An Adaptive Sleep Strategy for Energy Conservation in Wireless Sensor Networks

Giuseppe Anastasi, Marco Conti, Mario Di Francesco

***Abstract* - In recent years, wireless sensor network deployments for real life applications have rapidly increased. However, energy consumption still remains the main limitation of this technology. As communication typically accounts for the major power consumption, transceiver activity should be minimized, in order to prolong the network lifetime. To this end, we have developed an Adaptive Staggered sLeep Protocol (ASLEEP) for efficient power management in wireless sensor networks targeted to periodic data acquisition. The proposed protocol dynamically adjusts nodes' sleep schedules 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 and measurements in a real testbed. In both cases, the results obtained show that, under stationary conditions, the algorithm effectively reduces the energy consumption of sensor nodes, by dynamically adjusting their duty-cycle to the current traffic needs, thus increasing significantly the network lifetime. With respect to similar non-adaptive solutions, it also reduces the average message latency and increases the delivery ratio. Under time-varying conditions the algorithm is able to react quickly and adapt the duty-cycle of single nodes to the new operating conditions while keeping a consistent sleep schedule among sensor nodes.**

I. INTRODUCTION

A wireless sensor network (WSN) consists in a large 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, humidity, vibrations, accelerations) and process the acquired data locally, or send them to one or more collection points, usually referred to as sinks or base stations. Hence, WSNs can be viewed as an intelligent distributed instrumentation, which can be effectively used in many different contexts [1].

The set of potential WSN applications is extremely large. However, monitoring applications can particularly benefit from sensor networks as they allow a long-term data collection at scales and resolutions that are difficult, if not impossible, to achieve with traditional techniques [29]. Typically, data loggers are used to collect data. These devices are large and expensive, and require a recording and analysis equipment in place. This not only limit the accuracy of the recorded data, but can also disturb the natural behavior of the observed phenomena. In addition, these devices consume a lot of energy and require frequent human interventions. Instead, sensor nodes are very small and cheap devices so that they can be employed for fine-grained data collection. They can self organize into sensor networks and cooperate to perform an assigned task without human intervention. This makes WSNs particularly suitable for scenarios where the human presence is dangerous or impossible (i.e. a chemically contaminated field). In addition, the ease of deployment and the ability of unattended operations make WSNs capable to replace ordinary instrumentation also in industrial, medical and agricultural scenarios.

In recent years the number of sensor network deployments for real-life applications has rapidly grown up and, based on recent studies [33] [34], this trend is expected to increase dramatically in the next years, mainly in the fields of logistics, automation and control. However, energy consumption still remains the main obstacle to the diffusion of this technology, especially in application scenarios where a long network lifetime is required. The key issue is that sensor nodes are generally powered by batteries which have limited capacity and, often, cannot be replaced nor recharged, due to environmental or cost constraints. Therefore, efficient energy conservation strategies must be devised at sensor nodes in order to prolong the network lifetime.

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 [38]. On the other hand, it consumes significantly less power when it is put in 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 each other by agreeing on an appropriate sleep schedule in order to make communication feasible and efficient [11].

Unfortunately, designing efficient duty-cycling schemes is not straightforward. First, duty-cycling introduces additional

This work is funded partially by the European Community in the framework of the Memory project, and partially by the Italian Ministry for Education and Scientific Research (MIUR) in the framework of the FIRB ArtDeco project.

Giuseppe Anastasi is with Dept. of Information Engineering, Univ. of Pisa, Italy, (e-mail: g.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)

delays in the message delivery process, as messages cannot be transmitted until the next hop wakes up. Moreover, latency requirements are highly dependent on the application. For example, object tracking or event detection require quick response to the observed phenomena, 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. environmental monitoring [29], [45]), 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 any sensor platform. Second, it is able to quickly adapt 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, hence in a longer network lifetime as well.

A preliminary simulation analysis, carried out in a few representative scenarios, has shown that the performance of ASLEEP is very promising [4]. In this paper we extend the analysis to more general scenarios where sensor nodes are randomly deployed. In addition, we present a detailed analysis of the performance during steady state operations. Our findings are also supported by some experimental results obtained from measurements carried out in a real testbed [3]. Both simulation and experimental 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 as well. 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 introduces the reference system model and outlines the main design principles. Section IV describes the ASLEEP protocol. Section V presents the simulation setup and the results. Section VI describes a prototype implementation of ASLEEP in a real testbed and compares experimental and simulation results. Finally, Section VII concludes the paper.

II. RELATED WORK

A very large number of energy conservation schemes for WSNs have been proposed in the last years. The reader can refer to [5] for a detailed survey on the most relevant proposals. In the following we will focus on duty-cycling, i.e., techniques aimed at minimizing the energy consumption of a sensor node by adaptively switching off the radio subsystem during idle times.

According to [11], *duty-cycling* can be achieved through two different (and complementary) approaches: *topology control* and *power management*. Topology control protocols [8], [9] exploit nodes' redundancy and adaptively activate the minimum subset of nodes which allow network connectivity. Nodes which 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 [20] and [41] for detailed surveys on topology control protocols. However, even nodes selected by the topology control 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. Power management protocols are aimed at coordinating the sleep periods of neighboring nodes appropriately, so as to allow communication even in presence of a very low duty-cycle. As sensor nodes' activity is typically very limited, power management protocols can reduce the energy consumed by a node to some percent (with respect to the case without power management), thus increasing significantly the network lifetime.

In the following we will focus on power management 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). The first kind of approach allows to optimize medium access functions based on the specific sleep/wakeup pattern used for power management. On the other hand, general sleep/wakeup protocols permit 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 the two classes of power management techniques below.

II.A MAC protocols with a low duty-cycle

Several common MAC protocols for wireless sensor networks include a duty-cycling scheme for energy saving. For example, B-MAC (Berkeley MAC) [37] defines a duty-cycle through a channel sampling technique called Low Power Listening (LPL) and based on messages with long preambles. Nodes wake up periodically (i.e., every *check interval*) to check the channel for activity, and remain awake as long as the channel is being used. The preamble duration of each B-MAC frame is at least equal to the duration of the check interval, so that each node can always detect an ongoing transmission when it wakes up. The S-MAC (Sensor-MAC) protocol [49], instead, defines a distributed schedule dissemination scheme in order to form virtual clusters, i.e. set of nodes which agree on the same sleep schedule. The channel access time is split in a *listen period* (nodes exchange *sync* packets to coordinate their sleep/wakeup periods and

special control packets for collision avoidance) and a *data transfer* period. Nodes not concerned with the communication process can sleep until the next listen period. The IEEE 802.15.4 MAC protocol [17] supports a beacon-enabled mode based on a superframe structure. Each superframe consists of an *active period* and an *inactive period*. In the active period sensor nodes communicate with the coordinator node they associated with. During the inactive period nodes enter a low power mode to save energy.
All the above energy conservation schemes are static. An improvement over the aforementioned solutions is represented by MAC protocols with an adaptive duty-cycle scheme. T-MAC (Timeout MAC) [10], provides a timeout-based mechanism to tailor the activity period of nodes to their actual needs. Also a refinement of S-MAC proposed in [50] includes an adaptive listening mechanism which can adapt the activity of nodes to sudden changes in the network traffic. Although they present a basic form of adaptation, these protocols suffer from an additional latency in message forwarding. This sleep latency is introduced because a forwarding node has to wait until the next hop wakes up before transmitting a message, and it increases with the number of hops. This is a problem common to all protocols operating at the MAC layer. The most effective way to overcome this issue consists in exploiting upper layer information as in DMAC [25], which is an adaptive duty-cycle MAC protocol optimized for sensor-network routing trees. DMAC exploits the knowledge of the topology in order to stagger nodes' schedules according to their position in the routing tree. However, to the best of our knowledge, it has not been implemented on real sensor platforms.
Another approach for implementing duty-cycling at the MAC layer consists in using a TDMA (Time Division Multiple Access) approach for channel access [4], [12], [13], [22], [39]. In such schemes time is divided in slots that are assigned to nodes according to a certain algorithm. Slotted schemes are inherently energy efficient as nodes remain active only during slots assigned to them. On the other hand, they have a number of drawbacks that limit their usage in real WSNs [40]: (i) they lack *flexibility* in adapting to topology changes caused by time-varying channel conditions, physical environmental changes, nodes that run out of energy, and so on; (ii) they have limited *scalability*, (iii) they require tight *synchronization* among network nodes, which introduces a overhead in terms of control-message exchange and, thus, additional energy consumption, (iv) finding an *interference-free* schedule is a very hard task since interference ranges are typically larger than transmission ranges, i.e., many network nodes may interfere even if they are not in the transmission range of each other [2].
In conclusion, duty-cycled MAC protocols 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 an actual sensor platform. In addition, it might be unable to exploit information made available by the application. On the other side, general sleep/wakeup protocols are more flexible as they can be used on top of different MAC protocols. In addition, they can better exploit application-specific information.

II.B General sleep/wakeup schemes

We can broadly classify general sleep/wakeup schemes into three main categories: *on demand*, *asynchronous* and *scheduled rendezvous* schemes. *On-demand* schemes assume that destination nodes can be awakened somehow just before receiving data. To this end, two different radio transceivers are typically used [42], [48]. The first radio (*data radio*) is used during the regular packet 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 [19], [35], [48], [51]. In this case a node can just wakeup whenever it wants and still be able to communicate with its neighbors. Although being robust and easy to implement, asynchronous schemes generally present high latency in message forwarding and have problems with broadcast traffic. The last class of general sleep/wakeup protocols is represented by *scheduled rendezvous* schemes, which require that nodes are synchronized and all neighboring nodes wake up at the same time. Our ASLEEP protocol belongs to the last category.
By focusing on scheduled rendezvous schemes, a possible approach consists in establishing a coarse-grained TDMA 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) [15], [16], which includes an on-demand reservation mechanism capable to dynamically adapt to traffic demands. Since slots are relatively large a strict synchronization among nodes is not required. However, FPS borrows some drawbacks [40] from TDMA schemes, i.e. it has limited scalability and flexibility in adapting to traffic and topology changes.
Most solutions following a scheduled rendez-vous approach use plain duty-cycle based schemes. For instance, the well-known TinyDB query processing system [46] include 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) [27], 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 in the routing tree. Nodes located at different levels of the routing tree wake up at different, progressive times, like in a pipeline. Due to its nice properties this scheme has been considered and/or analyzed in many subsequent papers ([7], [24], [25], [26], [31] among others). In particular, the authors of [21] analyze the performance of the TAG staggered approach as well as several its variants.
Although providing 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 networks. This constraint simplifies the coordination among nodes, but results in low energy efficiency and high message latency. Like TAG, our proposal leverage a staggered approach. However, in our proposal nodes' active periods are dynamically adapted to the observed network conditions and can be 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. NETWORK MODEL AND DESIGN PRINCIPLES

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 used for data forwarding. This is very common in practice as many popular routing protocols rely on a routing tree [18], [25], [27], [30], [43], [46]. The routing tree may change over time due to link failures, node failures, or nodes running out of energy. Also, it may be re-computed periodically to better share energy consumption among nodes. However, as nodes are assumed to be static, we will assume that the routing tree – once established – remains stable for a reasonable amount of time. Note that ASLEEP do not require a specific algorithm to build the routing tree.

In the above network model, nodes can achieve both low energy consumption and low latency in transferring data to the sink if their active periods are *staggered* according to the position along the routing tree. Nodes at different tree levels wake up at progressive times starting from leaf nodes and proceeding upward to the root (i.e., sink). This optimizes the latency experienced by messages flowing from source nodes to the sink [21]. In [21] it is shown that messages delivered to the sink experience a maximum latency given by the sum of active periods of all traversed nodes, while messages flowing in the opposite direction can traverse only one level per data reporting period.

However, to minimize the energy consumption and message latency in the above staggered scheme, the following design principles should also be taken into account.

- *A sensor node should remain active for the minimum time required for receiving data from all its children and forwarding them to its parent.* According to [21], this minimizes both energy consumption and message latency. Since the optimal active period depends on the network operating conditions, it should be adjusted dynamically.
- *The optimal active period should be chosen on an individual basis.* As sensor nodes located at different locations typically handle different amounts of traffic and experience different network conditions, the optimal active period should be tailored to each individual sensor node.
- *The algorithm for deciding the node's active period should be local and simple.* This further reduces energy consumption at sensor nodes, because a global algorithm would require the exchange of information among nodes. In addition, a complex algorithm might not be suitable for sensor nodes with limited computational capacity.
- *Protocol operations should ensure a consistent network-wide schedule when local conditions change.* A variation in the active period of a single sensor node should not compromise the correctness and energy efficiency of the global schedule. As a consequence, a robust cooperation mechanism is required for nodes to manage the network-wide sleep schedule.

Our adaptive sleep protocol described below addresses all the above design principles. In fact, it is based on a staggered scheme, and allows each single sensor node to adjust dynamically its own active period based on local measurements of the current network activity. Finally, it includes a sleep coordination algorithm to re-organize the global sleep schedule when a variation in the talk interval of a node occurs.

IV. 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* each node uses for dynamically estimating its expected active period, and the *sleep coordination algorithm* used to propagate the new sleep schedule throughout the network. Finally, we will describe some optimizations to improve the robustness and the efficiency of the protocol.

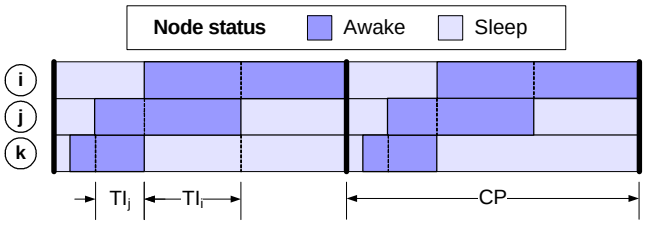


Figure 1. Sleep scheduling protocol parameters.

IV.A Protocol Overview

As mentioned in Section III, sensor nodes are assumed to form a logical *routing tree* (rooted at the sink) for data forwarding, which is recomputed periodically. Nodes' clocks are assumed to be synchronized by some time synchronization protocol (e.g. [36]). The communication between a parent and its children occurs in *communication periods* (CPs) that repeat periodically. Each communication period includes an *active interval* (AI) during which nodes communicate by using the underlying MAC protocol, and a *silence interval* (SI) during which nodes turn their radio off to save energy. As shown in Figure 1, active intervals are staggered so that nodes at lower levels in the routing tree wake up earlier than their ancestors. Each (intermediate) sensor node spans its activity over two adjacent *talk intervals* (TI), the first one with its children and the other one with its

parent[1]. Throughout, we will refer to the talk interval shared by a generic node j and all its children, during the *m-th* communication period CP^m, as TI_j^m.

The duration of the talk interval to be shared with children in the next communication period is dynamically estimated by each parent node, according to the algorithm described in Section IV.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 needs to 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 IV.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 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 – at any time during the talk interval. As direct beacon messages are critical for correctness, ASLEEP also includes mechanisms to (**i**) increase the probability of successful reception during direct beacon transmissions, and (**ii**) enforce a correct (even if non-optimal) behavior of nodes in case they miss a direct beacon. These mechanisms will be discussed in Section IV.D. We anticipate here that, to increase the probability of successful reception, direct beacons are transmitted in a specific time interval reserved only for direct beacon transmission (Beacon Period).

IV.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 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. For example, the user can query the maximum value of temperature, to be collected every two minutes. In this case the communication period is set to two minutes by each sensor node. 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 number of messages to be received, underlying MAC protocol, channel conditions, degree of contention, and so on. Furthermore, the number of messages to be received depends on the number of children, the message generation rate at source nodes, and the network topology.

From the above discussion it clearly emerges that computing the ideal talk interval would require the global knowledge of the network. Moreover, this value should be continuously updated as the network topology and operating conditions change over time. Since such an approach is not practical, we propose here an adaptive technique that approximates this ideal scheme. Our approach lets every parent node choose 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 minimize the message loss probability.

The above time interval should be increased appropriately 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 $TI_{est}^{m+1} = ceil((\overline{\Delta} \cdot n_{pkt}^{max} + BP)/q) \cdot q$, where BP denotes the *Beacon Period*, i.e., a time interval reserved for beacon transmission only (see Section IV.D.1), and q denotes the time slot. The q value should be chosen as a trade-off between efficiency and stability. From one hand, a low q value 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 q value 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 phases during which it had no traffic to send.

Advertising TI_{est}^{m+1} to children as the next talk interval might lead to some flapping of the protocol parameters. To smooth the variation of estimates, the talk interval for the next communication period, g_1 is determined as follows. If $TI_{est}^{m+1} - TI^m \geq g_1$ then TI^{m+1} is immediately set to the estimated value (i.e., $TI^{m+1} = TI_{est}^{m+1}$). If the predicted talk interval is below the current value and the difference is greater than, or equal to a guard threshold $g_{down} \geq 2q$ (i.e.,

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

$TI^m - TI_{est}^{m+1} \geq g_{down}$) then the talk interval is decreased by just *one* time slot ($TI^{m+1} = TI^m - q$). Finally, if $0 < TI^m - TI_{est}^{m+1} < g_{down}$, the talk interval is not immediately decreased. When the same condition persists for a number L_{down} of communication periods, then the talk interval is reduced anyway.

According to the above rules, an increase in the talk interval is managed less conservatively than a decrease. This is because an aggressive increase tends to minimize the probability that a node can miss messages from its children.

IV.C Sleep Coordination

As anticipated in Section IV.A, 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 node's talk intervals – 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 CP^{m+1};
- its next wakeup time $t_{parent,j}^{m+1}$;
- the length of the next talk interval to be shared with its children (TI_j^{m+1}).

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 the Appendix A for a in-depth description of the algorithm and its pseudo-code. In the following, we will describe the main operations performed during talk interval changes. There are two options: a talk interval increase or a talk interval reduction. 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 the 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 ensure that the new (transient) schedules are consistently handled during transient schedule propagation. 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 present in steady state conditions. Further details are given in the following discussion.

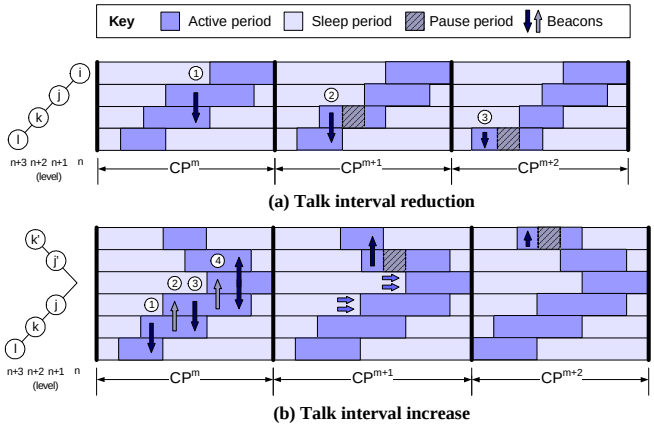


Figure 2. Talk interval adaptation examples

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 2a. 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 intervals with its children, i.e., nodes at the (*n+2*)-th level (including *k*) . In the *m-th* communication period node *j* announces the next talk-interval duration to its children by means of the direct beacon (1). The children receive the direct beacon and wait for the next communication period CP^{m+1} to inform their children – i.e., nodes at the (*n+3*)-th level – of the new scheduling parameters (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 (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 communication. To this end, the node makes use of reverse beacons, which are 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 2b will help us understanding. Let's 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 (1). Second, in the same communication period, the node sends the reverse beacon (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 (3), to its children. Because all ancestors have to shift their talk interval, node *j* also propagates the reverse beacon (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**) nodes' clocks are properly synchronized, and (**ii**) direct and reverse beacons never get lost, the following properties hold (the corresponding proofs are given in Appendix B):

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 couple of nodes* i *and* j *such that* j *is a child of* i*, the talk intervals* TI_i *and* TI_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.

The above properties hold under assumptions (**i**) and (**ii**). Clock synchronization is beyond the scope of this paper and can be achieved though any available clock synchronization protocol, e.g. the protocol described in [36]. Note that, ASLEEP operates on top of the MAC layer and use 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 devised a *Beacon Protection* mechanism to increase the probability of successful beacon transmission, and a *Beacon Loss Compensation* mechanism to offset the negative effects of beacon losses. We show below that, thanks to the latter mechanism, ASLEEP is able to maintain a correct schedule, even in the presence of beacon losses, at the cost of a decreased performance in terms of energy efficiency and delivery ratio.

IV.D Schedule robustness

The this section we describe the additional mechanisms devised for improving the protocol robustness.

1) Beacon Protection

Beacon messages are critical for the protocol correctness. When a node misses a direct beacon containing the new parameters, it cannot schedule its activity for the next communication period. In addition, until re-acquiring the correct schedule information, the node cannot send direct beacons to its children. As a consequence, the loss of coordination propagates along the routing tree to the descendants of the node which is out of coordination.

Direct beacons may get lost, for example, due to collisions with other beacons or regular messages transmitted by interfering nodes. In addition, as direct beacons are broadcast messages, mechanisms typically used for enhancing the communication reliability (i.e. retransmissions) cannot be used. Reverse beacons may be lost as well. However, as they are unicast messages, they are retransmitted by the MAC protocol.

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 copies of the direct beacon are transmitted back to back.

2) Beacon Loss Compensation

The Beacon Protection mechanisms increases the probability that a direct beacon is successfully received by its children, but does not eliminate the problem of beacon losses. Therefore, we have also devised the following compensation mechanism to overcome the negative effects of a direct beacon loss. Since talk intervals typically remain constant for a number of communication periods, nodes use the current schedule parameters also for the next communication period when they miss a direct beacon. The predicted value is used only for a single communication period, in which nodes expect to correctly receive a fresh beacon from their parent. If

this is not the case, the node remains awake until it re-acquires a fresh beacon.

Obviously, this heuristic produces a correct schedule if the parent node has not varied 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 whether the talk interval has been increased or decreased. If a parent has reduced its talk interval, the child node just wakes up earlier than the correct instant, thus wasting some energy and increasing the message loss probability. However, it 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.

There is also the possibility of incorrect schedules due to the loss of a reverse beacon. In this case the child node may wake up after the talk interval with the 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 lost.

V. SIMULATION ANALYSIS

To evaluate the performance of the ASLEEP protocol, we implemented it in the ns2 simulation tool [32]. We split the analysis in two distinct parts. In the first part we investigated how the protocol reacts to changes in the operating conditions. To this aim, we varied dynamically the network topology/traffic pattern and analyzed the response of the protocol. In the second part we analyzed the performance of ASLEEP in steady state conditions.

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 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 environmental monitoring applications.

As the ASLEEP protocol relies on a routing tree, we implemented the following simple routing tree formation algorithm in our simulator. The sink starts the tree-building phase setting itself as the root and broadcasting an HELLO message to neighboring nodes. Upon receiving an HELLO message, a node sets the source node as its parent, waits for a random backoff time, and forwards the HELLO message with an updated hop count. Nodes receiving HELLO messages from several nodes, choose the one with the minimum hop count as their parent.

TABLE 1. OPERATIONAL PARAMETERS

Parameter	Value
Communication Period (CP)	30s
Message rate	1 msg/CP
Message size	20 bytes
Observation window (L)	10 CPs
TI time slot (q)	100 ms
Beacon Period	60 ms
TI decrease time threshold (L_{down})	5 CPs
TI increase threshold (g_{down})	2q (200 ms)

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 both the maximum number of backoffs and retransmissions to 8, in order to increase the probability of successful message transmission. The radio propagation model was two-way ground; the transmission range was set to 15 m (according to the settings in [52]), while the carrier sense range was set to 30 m (according to the model presented in [2]). We carried out a preliminary simulation analysis to tune parameters such as the length of the timeslot (q), the observation window size (L), and the adaptation thresholds (L_{down} and g_2). Unless stated otherwise, we used the simulation 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, 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.

V.A Analysis in dynamic conditions

To investigate the ASLEEP behavior in dynamic conditions we considered the following two kinds of variations in the operating conditions.

- *Traffic pattern variation*. In this set of experiments sensor nodes start generating one message per communication period. Then, after some time, they increase the message rate to 3 messages per communication period, and finally they switch back to the original value. This scenario may occur when sensors are requested to report an event with better fidelity (i.e., including additional physical quantities or using a larger number of bits per sample) for a limited time.
- *Topology variation*. These experiments start with an initial configuration where only one half of the nodes deployed

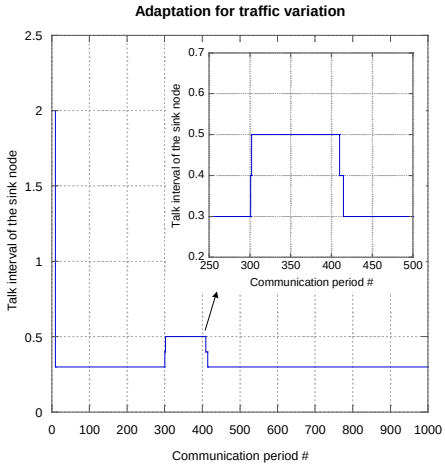


Figure 3. Talk interval adaptation for traffic variations

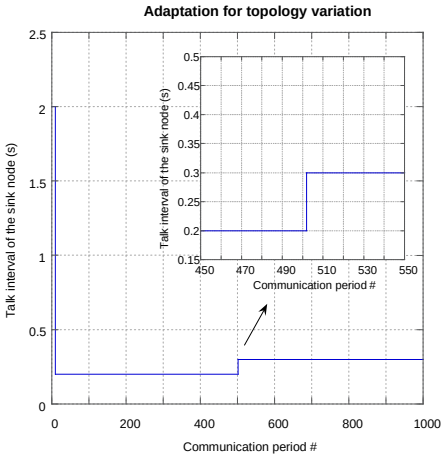


Figure 4. Talk interval adaptation for the topology variation

TABLE 2. TRANSIENT TIME FOR TRAFFIC VARIATION (MEAN AND STD DEV)

Metric	Transient (CPs)
Up transient (CPs)	4.1 (3.0)
Down transient (CPs)	15.8 (2.2)

TABLE 3. TRANSIENT TIME FOR TOPOLOGY VARIATION (MEAN AND STD DEV)

Metric	Transient (CPs)
Up transient (CPs)	1.3 (0.5)

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 considered the following performance metrics:

- *Talk interval*. Plotting the talk interval duration over time provides a graphical representation of the protocol's ability to adapt to changing operating conditions.
- *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.

1) Results

In the first set of experiments we varied the traffic pattern. All nodes started generating 1 message per communication period. After the 300th communication period, the rate increased to 3 messages and, finally, after the 400th communication period, it reverted back to the initial value.

Figure 3 shows the talk interval shared by the sink and its children as a function of time, referred to a specific topology (the trend is similar for the other topologies as well). 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 networks (the overall activity time of an intermediate node consists of the talk intervals with its children and parent).

The trend of the talk interval shown in Figure 3 deserves a detailed discussion. Initially, there is a sharp decrease in the talk interval. This is because the talk-interval prediction algorithm takes an observation window L before providing the first estimate. During this preliminary phase, a dummy talk interval is used so that nodes can refrain from being always-on. In our experiments we used an initial talk interval value of 2 seconds. Once the first prediction is available (L =10 in our experiments), the estimated value is set and, as shown in Figure 3, remains constant until the message generation increases to 3 messages per communication period. Then, the talk interval quickly adapts to new traffic conditions, i.e. it just takes two communication periods to reach the stability. The newly selected talk interval remains stable again for about one hundred communication periods. Then, it drops to a value which is one slot greater than that before the variation in message generation rate. Finally, after an additional number L_{down} of communication periods, it reverts back to the original value. This additional time is due to the heuristic used for estimating the talk interval, which is affected by the adaptation thresholds. Recall that, due to the heuristic we use (cfr. Table 1), the talk interval is decreased by one slot when the estimated talk interval is two slots lower than the previous value.

Table 2 shows the duration of the up and down transient time originated by the traffic variation, averaged over the different topologies. We can see that the up transient spans over about 4 communication periods. This is strictly related to data propagation process in a staggered scheme. To reach the sink, the new schedule parameters have to "climb up" the tree one level at 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 (recall that the talk interval is estimated based on statistics accumulated over the last L communication periods, and 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, initially only one half of the deployed sensor nodes (i.e., 15 nodes) is active and send messages to the sink. The other 15 nodes become active only starting from the 500th communication period. Figure 4 shows the variation over time in the talk interval shared by the sink and its children in a representative simulation run. Again, the resulting behavior matches the expected variation, although in this case the variation is limited. Nevertheless the figure shows the adaptive mechanism is effective in this scenario as well. We can see that the raising transient time is lower than in the previous experiment (Table 3). This is because the increase in the resulting traffic conditions is limited, so that the additional messages are likely to fit in the last (unused) part of the current talk interval. As a consequence, the incoming messages almost immediately trigger the adaptation.

V.B Analysis in stationary conditions

In this section we assess the performance of ASLEEP under stationary operating conditions, and compare it against other (non-adaptive) similar 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 never used in practice and is considered here only for comparison purposes.
- *TAG-like staggered scheme*. Sensor nodes uses a staggered scheme for sleep/wakeup period. The talk interval is fixed and equal for all sensor nodes. It is set to the value of used, for example, in TAG [27] where the talk interval is calculated as the communication period divided by the depth of the routing tree. This is the same rule as in TAG [27]. Therefore, throughout we will refer to such a scheme as *TAG*.
- *Fixed staggered scheme*. In this scheme the talk interval is fixed and equal for all nodes, like in the TAG scheme above. However, we used a different rule to calculate the value of the talk interval. Ideally, it should be set to the minimum time needed by any parent node to correctly receive all messages coming from its children. In practice the ideal value cannot be known in advance, so that this scheme is thus unfeasible. In our experiments, we set the talk interval by using the value estimated by our ASLEEP protocol. In detail, we first ran an experiment with our protocol and measured the maximum talk interval in that configuration. Then, we replaced our protocol with the fixed staggered scheme and set all talk intervals to the maximum value measured before. The rationale is to investigate the performance of a fixed staggered scheme when using the same parameters as ASLEEP. For brevity, in the following this scheme will be referred to as *Fixed.*

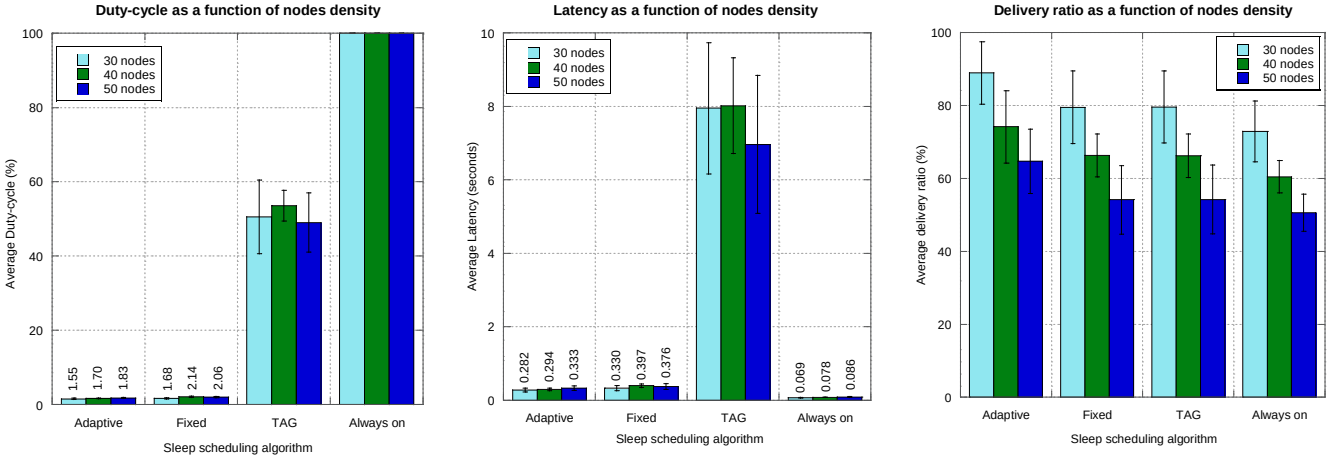
In all staggered schemes (TAG, Fixed and ASLEEP) messages are assumed to be generated just before the beginning of the talk interval. To make the comparison fair, especially in terms of message latency, when using the Always-on scheme we assumed that messages are generated at the same time instants as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

To compare the performance of our protocol 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 [23].
- *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 between the message generation time at the source node and the reception time of the same message at the sink.

1) Results

We started considering the basic scenario whose parameter settings have been specified at the beginning of Section V (basically, 30 nodes, 20-byte messages and no data aggregation). Then, we varied the node density, applied different data aggregation schemes, and investigated the influence of each 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.



(a) (b) (c)

Figure 5. Average Duty-cycle (left), average message latency (middle), and delivery ratio (right) for different nodes densities.

a) *Performance comparison in the basic scenario*

Table 4 summarizes the results obtained in the basic scenario. In terms of average duty-cycle, ASLEEP and Fixed perform much better than TAG. This is because in TAG the talk interval duration 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%. As expected, ASLEEP and Fixed schemes achieve approximately the same average duty-cycle. This is because the Fixed scheme sets the talk interval of *all nodes* to the maximum value measured when using ASLEEP. This also explains the slight discrepancy in favor of ASLEEP. We now clarify the impact of the energy consumption on the network lifetime. We consider the model used in [28] and adapted to the parameters of the Tmote Sky device [44], 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). With the Always-on scheme, the network lifetime is of about 6 days, which is not satisfactory for a long-term deployment. With ASLEEP, the network lifetime can be extended to about 390 days, which allows measurements to last for more than one year, making long-term deployments actually possible. There is also an improvement of about one month over the 360 days achievable with the Fixed scheme.

In terms of message latency, we can observe that 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. In fact, the average latency is in the order of ten seconds, which roughly corresponds to the ratio between the communication period and the tree depth. On the other hand, the Always-on scheme introduces a minimum latency as messages are forwarded as soon as they are received. The overall latency is thus limited to the sum of delays introduced for messages transmission in the various hops from the source to the sink. Finally, ASLEEP introduces lower latency than the Fixed scheme because it has the extra flexibility of tailoring each talk interval to the actual needs of individual nodes. It is important emphasizing that, in comparison to the Always-on scheme, in the basic scenario ASLEEP reduces the node activity by approximately 98%, while introducing only a moderate additional latency (in absolute value).

TABLE 4. PERFORMANCE IN THE BASIC SCENARIO (MEAN AND STD DEV).

	Average duty-cycle (%)	Average Latency (ms)	Delivery Ratio (%)
ASLEEP	1.55 (0.17)	282 (51)	88.9 (8.6)
Fixed	1.68 (0.20)	330 (68)	79.6 (10.0)
TAG	50.57 (9.92)	7951 (1788)	79.6 (9.8)
Always ON	100.00 (0.00)	69 (10)	72.9 (8.3)

Another important property of ASLEEP can be highlighted by looking at the delivery ratio. Our protocol exhibits the highest value among all schemes. In other word, ASLEEP delivers the highest percentage of messages while allowing the lower duty-cycle (i.e., energy consumption) at sensor nodes. This counter-intuitive behavior 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 the Fixed and TAG schemes because each node can set the talk interval with its children independently 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 of the collisions obtained by the TAG and Fixed schemes. Note that both TAG and Fixed have the same performance, which indicates that there is no clear benefit from keeping the 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.
To summarize, in comparison with non-adaptive staggered schemes, ASLEEP (i) minimizes the duty-cycle of sensor nodes, thus reducing their energy consumption; (ii) decreases the average message latency; and, (iii) increases the overall delivery ratio.

b) *Impact of node density*

In this section we evaluate the influence of the node density on the performance of the different 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)[2].
Figure 5a shows that, in general, the average duty-cycle tends to grow up with the 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). However, TAG has a different trend, which strictly depends on the mean depth of the routing tree. In fact, TAG uses a talk interval duration which is inversely proportional to the tree depth. While the mean routing tree depth – averaged over the 10 random topologies – is around 3.3 for the 30 and 50 nodes scenarios, it decreases to about 3.0 for the 40 nodes scenario[3]. This explains the higher energy consumption obtained by TAG when the network is formed by 40 nodes. As the message latency is tightly related to the talk interval (at least for staggered schemes) the above considerations can also be used to explain the impact of node density on the average message latency shown in Figure 5b. On the other hand, Figure 5c shows the delivery ratio achieved by the different schemes in the three scenarios under consideration. As expected, for all schemes the delivery ratio decreases when the node density grows up. This is mainly because the probability of correct delivery to the next hop reduces due to the increased collision probability. Figure 5c shows that ASLEEP always provides a delivery ratio significantly higher than the other schemes.

c) *Impact of message size*

To investigate the impact of the message size we varied the frame payload size in the range 10-100 bytes, while keeping the frame overhead fixed and equal to 20 bytes. The results obtained are summarized in Figure 6.
In principle, the message size may impact on the duty-cycle as larger message sizes result in a longer transmission times and, hence, in longer talk intervals predicted by the parent node. In practice, we observed that the increase in the estimated talk interval due to a larger message size is very limited. This is because the time taken for transmitting bits over the wireless medium is small compared to the total time required for sending a message. This includes additional delay components, such as queuing time before transmission, contention for channel access, collision resolution. Overall, these components predominate over the transmission time. However, the results in Figure6-left highlight that ASLEEP is able to adapt to small variations as well, so that its average duty-cycle is lower than that of the optimal fixed scheme. As in Figure 5, the TAG duty-cycle is not affected by the message size because of the rule used in this scheme to set talk intervals.
The average message latency increases with the message size, as expected. The trend is similar to the one observed in Figure 5b. Therefore, we decided to omit the related figure. Figure 6-right shows that the percentage of messages correctly delivered to the sink decreases when messages have larger sizes. This is because the channel wastage due to a collision is higher when messages have larger sizes. We can see that ASLEEP achieves a delivery ratio significantly larger than all other schemes, with any message size, due to its ability to reduce collisions. The optimal fixed and TAG schemes have the same delivery ratio as they experience the same collision probability (in both schemes nodes belonging to the same routing tree level start transmitting simultaneously).

d) *Impact of data aggregation*

Data aggregation is commonly used in sensor networks to reduce the amount of data (i.e., messages) to be forwarded to the sink. It may be worthwhile mentioning here that staggered schemes have been originally conceived to better support data aggregation. Therefore, it is extremely interesting to assess the impact of aggregation on the different sleep/wakeup schemes. In the following we will consider the following three aggregation schemes.

- *No Aggregation*. Each node forwards all received messages to its parent.
- *Min/Max Aggregation*. Each parent node aggregates all received messages into a single message of the same size.
- *Compact Aggregation*. Each parent node merges all received messages into a single data block. Then the merged data block is split across messages with maximum payload size. The message size depends on the specific MAC protocol that is used (127 bytes in our case). However, in general, messages obtained from aggregation have a larger size than messages originated by source nodes.

When enabling data aggregation, messages received by the sink contain data originated by different source nodes and, hence, message latency becomes meaningless. Therefore, in the following of this section we will just consider the average duty-cycle and delivery ratio. To derive the delivery ratio we assumed that, if a message is correctly received by the sink, all messages it aggregates are considered as correctly delivered.
In addition, we will limit our analysis to staggered schemes only, as data aggregation is performed in the same way in all such schemes, i.e., data are aggregated by a parent node at the end of the talk interval with its own children. Instead, in the *Always-on* scheme the performance is strongly dependent on

[2] In all plots shown in this and subsequent sections, we report values averaged over 10 simulation runs, each referring to a random network topology. Error bars represent standard deviations.

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

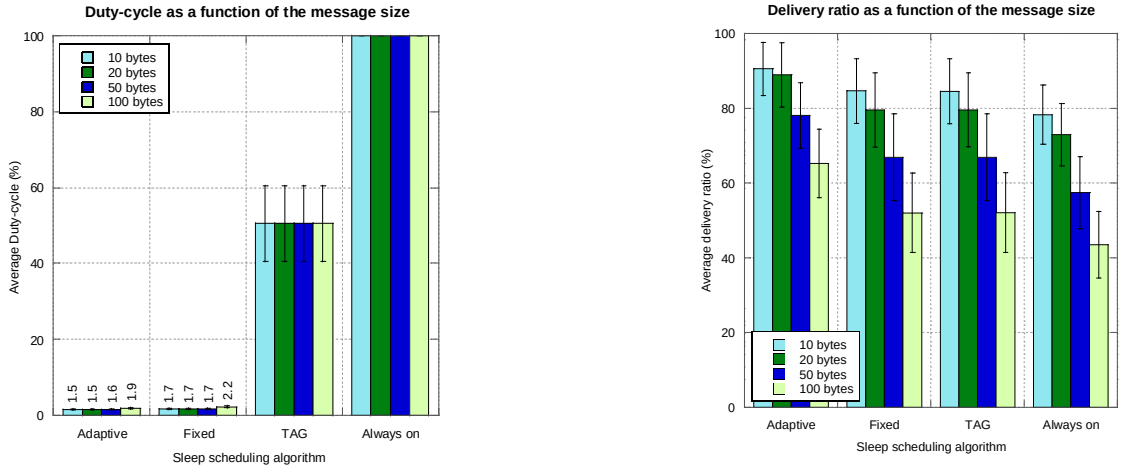


Figure 6. Average duty-cycle (left) and delivery ratio (right) for different message sizes.

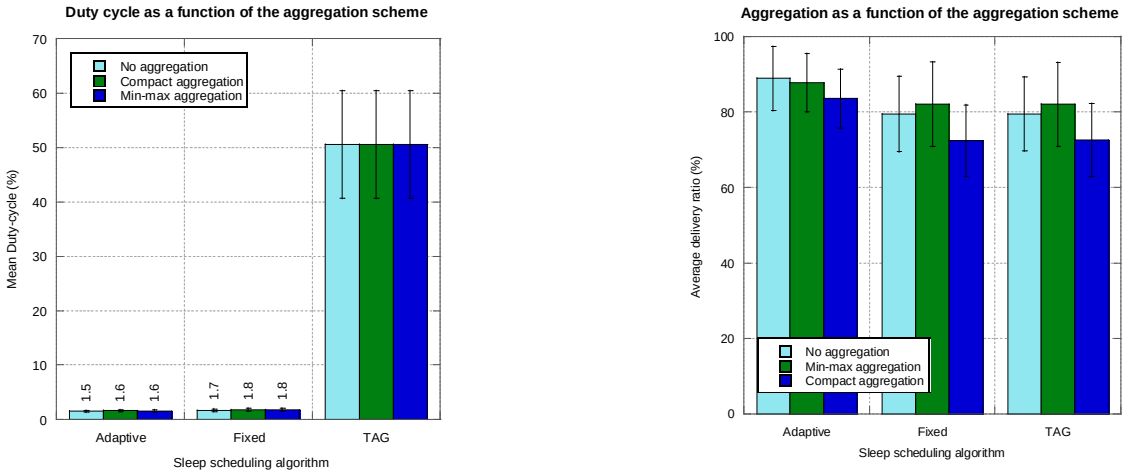


Figure 7. Average duty-cycle (left) and delivery ratio (right) different data aggregation schemes.

the time each parent node waits before aggregating data received from its children.

Figure 7-left shows that the (average) duty-cycle introduced by the different staggered schemes is not significantly influenced by data aggregation. For TAG this is because the length of the talk interval only depends on the routing tree depth. To understand the behavior of ASLEEP it may be worthwhile recalling here that the length of the estimated talk interval is expressed as an integer number of time slots. In the considered scenario the traffic reduction introduced by (Min/Max and Compact) aggregation is not enough to trigger a decrement of one slot in the estimated talk interval. Hence, the average duty-cycle remains constant. This also justifies the behavior of the Fixed scheme as its talk interval is set to the maximum value experienced by ASLEEP in the same scenario.

Figure 7-right shows the delivery ratio for the three aggregation schemes. We can observe that, irrespective of the aggregation scheme, ASLEEP provides a delivery ratio significantly higher than both Fixed and TAG (which get similar performance). In addition, it is less sensitive to the specific aggregation scheme. To better understand the results in Figure 7-right it is worthwhile to emphasize that data aggregation may have two different contrasting effects on the delivery ratio. On one hand, by reducing the number of messages to transmit, aggregation reduces the collision probability. This tends to increase the delivery ratio. On the other hand, data aggregation tends to decrease the delivery ratio as the loss of a message implies the loss of all data it aggregates. In addition, some aggregation schemes (e.g., the compact scheme), produces less messages but with increased size and, hence, with lower probability to reach the sink. Obviously, the actual delivery ratio results from the combination of all above effects.

In Figure 7-right the same performance achieved by Fixed and TAG can be easily explained by observing that in these schemes nodes behave in the same way, the only difference being the time interval nodes remain active. With respect to the *no-aggregation* case, the delivery ratio increases with Max/Min aggregation (there are less messages to manage), and decreases with Compact aggregation (the number of messages is not reduced significantly, and messages have larger size).

As far as ASLEEP, data aggregation does not provide a significant benefit in terms of delivery ratio. With respect to the no-aggregation case, we can observe a slight decrease with Max/Min aggregation, and more significant decrease with Compact aggregation. This is because the delivery ratio

provided by ASLEEP without aggregation is very high (almost 90%), i.e., the message loss probability is very low. With aggregation, message losses have a heavier impact on the delivery ratio. In addition, the Compact aggregation produces message of larger size.

In conclusion, from the above analysis it clearly emerges that, thanks to its ability to set the talk interval of each single node based on its real needs, ASLEEP not only reduces the average duty-cycle of sensor nodes and, thus, their energy consumption with respect to non-adaptive staggered schemes. It also reduce the average message latency and increases the delivery ratio. As a final remark, it should be pointed out that ASLEEP is more complex than non-adaptive staggered schemes as it requires a continuous coordination among nodes to maintain the network-wide sleep schedule. To this purpose, direct and reverse beacons are used, thus resulting in a communication overhead. The above results show that ASLEEP guarantees a lower duty-cycle despite this communication overhead.

VI. EXPERIMENTAL EVALUATION

Since simulation experiments might not take into account all factors that can occur in a real environment, we also carried out a set of experimental measurements. To this end we used a testbed consisting of 15 Tmote Sky [44] sensor nodes with TinyOS [14] operating system. The Tmote Sky platform 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.

VI.A Experimental Setup

We implemented ASLEEP in the TinyOS 1.1.15 operating system by following the protocol description given in Section IV (and Appendix A). We implemented our algorithm on top of the default CSMA/CA MAC protocol shipped with TinyOS[4] which is different from the IEEE 802.15.4 MAC protocol used in the above simulation experiments. This indicates that ASLEEP can be used with different MAC protocols. Before starting ASLEEP, we performed an initial time synchronization and we built the routing tree (these two operations are also repeated periodically). To this end we used a modified version of the synchronization protocol in [36], and the *MintRoute* tree-building scheme [47].

All experiments have been carried out in a vineyard. Tmote Sky sensor nodes were placed at about 50 cm from the ground. They periodically sampled the external temperature, and reported the acquired data to the sink node (node 0) which was connected to a laptop for data collection and analysis. We selected this application scenario as it is representative of a large set of monitoring applications based on periodic report of the sensed data.

When dealing with a real testbed one of the main difficulties is that experiments cannot be repeated exactly in the same way since external conditions may vary from time to time – sometimes during the same experiment – and there is no control on them. Therefore, successive experiments carried out with the same parameter settings may provide outcomes that differ from each other. To achieve more statistical accuracy, we replicated each experiment 5 times (each replica was 100 communication-period long). The results presented below are averaged over the entire set of 5 replicas. Standard deviations are also reported.

[4] We enabled MAC-layer acknowledgements and set the maximum number of retransmissions to 5.

VI.B Experimental Results

In the experimental analysis we focused on the steady state behavior of the protocol in the different scenarios. We used the same parameter settings shown in Table 1, with the following exceptions. We increased the slot time to 150 ms, due to timing constraints on the actual implementation. In addition, we set the message size to 38 bytes (28-byte payload plus 10-bytes overhead). Finally, to maintain the experiment duration within reasonable limits, we considered a communication period of 15s. However, we carried out additional experiments (not presented here) with larger communication periods that confirm the findings described in this section.

We started our experimental analysis by investigating the ASLEEP performance in the following three different scenarios.

- *All-in-Range Scenario*. All nodes are within the transmission range of node 0 which acts as the sink node. Specifically, as shown in Figure 8a, sensor nodes are deployed on a 5 by 3 grid, at a distance of approximately 10 m, and use the maximum allowed transmission power (which is needed to properly cover all nodes). Since the transmission range of all sensor nodes is larger than their distance from the sink, a star network topology is expected in this scenario, i.e., all nodes should become children of the sink during the routing tree formation process, and communicate directly to it.
- *Multi-hop Balanced Scenario*. Sensor nodes are deployed to form a T-shaped topology with approximately even branches, as illustrated in Figure 8b. We set the transmission power of sensor nodes to level 4 (corresponding to about -20 dBm) and measured a transmission range of approximately 12 m. Since the distance between neighboring nodes deployed along the same row is about 10 m, in theory we should expect a multi-hop topology with routing sub-trees having approximately the same depth.
- *Multi-hop Unbalanced Scenario*. Sensor nodes are deployed again to form a T-shaped topology, but sub-trees have now different depths (2, 4, and 8, as shown Figure 8c). Transmission power and distances between nodes are the same as the previous scenario. Therefore, in theory, nodes should form an unbalanced multi-hop topology.

Note that experiments are extremely sensitive to channel conditions due to different factors – i.e., weather conditions, time of day and so on – so that the actual topology may be different than expected. Therefore, as the routing tree impacts on performance, we reported the depth of the routing tree used during experiments in Table 5. For example, we noted

that in the all-in-range scenario, nodes located at the upper corners of the grid (see Figure 8a) actually associate with an intermediate node in all experiments, resulting in a routing tree of 2 hops. This is due to the link quality metrics used for building the data gathering tree [47].

Figure 9a compares the average duty-cycle allowed by the different sleep/wakeup schemes in the three scenarios under investigation. The trend is the same as in Figure 5a, i.e. ASLEEP allows the lower duty-cycle in all the three scenarios. As expected, with ASLEEP and the optimal fixed scheme sensor nodes experience higher duty-cycles when moving from the all-in-range to the multi-hop balanced, and to the multi-hop unbalanced scenario. For TAG the trend is the just the opposite, as in this scheme the length of the talk interval depends on the inverse of the routing tree's depth (see Table 5 for details).

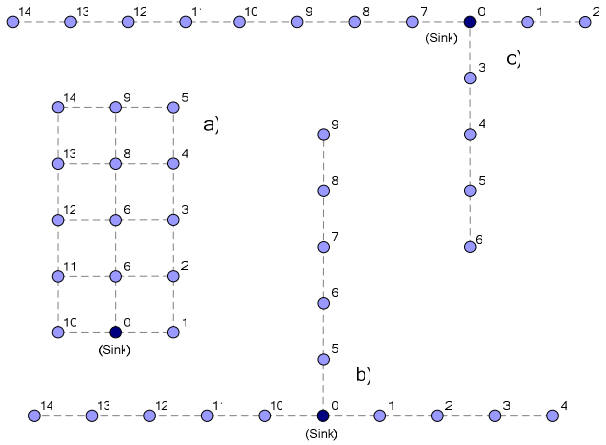


Figure 8. Scenarios considered for the experimental analysis: all-in-range (a), multi-hop balanced (b), multi-hop unbalanced (c).

TABLE 5. ROUTING TREE DEPTH FOR EXPERIMENTS (MEAN AND STD DEV).

	All-in-range Scenario	Multi-hop Balanced	Multi-hop Unbalanced
ASLEEP	2.0 (0)	3.8 (0.45)	5.4 (1.34)
Optimal Fixed	2.0 (0)	4.0 (0)	4.0 (0.71)
TAG	2.0 (0)	4.0 (0)	4.8 (1.10)
Always ON	2.2 (0.45)	3.6 (0.55)	4.4 (0.55)

Figure 9b shows the average message latency introduced by the different sleep/wakeup schemes in the three scenarios. Again, the trend is similar to the one observed in Figure 5b. In particular, the adaptive scheme introduces an average latency lower than the optimal fixed scheme (584 ms vs. 873 ms in the multi-hop unbalanced scenario).

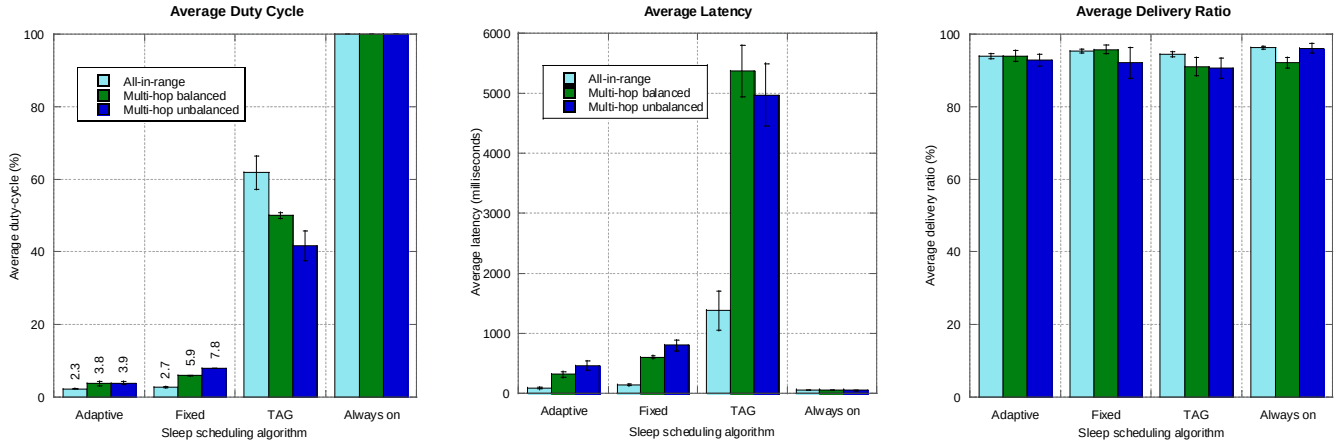
The delivery ratio is shown in Figure 9c. In this case the trend is different from that observed in Figure 5c. All the sleep/wakeup schemes provide approximately the same delivery ratio (above 90%), and there is no significant difference when passing from one scenario to another. This behavior can be explained by observing that, unlike in the simulation scenarios considered above, in the experimental scenarios nodes are located in such a way that collisions between neighboring nodes are not frequent. In addition, sensor nodes in the real testbed use a MAC protocol which is less prone to packet losses than the IEEE 802.15.4 MAC protocol considered in the simulation analysis.

VII. 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 nodes' schedules according to their position in the routing tree. This helps to reduce latency also when nodes are sleeping for the 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 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 in any available sensor platform.

We have analyzed the protocol performance in both dynamic and stationary conditions through an extended simulation campaign. 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 message latency with respect to an optimized staggered approach where, however, active periods are fixed and equal for all nodes in the network. We also observed that, as a side effect, our adaptive staggered approach increases the delivery ratio.

We also implemented our adaptive staggered protocol (and other sleep/wakeup schemes) in a real testbed and validated simulation results through experimental measurements. The experimental results confirm the benefits of using our protocol.



(a) (b) (c)

Figure 9. Average duty-cycle (left), average message latency (center), and delivery ratio in the different scenarios.

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] 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.

[3] G. Anastasi, M. Castronuovo, M. Conti, M. Di Francesco, "Experimental Evaluation of An Adaptive Staggered Sleep Protocol for Wireless Sensor Networks", *Proc. of the 3rd IEEE Workshop on advanced EXPerimental activities ON WIRELESS networks & systems (EXPONWIRELESS08)*, Newport Beach, CA, (USA), June 23, 2008.

[4] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, "An Adaptive and Low-latency Power Management Protocol for Wireless Sensor Networks", *Proc. of the ACM International Workshop on Mobility Management and Wireless Access (MobiWac 2006)*, Torremolinos (Spain), October 2, 2006.

[5] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, "Energy Conservation in Wireless Sensor Networks: a Survey", *in Ad hoc Networks*, to appear.

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

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

[8] A. Cerpa, D. Estrin, "Ascent: Adaptive Self-Configuring Sensor Network Topologies, *Proc. IEEE INFOCOM 2002*.

[9] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris. "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", *ACM Wireless Networks*, Vol. 8, N. 5, September 2002.

[10] 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.

[11] 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.

[12] J. Haartsen, "The Bluetooth Radio System", *IEEE Personal Communications*, Vol. 7, N. 1, pp. 28-36, February 2000.

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

[14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, "System architecture directions for networked sensors", *SIGPLAN* Not. 35, 11 (Nov. 2000), 93-104.

[15] 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.

[16] 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.

[17] 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), May 2003.

[18] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks", *Proc. ACM (MobiCOM 2000)*, Boston, USA, August 2000.

[19] 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.

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

[21] A. Keshavarzian, H. Lee, L. Venkatraman, "Wakeup Scheduling in Wireless Sensor Networks", *Proc. ACM MobiHoc 2006*, Florence, Italy, May 2006.

[22] 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.

[23] 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.

[24] 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.

[25] G. Lu, B. Krishnamachari and C.S. Raghavendra, "An Adaptive Energy-efficient and Low-latency Mac for Data Gathering in Wireless Sensor Networks", *in Proc. of PDSP '04*, Pages: 224, April 2004.

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

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

[28] S. Madden, "The Design and Evaluation of a Query Processing Architecture for Sensor Networks", UC Berkeley Ph.D. Thesis, 2003.

[29] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring", *Proc. of the First ACM Workshop on Wireless Sensor Networks and Applications*, Pages: 88-97 Atlanta, GA, USA, 28 September 2002.

[30] A. Manjeshwar and D. P. Agrawal, "APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks", *Proc. of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Ft. Lauderdale, Florida, April 2002

[31] D. Mirza, M. Owrang, C. Schurgers, "Energy-efficient Wakeup Scheduling for Maximizing Lifetime of IEEE 802.15.4 Networks", *Proc. International Conference on Wireless Internet (WICON'05)*, Budapest (Hungary), pp. 130 - 137, July 2005.

[32] Network Simulator Ns2, http://www.isu.edu/nsnam/ns.

[33] ON World Inc, "Wireless Sensor Networks – Growing Markets, Accelerating Demands", July 2005, available at http://www.onworld.com/html/wirelesssensorsrprt2.htm.
[34] Embedded WiSeNTs Consortium, "Embedded WiSeNts Research Roadmap (Deliverable 3.3)", available at www.embedded-wisents.org.
[35] V. Paruchuri, S. Basavaraju, R. Kannan, S. Iyengar, "Random Asynchronous Wakeup Protocol for Sensor Networks", *Proc. of BROADNETS '04*, 2004.
[36] S. Ping, "Delay Measurement Time Synchronization for Wireless Sensor Networks", *IRB-TR-03-013*, Intel Research Berkeley Lab, 2003.
[37] J. Polastre, J. Hill and D. Culler, "Versatile Low Power Media Access for Sensor Networks", *in Proc. of SenSys '04*, November, 2004.
[38] V. Raghunathan, C. Schurgers, S. Park and M. B. Srivastava, "Energy Aware Wireless Microsensor Networks", *IEEE Signal Processing Magazine*, Volume: 19, No: 2, Pages: 40-50, March 2002.
[39] 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.
[40] 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.
[41] P. Santi, "Topology Control in Wireless Ad Hoc and Sensor Networks, *ACM Computing Survey*, Vol. 37, n. 2, p. 164-194, June 2005.
[42] C. Schurgers, V. Tsiatsis, M. B. Srivastava, "STEM: Topology Management for Energy Efficient Sensor Networks", Proc. *of the IEEE Aerospace Conference '02*, Big Sky, MT, March 10-15, 2002.
[43] K. Sohrabi, J. Gao, V. Ailawadhi, G. J. Pottie, "Protocols for Self-organization of a Wireless Sensor Network", *IEEE Personal Communications*, Vol. 7 (5), October 2000.
[44] Tmote Sky Platform, MoteIV Corporation, http://www.moteiv.com/products/tmotesky.php
[45] G. Tolle et al., "A Macroscope in the Redwoods", *Proc. ACM SenSy)*, San Diego, 2-4 November 2005.
[46] TinyDB: a Declarative Database for Sensor Networks, http://telegraph.cs.berkeley.edu/tinydb/
[47] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multhop routing in sensor networks", *Proc. of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 03)*, Pages: 14-27, Los Angeles, California, November 2003.
[48] X. Yang, N. Vaidya, "A Wakeup Scheme for Sensor Networks : Achieving Balance between Energy Saving and End-to-end Delay", *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2004)*, pp. 19-26, 2004.
[49] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks", *in Proc. of the IEEE Infocom*, New York, NY, June 2002, pp. 1567–1576.
[50] W. Ye, J. Heidemann and D. Estrin, "Medium access control with coordinated adaptive sleeping for wireless sensor networks", *IEEE/ACM Trans. Netw*. n. 12 (3), Jun. 2004.
[51] 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.
[52] J. Zheng and M. J. Lee, "A Comprehensive Performance Study of IEEE 802.15.4", *IEEE Press Book*, 2004.

APPENDIX A. PSEUDO-CODE OF ASLEEP

In this appendix we present a formal description of the ASLEEP protocol. In the following we will first describe protocol behavior during the startup phase and, then, during the normal operational phase.

I. Startup phase

The startup phase is needed to build the initial schedules for nodes. Note that the startup phase of ASLEEP makes use of a default talk interval duration, which is the same for all nodes and is specified prior to deployment. This is needed because parent nodes cannot estimate the talk interval duration until the window used for deriving the statistics is full. However, as soon as a parent node has obtained the statistics, it can immediately produce the related estimate and start adapting schedules to network conditions. As ASLEEP can derive the correct talk interval autonomously, the proper setting of the default value used during startup is not critical. Nevertheless, it is convenient to choose a value large enough to accommodate the maximum required activity time, in order to reduce the number of subsequent communication periods needed to reach a steady state condition.

In the regular staggered scheme used by ASLEEP, nodes at a given level in the routing tree wake up earlier than their ancestors. As beacons are sent during the talk interval between the parent and its children, they take a communication period at a time to propagate toward the lower levels of the routing tree. Since all nodes in the tree must receive a beacon to acquire the schedule parameters, the initial beacon propagation phase lasts a number of communication periods as long as the depth of the tree. This is a serious drawback because the initial schedule establishing phase can last for a very long time. It's worth noting that nodes are always on in the initial schedule establishing phase, so the energy consumption is directly tied to the duration of this transient phase.

To reduce the energy consumption, a different approach is used to propagate beacons during the startup phase. Since nodes are initially always on, startup beacons are propagated to the lower levels of the tree in a row. Each node immediately forwards the received startup beacon to its children. Nodes receiving the startup beacon can therefore schedule their wakeup for the next communication period, so that the initial schedule propagation is reduced to a single communication period.

II. Steady state operations

In the following, we assume that ASLEEP has already completed the startup phase and established the initial schedules. We also assume that each source node senses the external environment and acquires one or more samples *before* the beginning of the talk intervals with its parent. Therefore, a child node has all the messages it is going to send during the talk interval in a local buffer.

Obviously, the specific actions performed by each single node strongly depend on its position along the routing tree, i.e. whether it is the root (sink), a leaf, or an intermediate node. Algorithm 1 shows the actions performed by a leaf node *k*, whose parent node is j, during the generic *m-th* communication period. The leaf node wakes up at the scheduled time $t^m_{child,k}$ and, as a preliminary step, schedules the end of the talk interval by setting up a timer (line 2). To this end the leaf node uses the talk interval value TI^m_j received from its parent (through the direct beacon) in the previous communication period. During the talk interval the leaf node sends data messages (if any) to its parent and, then, waits for the direct beacon from it (lines 3-6). The direct beacon includes the length of the communication period (CP^{m+1}) and talk interval (TI^{m+1}_j) to be used in the next communication period. It also contains the next wakeup time of the parent node j ($t^{m+1}_{parent,j}$). This value is used by the leaf node to set its next wakeup time (lines 7-8). If the direct beacon is missed, the leaf node assumes that the schedule will remain unchanged in the next communication period and uses the same parameters received with the previous direct beacon (lines 9-12). Finally, the leaf node goes to sleep until the next scheduled wakeup time.

Algorithm 1: Actions performed by a leaf node *k* during the *m-th* communication period

1. **upon** wakeup {
   // talk interval with parent node j, starting at $t^m_{child,k}$
2. **start** timer(TI^m_j);
3. **do** {
4. send data messages to parent node j;
5. wait for direct_beacon;
6. } **until** (timer expires **or** direct_beacon received);
7. **if** (direct_beacon (CP^{m+1}, TI^{m+1}_j, $t^{m+1}_{parent,j}$) received)
8. schedule next wakeup at $t^{m+1}_{child,k} = t^{m+1}_{parent,j}$;
9. **else** {
10. $CP^{m+1} = CP^m$; $TI^{m+1}_j = TI^m_j$;
11. schedule next wakeup at $t^{m+1}_{child,k} = t^m_{child,k} + CP^{m+1}$;
12. }
13. **start** timer($t^{m+1}_{child,k} - t_{now}$)
14. switch to sleep mode;
15. }

The sink node has only one talk interval with its children, because it is the root of the tree. Thus, it has only to receive messages from children, collect statistics on the incoming traffic, and adapt the scheduling parameters to the current network activity. Algorithm 2 shows the specific actions performed by the sink during its talk interval. Initially, it schedules the direct beacon transmission (line 3), by appropriately setting up a timer. To this end it uses the expected talk interval estimated in the previous communication period. During the talk interval the sink receives messages from its children. A message from a child

may be either a data message or a reverse beacon. Upon receiving a data message, the sink updates statistics on message inter-reception times and number of received messages that will be used later to estimate the next talk interval (line 10) according to the algorithm described in Section IV.B. Upon receiving a reverse beacon from a child node i, the sink realizes that node j requires to shift the talk interval by a quantity $rshift_j^{m+1}$ included in the reverse beacon (line 6). In case of multiple shift requests from children, the maximum shift value (line 7) will be used to derive the starting time of the next talk interval (line 11). Finally, the sink estimates the duration of next talk interval according to the algorithm described in Section IV.B (line 10), derives the starting time of the next talk interval (line 11), and broadcast the direct beacon with new schedule parameters to its children (line 12). Then, the sink pauses until the beginning of the next talk interval.

Algorithm 2: Actions performed by the sink node *s* during the *m-th* communication period

1. **upon** timer expiration {
   // talk interval with children starting at $t_{parent,s}^m$
2. $shift_s^{m+1} = 0$;
3. schedule direct_beacon transmission;
4. **do** {
5. **wait** for messages from children;
6. **if** (message==reverse_beacon($rshift_j^{m+1}$))
7. $shift_s^{m+1} = \max(shift_s^{m+1}, rshift_j^{m+1})$
8. **else if** (message==data) update statistics;
9. } **until** (direct_ beacon transmission time);
10. calculate TI_s^{m+1} from statistics;
11. schedule the beginning of next talk interval at:
    $t_{parent,s}^{m+1} = t_{parent,s}^m + CP^{m+1} + shift_s^{m+1} + \max\left(0, TI_s^m - TI_s^{m+1}\right)$;
12. **send** direct_beacon (CP^{m+1} , TI_s^{m+1} , $t_{parent,s}^{m+1}$)
13. **start** timer($t_{parent,s}^{m+1} - t_{now}$)
14. }

Finally, an intermediate node in the routing tree has both roles for the communication: as a parent it collects data from its children, as a child it forwards data to its parent. As a result, it must have two distinct, but adjacent, talk intervals. Algorithm 3 shows the actions performed during any communication period by a generic intermediate node j having node i as its parent. Upon wakeup, node j behaves as a parent, and acts in the same way as the sink. It schedules the direct beacon transmission (line 3) and waits for incoming messages (line 5). Data messages from children are first evaluated to update statistics and, then, stored in a queue for subsequent transmission to the parent node i (lines 8-11). Reverse beacons are managed by node j in the same way as the sink node (lines 6-7). At the end of the talk interval with its children, node j calculates the schedule parameters for the next communication period, and advertises them through the direct beacon (lines 13-16). This concludes the talk interval with children. Then, node j acts as a child of node i (this part of the algorithm is thus very similar to Algorithm 1). First of all, node *j* schedules the end of the talk interval with its parent (line 18). Then, it sends queued messages and, if needed, a reverse beacon (lines 20-22). When sending the reverse beacon, the intermediate node adds to the maximum shift requested by children (line 7) its own requested shift $TI_j^{m+1} - TI_j^m$, if this is greater than zero (lines 20-21). Upon receiving the direct beacon from parent node i, node *j* gets the parameters for the next communication period. If the direct beacon is missed, then node j uses the current parameters even in the next communication period (lines 27-30). In any case it can set the timer and switch to the sleep mode.

Algorithm 3: Actions performed at the *m-th* communication period by an intermediate node j having node i as its parent

1. **Upon** wakeup as a parent {
   // talk interval with children starting at $t^{m}_{parent,j}$
2. $shift_j^{m+1} = 0$;
3. schedule direct_beacon transmission;
4. **do** {
5. **wait** for messages from a generic child node *k*;
6. **if** (message==reverse_beacon($rshift_k^{m+1}$))
7. $shift_j^{m+1} = \max(shift_j^{m+1}, rshift_k^{m+1})$
8. **else if** (message==data) {
9. update statistics;
10. store message in the local queue;
11. }
12. } **until** (direct_ beacon transmission time);
13. calculate TI_j^{m+1} from statistics;
14. schedule the beginning of next talk interval at:
    $t^{m+1}_{parent,j} = t^{m}_{parent,j} + CP^{m+1} + shift_j^{m+1} + \max\left(0, TI_j^{m} - TI_j^{m+1}\right)$;
15. **send** direct_beacon (CP^{m+1} , TI_j^{m+1} , $t^{m+1}_{parent,j}$);
16. **}**

17. **Upon** wakeup as a child {
    // talk interval with parent node i starting at $t^{m}_{child,j}$
18. **start** timer(TI_i^{m});
19. **do** {
20. **if** ($shift_j^{m+1} > 0$ **or** $TI_j^{m+1} - TI_j^{m} > 0$)
21. **send** reverse_beacon($shift_j^{m+1} + \max(TI_j^{m+1} - TI_j^{m}, 0)$)
    to node i ;
22. **send** queued data messages to parent node i ;
23. **wait** for direct_beacon from node node i ;
24. } **until** (activity_timer expires **or** direct_beacon received);
25. **if** (direct_beacon (CP^{m+1} , TI_i^{m+1} , $t^{m+1}_{parent,j}$) received)
26. schedule next wakeup at $t^{m+1}_{child,j} = t^{m+1}_{parent,j}$;
27. **else** {
28. $CP^{m+1} = CP^{m}$; $TI_i^{m+1} = TI_i^{m}$;
29. schedule next wakeup at $t^{m+1}_{child,j} = t^{m}_{child,j} + CP^{m+1}$;
30. }
31. **start** timer($t^{m+1}_{child,j} - t_{now}$);
32. switch to sleep mode;
33. }

APPENDIX B. CORRECTNESS PROOFS

In this appendix we provide the proofs of the properties introduced in Section IV. In the following, we will assume that:

- The communication period remains constant over time (i.e., $CP^m = CP$, for any m)
- Beacons (either direct and indirect) are always received correctly;
- Nodes' clocks are properly synchronized (e.g., by using the synchronization protocol in [36]).

We are now in the position to prove the following properties.

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

Proof. Let us consider, at a generic communication period m, any two nodes i and j, such that j is a child of i. By hypothesis direct beacons are successfully received and, hence, node j correctly gets the values of CP, TI_j^{m+1} and $t_{parent,i}^{m+1}$. In addition, nodes' clocks are synchronized, so that when j sets $t_{child,j}^{m+1} = t_{parent,i}^{m+1}$, the correspondent instants agree. Hence, the two nodes wake up at the same time, thus proving our assert. □

Property 2 (Non overlapping schedules). *For any couple of nodes* i *and* j, *such that* j *is a child of* i, *the talk intervals* TI_i *and* TI_j *with the corresponding children are not overlapped.*

Proof. We prove by induction that parent-child schedules are not overlapped over time. Clearly, schedules are properly staggered during the initial communication period (m=0), as they are generated by the startup phase which uses a fixed talk interval for all nodes. We now assume as induction hypothesis that the activation instants of nodes i and j, are correct during the m-th communication period, i.e.

$$t_{parent,i}^{m} \geq t_{parent,j}^{m} + TI_j^m \quad (1)$$

Applying the expression in line 14 of Algorithm 3 to nodes i and j, respectively, yields

$$t_{parent,i}^{m+1} = t_{parent,i}^{m} + CP + shift_i^{m+1} + \max(TI_i^m - TI_i^{m+1}, 0)$$

$$t_{parent,j}^{m+1} = t_{parent,j}^{m} + CP + shift_j^{m+1} + \max(TI_j^m - TI_j^{m+1}, 0)$$

By introducing the above expressions in (1) we obtain

$$t_{parent,i}^{m+1} - CP - shift_i^{m+1} - \max(TI_i^m - TI_i^{m+1}, 0) \geq t_{parent,j}^{m+1} - CP - shift_j^{m+1} - \max(TI_j^m - TI_j^{m+1}, 0) + TI_j^m \quad (2)$$

Equation (2) can be rewritten as

$$t_{parent,i}^{m+1} \geq t_{parent,j}^{m+1} + shift_i^{m+1} - shift_j^{m+1} + \max(TI_i^m - TI_i^{m+1}, 0) - \max(TI_j^m - TI_j^{m+1}, 0) + TI_j^m \quad (3)$$

By lines 7 and 21 of Algorithm 3 it is

$$shift_i^{m+1} = \max_{j^*}(rshift_{j^*}^{m+1}) = \max_{j^*}\left(shift_{j^*}^{m+1} + \max_{j^*}(TI_{j^*}^{m+1} - TI_{j^*}^{m}, 0) \right)$$

where the max function is calculated over all children j^* of i (including j). Hence,

$$TI_j^m + shift_i^{m+1} - shift_j^{m+1} - \max(TI_j^m - TI_j^{m+1}, 0) =$$

$$TI_j^m + \max_{j^*}(rshift_{j^*}^{m+1}) - shift_j^{m+1} - \max(TI_j^m - TI_j^{m+1}, 0) \geq$$

$$TI_j^{m+1} + shift_j^{m+1} - shift_j^{m+1} + \max(TI_j^{m+1} - TI_j^m, 0) -$$

$$\max(TI_j^m - TI_j^{m+1}, 0) = TI_j^{m+1} + \max(TI_j^{m+1} - TI_j^m, 0) -$$

$$\max(TI_j^m - TI_j^{m+1}, 0)$$

The last inequality has been obtained by replacing the max function calculated over all children j^* of i, with the quantity[5] related to j.

In addition, it can be shown that

$$TI_j^m + \max(TI_j^{m+1} - TI_j^m, 0) - \max(TI_j^m - TI_j^{m+1}, 0) = TI_j^{m+1}$$

Hence, equation (3) can be rewritten as

$$t_{parent,i}^{m+1} \geq t_{parent,j}^{m+1} + TI_j^{m+1} + \max(TI_i^m - TI_i^{m+1}, 0),$$

$$\geq t_{parent,j}^{m+1} + TI_j^{m+1}$$

which proves our assert. □

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

Proof. It trivially follows from the above proof, by assuming $t_{parent,i}^{m} = t_{parent,j}^{m} + TI_j^m$ as inductive hypothesis and letting $shift_i^{m+1} - shift_j^{m+1} = 0$ and $TI_j^{m-1} - TI_j^m = 0$ as a consequence of the steady state conditions. □

[5] Recall that node j is a generic child of i, so that $\max_{j^*}(rshift_{j^*}^{m+1}) \geq rshift_j^{m+1}$ as the set of j^* includes j as well.