Analyzing the Energy-Latency Trade-off during the Deployment of Sensor Networks

Thomas Moscibroda, Pascal von Rickenbach, Roger Wattenhofer
{moscitho, vonrickenbach, wattenhofer}@tik.ee.ethz.ch
Computer Engineering and Networks Laboratory
ETH Zurich, 8092 Zurich, Switzerland

***Abstract*— The inherent trade-off between energy-efficiency and rapidity of event dissemination is characteristic for wireless sensor networks. Scarcity of energy renders it necessary for nodes to spend a large portion of their lifetime in an energy-efficient sleep mode during which they do neither receive nor send messages. On the other hand, the longer nodes stay in sleep mode, the slower will be the reaction time for disseminating an external event. The trade-off is prominently exhibited during the deployment phase of sensor networks, if some nodes are deployed earlier than others. In this paper, we study this fundamental trade-off by giving a formal model that enables us to compare the performance of different protocols and algorithms. Based on this model, we propose, analyze, and simulate two novel algorithms which significantly outperform existing solutions.**

I. Introduction

Wireless sensor networks have been envisioned in a growing number of application fields. The prospect of aggregating sensor nodes into sophisticated computation and communication infrastructures is bound to have a significant impact on a wide array of scientific, industrial, or military applications. One of the key characteristics of such *sensor networks* is that individual sensor nodes have a limited, typically non-renewable power supply and, once deployed, must work unattended. In view of the scarcity of energy, an economical and frugal management of this resource is essential for prolonging network lifetime and availability.

The search for energy-efficient solutions has lead to numerous algorithms and protocols that strike for the goal of reducing the energy-consumption of an operational sensor network. There have been, for instance, various proposals for energy-efficient medium access control (MAC) protocols [22], [25], [31], [32], routing algorithms [1], [6], topology control and clustering [3], [10], [20], [30], [17], or data gathering/dissemination [7], [9], [27], [33]. This impressive body of work has lead to new insights and several intriguing solutions. Clearly, continuous research on this *operational phase* of wireless sensor networks is needed. In this paper, we advocate studying also the *non-operational phase* of a sensor network with the same zeal. Specifically, in many applications, a crucial loss of energy occurs already *before* the sensor network reaches its operational state, i.e., during its deployment.

Consider for instance a water (or power, gas, etc.) metering network for an apartment complex. Each apartment is equipped with a water metering sensor. At midnight, all sensors wake up for a few seconds, the water consumption of each apartment is sent to a base station in multi-hop fashion, and all sensors go back to sleep for another 24 hours. In the operational phase such a sensor network features a gargantuan sleep/awake ratio, allowing even conventional batteries to last several years. In order to reach such a long lifetime the node's duty cycle must be significantlys below 1%. However, the deployment of the sensor nodes might take days or weeks. With a naive deployment protocol, say, when nodes stay awake until the entire system is deployed, the battery of the node deployed first might be drained before the network even becomes operational. This highlights a problem that is particularly pronounced in settings in which the node's duty cycle during the operational phase is small, but the deployment phase takes long. Many other applications featuring such a time-consuming deployment phase exist, e.g. vehicle tracking, or monitoring large-scale industrial processes.

Generally, once all sensor nodes are fully deployed, the network should make the transition from the deployment phase to the operational phase as quickly as possible. In particular, we might like to externally trigger a *network discovery* procedure that allows verifying the operability of the newly deployed network (e.g. detect faulty sensor nodes). Clearly, simple solutions to invoke such a system initialization would be to manually switch on all nodes once the deployment phase is completed, or to set a timer at the time of the node's deployment. Unfortunately, in many practical settings, neither of these hands-on solutions is practicable. First, nodes may be deployed in remote or hostile environment in which switching on nodes manually after all nodes are deployed may be impossible. Moreover, in application scenarios featuring a time-consuming deployment phase, predicting the exact duration of the deployment process is usually hard, hence ruling out the possibility of employing a solution based on predefined timers.

So how can the information about the beginning of the operational phase be distributed among the network nodes? Typically, this information is supposed to be *broadcasted by the nodes* in a multi-hop way through the entire network such that, eventually, every sensor node will know that the system is now ready to start its operational phase. Specifically, one or several nodes (in typical sensor network applications, this is usually the base station) are triggered externally. These nodes then try to inform their neighbors, who in turn inform their neighbors, and so forth. We call this externally triggered event that sets off the information broadcast the *launching point*. The trade-off studied in this paper is about saving energy *during* deployment, yet quickly going into operational mode *after* the launching point.

Ideally, each node should remain in some kind of energy-saving *sleep mode* for the entire duration of the deployment phase preceding the launching point. In sleep mode, nodes do neither send data packets nor listen for incoming messages [26]. The problem is however that individual nodes do not know the exact time of the launching point, or the duration of the deployment phase. As a consequence, in order to learn about the arrival of the launching point from neighbors, a node must periodically leave the sleep mode and listen for incoming messages.[1]

This observation establishes a trade-off between the energy consumption of nodes during the deployment phase and the rapidity of the transition to the operational phase. Neither of the two extremes, *always asleep* and *always awake* during the deployment, is satisfying; any decent protocol is in-between.

We believe that studying the trade-off delay vs. energy efficiency is practically important, even beyond the deployment problem. In particular, there are sensor networks that concentrate on discovering rare events, e.g. sensor networks for seismic surveillance in earthquake and rubble zones, or sensor networks monitoring enemy activity. The pronounced "event" character of such rare events leads to exactly the deployment problem trade-off. Namely, since events occur rarely, sensor nodes should be in sleep mode as often as possible to save energy. These energy savings, however, come at the cost of a prolonged reaction time once a rare event occurs. Hence, this conflict between energy-efficiency and the rapidity of information propagation is fundamental in sensor networks.

In this paper, we take a step towards understanding and analyzing the trade-off between energy-efficiency and propagation delay, particularly during the deployment phase. We model the problem in a way that allows to compare different protocols and algorithms and evaluate their respective strengths and weaknesses, independent of application specific parameters such as node distribution or deployment pattern. Specifically, we analyze the behavior of three different algorithms. The first algorithm [18] has originally been proposed for the purpose of neighbor discovery, but can be applied for the deployment problem as well. In addition, we present two novel algorithms that significantly outperform the algorithm by [18], for both worst-case *and* average-case scenarios. It is interesting to note that one of our algorithms is "semi-structured," in the sense that already deployed nodes structure themselves in a feeble way that allows to incorporate newly deployed nodes with a small energy overhead only. This semi-structured approach is in contrast to, say, tree-based dissemination algorithms in which during the deployment process, a lot of effort (and hence, energy) is required to recognize and integrate new nodes. We believe that constructing *semi-structures* is an interesting concept by itself, with potentially many applications beyond the scope of this paper.

The remainder of the paper is organized as follows. We define our model of computation and the problem in Sections II and III, respectively. In Section IV we analyze the behavior of the different algorithms. While this section derives fundamental results that hold even in *worst-case* scenarios, we investigate the algorithms' *average-case* efficiency in Section V using simulations. Section VI gives an overview over related work, before Section VII concludes the paper.

II. Model

Our model of computation is based on the *unstructured radio network model* as introduced in [15]. This model aims to capture the harsh characteristics of newly deployed ad hoc and sensor networks. It encompasses various critical aspects such as asynchronous wake-up, absence of a MAC layer, and scarce knowledge about the network graph. More specifically, the model makes the following assumptions.

- During the deployment phase, sensor nodes *wake up asynchronously* at any time. Moreover, they do not have access to a global clock and hence, upon waking up, they do not know whether or how many other nodes in their neighborhood have already been deployed. Once the launching point is reached, we assume that all nodes have been deployed and therefore, no new nodes join the network. In other words, after the launching point we consider a static network.
- We also assume that nodes have *no built-in knowledge* about other node's distribution or wake-up pattern. Specifically, nodes are completely clueless about the number of nodes in their neighborhood. The only knowledge a-priori given to the nodes is an upper bound n for the total number of nodes deployed in the network. It has been shown in [13] that without any such estimate of n, every algorithm requires at least time $\Omega(n/\log n)$ until one single message can be transmitted without collision. In practice, the number of nodes in a network may not be known exactly, but it can roughly be estimated in advance.
- If a node receives multiple messages at the same time, these messages become garbled and cannot be received properly. Moreover, nodes do not feature a reliable *collision detection mechanism*. That is, nodes are not capable of distinguishing the situation in which two or more neighbors are sending and the situation in which no neighbor is sending. Furthermore, a sending node does not know how many (if any at all) of its neighbors have correctly received its transmission. The unreliable collision detection model is the strongest possible model when analyzing wireless networks. Clearly, algorithms

[1]Obviously, the problem could be elegantly solved using very low power "trigger" circuits, which operate continuously on small power budgets, and wake up more power-hungry circuits only upon receipt of a suitable signal from a neighboring node. Unfortunately, currently available standard hardware such as the Mica2 [11] wireless sensor nodes do not offer this functionality, and we therefore do not consider this option in this paper.

designed for a model as restricted as this can also be employed by systems that are equipped with more sophisticated hardware.
- We model the multi-hop network as a unit disk graph (UDG). In a UDG $G = (V, E)$, with $n = |V|$, two nodes are connected by an edge if their Euclidean distance is at most 1. The network being multi-hop leads to well-known aspects such as the hidden-terminal problem.
- Finally, we assume that both the node's location and wake-up pattern is completely arbitrary, potentially even *worst-case*. Particularly, we do not assume any kind of uniform node distribution or Markovian wake-up pattern.

The various aspects of this model suggest that we deal with a particularly harsh model of computation; a model that captures many of the realistic characteristics of newly deployed sensor networks. We assume time to be divided into time-slots, the length of which are roughly the same at each node. In each time-slot a node can be in exactly one of the three following modes: *transmit* T, *listen* L, or *sleep* S. In sleep mode S, a nodes deactivates its radio subsystem in order to save energy. That is, a node does not overhear the shared medium in sleep mode and thus misses all messages sent by neighboring nodes. At the communication distances typical in sensor networks, listening for information on the radio channel is of a cost similar to transmission of data [23]. Therefore, the energy consumption $e(v)$ of a node v corresponds to the number of time-slots it spends in either transmit or listen mode. Consequently, reducing merely the node's *sending time* is not sufficient when designing energy efficient algorithms for sensor networks. Instead, the *listening time* must also be minimized, however slowing down event dissemination.

III. Problem Statement

Before the sensor network can start performing its intended task, nodes must be deployed, a process that may take several days or even weeks. We divide the *non-operational phase* of a sensor network into two parts, the *deployment phase* and *notification phase* as shown in Figure 1. In the deployment phase, sensor nodes are physically positioned at their intended locations. Once this is done for all sensor nodes, the notification phase is triggered, in which the aim is to inform all nodes about the system being up and running. The transition to this second phase is induced by an externally triggered event that is received by at least one node in the network. We call this moment when the first node becomes notified the *launching point* $\mathcal{LP}$. During the notification phase, we call a node *notified* if it has already received the notification message, and *unaware* otherwise. At the launching point, at least 1 node is notified whereas at most $n - 1$ nodes are unaware.

During the deployment phase, an algorithm may build an initial structure which can help speeding up the notification process later on. On the other hand, the building and maintenance (incorporating newly awakening nodes into a tree, for example) of such a structure requires the nodes to stay awake longer and thus spend more energy. In order to enable a fair comparison between different algorithmic approaches, our problem definition has to be general enough to account for these various possibilities.

The total energy consumption of a node v in a deployment algorithm $\mathcal{A}$ can be divided into two parts, the *initialization energy* and the *maintenance energy*. The initialization energy $e_{init}(v)$ is the total amount of energy used by v to initially join a desired structure (e.g., decide whether it is a clusterhead or become a part of a tree). A node's initialization energy accrues only once, regardless of the length of the deployment phase. In contrast, the maintenance energy $e_m(v)$ denotes the total amount of energy used by v once it has been properly initialized. Specifically, the maintenance energy $e_m(v)$ encompasses the node's periodic wake-up necessary to learn about the launching point. If $e_m(v)$ is small, the node will require a long time before learning about the $\mathcal{LP}$, thus slowing down the notification phase. Depending on the nature of the algorithm, $e_m(v)$ may comprise additional aspects. Consider for instance an algorithm that is based on maintaining a tree-structure which allows for rapid event dissemination during the notification phase. In this case, already initialized nodes periodically send messages in order to inform neighbors that may have woken up in the meantime, thus enabling their integration into the tree.

More formally, let T_D and T_N be the length of the deployment phase and notification phase, respectively. Further, $t_w(v)$ denotes the wake-up point of node v. The time v is active before the launching point is $\ell(v) = t(\mathcal{LP}) - t_w(v)$. Since we consider asynchronous wake-up with an imaginary adversary determining each node's wake-up point (and hence $\ell(v)$), we must consider the *average maintenance energy* $a_m(v) = e_m(v)/\ell(v)$. This value describes the maintenance energy used by a node v for a single time-slot between its wake-up and the LP. Note that $a_m(v)$ is independent of a node's $t_w(v)$, $\ell(v)$, or the time of the launching point, because $a_m(v)$ considers only periodical maintenance costs, i.e., no initialization costs.

We still have to come up with a measure for the algorithm's *energy efficiency* that takes into account both the maintenance and the initialization costs, but remains independent of the specific wake-up pattern. For that, we define the energy efficiency of an algorithm $\mathcal{A}$ with regard to a deployment phase of length T_D, denoted by $E(\mathcal{A}, T_D)$, as the *average energy of algorithm $\mathcal{A}$ per node and per time-slot.* That is, an algorithm in which all nodes listen in every time-slot has energy efficiency equal to 1, whereas the algorithm that lets all nodes sleep all the time has energy efficiency 0. With this definition, the measure of an algorithm's energy efficiency does not depend on the particular wake-up pattern of a given problem instance. Instead, it captures the characteristic of the algorithm itself, thus enabling a stringent and concise comparison between different approaches.

Formally, the two main quality measures of a deployment algorithm $\mathcal{A}$ are defined as follows.

Definition 1: Let $\mathcal{A}$ be a deployment algorithm and let T_D be the length of the deployment phase before the launching

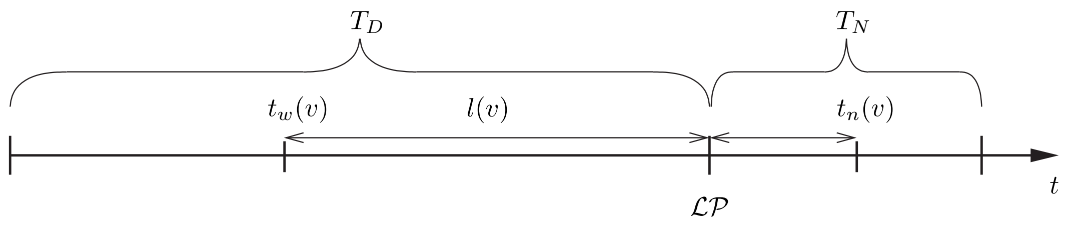


Fig. 1. The deployment phase is of length T_D, the notification phase is of length T_N.

point. Also, let $f(n)$ be a minimal function such that with probability at least $1 - 1/n$, it holds that $T_N \leq f(n)$. The algorithm's energy and time efficiency, $E(\mathcal{A}, T_D)$ and $T(\mathcal{A}, T_D)$, are defined as

$$\begin{aligned} E(\mathcal{A}, T_D) &:= \frac{1}{n \cdot T_D} \sum_{v \in V} (e_{init}(v) + T_D \cdot a_m(v)), \\ T(\mathcal{A}, T_D) &:= f(n). \end{aligned}$$

Note that the definition of $E(\mathcal{A}, T_D)$ corresponds to the intuitive notion of *energy efficiency* given above. Particularly, the terms $T_D \cdot a_m(v)$ and $e_{init}(v)$ describe a node v's maintenance and initialization energy during a deployment phase of length T_D, respectively. Adding up these values over all nodes and dividing by $\frac{1}{n \cdot T_D}$, the number of nodes and time-slots leads to the energy efficiency $E(\mathcal{A}, T_D)$. As for the second measure, an algorithm has time efficiency $f(n)$ (for instance n^2) if with high probability, all nodes are notified $f(n)$ time-slots after the launching point.

Definition 1 allows us to compare deployment algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ in two ways. First, we can fix the notification time $f(n)$ and compare both algorithm's energy requirements. That is, we demand two algorithms to finish the notification period within the same amount of time. We then compare which algorithm requires more energy during the deployment phase in order to ensure that all nodes are notified within $f(n)$, i.e., $T_N \leq f(n)$. Alternatively, we can fix the energy consumption $E(\mathcal{A}_1, T_D)$ and $E(\mathcal{A}_2, T_D)$, respectively, of both algorithms and then compare the resulting length of the notification phase. Clearly, both comparison methodologies are two sides of the same coin; they both describe the inherent trade-off between energy efficiency and the rapidity of information dissemination.

IV. Algorithms

In this section, we analyze three different algorithms under our model and derive their respective strengths and weaknesses. We begin our exposition by analyzing the so-called *birthday algorithm* proposed in [18] which can be employed as a algorithm for the deployment of sensor networks. In subsequent Sections IV-B and IV-C, we propose two novel algorithms that significantly outperform [18].

For the analysis of the algorithms we assume time to be divided into synchronized time-slots. However, notice that none of the algorithms relies on this assumption. This simplification of the analysis is justified due to the standard trick introduced in [24] for the study of slotted vs. unslotted ALOHA. In [24], it is shown that the realistic unslotted case and the idealized slotted case differ only by a factor of two. The basic intuition is that a single packet can only cause interference in two consecutive time-slots. By the same token, analyzing the algorithms in an "ideal" setting with synchronized time-slots, we obtain a result which is only a factor two better as compared to the more realistic unslotted setting.

Throughout the paper, we will denote by N_v the set of neighbors of node v, i.e., $N_v = \{u \in V \mid \{u, v\} \in E\}$. Finally, we will make use of the following two facts. The first can be found in standard mathematical textbooks and the second was proven in [13].

Fact 1: For all n,t, with $n \geq 1$ and $|t| \leq n$,

$$e^t \left(1 - \frac{t^2}{n}\right) \leq \left(1 + \frac{t}{n}\right)^n \leq e^t.$$

Fact 2: Given a set of probabilities $p_1 \cdots p_n$ with $\forall i : p_i \in [0, \frac{1}{2}]$, the following inequalities hold:

$$\left(\frac{1}{4}\right)^{\sum_{k=1}^n p_k} \leq \prod_{k=1}^n (1 - p_k) \leq \left(\frac{1}{e}\right)^{\sum_{k=1}^n p_k}.$$

A. Birthday Algorithm

The birthday algorithm $\mathcal{A}_{birth}$ proposed in [18] is conceptually simple. During the deployment phase, before being notified, a node v listens in each time-slot with probability p_L and sleeps with probability $1 - p_L$. Once v has learned about the launching point in the notification phase, it sends with probability p_T which is set to $1/n$ and listens with probability p_L. The choice of the sending probability is motivated by the goal to avoid interference in the case when several already notified nodes try to send a message to a common neighbor. Clearly, the broad idea of the algorithm is to let nodes sleep as long as possible. That is, we want to choose p_L as small as possible while still guaranteeing a speedy notification phase.

$\mathcal{A}_{birth}$ has been designed and analyzed for neighborhood discovery, i.e., not for the deployment problem as considered in this paper. In this section, we will analyze the birthday algorithm's performance in the context of the problem of sensor network deployment. Specifically, we analyze the trade-off exhibited by $\mathcal{A}_{birth}$ in accordance to the definitions given in Section III.

Let $f(n)$ be the time in which we require the notification procedure to finish with high probability, that is, let $f(n)$ be a function such that $T_N(\mathcal{A}_{birth}) \leq f(n)$ with high probability. Given this constraint, we want to optimize the algorithm's energy efficiency. The achievable trade-off is expressed in the following theorem.

Theorem 1: Let $f(n)$ be a function such that the *birthday algorithm* $\mathcal{A}_{birth}$ has time efficiency $T(\mathcal{A}_{birth}, T_D) \leq f(n)$. For arbitrary T_D, $\mathcal{A}_{birth}$'s energy efficiency is

$$E(\mathcal{A}_{birth}, T_D) \ \in \ \Theta\left(\frac{n^2}{f(n)}\right).$$

Proof: The birthday algorithm does not require any initialization and therefore, $e_{init}(v) = 0$, for all $v \in V$. The average maintenance energy for each node corresponds directly to the listening probability, i.e., $a_m(v) = p_L$. Hence, the algorithm's energy efficiency is

$$E(\mathcal{A}_{birth}, T_D) \ = \ \frac{1}{n \cdot T_D} \sum_{v \in V} (T_D \cdot p_L) \ = \ p_L.$$

Consider the network graph $G_b = (V_b, E_b)$ consisting of nodes $v_1, \ldots, v_n$ positioned in a line, i.e., v_i is a neighbor of v_j iff $j = i+1$ and $1 < j \leq n$. Recall that the nodes themselves have no knowledge about the topology of the network. Finally, let v_0 be the node which is externally triggered at the launching point.

By the construction of G_b, the information about the arrival of the launching point has to traverse the entire network in a hop-by-hop fashion. We call a time-slot t *successful*, if there is a notified node v_i that sends in t and its unaware neighboring node v_{i+1} listens at the same time. Informally, the notification information is passed on by one hop in each successful time-slot.

The probability P_{suc} that a time-slot t is successful is $P_{suc} = p_L \cdot p_T$. In order to pass the notification through the entire chain, a minimum of $n-1$ successful time-slots are required. In total, the algorithm is allowed to use $f(n)$ time-slots and the broadcast has to succeed with probability at least $1 - 1/n$. Given these constraints, we want to minimize p_L thus optimizing $E(\mathcal{A}_{birth}, T_D)$. In expectation, the number of successful rounds is $p_L p_T f(n)$. Since we want at least $n-1$ successes, it follows that

$$\frac{p_L f(n)}{n} \ = \ n - 1 \quad \Rightarrow \quad p_L \ \in \ \Omega\left(\frac{n^2}{f(n)}\right).$$

Finally, we show that for a large enough constant c, $p_L = cn^2/f(n)$ is enough to obtain the high probability argument. Let X be the number of successful rounds. The expected value of X is $\mu = p_L f(n)/n$. We bound the probability of having less than $n-1$ successful rounds using Chernoff Bounds as

$$\begin{aligned} P[X < n-1] \ &= \ P\left[X < \left(1 - \left(1 - \frac{n(n-1)}{p_L f(n)}\right)\right)\frac{p_L f(n)}{n}\right] \\ &< \ e^{-\frac{p_L f(n)}{2n}\left(1 - \frac{n(n-1)}{p_L f(n)}\right)^2} = e^{-\frac{cn}{2}\left(1-\frac{1}{c}\right)^2}, \end{aligned}$$

which is smaller than $1/n$ for a suitably large constant c. Notice that setting p_L to a value strictly smaller, i.e., $p_L \in o(n^2/f(n))$ renders the exponent positive thus not yielding the desired result. □

Algorithm $\mathcal{A}_{uni}$

```
upon wake-up do:
1: listen with probability $p_L$, otherwise sleep
upon notification do:
2: for $i := \lceil \log n \rceil + 1$ to 1 by $-1$ do
3:     $p_T := 1/2^i$
4:     for $\frac{c(\lceil \log n \rceil + 1)}{p_L}$ time-slots do
5:         send message with probability $p_T$
6:     end for
7: end for
```

Keep in mind that for the birthday algorithm, the notification phase T_N must be at least of length $\Omega(n^2)$ in order to guarantee a feasible solution. In the following two sections, we will propose algorithms featuring strictly better trade-offs.

B. Uniform Algorithm

In a way, the second algorithm $\mathcal{A}_{uni}$ shares the philosophy of the birthday algorithm, having in common that there are no initialization costs and all nodes perform the same procedure uniformly. Specifically, algorithm $\mathcal{A}_{uni}$ has one input parameter, the listening probability p_L; c is a constant to be defined later.

The main improvement is a simple idea originally stemming from the literature on broadcast in radio networks [2]. When trying to inform an unaware node, notified nodes will exponentially increase their sending probability thus reducing the average waiting time. Notice that the number of time-slots per sending probability is inversely proportional to the unaware node's listening probability p_L. In comparison with the birthday algorithm $\mathcal{A}_{birth}$ analyzed in Section IV-A, $\mathcal{A}_{uni}$ exhibits a strictly better performance trade-off as stated in Theorem 2.

Theorem 2: Let $f(n)$ be a function such that the *uniform algorithm* $\mathcal{A}_{uni}$ has time efficiency $T(\mathcal{A}_{uni}, T_D) \leq f(n)$. For arbitrary T_D, $\mathcal{A}_{uni}$'s energy efficiency is at most

$$E(\mathcal{A}_{uni}, T_D) \ \in \ O\left(\frac{n \log^2 n}{f(n)}\right).$$

Proof: Like $\mathcal{A}_{birth}$, $\mathcal{A}_{uni}$ does not require any initialization and all nodes are treated uniformly. Therefore, by the same argument as in Section IV-A, $E(\mathcal{A}_{uni}, T_D) = p_L$.

We define the listening probability p_L to be $p_L := cn(\lceil \log n \rceil + 1)^2/f(n)$. We seek to show that for a constant $c \geq 12$, the probability of the notification message advancing at least *one hop* in time $O(f(n)/n)$ is at least $1 - n^{-2}$. Since the diameter of the network is at most n, the theorem follows from $(1 - n^{-2})^n \geq e^{-1/n} \geq 1 - n^{-1}$.

Let $Z_{v,t}$ denote the event of node v hearing a notification message in time-slot t. Consider an unaware node $v \in V$ and let t_0 be the first time-slot in which at least one node in v's neighborhood N_v is notified. Starting from this round, the sum of sending probabilities $\sum_{w \in N_v} p_T(w)$ increases. Let t^* be the last time-slot in which the sum of sending probabilities is smaller than $1/2$. Notice that it takes at most $t^* - t_0 \leq (\lceil \log n \rceil + 1) \cdot \frac{c(\lceil \log n \rceil + 1)}{p_L}$ time-slots until t^* is reached.

Now, consider the time interval $\mathcal{I} = [t^* + 1, \ldots, t^* + \frac{c(\lceil \log n \rceil + 1)}{p_L}]$. During this interval, notified nodes can at most double their p_T and new nodes will send with the initial sending probability $p_T = \frac{1}{2n}$. At the end of this interval, the sum of sending probabilities is therefore at most

$$\sum_{w \in N_v} p_T(w) \ \leq\ 2 \cdot \frac{1}{2} + \sum_{w \in N_w} \frac{1}{2n} \ \leq\ \frac{3}{2}. \qquad (1)$$

Therefore, in each time-slot $t \in \mathcal{I}$, the sum of sending probabilities is at least $1/2$ and at most $3/2$. The probability $P[Z_{v,t}]$ that v receives the notification message from one of its neighbors is

$$\begin{aligned} P[Z_{v,t}] &= p_L \sum_{w \in N_v} \left(p_T(w) \cdot \prod_{\substack{q \in N_v \\ q \neq w}} (1 - p_T(q)) \right) \\ &\geq p_L \sum_{w \in N_v} p_T(w) \cdot \prod_{q \in N_v} (1 - p_T(q)) \\ &\underset{\text{Fact 2}}{\geq} p_L \sum_{w \in N_v} p_T(w) \cdot \left(\frac{1}{4}\right)^{\sum_{q \in N_v} p_T(q)} \\ &\geq \frac{3p_L}{2} \cdot \left(\frac{1}{4}\right)^{3/2} \ >\ \frac{p_L}{6}. \end{aligned}$$

For large enough functions $f(n)$ and $p_L = cn(\lceil \log n \rceil + 1)^2/f(n)$, the probability that none of the $\frac{c(\lceil \log n \rceil + 1)}{p_L}$ time-slots $t \in \mathcal{I}$ is successful is at most

$$\begin{aligned} P[\cap_{t \in \mathcal{I}} \overline{Z_{v,t}}] &\leq \left(1 - \frac{p_L}{6}\right)^{\frac{c(\lceil \log n \rceil + 1)}{p_L}} \\ &= \left(1 - \frac{cn(\lceil \log n \rceil + 1)^2}{6f(n)}\right)^{\frac{f(n)(\lceil \log n \rceil + 1)}{n(\lceil \log n \rceil + 1)^2}} \\ &\underset{\text{Fact 1}}{\leq} e^{-\frac{c}{6}(\lceil \log n \rceil + 1)} \ <\ n^{-2}. \end{aligned}$$

Therefore, with probability exceeding $1 - n^{-2}$, the notification message is passed on at least by one hop in time

$$t^* - t_0 \ \leq\ (\lceil \log n \rceil + 1) \cdot \frac{cf(n)(\lceil \log n \rceil + 1)}{cn(\lceil \log n \rceil + 1)^2} \ =\ \frac{f(n)}{n}.$$

Consequently, by the argument given at the beginning of the proof, the notification message reaches all n nodes within time $f(n)$ with probability at least $1 - \frac{1}{n}$. □

The trade-off obtained by $\mathcal{A}_{uni}$ is strictly better than the one obtained by the birthday algorithm in Section IV-A. Moreover, in the case $p_L = 1$, the algorithm allows a feasible solution for functions $f(n) \in \Omega(n \log^2 n)$ as opposed to $f(n) \in \Omega(n^2)$ for the birthday algorithm.

Algorithm $\mathcal{A}_{clu}$**:** Code for non-leader u

```
upon wake-up do:
 1: perform MIS algorithm of length $O(W + \log^2 n)$ $\rightarrow$ decide
    on leader $s(u)$, receive wake-up point $r_3$
 2: loop
 3:   sleep until next wake-up point $r_3$
 4:   for $\eta \log n$ time-slots listen for notification message $\mathcal{M}_n$
 5:   $r_3 := r_3 + I$
 6: end loop
upon notification do:
 7: loop
 8:   upon receiving $\mathcal{M}_a(r_2)$, wait until $r_2$                } $S_1$
 9:   for $i := \lceil \log n \rceil + 1$ to 1 by $-1$ do                  ⎫
10:     for $(\gamma + \eta)(\lceil \log n \rceil + 1)$ time-slots do      ⎪
11:       send message with probability                                    ⎬ $S_2$
          $p_T = 1/2^i$                                                    ⎪
12:       upon receiving $\mathcal{M}_r$, quit for-loops                   ⎪
13:     end for                                                            ⎪
14:   end for                                                              ⎭
15: end loop
```

C. Cluster Algorithm

Finally, our last algorithm is based on a different paradigm than the two previous ones. Instead of treating all nodes identically (uniformly), it forms a *semi-structure* that renders the notification of nodes during the notification phase quicker. On the other hand, installing and maintaining this structure requires additional energy during the deployment phase. Contrary to the first two algorithms, the cluster algorithm $\mathcal{A}_{clu}$ has non-zero initialization costs $e_{init}(v)$ and unequal energy requirements between different nodes. Therefore, $\mathcal{A}_{clu}$ uses the full potential of Definition 1.

The design of $\mathcal{A}_{clu}$ aims to mend the main dissipation of energy of the two previous algorithms, the lack of synchronization. If neighboring nodes had synchronized wake-up points, the notification phase would take significantly less time. Consequently, when demanding the same notification efficiency T_N, the nodes could sleep longer, thus saving energy during the deployment phase. The problem is that synchronization between neighboring nodes incurs additional set-up and maintenance costs and the question is whether these additional costs will equiponderate the gains stemming from the above mentioned notification speed-up.

Our approach is based on grouping neighboring nodes into synchronized clusters. Within such a cluster, nodes wake-up at the same time. In particular, the algorithm constructs a clustering based on a *maximal independent set* of the underlying network graph $G = (V, E)$. An independent set S of G is a subset of V such that $\forall u, v \in S, (u, v) \notin E$. S is a *maximal independent set* (MIS) if any node v not in S has a neighbor in S. In our particular case, we do not consider a MIS on the original graph G, but we consider a MIS of the graph G' in which two nodes are adjacent if their mutual distance is at most $1/2$. This corresponds to each node setting its transmission range to $1/2$.

Constructing a MIS efficiently in an unstructured radio network is a non-trivial task. Our clustering algorithm for the deployment problem uses a MIS algorithm proposed in [19]. It is important to note, however, that any other MIS algorithm in the unstructured radio network model can instead be used without affecting the asymptotic energy efficiency of algorithm $\mathcal{A}_{clu}$. We now introduce an adaptation of this algorithm to a level of detail necessary to understand our results.

Each node starts executing the algorithm upon waking up. Nodes that are located in a region which is already covered by an existing MIS node (leader) will learn about their being covered during an initial waiting period of length W. If this is not the case, v will decide whether it joins the MIS or not during the second phase of length $O(\log^2 n)$ time-slots. Hence, in total, every node needs to be awake for $W + O(\log^2 n)$ time-slots before deciding on whether it becomes a leader or not. Subsequently, for the entire duration of the deployment phase, leaders have to transmit with a sending probability of $\Theta(\log n/W)$ in order to inform newly awakening nodes of their being covered. This prevents nodes that wake up later from invalidating the MIS condition. Non-leader nodes do not have any duties and can sleep arbitrarily long. Let $s(u)$ denote the leader of node u and if $u \in S$ let $S(u)$ refer to the set of nodes having u as their leader, i.e., $S(u) = \{v | u = s(v)\}$ for all $u \in S$.

We incorporate a slightly stronger version of the result in [19] into our algorithm $\mathcal{A}_{clu}$. Particularly, we require that the set S is connected if we consider all two-hop paths in G. Note that this condition is automatically fulfilled if the network density is reasonably high (for instance, if there is at least one node in every disk of radius $1/4$ in the convex hull of the nodes). In $\mathcal{A}_{clu}$ each leader $v \in S$ coordinates the nodes in $S(v)$ and is responsible for their synchronized waking up. Specifically, a leader v decides on the timing of the *rendezvous windows* for its cluster; a time window during which the nodes $w \in S(v)$ are simultaneously awake. Every node $w \in S(v)$ learns the timing of these rendezvous windows from its leader v. The idea is that once a leader is notified, it can notify all nodes in its cluster at almost the same time.

Each rendezvous takes place in three steps as shown in Figure 2. In the *proclamation step* S_1, leader v announces the rendezvous interval to neighboring nodes which do *not* belong to $S(v)$. The reason is that once a node is notified, it remains listening on the channel. Such a node must be able to notify neighboring leaders, even if it is in a different cluster itself (otherwise, the notification message would not broadcast through the network). In other words, the proclamation step is intended for announcing the notification across cluster boundaries. The conveyance of the notification message in the opposite direction is the aim of the second step, the *leader-notification step* S_2. In this step, already notified nodes try to notify a neighboring unaware leader.

Finally, the rendezvous is concluded by the *notification step* S_3. A notified leader v attempts to notify all unaware nodes in

```
Algorithm $\mathcal{A}_{clu}$: Code for leader $v$

upon wake-up do:
 1: perform MIS algorithm of length $O(W + \log^2 n)$ $\rightarrow$
    become leader with cluster S(v)
 2: choose rendezvous point $r_1$
 3: $r_2 := r_1 + \eta\lceil\log n\rceil$, $r_3 := r_2 + (\gamma + \eta)\lceil\log^2 n\rceil$
 4: loop
 5:   sleep or send with probability
      $\log n/W$ until next wake-up point $r_1$
 6:   for $\eta \log n$ time-slots send $\mathcal{M}_a(r_2)$            } $S_1$
      with probability $p_{MIS} \in \Theta(1)$                         }
 7:   for $(\gamma + \eta)(\lceil\log n\rceil + 1)^2 - \eta(\lceil\log n\rceil)$   }
      time-slots listen for $\mathcal{M}_n$                            }
 8:   if $\mathcal{M}_n$ received then                                } $S_2$
 9:     send $\mathcal{M}_r$ for $\eta \log n$ time-slots with         }
        probability $p_{MIS}$                                          }
10:   end if                                                           }
11:   sleep until $r_3$
12:   if notified then                                                 }
13:     for $\eta \log n$ time-slots send $\mathcal{M}_n$              }
        with probability $p_{MIS}$                                     } $S_3$
14:     become non-leader                                              }
15:   end if                                                           }
16:   $r_1 := r_1 + I$
17:   $r_2 := r_1 + \eta\lceil\log n\rceil$
18:   $r_3 := r_2 + (\gamma + \eta)\lceil\log^2 n\rceil$
19: end loop
```

$S(v)$ during this step. Note that this is the only time-interval during which an unaware non-leader node must be awake. Summarizing, the actions during the rendezvous window are designed as to guarantee that a notification message in the neighborhood of a leader v is, first, passed to v, and second, passed from v to all nodes in $S(v)$. After the rendezvous window, a notified leader becomes a non-leader node in order to help informing other leaders located in its neighborhood. Finally, notice that all transmissions during a rendezvous are performed using the full transmission range. In the following, we give a more precise description of algorithm $\mathcal{A}_{clu}$ as performed by leaders and non-leaders, in which γ and η are suitably large constants.

Consider a rendezvous window of leader v. In the *proclamation step* S_1, v sends an announcement message $\mathcal{M}_a(r_2)$ containing the starting time of the second step of the rendezvous with a constant probability $p_{MIS} \in \Theta(1)$. Let u be a notified node with $(u, v) \in E$ and $u \notin S(v)$. Notified nodes remain listening in order to eavesdrop an announcement message of neighboring leaders. If node u receives such a message $\mathcal{M}_a(r_2)$ from v, it tries to notify v during the subsequent *leader-notification step*. In the analysis, we will show that with high probability *every* notified node in v's neighborhood will receive $\mathcal{M}_a(r_2)$ from v.

In the *leader-notification step* S_2 all notified neighbors of v try to send a notification message $\mathcal{M}_n$ to v. Notice that if there

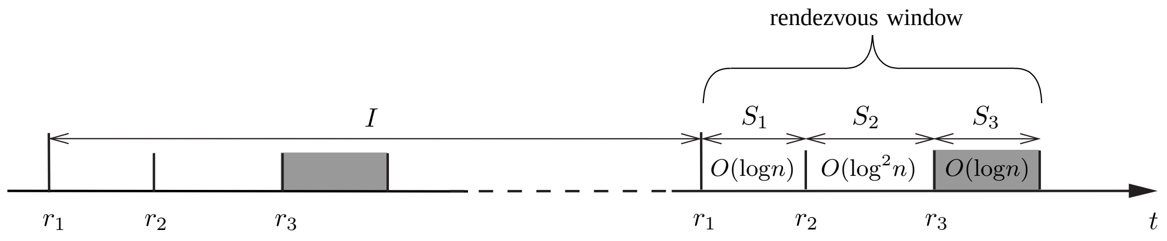


Fig. 2. Rendezvous interval I and rendezvous window

are no notified neighbors of v, nothing happens during the *leader-notification step*. The procedure of informing a leader follows along the lines of the uniform algorithm presented in Section IV-B. Starting with probability $\frac{1}{2n}$, notified nodes exponentially increase their sending probability to speed up the notification. In order to prevent too much "noise" (i.e., too many nodes sending with high probability at the same time), v starts sending a reception message $\mathcal{M}_r$ with probability p_{MIS} as soon as it has received $\mathcal{M}_n$. In the analysis, we show that the $O(\log^2 n)$ time-slots are sufficient to perform these tasks with high enough probability.

Finally, unaware nodes in $S(v)$ are only awake in the *notification step* S_3 starting from r_3. They are listening during these time-slots, waiting for a possible notification message $\mathcal{M}_n$ from a potentially notified v.

Analysis: In the following, we will sometimes omit calculating the exact values of the various constants involved for the sake of clarity and due to lack of space. Instead, we focus our attention on portraying the main ideas and concepts of our algorithm and proofs. Exact constants can be derived by a more rigorous analysis in a straightforward way.

We begin with a simple geometric lemma, saying that the number of leaders (and corresponding clusters) in any disk of radius 1 is bounded by a constant.

Lemma 3: Let v be an arbitrary node. Let $Q := \{s(u) \mid u \in N_v\}$ be the set of all leaders that lead at least one node in v's neighborhood. It holds that $|Q| \leq \varphi$ for a constant φ.

Proof: The proof follows from a simple area argument. There cannot be more than a constant number of disks of radius $1/4$ packed into a disk of radius 1 such that no two disks overlap. □

In the following, let $p_v(t)$ be the sending probability of node v in time-slot t. Further, $\Phi_v(t)$ denotes the sum of the sending probabilities of neighbors of v that are not leaders, formally

$$\Phi_v(t) := \sum_{u \in N_v \setminus S} p_u(t).$$

In the next lemma, we show that given an upper bound on $\Phi_v(t)$, $\eta \log n$ time-slots are sufficient to let a leader inform all its neighbors. Because of cyclic dependencies, it is convenient to formulate this upper bound on $\Phi_v(t)$ as an invariant.

Invariant 1: Let t be an arbitrary time-slot. For all leaders $v \in S$, it holds that $\Phi_v(t) \leq \chi$, for a constant $\chi \leq \frac{3\varphi}{2}$.

Lemma 4: Let v be a leader and consider a time interval $\mathcal{J}$ of length $\eta \log n$ during which v sends with probability p_{MIS}. Under the condition that Invariant 1 holds, all nodes $w \in S(v)$ receive the message during $\mathcal{J}$ with probability $1 - n^{-3}$.

Proof: Let N_v^2 denote the set of nodes which are in distance at most 2 of v. We call a time-slot successful if v sends, but no other node in N_v^2 sends. In a successful time-slot, all nodes in N_v receive the message from v without collision. The probability $P_{suc}(t)$ that a single time-slot t is successful is at least

$$\begin{aligned} P_{suc}(t) &\geq p_{MIS} \cdot \prod_{\substack{w \in N_v^2 \\ w \neq v}} (1 - p_w(t)) \\ &\underset{\text{Lm 3}}{\geq} p_{MIS} \cdot (1 - p_{MIS})^{\varphi - 1} \prod_{w \in N_v^2 \setminus S} (1 - p_w(t)) \\ &\underset{\text{Fact 2}}{\geq} p_{MIS} \cdot (1 - p_{MIS})^{\varphi - 1} \left(\frac{1}{4}\right)^{\sum_{w \in N_v^2 \setminus S} p_w(t)} \\ &\geq p_{MIS} \cdot (1 - p_{MIS})^{\varphi - 1} \left(\frac{1}{4}\right)^{\chi\varphi} \in \Theta(1). \end{aligned}$$

where the last inequality follows from Lemma 3 and Invariant 1 which holds by assumption. Finally, the probability P_{no} that none of the $\eta \log n$ time-slots is successful is bounded by

$$\begin{aligned} P_{no} &\leq \left(1 - p_{MIS}(1 - p_{MIS})^{\varphi - 1} \left(\frac{1}{4}\right)^{\chi\varphi}\right)^{\eta \log n} \\ &\leq \frac{1}{2n^3} \end{aligned}$$

for a suitably large constant η. □

Unfortunately, Lemma 4 holds only conditionally; based on the assumption that Invariant 1 holds. In the following, we prove this invariant by placing an upper bound on $\Phi_v(t)$ that holds throughout the execution of the algorithm with high probability.

Lemma 5: With probability $1 - n^{-2}$, it holds for all t and for all leaders $v \in S$ that $\Phi_v(t) \leq \chi$, where $\chi \leq \frac{3\varphi}{2}$ is a constant, i.e., Invariant 1 holds.

Proof: At the beginning of the notification phase, Invariant 1 clearly holds. For the sake of contradiction, assume that leader v is the first to violate the invariant. Further, notice that Φ_v can only increase if some of its neighboring non-leader nodes are in the *leader-notification step* S_2. The idea is that as soon as v receives the notification message, it starts sending a reception message $\mathcal{M}_r$. We will show that the nodes in N_v receive this message and stop sending. This prevents Φ_v from increasing too much.

We define time-slots t_v^* for a leader v, such that, $\Phi_v(t_v^*) < 1/2$ and $\Phi_v(t_v^*+1) \geq 1/2$. By the same argument as in the proof of Theorem 2 (cf. Inequality (1)), we can bound $\Phi_v(t_v^*+(\gamma+\eta)(\lceil\log n\rceil+1)) \leq 3/2$. That is, for all time-slots t in the interval $\mathcal{J} = [t_v^*+1, \ldots, t_v^*+(\gamma+\eta)(\lceil\log n\rceil+1)]$, it holds that $1/2 < \Phi_v(t) \leq 3/2$. The probability $P_{suc}(t)$ that v receives a message without collision in an arbitrary time slot $t \in \mathcal{J}$ is at least

$$\begin{aligned} P_{suc}(t) \;&\geq\; \prod_{w\in S\cap N_v}(1-p_w(t)) \\ &\quad\cdot \sum_{w\in N_v\setminus S}\left(p_w(t)\cdot \prod_{\substack{q\in N_v\setminus S\\ q\neq w}}(1-p_q(t))\right) \\ &\geq\; (1-p_{MIS})^{\varphi}\cdot \Phi_w(t)\left(\frac{1}{4}\right)^{\Phi_w(t)} \\ &\geq\; (1-p_{MIS})^{\varphi}\cdot \frac{3}{2}\left(\frac{1}{4}\right)^{3/2} > \frac{(1-p_{MIS})^{\varphi}}{6}. \end{aligned}$$

We continue the proof by showing that with high probability, the first $\gamma(\lceil\log n\rceil+1)$ time-slots of $\mathcal{J}$ suffice such that v receives $\mathcal{M}_n$. Specifically, the probability P_{no} that none of these time-slots is successful is

$$P_{no} \;\leq\; \left(1-\frac{(1-p_{MIS})^{\varphi}}{6}\right)^{\gamma(\lceil\log n\rceil+1)}, \qquad (2)$$

which again can be made $P_{no} \leq n^{-3}/2$ for large enough constants γ. Once, node v receives $\mathcal{M}_n$, it will try to acknowledge by sending $\mathcal{M}_r$. Notice that there are at least $\eta(\lceil\log n\rceil+1)$ time-slots in $\mathcal{J}$ left during which $1/2 < \Phi_v(t) \leq 3/2$. By the assumption that v is the *first* leader to violate Invariant 1, we know that until the end of $\mathcal{J}$, Invariant 1 and consequently Lemma 4 hold. That is, with probability at least $1-n^{-3}$, the message $\mathcal{M}_r$ will be received by all nodes in N_v within the $\eta(\lceil\log n\rceil+1)$ time-slots. Hence, the probability that v is the first node to violate Invariant 1 is bounded by $2\cdot n^{-3}/2$ for suitably large constants γ and η. Because there are at most n leaders in the network and every leader needs to be notified only once, the Lemma holds with probability $1-n^{-2}$. □

The following Corollary is implicit in the proof of Lemma 5 (cf, Inequality (2)).

Corollary 6: Consider a leader v and the leader-notification step S_2 of a notification window. If there exists a notified node in $N_v \setminus S$, v will be notified at the end of S_2.

Thanks to Lemma 5, we can now apply Lemma 4 throughout the algorithm with high probability.

Theorem 7: With probability at least $1-1/n$, the algorithm works as demanded, that is, each leader v successfully announces to all its neighbors about the proclamation step S_1 for the entire duration of the notification phase. Furthermore, as soon as there exists a notified non-leader in N_v, v will be notified in the following leader-notification step S_2. And finally, a notified leader v will inform all its neighbors $u \in N_v$ in the notification-step S_3 following v's notification.

Proof: The steps S_2 and S_3 follow directly from Lemma 4, Corollary 6, and the fact that there are at most n leaders, each of which is notified at most once. By Lemma 4, every attempt of sending a $\mathcal{M}_a$ message is successful with probability $1-n^{-3}$. Each of the n nodes needs to send at most n messages $\mathcal{M}_a$ during the notification phase. The proof is concluded because the set of leaders is connected if we consider all two-hop paths in G. □

Of particular interest is the energy efficiency and its comparison to the two previous algorithms. Let $m \leq n$ be the number of leader nodes in the network and let ξ denote the energy efficiency $E(\mathcal{A}_{clu}, T_D)$. Clearly, the ratio m/n depends on the *density* of the network. The following theorem quantifies the achieved trade-off.

Theorem 8: Let $f(n)$ be a function such that algorithm $\mathcal{A}_{clu}$ has time efficiency $T(\mathcal{A}_{clu}, T_D) \leq f(n)$. Let m be the number of *leaders* chosen by $\mathcal{A}_{clu}$. For a given T_D, $\mathcal{A}_{clu}$'s energy efficiency $\xi = E(\mathcal{A}_{clu}, T_D)$ is bounded by

$$\xi \;\in\; O\left(\frac{\frac{f(n)}{n\log n}+\log^2 n}{T_D}+\frac{n\log n}{f(n)}+\frac{m\log^2 n}{f(n)}\right).$$

Proof: The choice of I's length determines the trade-off between energy-efficiency and the speed of notification. We have to choose I such that with high probability, the notification broadcast is finished within time $f(n)$. We do so by setting I to a value guaranteeing that the notification proceeds at least one hop in time $f(n)/n$ with high probability. That is, we set $I := \lceil f(n)/n\rceil$.

Upon waking up, each node v has initialization costs $e_{init}(v) \in O(W+\log^2 n)$. For the maintenance costs during the deployment phase, we distinguish between leaders and non-leader nodes. Non-leaders are awake for the duration of $\eta\log n$ during each rendezvous interval of length I. Thus, for non-leaders, $a(v) = \eta\lceil\log n\rceil/I$. Leaders must be awake longer in each rendezvous interval, namely $2\eta\log n + (\gamma+\eta)(\lceil\log n\rceil+1)^2 \in O(\log^2 n)$ time-slots. Additionally, leaders need to send with probability $\log n/W$ in each time-slot.

Therefore, for appropriate constants α, $\delta > \gamma+\eta$, and η the energy efficiency $E(\mathcal{A}_{clu}, T_D)$ of $\mathcal{A}_{clu}$ is at most

$$\begin{aligned} \xi \;=\; &\frac{1}{nT_D}\left[\sum_{v\in S}\alpha\left(W+\log^2 n\right)\right. \\ &+ T_D\sum_{v\in S}\left(\frac{\log n}{W}+\frac{\delta\log^2 n}{I}\right) \\ &\left.+ \sum_{v\in V\setminus S}\left(\alpha(W+\log^2 n)+\frac{T_D\eta\log n}{I}\right)\right]. \end{aligned}$$

Setting $I = \lceil f(n)/n \rceil$ and $W = I/\log n$, we obtain

$$\begin{aligned}
\xi &= \frac{\alpha(W + \log^2 n)}{T_D} + \frac{m}{n}\left(\frac{\log n}{W} + \frac{\delta \log^2 n}{I}\right) \\
&\quad + \frac{n-m}{n} \cdot \frac{\eta \log n}{I} \\
&\leq \frac{\alpha\left(\frac{f(n)}{n \log n} + \log^2 n\right)}{T_D} + \frac{m}{n} \cdot \frac{(\delta+1) n \log^2 n}{f(n)} \\
&\quad + \frac{n-m}{n} \cdot \frac{\eta n \log n}{f(n)} \\
&\in O\left(\frac{\frac{f(n)}{n \log n} + \log^2 n}{T_D} + \frac{m \log^2 n}{f(n)} + \frac{n \log n}{f(n)}\right).
\end{aligned}$$

□

Observe that the first asymptotic term of Theorem 8 contains T_D in the denominator. This highlights the notion that the amount of energy spent on initializing a structure weighs more or less heavily, depending on the respective length of the deployment phase. Specifically, this term can be neglected if the deployment phase is long. As for the two remaining terms, they express the energy efficiency of leaders and non-leaders, respectively.

D. Discussion

In this section, we discuss the results obtained in Theorems 1, 2, and 8. These theorems yield a concise comparison between the three algorithms analyzed in this paper.

For the comparison, we demand all three algorithms to finish their notification phase within a fixed amount of time $f(n) \in \Theta(n^2)$, $f(n)$ being the same for all algorithms. This allows us to compare the energy efficiency $E(\mathcal{A}, T_D)$ each algorithm is required to invest in order to ensure that the notification is finished within time $f(n)$. As mentioned in Section III, we obtain the same results when asking the question the other way around, i.e., when fixing the algorithm's energy efficiency and comparing the resulting time efficiency $T(\mathcal{A}, T_D) = f(n)$. Table I shows the results derived from Theorems 1, 2, and 8 under the assumption that the length of the deployment phase T_D is long enough compared to $f(n)$[2].

First, we emphasize that both $\mathcal{A}_{clu}$ and $\mathcal{A}_{uni}$ significantly outperform $\mathcal{A}_{birth}$, regardless of the network density or, generally, the ratio between leaders vs. non-leaders. It is interesting to study the relative strengths of $\mathcal{A}_{clu}$ and $\mathcal{A}_{uni}$. Asymptotically, the trade-off achieved by $\mathcal{A}_{clu}$ is strictly better than $\mathcal{A}_{uni}$ if $m \in o(n)$, that is, if less than a constant fraction of the nodes are leaders. If, for instance, $m \in O(n/\log n)$, the resulting asymptotic energy-efficiency is $E(\mathcal{A}_{clu}, T_D) \in O(n \log n / f(n))$, which is better than $\mathcal{A}_{uni}$ by a $O(\log n)$ factor. In case the number of leaders is a constant fraction of n, the asymptotic energy efficiency is $O(n \log^2 n / f(n))$, which equals the trade-off achieved by $\mathcal{A}_{uni}$. Hence, depending on

[2]Note that $f(n) \in \Theta(n^2)$ is the smallest value for $f(n)$ so that $\mathcal{A}_{birth}$ is capable of finishing its notification phase within $f(n)$ in arbitrary networks. Similar results as shown in Table I can be obtained for higher values of $f(n)$ in a straightforward way.

Algorithm $\mathcal{A}$	Energy Efficiency $E(\mathcal{A}, T_D)$
$\mathcal{A}_{birth}$ [18]	$\Theta(1)$
$\mathcal{A}_{uni}$	$\Theta(\frac{\log^2 n}{n})$
$\mathcal{A}_{clu}$	$\Theta(\frac{\log n}{n} + \frac{m \log^2 n}{n^2})$

TABLE I

A COMPARISON OF THE ENERGY EFFICIENCY OF THE THREE ALGORITHMS FOR A FIXED $T(\mathcal{A}, T_D) = f(n) \in \Theta(n^2)$ AND LARGE ENOUGH T_D.

the *network density* and the resulting number of leaders, the asymptotic energy efficiency of $\mathcal{A}_{clu}$ is either better or equal than that of $\mathcal{A}_{uni}$. Intuitively, high network densities render the number of leaders m small relative to n and hence, $\mathcal{A}_{clu}$ is more efficient than $\mathcal{A}_{uni}$. That is, the higher the network density, the more worthwhile it becomes to invest initial energy on obtaining a cluster-based semi-structure.

V. Simulations

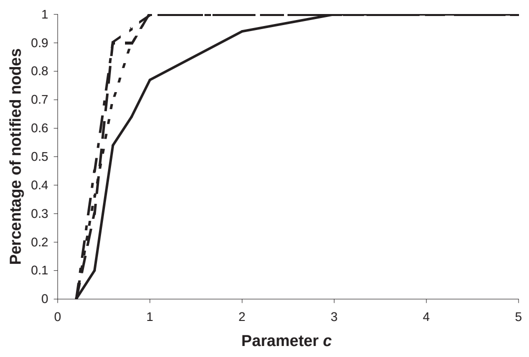
In this section we evaluate the performance of the three algorithms proposed in Section IV on average-case Euclidean graphs, that is on graphs with randomly placed nodes. In particular networks were constructed by placing nodes randomly and uniformly on a square field of size 10 by 10 units and subsequently computing for each node set the Unit Disk Graph—defined such that an edge exists if and only if its Euclidean length is at most one unit. The resulting Unit Disk Graphs were then employed as input networks for the algorithms under consideration.

The two newly introduced algorithms $\mathcal{A}_{uni}$ and $\mathcal{A}_{clu}$ make use of several parameters. In Section IV, an exact value for the parameter c of $\mathcal{A}_{uni}$ is given whereas minute bounds for the parameters of $\mathcal{A}_{clu}$ are omitted for the sake of clarity. However, it is important to notice that Section IV considers a worst-case scenario while we assume average-case networks in this section. Hence, we can presumably set the parameters for the two algorithms to lower values than determined in the previous section.

Figure 3 shows the mean percentage of notified nodes if algorithm $\mathcal{A}_{uni}$ is executed on networks with density 5, and density 20 respectively, against the parameter c ranging from 0 to 5 and various input parameters p_L. The network density is thereby defined as the number of nodes per unit square throughout the rest of this section. Figure 3 leads to the important observation that we can lower the parameter c with decreasing listening probability p_L on condition that all nodes are notified after the termination of $\mathcal{A}_{uni}$. Furthermore, as apparent in Figure 3(a) and 3(b), the parameter c is quite independent of the network density and can thus be chosen exclusively dependent on the input parameter p_L of $\mathcal{A}_{uni}$. As a consequence, we define the parameter c as follows. If the listening probability p_L of algorithm $\mathcal{A}_{uni}$ is above 0.75 we set $c = 3$. If p_L is between 0.5 and 0.75 c is set to 2 and $c = 1$ if p_L is less than 0.5.

Contrary to $\mathcal{A}_{uni}$, algorithm $\mathcal{A}_{clu}$ has more than one parameter, namely γ, η, and p_{MIS}, which leads to multi-

(a)



(b)

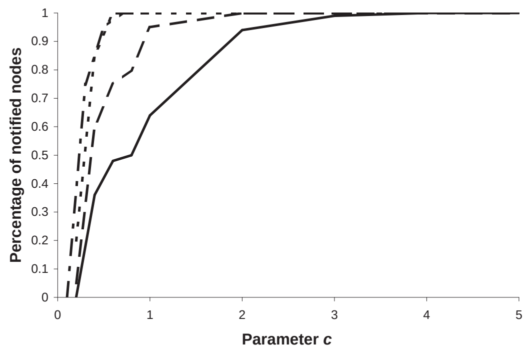
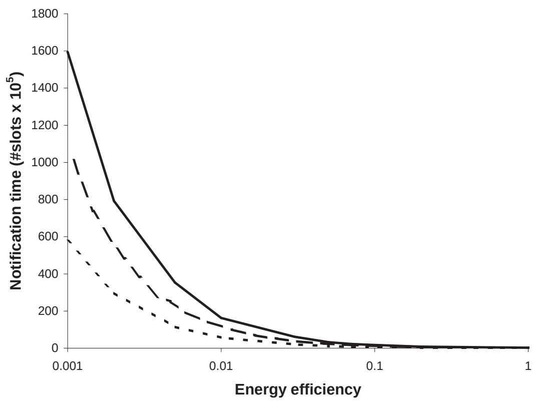
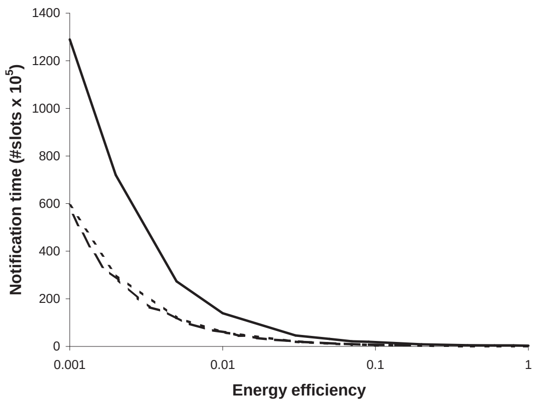


Fig. 3. Percentage of notified nodes for a given c of algorithm $\mathcal{A}_{uni}$. $\mathcal{A}_{uni}$ is thereby simulated with p_L = 1 (solid), 0.5 (dashed), 0.1 (dotted), and 0.01 (dash-dotted) at network densities 5 (a) and 20 (b).

(a)



(b)



(c)

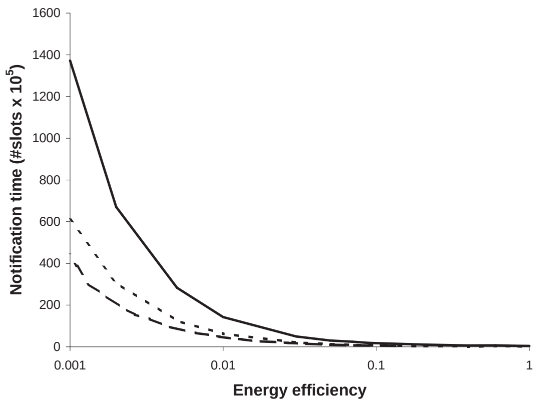


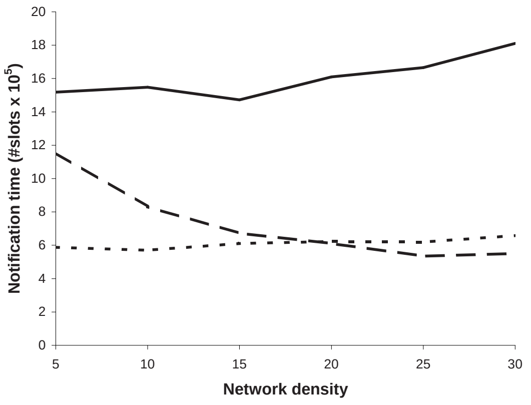
Fig. 4. Mean values of the number of time-slots required to notify a network given a particular energy efficiency. The algorithms $\mathcal{A}_{birth}$ (solid), $\mathcal{A}_{uni}$ (dotted), and $\mathcal{A}_{clu}$ (dashed) are simulated at network densities 5 (a), 15 (b), and 30 (c).

dimensional optimization. By starting with relatively high values for these parameters and reducing them individually until full notification could not be guaranteed with high probability we determined the following values for γ, η, and p_{MIS}: $\gamma = 5$, $\eta = 5$, and $p_{MIS} = 0.2$.

Using the above determined parameters, the respective performance of the three algorithms of Section IV was evaluated by simulating their corresponding notification phase for different node densities and given a particular energy efficiency. The node nearest to the top-left corner was notified by an externally triggered event at the outset of the simulation, i.e., at the launching point. Notice that algorithm $\mathcal{A}_{birth}$, as described in [18], does not terminate after a fixed number of time-slots once a node is notified. We therefore executed $\mathcal{A}_{birth}$ without termination criterion and stopped the simulation series once all node were notified. On the other hand we assumed the deployment phase to be much longer than the notification phase. As a consequence we did not consider the initialization energy of a node since the maintenance energy becomes the dominant factor of energy consumption.

We found that the two newly proposed algorithms $\mathcal{A}_{uni}$ and $\mathcal{A}_{clu}$ outperform algorithm $\mathcal{A}_{birth}$ not only in the worst-case consideration as described in Section IV but also in average case networks (cf. Figure 4). Figure 4(a) depicts the performance of the algorithms on networks with density 5. Algorithm $\mathcal{A}_{uni}$ is able to notify all nodes more than twice as fast than $\mathcal{A}_{birth}$ with high probability. $\mathcal{A}_{clu}$ lies roughly in the middle of the other two. If we consider a network density of 15, $\mathcal{A}_{clu}$ and $\mathcal{A}_{uni}$ need approximately the same number of time-slots to notify all nodes (cf. Figure 4(b)). That is, in comparison to the simulations at network density 5, $\mathcal{A}_{clu}$ is now able to notify the nodes faster with the same energy efficiency. This is due to the fact that with increasing density the ratio between leaders and non-leaders in $\mathcal{A}_{clu}$ decreases and non-leaders spend less energy during the deployment phase than nodes in algorithm $\mathcal{A}_{uni}$. This is exactly the same behavior that we analytically derived in Theorem 8. Considering network density 30, Figure 4(c) shows that the tide has turned in favor of $\mathcal{A}_{clu}$, which now outperforms the

(a)



(b)

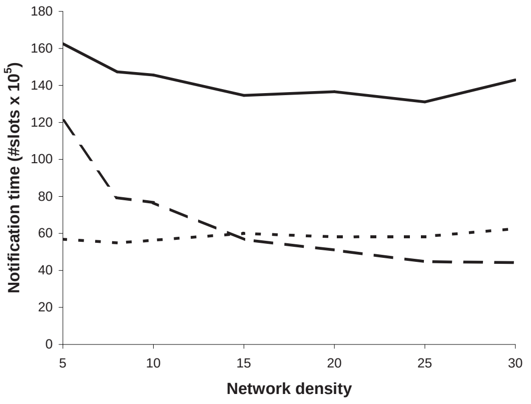


Fig. 5. The number of time-slots required to notify all nodes for algorithm $\mathcal{A}_{birth}$ (solid), $\mathcal{A}_{uni}$ (dotted), and $\mathcal{A}_{clu}$ (dashed) for different network densities. The energy efficiency of the algorithms is thereby 0.1 (a) and 0.01 (b).

other two algorithms.

This results confirm the assumption that algorithm $\mathcal{A}_{uni}$ is well suited for low network densities while $\mathcal{A}_{clu}$ is dedicated for dense networks. In order to investigate more closely the critical network density where $\mathcal{A}_{clu}$ starts to outperform $\mathcal{A}_{uni}$ we simulated the algorithms for a fixed energy efficiency against increasing network densities ranging from 5 to 30 (cf. Figure 5). What strikes from Figure 5 is that the performance of algorithms $\mathcal{A}_{birth}$ and $\mathcal{A}_{uni}$ is more or less independent from the given network density. This is also shown in Figure 4 where the curves for both algorithms, $\mathcal{A}_{birth}$ and $\mathcal{A}_{uni}$, look approximately the same in all three plots. In contrast, the behavior of $\mathcal{A}_{clu}$ is quite different. Since the network density has a direct impact on the ratio of leader to non-leader nodes, its performance is highly dependent on the network density. Figure 5(a) depicts the performance of the three algorithms under the constraint of attaining an energy efficiency of 0.1. If the network density exceeds 19, algorithm $\mathcal{A}_{clu}$ needs less time-slots than $\mathcal{A}_{uni}$ to notify the entire network and should thus be preferred. If we require the algorithms to attain energy efficiency 0.01, $\mathcal{A}_{clu}$ outperforms $\mathcal{A}_{uni}$ already at density 15 (see Figure 5(b)).

These simulations complement the worst-case results derived in Section IV, showing that our algorithms $\mathcal{A}_{uni}$ and $\mathcal{A}_{clu}$ are also efficient in average-case scenarios. Moreover the simulations, in particular Figure 5, give a clear indication as to when the concept of clustering or the usage of semi-structures is worthwhile in the deployment process.

VI. Related Work

To the best of our knowledge, the only previous paper to explicitly address the problem of saving energy *during* the deployment of wireless ad hoc and sensor networks has been [18]. McGlynn and Borbash [18] propose an energy-saving method for performing adjacent neighbor discovery after the deployment of a network. Their algorithm is inspired by the well known *birthday paradox*. Using a similar idea to access the shared medium, a node randomly schedules its periodic wake-up in order to listen for incoming messages. The rest of the time the node powers down its radio subsystem to reduce energy consumption. In Section IV-A, we have given a succinct analysis of the birthday algorithm's performance in the context of the deployment problem.

In the literature on wireless sensor networks, various other problems in the context of *deployment* have been studied. Most notably, several papers have investigated problems related to the *placement* of nodes such that certain coverage requirements be fulfilled. The deployment of mobile nodes for coverage of a sensing field has been considered in [12], [28], [34]. In [4], the problem of covering and exploring an unknown dynamic environment using a mobile robot is addressed. An algorithm for this problem is presented that makes use of a deployed network of radio beacons which assists the robot in coverage. Other work in this area associated with the term *deployment* includes the placement of a given number of sensor nodes to reduce communication cost [14] or an optimal sensor placement for a given target distribution [21].

The model of computation used throughout this paper was introduced in the domain of the *initialization* of wireless radio networks, and in particular ad hoc and sensor networks. Early works on radio networks can for example be found in [2], [13]. Most recently, fast algorithms for computing initial structures from scratch, based on which more sophisticated algorithm can subsequently be applied have been given in [15] and [19].

VII. Conclusions

The trade-off between energy-efficiency and the rapidity of event dissemination lies at the heart of wireless sensor network design. In this paper, we have analyzed this key trade-off in the important non-operational phase by formalizing the problem of sensor network deployment, thus allowing a stringent analysis and comparison of different protocols.

Specifically, we have presented two algorithms, the first being entirely unstructured, the second using the idea of clustering. These algorithms can be regarded as archetypal representatives of an *unstructured* and *semi-structured* approach to the deployment problem, respectively. Interestingly, currently used standard MAC protocols such as B-MAC [22] or S-MAC [32] can be classified into these two approaches.

Specifically, while the B-MAC approach is unstructured, S-MAC sets up some weak notion of clustering during the deployment, i.e., it uses a semi-structure.

Having a formal model that allows comparing these two schemes yields results that bear relevance to theoreticians and practitioners alike, because they give concise and sound answers to the question which deployment algorithm should be employed in a certain application scenario. Furthermore, notice that our results also shed new light into the intriguing question whether and in which cases *clustering* (as opposed to unstructured solutions that do not require any maintenance costs) is really worthwhile. This is of particular interest in view of the multiplicity of clustering algorithms proposed in the recent literature, e.g., [5], [8], [15], [16], [29].

References

[1] S. Banerjee and A. Misra. Minimum Energy Paths for Reliable Communication in Multi-Hop Wireless Networks. In *Proceedings of the 3rd ACM Int. Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, pages 146–156. ACM Press, 2002.

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the Time-Complexity of Broadcast in Radio Networks: an Exponential Gap between Determinism and Randomization. In *Proceedings of the 6th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 98–108, 1987.

[3] S. Basagni, M. Mastrogiovanni, and C. Petrioli. A Performance Comparison of Protocols for Clustering and Backbone Formation in Large Scale Ad Hoc Networks. In *Proceedings of the 1st IEEE Int. Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, 2004.

[4] M. A. Batalin and G. S. Sukhatme. Coverage, Exploration, and Deployment by a Mobile Robot and Communication Network. In *Proceedings of the 2nd Int. Workshop on Information Processing in Sensor Networks (IPSN)*, 2001.

[5] M. Chatterjee, S. K. Das, and D. Turgut. WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. *Journal of Cluster Computing (Special Issue on Mobile Ad Hoc Networks)*, 5(2):193–204, 2002.

[6] C.-F. Chiasserini and R. R. Rao. Routing Protocols to Maximize Battery Efficiency. In *IEEE Milcom 2000*, 2000.

[7] C. F. Chiasserini and R. R. Rao. Improving Energy Saving in Wireless Systems by Using Dynamic Power Management. *IEEE Transactions on Wireless Communications*, 2(5):1090–1100, 2003.

[8] F. Dai and J. Wu. On Constructing k-Connected k-Dominating Set in Wireless Networks. In *Proceedings of the 19th International Parallel & Distributed Processing Symposium (IPDPS)*, 2005.

[9] A. Goel and D. Estrin. Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk. In *Proc. of the 14th Symposium on Discrete Algorithms (SODA)*, 2003.

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33rd Hawaii Int. Conference on System Sciences (HICSS)*, page 8020. IEEE Computer Society, 2000.

[11] J. Hill and D. Culler. Mica: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro.*, 22(6):12–24, 2002.

[12] A. Howard, M. J. Matarić, and G. S. Sukhatme. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem. In *Proceedings of the 6th Int. Conference on Distributed Autonomous Robotic Systems (DARS)*, 2002.

[13] T. Jurdziński and G. Stachowiak. Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks. In *Proceedings of the 13th Int. Symposium on Algorithms and Computation (ISAAC)*, 2002.

[14] T. Kasetkasem and P. K. Varshney. Communication structure planning for multisensor detection systems. *IEE Proc. Radar, Sonar and Navigation*, 148:2–8, 2001.

[15] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proceedings of the 10th Annual Int. Conference on Mobile Computing and Networking (MOBICOM)*, pages 260–274, 2004.

[16] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the Locality of Bounded Growth. In *Proceedings of the 24th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 60–68, 2005.

[17] X.-Y. Li, W.-Z. Song, and W. Wang. A Unified Energy-Efficient Topology for Unicast and Broadcast. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 1–15, 2005.

[18] M. J. McGlynn and S. A. Borbash. Birthday Protocols for Low Energy Deployment and Flexible Neighborhood Discovery in Ad Hoc Wireless Networks. In *Proceedings of the 2nd ACM Int. Symposium on Mobile Ad Hoc Networking & Computing (MOBIHOC)*, 2001.

[19] T. Moscibroda and R. Wattenhofer. Maximal Independent Sets in Radio Networks. In *Proceedings of the 24th ACM Symp. on Principles of Distributed Computing (PODC)*, pages 148–157, 2005.

[20] T. Moscibroda and R. Wattenhofer. Maximizing the Lifetime of Dominating Sets. In *Proceedings of the 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, 2005.

[21] D. E. Penny. The Automatic Management of Multi-Sensor Systems. In *Proceedings of the 1st Int. Conference on Information Fusion (FUSION)*, 1998.

[22] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the 2nd Int. Conference on Embedded Networked Sensor Systems*, pages 95–107. ACM Press, 2004.

[23] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava. Energy-Aware Wireless Microsensor Networks. *IEEE Signal Processing Magazine*, 19(2):40–50, 2002.

[24] L. G. Roberts. Aloha Packet System with and without Slots and Capture. *ACM SIGCOMM, Computer Communication Review*, 5(2):28–42, 1975.

[25] S. Singh and C. S. Raghavendra. PAMAS - Power Aware Multi-Access Protocol with Signalling for Ad Hoc Networks. *SIGCOMM Comput. Commun. Rev.*, 28(3):5–26, 1998.

[26] A. Sinha and A. Chandrakasan. Dynamic Power Management in Wireless Sensor Networks. *IEEE Design and Test*, 18(2):62–74, 2001.

[27] P. von Rickenbach and R. Wattenhofer. Gathering Correlated Data in Sensor Networks. In *Proceedings of the ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2004.

[28] G. Wang, G. Cao, and T. L. Porta. Movement-Assisted Sensor Deployment. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[29] Y. Wang, W. Wang, and X.-Y. Li. Distributed Low-Cost Backbone Formation for Wireless Ad Hoc Networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 2–13, 2005.

[30] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2001.

[31] A. Woo and D.-E. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proceedings of the 7th Annual Int. Conference on Mobile Computing and Networking (MOBICOM)*, pages 221–235. ACM Press, 2001.

[32] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1567–1576, 2002.

[33] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks. In *Proceedings of the 23st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2004.

[34] Y. Zou and K. Chakrabarty. Sensor Deployment and Target Localization Based on Virtual Forces. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.