A distributed approximation scheme for sleep scheduling in sensor networks

Patrik Floréen, Petteri Kaski, and Jukka Suomela
Helsinki Institute for Information Technology HIIT
University of Helsinki, Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
{firstname.lastname}@cs.helsinki.fi

***Abstract*—We investigate the theoretical feasibility of near-optimal, distributed sleep scheduling in energy-constrained sensor networks with pairwise sensor redundancy. In this setting, an optimal sleep schedule is equivalent to an optimal fractional domatic partition of the associated redundancy graph. We present a set of realistic assumptions on the structure of the communication and redundancy relations; for the family of networks meeting these assumptions, we develop an efficient distributed approximation scheme for sleep scheduling. For any $\epsilon > 0$, we demonstrate that it is possible to schedule the sensing activities of the nodes in a local and distributed manner so that the ratio of the optimum lifetime to the achieved lifetime of the network is at most $1+\epsilon$. The computational effort (time, memory and communication) required at each node depends on ϵ and the parameters of the network family, but given so-called anchor nodes (a set of nodes meeting certain density constraints) and locally unique node identifiers, the effort is *independent* of the actual network at hand; in particular, the required effort at each node remains constant as the size of the network is scaled up.**

I. Introduction

This work discusses the problem of scheduling sensing activities in large-scale wireless sensor networks [1]. The objective is to maximise the lifetime of a battery-powered sensor network by letting each node sleep occasionally, subject to the constraint that the active nodes at all times suffice to observe the phenomenon of interest.

Our work shows that under realistic assumptions on the problem structure, near-optimal sleep scheduling is possible in a distributed manner using only local information and coordination. Given so-called anchor nodes and locally unique identifiers, the amounts of local information, computation, memory and communication in each node are bounded by constants. Throughout this work, a "constant" refers to a value that may depend on the parameters of the problem family, but not on the particular problem instance; in particular, a constant is independent of n, the number of nodes in the network.

Distributed approximation schemes have been proposed for other problems related to ad hoc and sensor networks. For example, Kuhn et al. [2] present a distributed algorithm for finding a near-optimal minimum dominating set and maximum independent set in graphs motivated by practical wireless networks. However, to our knowledge there is no previous work on distributed approximation schemes with provable approximation guarantees for sleep scheduling in sensor networks.

A. Redundancy model

We make very few assumptions on the sensor nodes. We do not assume that the sensors know their geographic positions. Neither do we assume any particular knowledge on the monitored environment or a specific model of sensor coverage or radio propagation. In addition to sensors such as motion detectors, for which it makes sense to define the geographic coverage of a particular sensor, we are also interested in applying sleep scheduling to commonly used sensors such as thermometers, for which there is no well-defined area of coverage.

Naturally, this prevents us from using the traditional geometric formulation of sleep scheduling problems, where one ensures that every single point in a two- or three-dimensional space is covered by the disks or balls that represent the ranges of the sensors. However, it is still possible to use sleep scheduling to improve the lifetime of the sensor network. Instead of geometric coverage, we focus on pairwise redundancy [3]–[7]. A pairwise redundancy of nodes u and v means that if node u is active, node v can be asleep and vice versa. For example, measurements at u can be used to accurately predict measurements at v and vice versa [4].

We assume that nodes that can communicate with

each other can also determine whether they are pairwise redundant. The details are beyond the scope of this work, and we simply assume that this redundancy information is available. To give some intuition on this part, we present two examples of possible approaches for finding the pairwise redundancies:

1) A node listens to the radio transmissions of its neighbours, compares the local measurements with the measurements reported by the neighbours, and determines whether the measurements are highly correlated (cf. Koushanfar et al. [4]).
2) A pair of nodes declares that they are pairwise redundant if they seem to be physically close to each other based on received radio signal strength or similar indicator.

B. Redundancy graph and communication graph

To formalise the distributed sleep scheduling problem, we define two undirected graphs, the *communication graph* G and the *redundancy graph* H. The set of nodes V is the same for both graphs; each node $v \in V$ corresponds to a sensor device. The edge sets are denoted by $E(G)$ and $E(H)$, respectively.

In the communication graph G, an edge $\{u, v\} \in E(G)$ indicates that u and v can communicate with constant effort. In the redundancy graph H, an edge $\{u, v\} \in E(H)$ indicates that the nodes u and v are pairwise redundant: if v is active, then u can be asleep and vice versa. We assume that the graph H is a subgraph of G, that is, $E(H) \subseteq E(G)$, reflecting the approaches for detecting redundancy sketched in Sect. I-A.

We say that a set $K \subseteq V$ *dominates* the node $v \in V$ in H if $v \in K$ or there is a node $u \in K$ with $\{u, v\} \in E(H)$. A set $D \subseteq V$ is called a *dominating set* of H if D dominates each $v \in V$. In this work, domination always refers to the redundancy graph H.

In the pairwise redundancy model, the valid sets of active nodes are precisely the dominating sets of H. Indeed, if the nodes $D \subseteq V$ are active and the remaining nodes $V \setminus D$ are asleep, then the set D must be a dominating set of H; conversely, any dominating set of H is a valid set of active nodes.

C. The sleep scheduling problem

The problem of sleep scheduling under pairwise redundancy corresponds to the problem of scheduling dominating sets. That is, the task is to find a collection of dominating sets and associated time periods such that (i) the total length of the time periods is maximised, and (ii) for each node v, the total length of the time periods

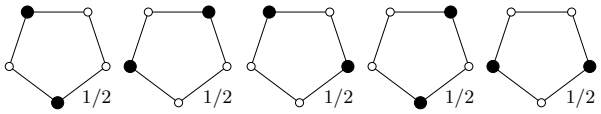

Fig. 1. A sleep schedule of length $5/2$ for a ring of 5 nodes.

associated with the dominating sets that contain v is at most 1. Here 1 is an arbitrary constant; we have chosen the time units so that the battery of a single node lasts for 1 time unit of sensor activity (cf. Sect. VI-C).

The problem can be formulated as a linear program. For a dominating set D, we write $D(v) = 1$ if $v \in D$ and $D(v) = 0$ if $v \notin D$. Denoting by $x(D)$ the length of the time period associated with the dominating set D, the objective is to

$$\begin{aligned} \text{maximise} \quad & \textstyle\sum_D x(D) \\ \text{subject to} \quad & \textstyle\sum_D D(v)x(D) \leq 1 \quad \text{for all } v, \\ & x(D) \geq 0 \quad \text{for all } D, \end{aligned} \tag{1}$$

where v ranges over all nodes and D ranges over all dominating sets in H. In this work, a feasible solution of the above LP is called a *fractional domatic partition* of H and the maximum value of $\sum_D x(D)$ is called the *fractional domatic number* of H [7].

If we require $x(D) \in \mathbb{Z}$, the resulting integer program corresponds to the problem of *domatic partition*, and the optimal value of $\sum_D x(D)$ is the *domatic number* of the graph H. While this classic integral formulation (as well as its generalisation, *set cover packing*) is widely used in the literature in the context of sleep scheduling, it should be noted that sleep schedules obtained by domatic partitions may be suboptimal. For example, a ring of 5 nodes admits a sleep schedule of length $5/2$ (that is, its fractional domatic number is at least $5/2$, see Fig. 1), but the domatic number is 2 because each dominating set has at least 2 nodes.

Determining the domatic number is a well-known NP-hard problem [8, problem GT3]. The domatic number in general graphs can be approximated in polynomial time within a logarithmic factor but, under plausible complexity-theoretic assumptions, no better [9]. The fractional domatic number is as hard to approximate as the domatic partition in general graphs [7], [9]. The fractional domatic partition is an LP relaxation of the domatic partition, but the size of the LP (1) can be exponential in $n = |V|$.

We note that there is an unfortunate conflict in the terminology. In our case, fractional domatic partition refers to a fractional packing of integral dominating sets

[7]; the same terms have also been used to refer to an integral packing of fractional dominating sets [10], [11].

D. Contribution

This work shows that the sleep scheduling problem (fractional domatic partition), although hard in the general case, admits an efficient distributed approximation algorithm in a family of problems relevant to practical sensor networks.

We formalise the features of what we regard as natural problem instances in detail in Sect. III. In essence, we assume that the sensor nodes are located in an Euclidean space such that (1) they are not packed in an arbitrarily dense manner; (2) there is some upper bound on the range of communication links; and (3) the communication graph G is a geometric spanner, that is, there are no pathological cases where the shortest communication path between two nodes can be arbitrarily long in comparison with their Euclidean distance.

While the problem structure is formulated in terms of geometric constraints, we emphasise that the nodes need not know their coordinates or even their pairwise distances. It suffices that there *exists* an embedding of the nodes in a low-dimensional Euclidean space such that the above constraints are satisfied.

Under these assumptions, for any constant $\epsilon > 0$, our distributed algorithm achieves the approximation ratio of $1 + \epsilon$. Specifically, the algorithm guarantees that if the maximum lifetime of the network is q^*, the entire network operates for at least $q^*/(1 + \epsilon)$ units of time.

Our previous work [7] shows that there is a centralised approximation algorithm in the case where the nodes know their coordinates (the problem formulation is slightly different, as the structure of the communication network is not an issue in the centralised case). The present work extends this towards more practical applications in two ways: (i) the algorithm is distributed, and (ii) the nodes need not know their coordinates. Our algorithm can be seen as a distributed, coordinate-free variant of the *shifting strategy* [12].

E. Time, space and communication complexity

To analyse the complexity of our distributed approximation scheme in terms of time, space and communication requirements, we have divided the algorithm in two phases: initialisation (Sect. IV) and sleep scheduling (Sect. V).

The initialisation guarantees that the sensor nodes have locally unique identifiers (of constant size), and that there are so-called anchor nodes appropriately distributed in the network. These constraints can be satisfied by the design of the network (say, the hardware addresses of the nodes as locally unique identifiers and the base stations of a two-tier network as anchor nodes); or these may be already determined for other purposes (for example, for routing, data gathering and in-network processing of information); or these can be determined by standard distributed algorithms in a relatively efficient manner (however, not necessarily in a deterministic way, as there may be a need to break the symmetry, and not necessarily in a strictly constant time per node).

Our main focus is on the additional computational overhead of sleep scheduling after these initial steps. We will see that our deterministic algorithm is able to find a sleep schedule arbitrarily close to the optimum in constant time per node. This implies that the memory requirement and communication complexity for each node is constant. Furthermore, also executing the schedule can be performed in a constant time, implying, among others, that a node is switched on and off at most a constant number of times.

In the context of distributed systems, it is a common practice to add hidden $\log n$ factors; for example, one commonly assumes that the size of each message transmitted in the network is large enough to hold a globally unique address of a node [13]. In this work we do not make such assumptions. For example, a constant space or a constant amount of communication in this work means a strictly constant number of bits independent of n. In fact, if the conditions of the initialisation are satisfied by network deployment, the approximation scheme finds a provably near-optimal sleep schedule even in an infinite network.

II. Related work

Feige et al. [9] study the approximability of the domatic partition. They prove that the domatic partition in general graphs can be approximated in polynomial time within a logarithmic factor but, under plausible complexity-theoretic assumptions, no better. This result directly extends to the case of the fractional domatic partition [7].

Domatic partitions have been applied to maximising the lifetime of ad hoc and sensor networks. For example, Cardei et al. [3] present a heuristic algorithm. Moscibroda and Wattenhofer [5] present a distributed, randomised approximation algorithm for arbitrary redundancy graphs; thus, their analysis achieves only a logarithmic approximation ratio. Koushanfar et al. [4] find optimal solutions by using a centralised algorithm

with superpolynomial time complexity. Pemmaraju and Pirwani [6] study special cases such as unit disk graphs and their generalisations. However, they do not obtain a constant-factor approximation algorithm for domatic partitions; instead, they study a more general problem of k-domatic partition, which is a domatic partition in the kth power of the graph, and bound the performance of their algorithms in terms of the optimal $(k-1)$-domatic partition.

The redundancy graph is often assumed to be identical to or derived from the communication graph [5], [6]. Koushanfar et al. [4] explicitly consider redundancy graphs (called the *prediction graph* in their work) and describe a method for constructing the graph.

Instead of a packing of dominating sets, one can also consider a packing of more general set covers. This leads into the problem of *set cover packing*. Naturally such problems are at least as hard to solve and approximate as domatic partition. This problem has also been considered in the context of sensor networks; Slijepcevic and Potkonjak [14] call it the *set K-cover problem* and propose a heuristic algorithm for solving it. Cardei and Du [15] call it the *disjoint set covers problem* and propose different heuristics. Gu and He [16] consider a particular special case in which a minimum set cover can be found in a polynomial time. However, their problem formulation does not require full coverage at every point in time. Lin and Chiu [17] study a related problem in sensor network deployment: finding positions of sensor nodes that admit a good set cover packing.

Berman et al. [18] is one of the few works that explicitly considers the fractional version of the set cover packing problem. However, they obtain only a logarithmic approximation ratio, as they focus on the general case. Cardei et al. [19] present some heuristic algorithms for fractional set cover packing, and verify the algorithms by simulations. Wang et al. [20] study a problem that can be interpreted as a generalisation of fractional set cover packing. They consider sets of sensors which provide a desired level of so-called information coverage on every point of the monitored area. They study the problem of scheduling such sets and present a heuristic algorithm.

III. The problem family

In this section, we define the family of problems that can be solved by our approximation algorithm. Before describing the constraints on the problem structure, we introduce some notation. Each node $v \in V$ is associated with a position $p(v)$ in the d-dimensional Euclidean space $\mathbb{R}^d$; the positions are not used in the algorithm, they are only used to describe the problem structure. The number of edges in G on the shortest path between nodes $u, v \in V$ is denoted by $d_G(u, v)$; this is extended to sets in the natural way: $d_G(U, v) = \min_{u \in U} d_G(u, v)$. In the communication graph, balls of radius $R \in \mathbb{N}$ are denoted by $B_G(v, R) = \{u \in V \mid d_G(u, v) \leq R\}$, and in the Euclidean space, balls of radius $\rho > 0$ are denoted by $B(y, \rho) = \{z \in \mathbb{R}^d \mid \|y - z\| \leq \rho\}$. Table I summarises the notation that we use throughout this work.

A. Assumptions on the problem structure

A problem instance consists of the graph G and its subgraph H; the graph G defines the communication links that can be used in the distributed algorithm, and the graph H defines the set of feasible solutions as described in Sect. I-C.

Fixed values of $d \in \mathbb{N}$, $N \in \mathbb{N}$, $\sigma > 1$ and $\alpha \geq 1$ define a collection of problem instances. A graph G and its subgraph H are members of this collection if there exists a position $p(v) \in \mathbb{R}^d$ for each $v \in V$ such that G satisfies the following constraints:

1) The density of the nodes is bounded: for any point $x \in \mathbb{R}^d$, there are at most N nodes v with $p(v) \in B(x, 1)$.
2) There is an upper bound on the length of the communication links: if $\{u, v\} \in E(G)$ then $\|p(u) - p(v)\| < 1$.
3) The graph G is a geometric σ-spanner: $d_G(u, v) \leq \sigma \lceil \|p(u) - p(v)\| \rceil$ for all $u, v \in V$.
4) The parameter $\alpha \geq 1$ controls the initialisation phase, and is described in detail in Sect. IV.

Observe that the parameters do not constrain the sizes of G and H, only their structure.

The rounding up in assumption 3 is a technicality due to the fact that G is unweighted and $d_G(\cdot, \cdot)$ takes integral values only. For example, the distance $\|p(u) - p(v)\|$ may be arbitrarily close to 0 while $d_G(u, v)$ is at least 1 for distinct nodes u and v.

Assumptions 1–2 characterise what we call (d, N)-local graphs [7], [21]; cf. *civilised graphs* [22, §8.5]. The intuition is that there is an upper bound on the range of the radio and that scaling up the number of sensor nodes typically means that a larger network covers a larger geographic area. Note that neither of these two assumptions alone restricts the collection of the graphs; both are needed together. These assumptions imply that G is a bounded-degree graph. Observe that these assumptions do not imply that G is a unit-disk graph: while there is no edge between distant nodes, it

TABLE I
NOTATION

Parameters of the problem family	
$\epsilon > 0$	controls the desired approximation ratio
$d \in \mathbb{N}$	dimensionality of the Euclidean space
$N \in \mathbb{N}$	density bound
$\sigma > 1$	stretch factor of the communication graph
$\alpha \geq 1$	anchor distance multiplier
Constants derived from the parameters	
$k \in \mathbb{N}$	controls the number of partitions
$\delta > 0$	cycle length
$m \in \mathbb{N}$	bound on the number of nearby anchors
$r \in \mathbb{N}$	anchor minimum distance
$L \in \mathbb{N}$	radius of local neighbourhood of each node
S	finite, totally ordered set of identifiers
Problem instance	
V	sensor nodes
n	number of nodes, $\lvert V \rvert$
G	communication graph on nodes V
H	redundancy graph on nodes V
Assumed to exist, but not used in the algorithm	
$p(v) \in \mathbb{R}^d$	position of node $v \in V$
Defined during initialisation	
$A \subseteq V$	the set of anchors
$s(v) \in S$	locally unique identifier of node $v \in V$
Defined in the algorithm	
i	configuration, $i = 0, 1, \ldots, km - 1$
$a(v, i) \in A$	anchor of node v in configuration i
$C(a, i) \subseteq V$	cell of anchor $a \in A$ in configuration i
$\partial C(a, i) \subseteq V$	boundary of cell $C(a, i)$
$\bar{C}(a, i) \subseteq V$	$C(a, i) \cup \partial C(a, i)$
$K \subseteq V$	a set which dominates a cell
$x_{a,i}$	solution of the local LP
Additional notation	
$d_G(u, v)$	distance between u and v in G
$d_G(U, v)$	$\min_{u \in U} d_G(u, v)$
$B_G(v, R)$	$\{u \in V \mid d_G(u, v) \leq R\}$
$B(y, \rho)$	$\{z \in \mathbb{R}^d \mid \lVert y - z \rVert \leq \rho\}$
$M(d, \rho)$	max. number of 1-separated points in $B(y, \rho)$
D	a dominating set in H
q^*	fractional domatic number of H
x	fractional domatic partition of H
$x(D)$	time allocation for D

is not required that there is an edge between two nodes close to each other.

Assumption 3 captures the intuition that in order to communicate with a nearby node, arbitrarily long paths are not needed. As this is a desirable feature in practical networks and as this condition is easily satisfied in cases where nodes are deployed in a dense manner on an approximately convex area, we claim that this is a realistic assumption in many applications where sleep scheduling is relevant. We emphasise that this assumption is imposed only on the communication graph; the redundancy graph H, which is the graph where fractional domatic partitions are to be found, does not need to be a geometric spanner.

B. Derived constants

Given the desired $\epsilon > 0$ and the parameters d, N, σ and α, we derive a few constants that are needed in the approximation algorithm. First, choose constants $k \in \mathbb{N}$ and $\delta > 0$ that satisfy $1/(1 + \epsilon) \leq 1/(1 + 4N/k) - \delta$.

We write $M(d, \rho)$ for the maximum number of points that can be placed in a d-dimensional ball of radius ρ such that the Euclidean distance between any pair of points is at least 1. Naturally, this is a finite constant for any constant d and ρ.

In order to choose the values of the remaining constants, we need an upper bound for $M(d, \rho)$. For our purposes, it suffices to use a simple volume bound for sphere packing. If there are $M(d, \rho)$ points in a ball of radius ρ such that the pairwise distances are at least 1, we can add an open ball of radius $1/2$ centred at each such point and the balls are non-intersecting. Furthermore, all such balls are located within a ball of radius $\rho + 1/2$. By the ratio of the volumes of the balls, $M(d, \rho) \leq ((\rho + 1/2)/(1/2))^d = (2\rho + 1)^d$.

Second, choose integral constants $m \geq M(d, 2\alpha\sigma)$, $r \geq (km + 1)/\alpha + 2\sigma$, and $L = \lceil 2\alpha(r - \sigma) - 1 \rceil$.

Finally, choose a finite, totally ordered set of identifiers S, for example, a range of integers. The set has to be large enough that distinct identifiers can be found for any pair of nodes at most $2L$ hops from each other; such a finite set exists due to the density bound N.

C. Local information

To perform scheduling, the nodes must have access to a clock. We assume that the clocks are (approximately) synchronised. For example, the nodes may be initially switched on approximately at the same time, or they may use a radio controlled clock that uses a public radio station such as WWVB [23] or DCF77 [24] as a reference.

In addition to the clock, each node v needs to know its neighbours in G and H and the parameters ϵ, d, N, σ and α. No other information is required. We emphasise that node v does *not* need to know its position $p(v)$, distances or directions to its neighbours, any globally unique identifier, the global network topology of G or H, or the number of nodes n.

IV. Initialisation

This section discusses the choice of the locally unique identifiers $s(v) \in S$ and the set of anchor nodes $A \subseteq V$. Depending on the application, the material in this section can be seen either as an additional assumption on the problem structure or as a computational step to be carried out by the network upon initialisation.

The locally unique identifiers $s(v) \in S$ and the set of anchors $A \subseteq V$ are chosen in an arbitrary manner, as long as they satisfy the following conditions:

1) Identifiers are locally unique within distance $2L$, that is, $s(u) \neq s(v)$ whenever $u \neq v$ and $d_G(u, v) \leq 2L$.
2) For any two distinct anchors $a, b \in A$, it holds that $d_G(a, b) > r$, and for any $v \in V$, there exists an anchor $a \in A$ with $d_G(a, v) \leq \alpha r$.

These assumptions can be satisfied by using suitable hardware and by planning the network deployment properly: for example, the node hardware may have identifiers such as MAC addresses (which can be used as locally unique identifiers) and some of the nodes can be more powerful base stations (which can also act as anchor nodes for our purposes). Values $\alpha \gg 1$ can be used to allow for an imperfect placement of base stations.

These assumptions may also be satisfied because other processes running in the sensor network have similar requirements. For example, routing and data gathering may be organised in a hierarchical fashion by employing local clusters; again, cluster heads may act as anchors for our purposes.

Finally, these assumptions can be satisfied by efficient (although not deterministic constant-time) distributed algorithms. To choose locally unique identifiers, it suffices to colour the graph G^{2L}, the graph on V where $u, v \in V$ are neighbours iff $d_G(u, v) \leq 2L$. To choose a valid set of anchor nodes, it suffices to find any maximal independent set in the graph G^r. There are several distributed algorithms for colouring a bounded-degree graph (see e.g. Linial [25]) and for finding a maximal independent set (see e.g. Peleg [13, §8] and Kuhn et al. [26]).

A. Properties of the anchors

The following two lemmata show that even though the neighbourhood $B_G(v, L)$ for a node v extends at least km hops beyond the nearest anchor, it still contains at most m anchors in total.

Lemma 1: $L \geq \alpha r + km$.

Proof: By the choice of r and L, we obtain $L \geq 2\alpha(r - \sigma) - 1 = \alpha r + \alpha r - 2\alpha\sigma - 1 \geq \alpha r + km$. ∎

Lemma 2: For any $v \in V$, there are at most m anchors in $B_G(v, L)$.

Proof: The hop-count distance in G between two distinct anchors is more than r, and G is a geometric σ-spanner. Thus, the pairwise Euclidean distance between two distinct anchors $a, b \in A$ is at least $\|p(a) - p(b)\| > r/\sigma - 1 = (2\alpha(r - \sigma))/(2\alpha\sigma) \geq L/(2\alpha\sigma)$. Thus, in any $B(p(v), L)$ there are at most $M(d, 2\alpha\sigma) \leq m$ anchors. As the length of each edge is bounded by 1, $a \in B_G(v, L)$ implies $p(a) \in B(p(v), L)$. Thus, in any $B_G(v, L)$ there are at most m anchors. ∎

V. Sleep scheduling

This section presents the distributed approximation algorithm for sleep scheduling. For fixed values of ϵ, σ, α, d and N, all operations described in this section require only a constant amount of time, space and communication per node. We will see that it suffices for each node v to know its constant-size neighbourhood $B_G(v, L)$.

In essence, we partition the set of nodes V into small *cells*. The size of each cell is bounded by a constant. Then, we solve the constant-size sleep scheduling problem within each cell optimally or near-optimally and combine the local solutions. However, as there is no global coordination, the nodes that are near the boundary of a cell (that is, the nodes with neighbours in different cells) may operate suboptimally. Our solution is to consider multiple *configurations*. Each configuration corresponds to one partition of V, and we apply each configuration in turn. Our choice of the configurations guarantees that no single node is too often at cell boundary; we will formalise this property in Lemma 4 below.

A. Partitions

First, each node v finds the distance to its nearest anchor, that is, the distance $d_G(A, v)$. It holds that $d_G(A, v) \leq \alpha r$. Then, for each configuration $i \in \{0, 1, \ldots, km - 1\}$, the node v selects an anchor $a \in A \cap B_G(v, d_G(A, v) + i)$ with the smallest identifier $s(a)$; let $a(v, i) = a$. We say that $a(v, i)$ is the anchor of node v in configuration i, and we also say that v is in the cell of anchor $a(v, i)$ in configuration i.

By the choice of anchors and by Lemma 1, it holds that $d_G(A, v) + i \leq \alpha r + km - 1 < L$; thus, local information in constant-size neighbourhood $B_G(v, L)$ suffices. Furthermore, as the identifiers are locally unique within distance $2L$, there is exactly one possible selection for $a(v, i)$.

We define the cell of anchor a in configuration i by $C(a,i) = \{v \in V \mid a(v,i) = a\}$, the *boundary* of the cell by $\partial C(a,i) = \{v \in V \mid d_G(C(a,i),v) = 1\}$, and the closure of the cell by $\bar{C}(a,i) = C(a,i) \cup \partial C(a,i)$. Observe that the nodes in the boundary of a cell are not members of the cell.

Finally, a node $v \in V$ is a *boundary node* in configuration i if there is an anchor $a \in A$ such that $v \in \partial C(a,i)$. Equivalently, v is a boundary node in configuration i if v has a neighbour u in G such that $a(v,i) \neq a(u,i)$.

B. Properties of the partitions

For each fixed i, the nonempty cells $C(\cdot,i)$ partition the set V. At configuration 0, the cells $C(\cdot,0)$ correspond to the cells of a Voronoi diagram generated by the anchors (with ties broken by the locally unique identifiers); in configurations $1,2,\ldots,km-1$, the boundaries of the cells are shifted towards the anchors with larger locally unique identifiers. Bounded density of the nodes in the Euclidean space implies that $|C(\cdot,\cdot)|$ is bounded by a constant.

The careful choice of the derived parameters enables us to prove that no node is a boundary node too often. We begin by analysing how the function $a(v,i)$ changes its value as i increases from 0 to $km-1$.

Lemma 3: For any node $v \in V$, there are at most $m-1$ configurations i such that $a(v,i) \neq a(v,i+1)$.

Proof: Fix any v. By Lemma 2, there are at most m anchors in $B_G(v,L)$, implying that $a(v,i)$ takes at most m different values. To complete the proof, it suffices to show that each distinct value of $a(v,i)$ corresponds to a single interval of configurations i; once $a(v,i)$ changes its value from a_1 to $a_2 \neq a_1$, it never changes back to a_1.

Assume that $a(v,i_1) = a(v,i_2) = a$ for arbitrary a and $i_1 \leq i_2$. Then a is a member of the ball $B_G(v, d_G(A,v) + i_1)$, and a is the anchor with the smallest identifier in the larger ball $B_G(v, d_G(A,v)+i_2)$. Thus, for any $i_1 \leq i \leq i_2$, it holds that a is the anchor with the smallest identifier in $B_G(v, d_G(A,v) + i)$, implying $a(v,i) = a$ for all $i_1 \leq i \leq i_2$. ■

Now we can prove the key lemma.

Lemma 4: For any $v \in V$, there are at most $4m$ configurations i such that v is a boundary node in i.

Proof: Fix any v. By Lemma 3, we can divide the list of configurations $(0,1,\ldots,km-1)$ into at most m intervals, such that $a(v,i)$ is constant within each interval. We will prove that v can be a boundary node at most 4 times on each interval.

This clearly holds for intervals of length at most 4. Next, consider an interval from i_1 to i_2 with $i_2 \geq i_1 + 4$ such that $a(v,i) = a$ for all $i_1 \leq i \leq i_2$. Consider any neighbour $u \in V$ with $\{u,v\} \in E(G)$.

By definition, $a = a(v,i_1) \in B_G(v, d_G(A,v) + i_1)$, implying $a \in B_G(u, d_G(A,v) + i_1 + 1)$ and $a \in B_G(u, d_G(A,u) + i_1 + 2)$. The last step uses the fact that $|d_G(A,v) - d_G(A,u)| \leq 1$, as $d_G(u,v) = 1$.

Similarly, $a = a(v,i_2)$ implies that a is the anchor with the smallest identifier in $B_G(v, d_G(A,v) + i_2)$, implying that the same holds for $B_G(u, d_G(A,v)+i_2-1)$ and $B_G(u, d_G(A,u) + i_2 - 2)$.

Thus, $a(u,i) = a = a(v,i)$ for $i_1+2 \leq i \leq i_2-2$. As this holds for any neighbour u, the node v cannot be a boundary node in the configurations $i_1+2 \leq i \leq i_2-2$. Therefore, there are at most 4 configurations in the ends of the interval such that v may be a boundary node. ■

C. Finding the local schedules

For each anchor a and configuration i, solve the LP

$$\begin{aligned} &\text{maximise} && \textstyle\sum_K x_{a,i}(K) \\ &\text{subject to} && \textstyle\sum_K K(v) x_{a,i}(K) \leq 1 \quad \text{for all } v, \\ &&& x_{a,i}(K) \geq 0 \quad \text{for all } K, \end{aligned} \tag{2}$$

where v ranges over all nodes in $\bar{C}(a,i)$ and K ranges over all subsets $K \subseteq \bar{C}(a,i)$ such that K dominates $C(a,i)$ in H. Note that the boundary nodes may take part in domination, but they need not be dominated. The LP is of constant size and it depends on the local information only.

All nodes in $\bar{C}(a,i)$ need to know the solution $x_{a,i}$. In practice, this can be implemented by one of the following approaches:

1) Each node solves the LP. In this case, there is very little communication, making this approach ideal if communication is expensive in comparison with computation.
2) Each anchor a solves the LP and informs everyone else in its cell. This approach is suitable if the anchors are more powerful base stations.
3) For each i, each anchor a chooses a node v in its local neighbourhood in a round-robin fashion and lets the node v solve the LP. This leads into a more even distribution of the computational load.

D. Executing the schedule

Each local schedule $x_{a,i}$ was determined in a constant time, it is of constant size, and there is a constant number of such schedules. However, some care is needed to guarantee that also executing the schedule can be done in a constant number of operations.

We use the synchronised clocks to proceed in cycles of length δ time units. Each cycle is further divided into km steps of length $\delta/(km)$. We label the steps within each cycle by $0, 1, \ldots, km-1$. At step i, we apply the schedule of configuration i as follows.

Consider a node v. If v is a boundary node in configuration i, then v is active for the entire step. Otherwise, v is scheduled according to $x_{a,i}$ where $a = a(v,i)$. All nodes in $C(a,i)$ consider the sets K with a nonzero $x_{a,i}(K)$ in the same order $K_1, K_2, \ldots$ (say, the lexicographic order). Let $t_j = \delta x_{a,i}(K_j)/(km \sum_K x_{a,i}(K))$. First, if $v \in K_1$, the node is active for t_1 time units; otherwise it is asleep for t_1 time units. Then, if $v \in K_2$, the node is active for t_2 time units, and so on. This way we have scaled down the entire schedule $x_{a,i}$ into one time step of length $\delta/(km)$.

At every configuration i, each node is a member of $C(a,i)$ for some a, and the local schedule $x_{a,i}$ guarantees that $C(a,i)$ is dominated at every point in time. Switching on all boundary nodes does not affect the domination. Thus, this procedure is correct in the sense that V is dominated at every point in time, as long as no node runs out of battery. In the following section, we prove that the batteries last for a near-optimal time.

E. Proof of near-optimality

Let the fractional domatic number of H be q^*. A trivial feasible solution of (1) with length 1 can be obtained by choosing $x(D) = 1$ for $D = V$ and $x(D) = 0$ for $D \neq V$. The graph H is a bounded-degree graph: the degree of a node is at most $N-1$, which also bounds the fractional domatic number. Thus, $1 \leq q^* \leq N$.

Any fractional domatic partition x provides a feasible solution to the local LP (2): for each dominating set D in the fractional domatic partition, add $x(D)$ units to $x_{a,i}(K)$ where $K = D \cap \bar{C}(a,i)$; as D dominates all nodes, K dominates $C(a,i)$. In particular, this applies if x is an optimal domatic partition. Thus, $\sum_K x_{a,i}(K) \geq q^*$, as $x_{a,i}$ is an optimal solution to the local LP.

Fix a node $v \in V$. Whenever v is a boundary node, it is active for the entire step, and whenever v is not a boundary node, it is active for at most the fraction $1/q^*$ of the step. By Lemma 4, the node v is a boundary node in at most $4m$ configurations out of km. Thus, during an entire cycle of length δ, the node v is active for at most $\delta(4/k+1/q^*)$ units of time. During $\lfloor 1/(\delta(4/k+1/q^*)) \rfloor$ cycles, each node is active for at most 1 time unit. Thus, each node survives for at least $\lfloor 1/(\delta(4/k+1/q^*)) \rfloor \delta \geq (1/(\delta(4/k+1/q^*)) - 1)\delta = q^*/(1+4q^*/k) - \delta \geq q^*/(1+4N/k) - \delta \geq q^*(1/(1+4N/k) - \delta) \geq q^*/(1+\epsilon)$ time units, which is within a factor $(1+\epsilon)$ of what can be achieved by an optimal schedule.

Note that the entire network does not know the length of the schedule. Many local parts of the network may run for a much longer time than q^*. However, the bounded degree of the graph guarantees that the schedule runs for at most N time units or $\lceil N/\delta \rceil$ cycles, which is a constant. Thus, executing the schedule requires a constant number of operations. Note that this also bounds the number of the times a given node is switched on or off, bounding the overhead in these operations.

VI. Discussion

A. Connectivity

Much work on sleep scheduling focuses on the issue of preserving the connectivity of a wireless network [1, §7]. Extending our algorithms to connected dominating sets offers directions for future research. However, there are several applications where the connectivity need not be taken into account:

(1) Applications in which using the sensor, processing the sensor data and transmitting the sensor data consumes considerable amounts of energy. In these cases, the entire node does not need to be asleep in order to conserve energy; it is enough to switch off the sensor.

(2) Applications in which latency is not critical. In the simplest case, all nodes wake up periodically in order to transmit the latest data from the local buffers to the sink.

(3) Applications in which the range of radio communication is much larger than the range of redundancy relations. Typically, all sets which dominate H are also connected in G [27], [28].

(4) Multi-tier sensor networks. Base stations form a backbone network which provides connectivity from any sensor node to the sink.

(5) Multi-radio networks, for example, sensor networks based on mobile phones. While G may describe connectivity by a low-power short-range radio such as Bluetooth, alerts and other information can be sent to the sink over a mobile data service such as GPRS.

B. Fault tolerance

Fault tolerance in sleep scheduling is a conceptually complicated problem. If all other parts of a node besides a real-time clock are completely switched off in order to save the batteries, how can the node know that its active neighbour has silently failed? A thorough discussion of this problem is beyond the scope of this work; we merely present some examples of how one may add

fault tolerance to our algorithm and how the algorithm behaves in the presence of faults.

One obvious approach for recovering from failures in sleep scheduling is to periodically check whether there are faulty nodes in the neighbourhood. This approach fits into our framework. First, choose a small enough δ so that $\delta/(km)$ is below the desired interval of periodic checks. Then, at the beginning of each step, each node v checks its neighbourhood $B_G(v, L)$ for topology changes. If any changes are detected, the node re-calculates the schedule; all other nodes which are affected by the change will perform similar calculations at the same time. After recalculation, the new schedule is applied from this point on.

The recalculation is a constant overhead and the failure of a single node requires recalculations in its constant-size neighbourhood only. The constant overhead may be relatively large, but it should be noted that in configuration i, only the schedules $x_{a,i}$ for this particular i are needed; the remaining schedules may be calculated later. During the calculations, all affected nodes can simply keep their sensors active, in order to avoid missing monitored data.

The fact that the operation of the nodes is based on a strictly constant-size neighbourhood provides a high degree of fault tolerance even in cases where large fractions of the nodes fail at the same time. For example, if a catastrophic event eliminates all nodes in a certain geographic area, the remaining network continues to operate correctly and maintains a near-optimal sleep schedule for all parts not near the faulty area. The fault-recovery scheme sketched above only causes recalculations at nodes near the faulty area; information on the fault does not propagate to more distant areas.

Interestingly, even if the topology changes imply that the maximum lifetime of the network improves, this new information does not need to be propagated to distant parts of the network. In our algorithm, the nodes do not even know the total lifetime achieved by the distributed schedule: they do locally their best and keep working even if nodes in some remote parts may have run out of their battery.

C. Non-homogeneous nodes

The algorithm assumed homogeneous sensor nodes, that is, the battery capacities of the nodes were identical. Extensions to the non-homogeneous case are relatively straightforward. Essentially, the upper bounds 1 in both global LP (1) and local LP (2) need to be replaced by the battery capacities. This leads into a more general sleep scheduling problem that is no longer equal to the fractional domatic partition.

Some information on minimum and maximum battery capacities is needed, however, in order to obtain a strictly constant-time algorithm. The bound $1 \leq q^* \leq N$ used in Sect. V-E assumes that the minimum and maximum battery capacities are equal to 1.

D. Approximate solutions of the local LP

If necessary, the local LP (2) can be solved by using a fast approximation scheme. It can be shown that if k, δ and ϵ_{LP} satisfy $1/(1+\epsilon) \leq 1/((1+4N/k)(1+\epsilon_{\mathrm{LP}})) - \delta$ then a $(1+\epsilon_{\mathrm{LP}})$-approximation of the local LP yields a $(1+\epsilon)$-approximate sleep schedule.

E. Concluding remarks

This work shows that there is a set of realistic assumptions on the structure of the communication and redundancy relations in a sensor network, under which near-optimal, distributed sleep scheduling is feasible. We have demonstrated that for any $\epsilon > 0$, given appropriate anchor nodes and locally unique identifiers, it is possible to schedule the sensing activities of the nodes in a local and distributed manner so that the ratio of the optimum lifetime to the achieved lifetime of the network is at most $1+\epsilon$. The total computational effort (time, memory, communication) required at each node is constant. However, it should be pointed out that the constants are arguably unfeasibly large for practical purposes. In particular, solving the local LP (2) resulting from small values of ϵ is likely to prove an arduous task in practice. Thus, the present contribution should primarily be viewed as a theoretical feasibility result, whereby considerable further work is required to bring near-optimal scheduling to the sensor networking practice.

To our knowledge the present feasibility result is the first rigorous demonstration that local information and coordination suffice to arrive at a near-optimum solution of a global scheduling task. It is of theoretical and practical interest to investigate further (a) what other basic network control tasks admit near-optimal solution via local information and coordination, and (b) to what extent can the structural assumptions on the network be alleviated while still retaining local solvability.

Acknowledgments

We thank Topi Musto for implementation work and an anonymous referee for valuable suggestions.

This research was supported in part by the Academy of Finland, Grants 117499 and 116547, by the IST

Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, and by Helsinki Graduate School in Computer Science and Engineering (Hecse).

REFERENCES

[1] B. Krishnamachari, *Networking Wireless Sensors*. Cambridge, UK: Cambridge University Press, 2005.

[2] F. Kuhn, T. Nieberg, T. Moscibroda, and R. Wattenhofer, "Local approximation schemes for ad hoc and sensor networks," in *Proc. Joint Workshop on Foundations of Mobile Computing (DIALM-POMC, Cologne, Germany, September 2005)*. New York, NY, USA: ACM Press, 2005, pp. 97–103.

[3] M. Cardei, D. MacCallum, M. X. Cheng, M. Min, X. Jia, D. Li, and D.-Z. Du, "Wireless sensor networks with energy efficient organization," *Journal of Interconnection Networks*, vol. 3, no. 3–4, pp. 213–229, 2002.

[4] F. Koushanfar, N. Taft, and M. Potkonjak, "Sleeping coordination for comprehensive sensing using isotonic regression and domatic partitions," in *Proc. 25th Conference on Computer Communications (INFOCOM, Barcelona, Spain, April 2006)*. Piscataway, NJ, USA: IEEE, 2006, paper 48.3.

[5] T. Moscibroda and R. Wattenhofer, "Maximizing the lifetime of dominating sets," in *Proc. 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS, Denver, CO, USA, April 2005)*. Washington, DC, USA: IEEE Computer Society, 2005, p. 242b.

[6] S. V. Pemmaraju and I. A. Pirwani, "Energy conservation via domatic partitions," in *Proc. 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc, Florence, Italy, May 2006)*. New York, NY, USA: ACM Press, 2006, pp. 143–154.

[7] J. Suomela, "Locality helps sleep scheduling," in *Working Notes of the Workshop on World-Sensor-Web: Mobile Device-Centric Sensory Networks and Applications (WSW, Boulder, CO, USA, October 2006)*, 2006, pp. 41–44. [Online]. Available: http://www.sensorplanet.org/wsw2006/

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman and Company, 1979.

[9] U. Feige, M. M. Halldórsson, G. Kortsarz, and A. Srinivasan, "Approximating the domatic number," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 172–195, 2002.

[10] D. F. Rall, "A fractional version of domatic number," *Congressus Numerantium*, vol. 74, pp. 100–106, 1990.

[11] P. J. Slater and E. L. Trees, "Multi-fractional domination," *Journal of Combinatorial Mathematics and Combinatorial Computing*, vol. 40, pp. 171–181, 2002.

[12] D. S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *Journal of the ACM*, vol. 32, no. 1, pp. 130–136, 1985.

[13] D. Peleg, *Distributed Computing – A Locality-Sensitive Approach*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000.

[14] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *IEEE International Conference on Communications (ICC, Helsinki, Finland, June 2001)*. Piscataway, NJ, USA: IEEE, 2001, pp. 472–476.

[15] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks*, vol. 11, no. 3, pp. 333–340, 2005.

[16] Y. Gu and T. He, "uScan: A lightweight two-tier global sensing coverage design," in *Proc. 4th ACM Conference on Embedded Networked Sensor Systems (SenSys, Boulder, CO, USA, October–November 2006)*. New York, NY, USA: ACM Press, 2006, pp. 399–400.

[17] F. Y. S. Lin and P. L. Chiu, "Energy-efficient sensor network design subject to complete coverage and discrimination constraints," in *Proc. 2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON, Santa Clara, CA, USA, September 2005)*. Piscataway, NJ, USA: IEEE, 2005, pp. 586–593.

[18] P. Berman, G. Calinescu, C. Shah, and A. Zelikovsky, "Power efficient monitoring management in sensor networks," in *IEEE Wireless Communications and Networking Conference (WCNC, Atlanta, GA, USA, March 2004)*. Piscataway, NJ, USA: IEEE, 2004, pp. 2329–2334.

[19] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM, Miami, FL, USA, March 2005)*. Piscataway, NJ, USA: IEEE, 2005, pp. 1976–1984.

[20] B. Wang, K. C. Chua, V. Srinivasan, and W. Wang, "Scheduling sensor activity for point information coverage in wireless sensor networks," in *Proc. 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt, Boston, MA, USA, April 2006)*. Piscataway, NJ, USA: IEEE, 2006.

[21] J. Suomela, "Approximability of identifying codes and locating-dominating codes," *Information Processing Letters*, 2007, to appear, doi: 10.1016/j.ipl.2007.02.001.

[22] P. G. Doyle and J. L. Snell, *Random Walks and Electric Networks*. Washington, DC, USA: The Mathematical Association of America, 1984.

[23] M. A. Lombardi, "NIST time and frequency services," National Institute of Standards and Technology, NIST Special Publication 432, Jan. 2002.

[24] D. Piester, A. Bauch, J. Becker, and T. Polewka, "Time and frequency activities at the Physikalisch-Technische Bundesanstalt," in *Proc. 36th Annual Precise Time and Time Interval Systems and Applications Meeting (PTTI, Washington, DC, USA, December 2004)*, 2005, pp. 179–188. [Online]. Available: http://tycho.usno.navy.mil/ptti/

[25] N. Linial, "Locality in distributed graph algorithms," *SIAM Journal on Computing*, vol. 21, no. 1, pp. 193–201, 1992.

[26] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer, "Fast deterministic distributed maximal independent set computation on growth-bounded graphs," in *Proc. 19th International Conference on Distributed Computing (DISC, Cracow, Poland, September 2005)*, ser. Lecture Notes in Computer Science, vol. 3724. Berlin, Germany: Springer-Verlag, 2005, pp. 273–287.

[27] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proc. 1st International Conference on Embedded Networked Sensor Systems (SenSys, Los Angeles, CA, USA, November 2003)*. New York, NY, USA: ACM Press, 2003, pp. 28–39.

[28] T. Yan, T. He, and J. A. Stankovic, "Differentiated surveillance for sensor networks," in *Proc. 1st International Conference on Embedded Networked Sensor Systems (SenSys, Los Angeles, CA, USA, November 2003)*. New York, NY, USA: ACM Press, 2003, pp. 51–62.