sensor nodes over a given area for periodic reporting of sensed data, which is a typical case in monitoring applications. In such applications, data of interest (e.g., temperature, vibrations) are sensed and reported periodically to the sink node – data messages are typically short e.g., 10-20 bytes [11]. The sensing/communication period depends on the specific application. For many applications (e.g., remote monitoring of machinery health) a communication period of 30 s, or even larger, may be enough. For more critical applications where alarm messages have to be delivered with a tight timing (e.g., detection of liquid/gas leakage) a very short communication period may be required (e.g., few seconds) [6]. In our experiments we considered a communication period of 30 s. However, we also analyzed the effects of using lower communication periods.

In all our experiments we used the IEEE 802.15.4/Zigbee MAC protocol in non-beacon enabled mode. We used the 2.4 GHz physical layer and enabled MAC layer acknowledgements. We set the maximum number of retransmissions to 8, in order to increase the probability of successful message transmission. The transmission range was set to 15 m (according to the settings in [47]), while the carrier sense range was set to 30 m (according to the model in [48]). We used the Gilbert-Elliot model to simulate correlated message errors. Previous studies have shown that this model provides a good approximation of fading behavior in industrial environments and is, thus, a valuable tool for simulating errors in industrial wireless communication scenarios [49],[50],[51]. In addition, the Gilbert-Elliot model has been previously used in several performance studies referring to industrial wireless systems and networks (e.g., [52],[53]). In our experiments we took an approach similar to [53] and used values inspired from real measurements as in [49]. In detail, in most of experiments we considered a message error rate of approximately 10%, and average error-burst and error-free burst sizes (i.e., average sojourn times in the bad and good state of the Gilbert-Elliot model) of 5.7 and 46.2 ms, respectively. However,

for broadening the analysis of the impact of communication errors on the performance of ASLEEP, we considered different message error rates and error burst sizes[4].

TABLE 1. SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
Communication Period (CP)	30 s	Average Error-Burst Size	5.7 ms
Message rate	1 msg/CP	Average Error-free Burst Size	46.2 ms
Message size	20 bytes	Observation window (L)	10 CPs
MAC frame size	40 bytes	TI time slot (q)	100 ms
Transmission Range	15 m	Beacon Period	60 ms
Carrier Sensing Range	30 m	TI decrease time threshold (L_{down})	5 CPs
Message Error/Loss Rate	10%	TI decrease threshold (g_{down})	2q (200 ms)

We carried out a preliminary simulation analysis (not shown here) to tune parameters such as the observation window size (L), and the adaptation thresholds (L_{down} and g_{down}). Unless stated otherwise, we used the operation parameters shown in Table 1, and did not consider any form of data aggregation at intermediate nodes (i.e., intermediate nodes forward all messages coming from descendants to their parent). For each scenario, we generated 10 different random topologies and, for each topology, we performed a simulation run consisting of 1000 communication periods. The results shown below are averaged over the 10 different topologies. We also show the related standard deviations.

B. Analysis in dynamic conditions

To investigate the behavior of ASLEEP in dynamic conditions we considered two different kinds of variation in the operating conditions.

- *Traffic pattern variation*. Sensor nodes start with a given message generation rate. Then, after some time, they increase significantly their message rate and, finally, they switch back to the original rate. This scenario may occur when sensors are requested to report an event with better fidelity (i.e., using a higher sampling rate or including additional physical quantities) for a limited time.
- *Topology variation*. These experiments start with an initial configuration where only one half of the nodes deployed in the sensing area report data. After some time, also the remaining nodes start reporting data. This scenario may occur when additional nodes are required to report data so as to observe the sensed phenomenon with increased spatial resolution.

We measured the following performance indices:

[4] We performed this set of experiments also because the results in [49], although based on real measurements, have been obtained by using a wireless technology which is different from the IEEE 802.15.4 standard we are considering here.

- *Talk interval*. Plotting the talk interval duration over time provides a graphical representation of the ability of the protocol to adapt to changing operating conditions.
- *Duty-cycle*, denotes the fraction of time a sensor node is active within a communication period (it is given by the talk interval divided by the communication period).
- *Transient time duration*, defined as the number of communication periods from when the variation occurs to when the new talk interval stabilizes. We considered a talk interval as stable when it remains constant for more than L communication periods. This metric gives a measure of how quickly the protocol adapts to the new operating conditions.

Results

Since the analysis in dynamic conditions is aimed at investigating how the protocol reacts to changes in the traffic pattern and network topology, we did not consider the effects of message errors/losses in this part. This allow us to better understand the behavior of the protocol.

In the first set of experiments we varied the traffic pattern. All nodes started generating 1 message per communication period. After the 300^{th} communication period, the rate increased to 3 messages per communication period and, finally, after the 400^{th} communication period, it reverted back to the initial value. Figure 3-a shows the talk interval shared by the sink and its children as a function of time, in a specific topology (the trend was similar for all other topologies as well[5]). Specifically, the bottom curve represents the talk interval required by the sink node for receiving all messages from its children when using an *ideal* MAC protocol that guarantees non-overlapping transmission times (i.e., no collisions) and does not introduce any overhead. The fluctuating curve represents the talk interval *actually* required by the sink node, at each communication period, to correctly receive all messages from all its children in a staggered sleep/wakeup scheme. The difference, with respect to the previous curve, is due to the overhead introduced by the MAC protocol. Finally, the three remaining curves show the talk interval dynamically set by ASLEEP, and refer to different values of the time slot parameter (q). Ideally, the talk interval selected by ASLEEP should be as close as possible to the actual talk interval (i.e., the fluctuating curve). In practice, we can see that, initially, there is a sharp decrease in the talk interval estimated by ASLEEP. This is because the prediction algorithm takes an observation window of L communication periods before providing the first estimate (we used $L=10$ in our experiments).

[5] Although the sink node may be always on (typically it is not energy constrained), its talk interval contributes to define the activity time of its children which are the most loaded nodes in the network.

During this initial phase, a default value is used for the talk interval so that nodes can refrain from being always on. In our experiments we used an initial talk interval of 2 seconds. After this preliminary phase, ASLEEP is able to track very closely the actual talk interval required by nodes, even if it introduces some extra time. This is due to the Beacon Period needed for ensuring sleep coordination among nodes (β=60 ms in our experiments), and the extension of the estimated talk interval to an integer number of time slots (of duration q) to avoid frequent variations. As expected, a low value of q reduces this extra-time (and, thus the energy consumption) at the cost of an increased number of adaptations.

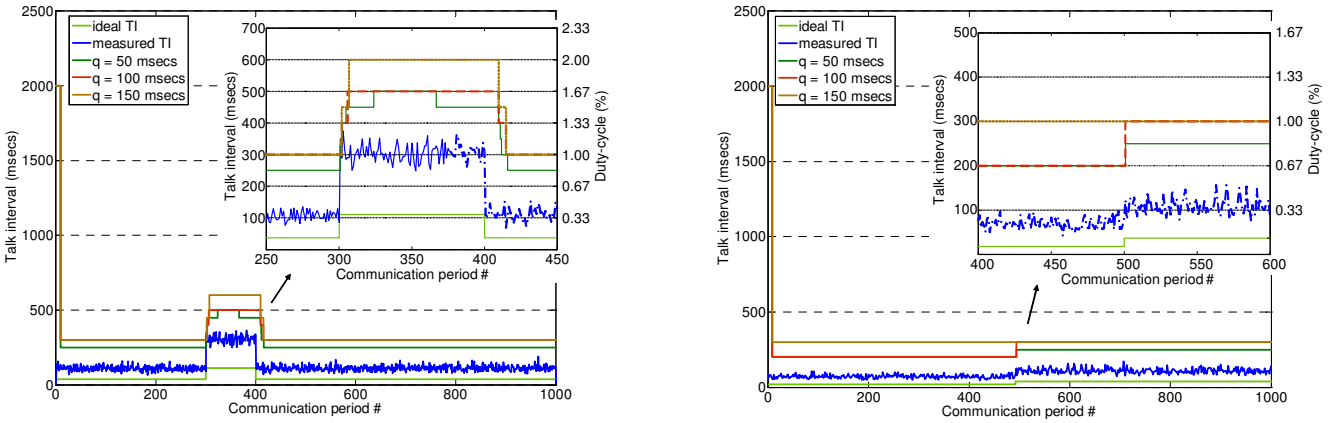


Figure 3. Talk interval adaptation to variations in the traffic pattern (left) and topology.

TABLE 2. TRANSIENT TIME FOR TRAFFIC VARIATION

Metric (q=100 ms)	Transient Duration (CPs)	
	MEAN	STD DEV
Up transient	4.1	3.0
Down transient	15.8	2.2

TABLE 3. TRANSIENT TIME FOR TOPOLOGY VARIATION

Metric (q=100 ms)	Transient Duration (CPs)	
	MEAN	STD DEV
Up transient	1.3	0.5

TABLE 2 shows the duration of the up and down transient times in the ASLEEP talk interval adaptation (when q=100 ms) originated by variations in the traffic pattern. Transient times have been averaged over the different topologies. The up transient spans over about 4 communication periods. This is strictly related to the data propagation process in a staggered scheme. To reach the sink, the new schedule parameters have to "climb up" the tree, one level at a communication period (recall that direct beacons advertise the parameters for the next communication period). Apart from the time required to trigger adaptation at the different levels, the results show that the protocol reacts quickly to increases in the message rate. Obviously, the actual values strongly depend on the given topology which, in our experiments, changes at every run because nodes are randomly re-deployed at the beginning of a new run. For example, it is clear that the transient time is conditioned by the number of levels in the tree,

which in our simulation runs varies between 3 and 4. This also explains the relatively high deviation in the obtained results. On the other hand, we can see that the down transient time is longer, i.e. about 16 communication periods. This is the joint effect of two factors. First, the system needs L communication periods to completely "forget" the previous traffic conditions (the talk interval is estimated based on statistics accumulated over the last L communication periods, where L=10 in our case). Second, the adaptation heuristic we have adopted is quite conservative in reducing talk intervals, since it also includes the additional L_{down} threshold (L_{down}=5 in our experiments). Clearly, this heuristic is a major driving factor of the performance in terms of transient times.

In the second set of experiments we investigated how the algorithm reacts to variations in the network topology. In these experiments, only one half of the deployed sensor nodes (i.e., 15 nodes) are initially active and send messages to the sink. The other 15 nodes become active only starting from the 500th communication period. Figure 3-b shows, for a representative simulation run, the variation over time in the talk interval shared by the sink and its children. The ASLEEP adaptive mechanism is effective also in this scenario. Assuming q=100 ms, we can see that the raising transient time is lower than in the previous experiment (see TABLE 3). This is because the increment in the resulting traffic conditions is limited, so that the additional messages do not need multiple increases of the talk interval to be properly estimated. As a consequence, the incoming messages almost immediately trigger the adaptation.

C. Analysis in stationary conditions

In this section we assess the performance of ASLEEP under stationary operating conditions, and compare it against other (non-adaptive) independent sleep/wakeup schemes, i.e. implemented above the MAC layer. Specifically, we consider three additional schemes, which are shortly described below.

- *Always-on*. In this scheme there is no duty-cycle: nodes are always active and forward messages as soon as they receive them. Obviously, this approach is rarely used in practice, and it is considered here only for comparison.
- *TAG-like staggered scheme*. Sensor nodes use a staggered sleep/wakeup scheme. The talk interval is fixed and equal for all sensor nodes. It is set to the value of the communication period divided by the depth of the routing tree. This is the same rule used in TAG [37]. Therefore, throughout we will refer to such a scheme as TAG.
- *Fixed staggered scheme*. In this scheme the talk interval is still fixed and equal for all nodes, like in TAG. However, the talk interval is approximately equal to the minimum value required in that configuration. Ideally, this minimum value corresponds to the time needed by any parent node to

correctly receive all messages coming from its children. In practice, this time cannot be known in advance and, thus, this scheme is unfeasible. In our experiments, for each configuration, we thus performed an additional preliminary simulation run to measure the maximum talk interval to be used. The rationale behind this choice is to compare the performance of ASLEEP against those of an ideal (but unfeasible) fixed scheme.

In all staggered schemes (TAG, Fixed and ASLEEP) messages are assumed to be generated just before the beginning of the talk interval. To make the comparison fair, especially in terms of message latency, when using the Always-on scheme we assumed that messages are generated at the same time instants as in the Fixed scheme.

To compare the performance of ASLEEP with those of the above schemes, we considered the following performance indices.

- *Average Duty-cycle*, defined as the average fraction of time a node at one hop from the sink remains active. This index gives a measure of the energy consumed by 1-hop sensor nodes. Since all traffic originated by source nodes must pass through these nodes, they determine the network lifetime [54].
- *Delivery ratio*, defined as the ratio between the number of messages successfully received by the sink and the total number of messages generated by all sensor nodes.
- *Average message latency*, defined as the average time interval between the generation time of a message at the source node and the reception time of the same message at the sink.

We started considering the basic scenario whose parameter settings have been specified in Section IV-A (basically, 30 nodes, 10% message error rate, average error-burst size of 5.6 ms). Then, we varied the node density, the duration of the communication period, the message error rate, and the average error-burst size so as to investigate the influence of each single parameter on the performance of the different duty-cycling schemes. As a preliminary remark, we have to emphasize that performance indices depends on the specific parameter settings. Therefore, their absolute values are of relative interest for us. What is really important is to *compare* the performance of the different sleep/wakeup schemes under the same operating conditions.

Performance comparison in the basic scenario

TABLE 4 summarizes the results obtained in the basic scenario. In terms of average duty-cycle, as expected, ASLEEP and Fixed perform much better than TAG. This is because in TAG the talk interval

size is equal to the communication period divided by the depth of the routing tree. In the basic scenario the tree depth is 3-4 in all random topologies generated by the simulator. This justifies an average duty-cycle of approximately 50% for TAG. When using ASLEEP the average duty-cycle is slightly higher than that experienced with the Fixed scheme (which is, however, unfeasible). This is due to the overhead required by ASLEEP to guarantee the coordination among nodes. In detail, as described in Section III-D, when a node misses two consecutive direct beacons, it is forced to remain active for the entire communication period in order to get a fresh direct beacon. Obviously, this increases the average duty-cycle of that node. Although several optimizations could be further introduced to reduce the energy wastage due to beacon losses, we nevertheless do not address them in this paper.

We now clarify the impact of the average duty-cycle (i.e., energy consumption) on the network lifetime. We consider the model used in [55] and adapt it to the parameters of the Tmote Sky device [56], which is assumed to be powered with a pair of 3000 mAh AA batteries. For a rough estimate, we only consider the contribution of the radio by using only the current draw in the transmit/receive states (19.6 mA) and neglect all other factors (i.e. sensing and processing). In the basic scenario, if one uses the Always-on scheme the network lifetime is approximately 6 days, which is not satisfactory for a long-term deployment. With TAG the network can survive up to approximately two weeks, which is still unsatisfactory. Thanks to its adaptation mechanism, ASLEEP is able to extend the network lifetime to approximately 203 days, which allows measurements to last for approximately 7 months, thus making long-term deployments actually possible. The network longevity allowed by ASLEEP is shorter than the one potentially achievable by using the Fixed scheme, which is approximately 290 days (about 10 months). However, the latter scheme requires to know in advance the talk intervals of sensor nodes, which is clearly unfeasible in practice.

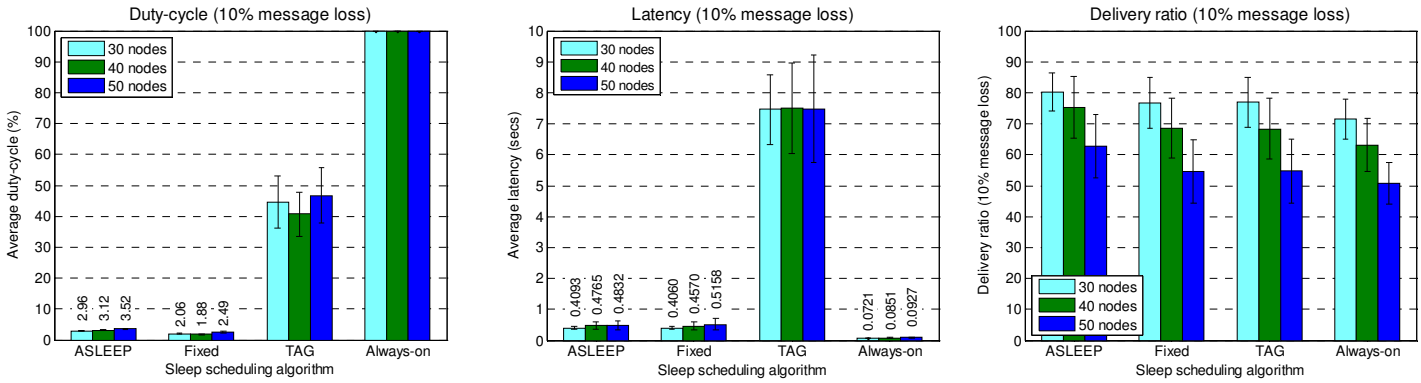
TABLE 4. PERFORMANCE IN THE BASIC SCENARIO: AVERAGE VALUE AND STANDARD DEVIATION (IN BRACKETS).

	Average duty-cycle (%)	Average Latency (s)	Delivery Ratio (%)
ASLEEP	2.96 (0.17)	0.4093 (0.0498)	80.32 (6.08)
Fixed	2.06 (0.22)	0.4060 (0.0543)	76.76 (8.06)
TAG	46.38 (8.91)	7.4564 (1.1253)	76.95 (8.03)
Always ON	100.00 (0.00)	0.0721 (0.0073)	71.45 (6.53)

In terms of average message latency, we can observe that the Always-on scheme introduces a very small latency as messages are forwarded as soon as they are received. The overall latency is thus limited to the sum of delays introduced for message transmissions in the various hops from the source to the sink. On the other hand, TAG exhibits extremely poor performance. Again, this is due to the way talk intervals are set in TAG, which strongly depend on the routing tree. The average latency is in the

order of 7.5 seconds, which roughly corresponds to the ratio between the communication period and the tree depth. ASLEEP and Fixed introduce approximately the same latency (but Fixed is unfeasible). We can observe that, by reducing the activity time of sensor nodes, ASLEEP also reduces accordingly the latency experienced by messages to reach the sink node.

Another important property of ASLEEP can be highlighted by looking at the delivery ratio. Our protocol exhibits the highest value among all schemes, including TAG and Always-on which, however, consume much more energy. This interesting result can be explained as follows. First, all staggered approaches (including ASLEEP) have a clear advantage over the Always-on approach, in terms of delivery ratio, as in staggered schemes nodes with a parent-child relationship never contend for forwarding messages to their corresponding parents. Instead, in the Always-on scheme nodes forward data as soon as they receive them, so that each parent contends with its children too. Second, ASLEEP achieves better performance than Fixed and TAG because each node can set the talk interval with its children independent of other nodes. As a side effect, children of sibling nodes wake up at different times which, in turn, reduces the probability of collisions in message transmissions. We found that, in the basic scenario, ASLEEP experiences – on average – about half the collisions occurring with TAG and Fixed. Note that both fixed staggered schemes (i.e., TAG and Fixed) have approximately the same performance, which indicates that there is no clear benefit from keeping nodes awake over a certain threshold. In both cases, nodes located at the same level of the routing tree start transmitting simultaneously and, thus, experience a large number of collisions and retransmissions.



(a) (b) (c)

Figure 4. Average duty-cycle (left), average message latency (middle), and delivery ratio (right) for increasing number of nodes (i.e., density).

Impact of node density

In this section we evaluate the impact of node density on the performance of the different sleep/wakeup schemes. To this end we varied the number of nodes in the range [30-50] by keeping the size of the

sensing area constant (50m x 50m). All other parameters are as in the basic scenario. In particular the message error rate and error burst size are equal to 10% and 5.7 ms, respectively.

Figure 4-a shows that, in general, the average duty-cycle tends to increase with node density. This is because a higher number of nodes in the same sensing area implies a larger number of interferences (i.e., a larger collision probability). The different behavior exhibited by TAG depends on the depth of the routing tree in the various scenarios. While the mean routing-tree depth – averaged over the 10 random topologies – is around 3.4 for the 30 and 50 nodes scenarios, it increases to about 3.6 for the 40 nodes scenario[6]. This explains the lower energy consumption obtained by TAG when the network is formed by 40 nodes. As in the basic scenario, the average duty-cycle of ASLEEP is slightly larger than that of the Fixed scheme. Since message latency is tightly related to the talk interval, in particular for ASLEEP and Fixed, these considerations also explain the increase of the average message latency – for increasing number of nodes – when using the above-mentioned two protocols (Figure 4-b).

Finally, Figure 4-c shows that the delivery ratio of all schemes decreases when the node density increases, as expected. This is because the probability of correct delivery to the next hop reduces due to the increased collision probability. Figure 4-c shows that ASLEEP still provides a delivery ratio significantly higher than all other schemes, even with a higher number of nodes.

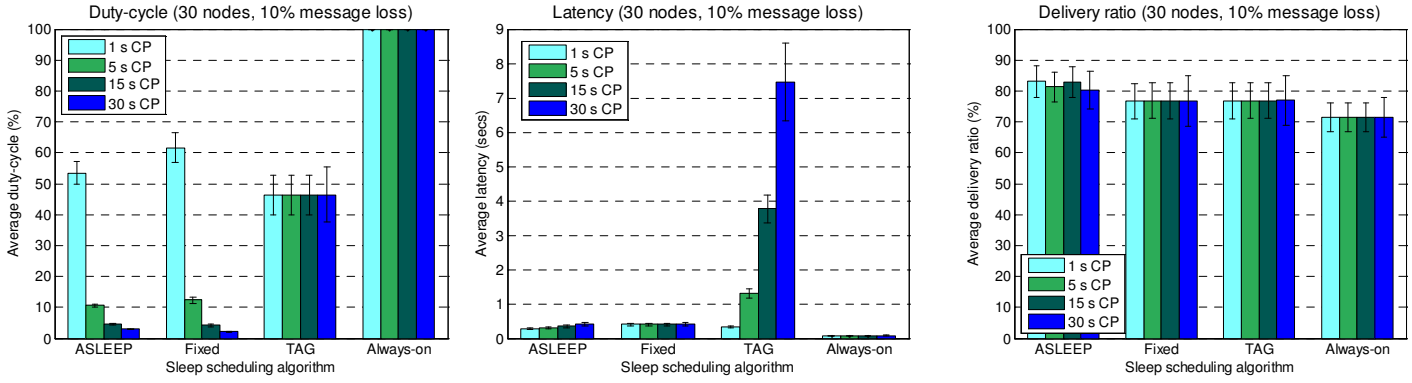
As a final remark, it may be worthwhile emphasizing that, since we are considering a CSMA/CA protocol, interferences may be originated not only by sensor nodes belonging to the same sensor network, but also by nodes belonging to other wireless (sensor) networks operating in the same frequency band. If the other networks use the same CSMA/CA protocol and are within the same transmission range, their impact is similar to what we observed by increasing the network density, i.e., an increase of the collision probability. On the other hand, if the other networks use a different MAC protocol and/or are outside the transmission range, their impact is measured through an increase in the message loss probability. This aspect will be investigated below.

Impact of offered load

In the basic scenario each sensor node transmits a 40-byte MAC frame – corresponding to a 20-byte message – every communication period (i.e., 30 s). Since there are 30 nodes in the network, the resulting total offered load is 320 bps. This is an extremely low load which, however, can be considered typical of many WSNs applications. In this section we want to investigate how the performance of the different sleep/wakeup schemes are impacted when the workload increases. This can achieved either by

[6] Since topologies are generated randomly, there is no control on the depth of the tree.

decreasing the size of the communication period γ (while leaving the number of messages per communication period unchanged), or by increasing the number of messages generated at each communication period (while leaving γ fixed)[7]. Thus, we performed two different sets of experiments. In the first set we varied the value of γ in the range [1-30] s, while leaving all other parameters as in the basic scenario. Figure 5-a shows that, when γ decreases, the average duty-cycle remains approximately constant for TAG (as the average number of levels in the routing tree does not change), while it increases for ASLEEP and Fixed (because the active time required for message exchange is unchanged but sensor nodes activate more frequently). When the size of the communication period is very small (e.g., 1s), TAG is even more energy-efficient than ASLEEP and Fixed. The minimum communication period γ_{min}, under which there is no real advantage in using ASLEEP with respect to TAG, can be easily calculated as follows. Let's denote by $\overline{\tau}_1$ the average talk interval of sensor nodes at 1-hop distance from the sink[8] when using ASLEEP, and λ be the number of levels in the routing tree. Clearly, ASLEEP will exhibit an average duty-cycle larger than TAG if $\overline{\tau}_1$ is larger than the (fixed) talk interval used in TAG i.e., $\overline{\tau}_1 > \frac{\gamma}{\lambda}$. Hence, $\gamma_{min} = \overline{\tau}_1 \cdot \lambda$. In the basic scenario considered here, $\overline{\tau}_1$ is less than 0.4 s, and the value of λ (averaged over all the topologies generated in the simulation experiment) is 3.4. Hence, for this scenario γ_{min} is about 1.4 s. Note that we limited our analysis to communication periods above 1 s to prevent TAG from using a talk interval too short with respect to the network demands. In fact, a very low value of the talk interval would prevent sensors from sending all their messages, thus making the comparison with the other sleep scheduling schemes unfair.



(a) (b) (c)

[7] The effect of increasing the message/frame size is less relevant and, in addition, has been already investigated in [43].

[8] It is worth recalling here that the average duty-cycle is based on the energy consumption of the 1-hop neighbors of the sink, therefore we will consider the impact of these nodes only in the following discussion.

Figure 5. Average duty-cycle (left), average message latency (middle), and delivery ratio (right) for different values of the communication period.

Figure 5-a also shows that ASLEEP can be even more energy-efficient than Fixed, especially for low sizes of the communication period. This is because with Fixed all sensor nodes use the same talk interval, while with ASLEEP different nodes may use different talk intervals, depending on their operating conditions. Obviously, the impact of this difference on the average duty-cycle is more apparent when the communication period is shorter.

Figure 5-b shows that – in terms of message latency – only TAG is significantly influenced by the size of the communication period (the talk interval of nodes changes accordingly). Finally, as expected, the delivery ratio of all schemes does not exhibit significant variations (see Figure 5-c).

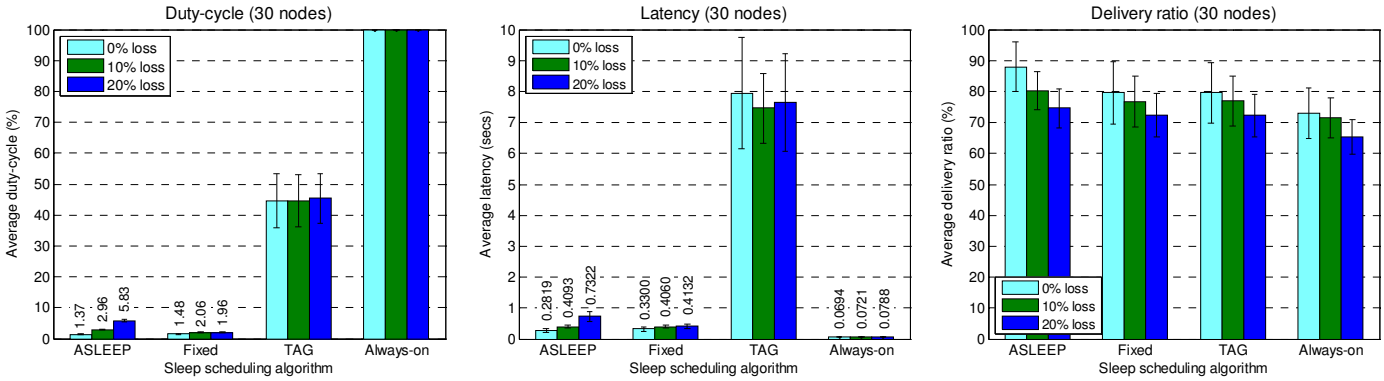
In the second set of experiments we varied the number of messages per communication period in the range [1,5] while keeping the value of γ unchanged and equal to 30 s. The obtained results are omitted for the sake of space, but they can be summarized as follows. When the number of messages to be transmitted increases, the average message latency increases accordingly with ASLEEP and Fixed (up to 0.9 s in the worst case), while it does not vary significantly with TAG. On the other side, the delivery ratio decreases (the worst result being for the always-on scheme with a delivery ratio of 60% with the highest load). The average duty-cycle obviously exhibits a trend similar to latency. However, in this case, the performance of ASLEEP decreases more than Fixed for the highest loads. For instance, the average duty-cycle of ASLEEP is 4.89%, compared to the 3.95% of the Fixed scheme, when 5 messages are generated per communication period. This is because of the higher level of contention in the network, which increases the probability of beacon losses. More details on the impact of the beacon loss on the performance of ASLEEP are given in the next section.

Impact of message error rate

Since ASLEEP relies on direct and reverse beacons for maintaining sleep coordination among sensor nodes, it is extremely important to assess its robustness and performance in the presence of communication errors introduced by the wireless link. In this section we investigate the influence of the message error rate, while in the next section we will look at the impact of the error burst size (i.e. the sojourn time in the bad state). In the following experiments we vary the message error rate in the range [0-20%], while keeping all other parameters as in the basic scenario. In particular, we consider 30 nodes and an average error burst size of 5.7 ms.

Figure 6-a shows that the average duty-cycle of both ASLEEP and Fixed tends to increase with the message error rate. This is because a higher number of errors requires more retransmissions at the MAC layer (in TAG there is no significant effect as the average duty-cycle only depends on the routing tree). However, the increase is much more apparent in ASLEEP, rather than Fixed, due to effects of beacon losses. In the adaptive scheme the miss of two consecutive direct beacons forces the sensor node to remain active for the entire next communication period, thus increasing significantly the average duty-cycle. In addition, reverse beacons not correctly received by the destination node must be retransmitted, just like normal messages. These remarks also justify the increase in the average message latency as the wireless link becomes more and more unreliable (see Figure 6-b). Finally, Figure 6-c shows the effects of communication errors on the delivery ratio. As expected, for all schemes the percentage of (correctly) delivered messages decreases as the message error rate increases. However, ASLEEP still experience the highest delivery ratio, for the same reasons highlighted above (i.e., lower number of collisions with respect to the other schemes).

The above results highlight that ASLEEP is more sensitive to packet errors than non-adaptive schemes (i.e., TAG and Fixed), due to the distributed sleep coordination algorithm. However, even when the message error rate (in each single wireless link) is 20%, the average duty-cycle is around 6%. In any case, this is the necessary cost to be paid for achieving adaptive talk intervals.



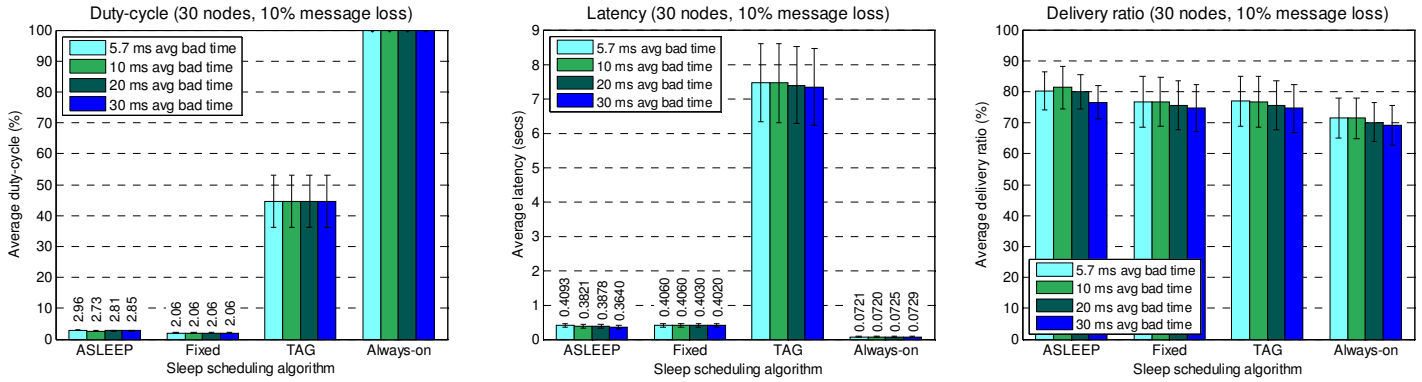
(a) (b) (c)

Figure 6. Average duty-cycle (left), message latency (middle), and delivery ratio (right) for different message error rates.

Impact of message error-burst size

In this section we complete the analysis started in the previous section by investigating the impact of the error burst size. To this end, we vary the average error-burst size (i.e., the average holding time in the bad state of the Gilbert-Elliot model) in the range [5.7 - 30] ms. At the same time, we vary accordingly the average packet error-free burst size (i.e., the average holding time in the good state) so as to maintain the message error rate at 10%. All other parameters are as in the basic scenario.

Figure 7-a and Figure 7-b show that the average duty-cycle and message latency are not significantly affected by the average error-burst size, provided that the message error rate remains constant. This is because we assumed that messages not successfully transmitted to the parent node before the end of the talk interval are discarded (i.e., they are not deferred to the next communication period). The MAC protocol uses retransmissions to recover from possible communication errors. However, as bad periods become longer and longer, the retransmission mechanism tends to become ineffective, and an increasing fraction of messages is discarded. However, since bad communication period occurs sporadically[9] (and tend to become more and more sporadic as the average error-burst size increases), discarded messages do not affect significantly the estimate of the next talk interval size, as this is calculated over a certain number of communication periods. Instead, they affect the delivery ratio (see Figure 7-c). As the error burst size increases, the probability that a message cannot be successfully transmitted before the end of the current talk interval increases accordingly. Hence, the delivery ratio generally tends to decrease for all schemes. Obviously, ASLEEP is much more sensitive to lost messages than the other schemes, as beacons may get lost as well during bad periods. This justifies the more significant dependency of ASLEEP on the average error-burst size.



(a) (b) (c)|

Figure 7. Average duty-cycle (left), message latency (middle), and delivery ratio (right) for different packet error-burst sizes.

Discussion

From the analysis presented above it clearly emerges that, thanks to its ability to tailor the talk interval of each single node to its real needs, ASLEEP not only reduces the average duty-cycle of sensor nodes – and, thus, energy consumption – with respect to non-adaptive (feasible) staggered schemes. It decreases accordingly the average message latency, and may also increase the delivery ratio. These are

[9] Obviously, bad periods occurring while the node is sleeping have no effect.

very important features in an industrial perspective as they allow for long term deployment of sensor networks.

Obviously, ASLEEP is more complex than non-adaptive staggered schemes, as it requires a continuous coordination among nodes to maintain the network-wide sleep schedule. Therefore, it is more sensitive to message errors/losses which are quite inevitable in industrial environments. However, the analysis above has shown that the protocol can work correctly and efficiently even when the wireless communication is very unreliable (e.g., the link message error rate is 20%). In addition, its robustness against communication errors could be further enhanced by means of traditional techniques for increasing wireless reliability (e.g., using FEC techniques in beacon transmissions).

Since simulation experiments might not take into account all factors that can occur in a real environment, we also implemented our ASLEEP protocol (and the other considered sleep/wakeup schemes) for the Tmote Sky sensor platform [56] under the TinyOS 1.1.15 operating system [57]. Tmote Sky sensor nodes use the Chipcon CC2420 radio transceiver which is compliant to the IEEE 802.15.4 physical layer and enables 250Kbps bit rate over the unlicensed 2.4 GHz ISM band. In this implementation ASLEEP rely on the CSMA/CA protocol shipped with TinyOS which is different from the IEEE 802.15.4 MAC protocol considered in the simulation experiments. The experimental results are not shown here for the sake of space (the reader can refer to [45]). However, they confirm the simulation results discussed above and show that ASLEEP can be effectively employed on top of different CSMA/CA MAC protocols.

V. CONCLUSIONS

In this paper we have defined an Adaptive Staggered sLEEp Protocol (ASLEEP) for efficient power management in wireless sensor networks targeted to periodic data acquisition. The proposed protocol has several strengths. It staggers the schedules of nodes according to their position in the routing tree. This helps to reduce latency also when nodes are sleeping for most of the time and favors data aggregation. Unlike traditional staggered schemes, however, in the proposed protocol the active period of each sensor node is adjusted dynamically based on the traffic pattern and the operating conditions experienced by *that* node. ASLEEP is thus able to adapt to variations in the message generation rate, network topology, external conditions, and so on. In addition, as the active periods are tailored to the actual needs of each single node, the proposed protocol tends to minimize both energy consumption (thus extending the network lifetime) and message latency. Finally, the ASLEEP protocol is conceived

as an independent sleep/wakeup protocol operating above the MAC layer. Thus, it is independent from the underlying MAC protocol and can be used with different sensor platforms.

The performance of the protocol has been investigated in both dynamic and stationary conditions, through an extended analysis based on simulations and real measurements. The results obtained show that the protocol is able to react quickly to variations in the traffic pattern and network topology. In stationary conditions, we observed a significant reduction in the duty-cycle of sensor nodes, and, hence, an increase in the network lifetime, with respect to fixed staggered approaches where active periods are fixed and equal for all nodes in the network. As a side effect, we also observed an increase of the delivery ratio. Finally, we verified that ASLEEP is able to work effectively even in the presence of correlated communication errors which are typical in industrial environments.

ACKNOWLEDGEMENTS

Work funded partially by the European Commission under the FP6-2005-NEST-PATH MEMORY project, and partially by the Italian Ministry for Education and Scientific Research (MIUR) under the FIRB ArtDeco and PRIN WiSe DeMon projects.

VI. REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, “Wireless Sensor Networks: a Survey”, *Computer Networks*, Vol.38, No: 4, March 2002.

[2] D. Miorandi, E. Uhlemann, S. Vitturi and A. Willig, “Guest Editorial: Special Section on Wireless Technologies in Factory and Industrial Automation, Part I”, *IEEE Transactions on Industrial Informatics*, Vol. 3, N. 2, May 2007.

[3] M. D. Lemmon, Q. Ling, and Y. Sun, “Overload management in sensor-actuator networks used for spatially-distributed control systems,” *Proc. Conf. Embedded Networked Sensor Systems (SenSys)*, pp. 162–170, Los Angeles, CA, Nov. 2003.

[4] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. S. Sastry, “Distributed Control Applications within Sensor Networks,” *Proc. of the IEEE*, vol. 91, no. 8, pp. 1235–1246, Aug. 2003.

[5] G. Platt, M. Blyde, S. Curtin, J. Ward, “Distributed Wireless Sensor Networks and Industrial Control Systems - a New Partnership**”,** *Proc. IEEE workshop on Embedded Networked Sensors (EmNetS-II)*, April 30 - May 01, 2005.

[6] K. S. Low, W. N. N. Win, and M. J. Er, “Wireless Sensor networks for Industrial Environments,” *Proc. Int. Conf. Computational Intelligence for Modeling, Control and Automation (CIMCA 2005)*, Nov. 2005.

[7] Embedded WiSeNTs Consortium, “Embedded WiSeNts Research Roadmap (Deliverable 3.3)”, available at www.embedded-wisents.org.

[8] ON World Inc, “Wireless Sensor Networks – Growing Markets, Accelerating Demands”, July 2005, available at http://www.onworld.com/html/wirelesssensorsrprt2.htm.

[9] IEEE Pervasive Computing, “Energy Harvesting and Conservation”, Vol. 4, Issue 1, Jan-Mar. 2005.

[10] A. Kansal, J. Hsu, S. Zahedi, M. Srivastava, “Power Management in Energy Harvesting Sensor Networks”, ACM Transactions on Embedded Computing Systems, Vol. 6, N. 4, 2007.

[11] A. Willig, “Recent and Emerging Topics in Wireless Industrial Communications: a Selection”, *IEEE Transactions on Industrial Informatics*, Vol. 4, N. 2, May 2008.

[12] V. Raghunathan, C. Schurgers, S. Park and M. B. Srivastava, “Energy Aware Wireless Microsensor Networks”, *IEEE Signal Processing Magazine*, Vol. 19, No: 2, March 2002.

[13] D. Ganesan, A. Cerpa, W. Ye, Y. Yu, J. Zhao, D. Estrin, “Networking Issues in Wireless Sensor Networks”, *Journal of Parallel and Distributed Computing*, Vol. 64 (2004) , pp. 799-814.

[14] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, "An Adaptive and Low-latency Power Management Protocol for Wireless Sensor Networks", Proc. *ACM International Workshop on Mobility Management and Wireless Access (MobiWac 2006)*, Torremolinos (Spain), October 2006.

[15] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, "Energy Conservation in Wireless Sensor Networks: a Survey", *Ad Hoc Networks*, Vol. 7, N.3, May 2009 (available at http://dx.doi.org/10.1016/j.adhoc.2008.06.003.

[16] H. Karl and A. Willig, "Protocols and Architectures for Wireless Sensor Networks", Chapter 10 (Topology Control), Wiley, 2005.

[17] P. Santi, "Topology Control in Wireless Ad Hoc and Sensor Networks", *ACM Computing Survey*, Vol. 37, N. 2, pp. 164-194, June 2005.

[18] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient Communication Protocol for Wireless Microsensor Networks", *Proc. HICSS-34*, January 2000.

[19] W. Ye, J. Heidemann, and D. Estrin, "An Energy-efficient MAC Protocol for Wireless Sensor Networks", *Proc. IEEE INFOCOM 2002*, New York, USA, June 23-27, 2002.

[20] K. Arisha, M. Youssef, M. Younis "Energy-aware TDMA-based MAC for Sensor Networks", *Proc. IEEE IMPACCT '02*, New York City (USA), May 2002.

[21] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks", *Proc. ACM SenSys 2003*, Los Angeles, USA, Nov. 2003.

[22] IEEE 802.15.4, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), 2006.

[23] V. Rajendran, K. Obracza, J. J. Garcia-Luna Aceves, "Energy-efficient, Collision-free Medium Access Control for Wireless Sensor Networks", *Proc. ACM SenSys 2003*, Los Angeles, USA, November 2003.

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

[25] W. Ye, J. Heidemann and D. Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks", *IEEE/ACM Transactions Netw*orking, Vol. 12 (3), Jun. 2004.

[26] G. Lu, B. Krishnamachari and C.S. Raghavendra, "An Adaptive Energy-efficient and Low-latency MAC for Data Gathering in Wireless Sensor Networks", *Proc. PDSP 2004*, April 2004.

[27] J. Li, G. Lazarou, "A Bit-map-assisted energy-efficient MAC Scheme for Wireless Sensor Networks", *Proc. Int'l Symp. on Information Processing in Sensor Networks (IPSN 2004)*, Berkeley, USA, 2004.

[28] C. Schurgers, V. Tsiatsis, M. B. Srivastava, "STEM: Topology Management for Energy Efficient Sensor Networks", Proc. *IEEE Aerospace Conference 2002*, Big Sky, USA, March 10-15, 2002.

[29] X. Yang, N. Vaidya, "A Wakeup Scheme for Sensor Networks : Achieving Balance between Energy Saving and End-to-end Delay", *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004)*, 2004.

[30] R. Zheng, J. Hou, L. Sha, "Asynchronous Wakeup for Ad Hoc Networks", *Proc. ACM MobiHoc 2003*, pp 35-45, Annapolis (USA), June 1-3, 2003.

[31] V. Paruchuri, S. Basavaraju, R. Kannan, S. Iyengar, "Random Asynchronous Wakeup Protocol for Sensor Networks", *Proc. of BROADNETS '04*, 2004.

[32] R. Jurdak, P. Baldi and C. V. Lopes, "Adaptive Low Power Listening for Wireless Sensor Networks", *Transactions on Mobile Computing*, Vol. 6 (8), pp. 988-1004, Aug. 2007.

[33] B. Hohlt, L. Doherty and E. Brewer, "Flexible Power Scheduling for Sensor Networks", *IEEE and ACM International Symposium on Information Processing in Sensor Networks*, April 2004.

[34] B. Hohlt and E. Brewer, "Network Power Scheduling for TinyOS Applications", *Proc. IEEE Int'l Conf. on Distributed Computing in Sensor Systems (DCOSS 2006)*, San Francisco (USA), 2006.

[35] I. Rhee, A. Warrier, M Aia, J. Min, "Z-MAC: a Hybrid MAC for Wireless Sensor Networks", *Proc. ACM SenSys 2005*, S. Diego, USA, November 2005.

[36] TinyDB: a Declarative Database for Sensor Networks, http://telegraph.cs.berkeley.edu/tinydb/

[37] S. Madden, M. Franklin, J. Hellerstein and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks", *Proc. OSDI*, 2002.

[38] Q. Cao, T. Abdelzaher, T. He, J. Stankovic, "Toward Optimal Sleep Scheduling in Sensor Networks for Rare Event Detection", *Proc. IPSN 2005*, April 2005.

[39] Y. Li, W. Ye, J. Heidemann, "Energy and Latency Control, in Low Duty-cycle MAC Protocols", *Proc. IEEE Wireless Communication and Networking Conference*, New Orleans , USA, March 2005.

[40] G. Lu, N. Sadagopan, B. Krishnamachari, A. Goel, "Delay Efficient Sleep Scheduling in Wireless Sensor Networks", *Proc. IEEE Infocom 2005*, March 2005.

[41] D. Mirza, M. Owrang, C. Schurgers, "Energy-efficient Wakeup Scheduling for Maximizing Lifetime of IEEE 802.15.4 Networks", *Proc. Int'l Conference on Wireless Internet (WICON'05)*, Budapest (Hungary), July 2005.

[42] A. Keshavarzian, H. Lee, L. Venkatraman, "Wakeup Scheduling in Wireless Sensor Networks", *Proc. ACM MobiHoc 2006*, Florence, Italy, May 2006.
[43] G. Anastasi, M. Conti, M. Di Francesco, "An Adaptive Sleep Strategy for Energy Conservation in Wireless Sensor Networks", *Technical Report DII-TR-2009-03*, http://info.iet.unipi.it/~anastasi/papers/DII-TR-2009-03.pdf
[44] S. Ping, "Delay Measurement Time Synchronization for Wireless Sensor Networks", *IRB-TR-03-013*, Intel Research Berkeley Lab, 2003.
[45] G. Anastasi, M. Conti, M. Castronuovo, M. Di Francesco, "Experimental Evaluation of an Adaptive Staggered Sleep Protocol for Wireless Sensor Networks", Proc. *IEEE WoWMoM 2008*, *International Workshop on Experimental Activities in Wireless Networks and Systems (ExpOnWireless 2008)*, Newport Beach (USA), June 23, 2008.
[46] Network Simulator Ns2, http://www.isu.edu/nsnam/ns.
[47] J. Zheng, M. J. Lee, "A Comprehensive Performance Study of IEEE 802.15.4", *IEEE Press Book*, 2004.
[48] G. Anastasi, E. Borgia, M. Conti, E. Gregori and A. Passarella, "Understanding the Real Behavior of 802.11 and Mote Ad hoc Networks", *Pervasive and Mobile Computing*, Vol. 1, N. 2, 2005.
[49] A. Willig, M. Kubisch, C. Hoene, and A. Wolisz, "Measurements of a Wireless Link in an Industrial Environment using an IEEE 802.11-compliant Physical Layer," *IEEE Transactions on Industrial Electronics*, Vol. 49, No. 6, December 2002.
[50] D. Brevi, D. Mazzocchi, R. Scopigno, A. Bonivento, R. Calcagno, and F. Rusina, "A methodology for the Analysis of 802.11a Links in Industrial Environments", *Proc. WFCS 2006*, pp. 165–174.
[51] E. Tanghe, W. Joseph, L. Verloock, L. Martens, H. Capoen, K. V. Herwegen, and W. Vantomme, "The Industrial Indoor Channel: Large–scale and Temporal fading at 900, 2400 and 5200 mhz,", *IEEE Transactions on Wireless Communications*, Vol. 7, no. 7, July 2008.
[52] A. Willig, "Polling-based MAC Protocols for Improving Real-Time Performance in a Wireless Profibus", *IEEE Transactions on Industrial Electronics*, Vol. 50, No. 4, August 2003.
[53] F. De Pellegrini, D. Miorandi, S. Pitturi, A. Zanella, "On the Use of Wireless Networks at Low Level of Factory Automation", IEEE Transactions on Industrial Informatics, Vol. 2, N. 2, May 2006.
[54] J. Li, P. Mohapatra, "Analytical Modeling and Mitigation Techniques for the Energy Hole Problem in Sensor Networks", *Pervasive and Mobile Computing*, Vol. 3, N. 3, pp: 233-254 , June 2007.
[55] S. Madden, "The Design and Evaluation of a Query Processing Architecture for Sensor Networks", UC Berkeley Ph.D. Thesis, 2003.
[56] Tmote Sky Platform, MoteIV Corporation, http://www.moteiv.com/products/tmotesky.php
[57] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, "System Architecture Directions for Networked Sensors", *SIGPLAN* Not. 35, 11, pp. 93-104, Nov. 2000.