Lifetime and Coverage Guarantees Through Distributed Coordinate-Free Sensor Activation

Gaurav S. Kasbekar, Yigal Bejerano and Saswati Sarkar

***Abstract*—In wireless sensor networks, a large number of sensors perform distributed sensing of a target field. A sensor cover is a subset of the set of all sensors that covers the target field. The lifetime of the network is the time from the point the network starts operation until the set of all sensors with non-zero remaining energy does not constitute a sensor cover any more. An important goal in sensor networks is to design a schedule, that is, a sequence of sensor covers to activate in every time slot, so as to maximize the lifetime of the network. In this paper, we design a polynomial-time, distributed algorithm for maximizing the lifetime of the network and prove that its lifetime is at most a factor $O(\log n * \log nB)$ lower than the maximum possible lifetime, where n is the number of sensors and B is an upper bound on the initial energy of each sensor. Our algorithm does not require knowledge of the locations of nodes or directional information, which is difficult to obtain in sensor networks. Each sensor only needs to know the distances between adjacent nodes in its transmission range and their sensing radii. In every slot, the algorithm first assigns a weight to each node that is exponential in the fraction of its initial energy that has been used up so far. Then, in a distributed manner, it finds an $O(\log n)$ approximate minimum weight sensor cover, which it activates in the slot.**

***Index Terms*—Wireless Sensor Networks, Network Lifetime, Coverage, Approximation Algorithms, Distributed Algorithms, Coordinate-Free**

I. Introduction

Recent advances in wireless communications and electronics have enabled the development of *low-cost sensor nodes* [11]. Each sensor node is capable of sensing specific events in its vicinity and of communicating with adjacent nodes. Thus, for event sensing applications, a large number of sensor nodes are deployed in a *distribution area* and they collaborate to form an ad-hoc network, referred to as a *wireless sensor network* (WSN). WSNs have the potential to become the dominant sensing technology in many civilian and military applications, such as intrusion detection, environmental monitoring, object tracking, traffic control, and inventory management. In many of these applications, WSNs need to monitor the target field for detecting events of interest, e.g., entrance of an intruder in an intrusion detection application.

Wide-spread deployment of WSNs in target field monitoring is being deterred by the energy consumed in the monitoring process. The challenge is compounded by the fact that the sensors are battery-powered, and owing to size limitations, the sensors can only be deployed with low-lifetime batteries, most of which are not rechargeable. Thus, a sensor ceases to function (e.g., monitor) once its battery expires, and oftentimes, sensors whose batteries have expired can not be easily replaced owing to logistics issues such as remoteness or inaccessibility of distribution areas. Thus, the success of the WSN technology is contingent upon developing strategies for intelligently using the available sensors so as to maximize the duration for which the entire target field is monitored by sensors. This duration, referred to as the *network lifetime*, is an important performance metric for the network as the *coverage* of the entire target field is essential for reliable detection of events of interest.

Owing to large scale availability of low cost sensors, sensors are often deployed with some redundancy, that is, several locations in the target field can be monitored by multiple sensors. Lifetime of the WSNs can be substantially enhanced by intelligently activating the sensors that monitor the target field at any given time. We seek to maximize the lifetime of sensor networks by designing algorithms that dynamically activate sensors based on their residual energy content. The algorithm we develop is completely distributed, does not need to know the coordinates of any sensor, and provides provable guarantees on the attained lifetimes.

A. Related Literature

Coverage, connectivity and lifetime maximization for WSNs have received considerable attention in the last few years. Comprehensive surveys can be found in [13], [14]. Most of the existing papers focus on the coverage and connectivity aspects [2], [3], [6], [9], [15], [16], [17], [18], [19], [20], and typically propose computational geometry based approaches for discovering coverage holes and ensuring connectivity. An interesting *connectivity property* has been proved in [19], [20] that shows that if the transmission radius of each node is at least twice of its sensing radius, then coverage implies connectivity of the sensor network. We make the same assumption, and therefore seek to maximize lifetime while guaranteeing coverage without explicitly considering connectivity.

We now summarize the papers that propose topology control solutions that maximize the network lifetime by scheduling the active periods of the sensors, while preserving coverage and connectivity requirements. In [12], Cardei *et al.* addressed the problem of lifetime maximization when only a given set of targets needs to be covered. They showed that the problem is NP-hard and provided heuristic sensor activation algorithms based on linear programming relaxations. They also proposed a greedy heuristic activation scheme that at each round seeks the minimal set of sensors that covers all the targets. They evaluated the lifetimes attained by the heuristic solutions using simulations, but did not provide provable guarantees on the lifetimes of these schemes. Wang *et al.* [20] showed that the monitoring area is covered if all intersection points between sensing borders of sensors and those between sensing borders

G. Kasbekar and S. Sarkar are with the Department of Electrical and Systems Engineering at University of Pennsylvania, Philadelphia, PA, U.S.A. Their email addresses are kgaurav@seas.upenn.edu and swati@seas.upenn.edu respectively. Y. Bejerano is with Bell-Labs, Alcatel-Lucent, Murray Hill, N.J., U.S.A. His email address is Yigal.Bejerano@alcatel-lucent.com.

Part of this paper was presented at MobiCom'09.

of sensors and the monitoring area are covered. They also provided a distributed algorithm to activate a minimum set of sensors, while ensuring coverage and connectivity. However, the algorithm in [20] assumes knowledge of coordinates of nodes and does not provide provable guarantees on the network lifetime. The scheme proposed by Berman *et al.* [10] provides provable guarantees on the network lifetime while ensuring coverage of the target field. They have provided a centralized algorithm that attains a network lifetime which is within $O(\log n)$ of the maximum possible lifetime, where n is the number of sensors. This algorithm determines how to activate sensors based on an approximate solution of a linear program that requires complete knowledge of network topology, coordinates of sensor locations and initial energy of sensors. Such linear programs can clearly be solved only by a central entity that knows all of the above, which is hard to realize in practice. Also, the sensors rarely know their precise locations since WSNs usually do not have access to global positioning systems (GPS). Several sensor positioning systems [23], [24] have been proposed in the literature for learning the locations, without manual configuration or the use of GPS receivers. However, they provide only coarse location estimations in practical settings [25]. Note that several coverage verification algorithms that do not assume knowledge regarding the locations of the sensors exist [3], [9], [15], [16], but these papers do not provide any guarantee on the network lifetime. Our contribution is to provide a distributed, coordinate-free sensor activation scheme that provides provable guarantees on the network lifetime.

A centralized approximation algorithm similar to that in [10] has been proposed by Zhao *et al.* in [21] for the connected target coverage problem, *i.e.*, the problem of maximizing lifetime while ensuring coverage of a given set of target points and connectivity of the network. Thai et al [35] have proposed a distributed algorithm to maximize the network lifetime up to an $O(\log n)$ factor, while ensuring coverage of a given set of targets. However, the paper does not provide a coordinate-free algorithm for the area coverage problem, which we focus on. Also, the coverage and lifetime guarantees in [35] are probabilistic, whereas we provide deterministic guarantees on both coverage and lifetime.

Finally, Wu *et al.* [22] considered a different notion of lifetime in a recent paper: the maximum time until which all nodes in the data aggregation tree of choice remain operational (a node in this case consumes energy only during communication). Since we focus on the energy consumed in sensing, our notion of lifetime, the problem formulation and solution techniques differ substantially.

B. Our Contribution

The contribution of this paper is two-fold.

First, we present the *first coordinate-free distributed scheme that provides provable approximation guarantees on network lifetime, while providing strict coverage guarantees*. This is a surprising result since the sensors are not aware of their coordinates in a global coordinate system, and are therefore oblivious to their locations relative to each other and to the target field. To overcome this challenge, we assume that the sensor distribution area is slightly larger than the area that needs to be monitored. The sensors are divided into *periphery nodes* that are located near the boundary of the distribution area and *internal nodes* that are internal to this area. The target field that our scheme is committed to monitor is taken as the closure of the area covered by the internal nodes. Our scheme at each time slot selects a subset of sensors for monitoring the target field that ensure k-coverage of the entire target field, for a given integer $k \geq 1$, and different subsets may be selected in different slots. The selection process relies on two key steps: (i) each sensor is assigned a weight that is an exponentially increasing function of the energy it has consumed so far (ii) the set of sensors that has the minimum total weight, or an approximation thereof, among all those that cover the entire target field is activated. This selection process balances the monitoring load on all the sensors, and preferentially selects in each slot, the sensors with high residual energy. We demonstrate that the algorithm can be executed using distributed computations that do not need to know the locations of the sensors.

Second, we prove that the lifetime of the network when this algorithm is used is at least $1/O((\log n)(\log nB))$ of the optimal solution, where n is the number of sensors and B is a bound on the initial energy level of the nodes. We prove this approximation ratio, by extending to this problem the exponential-function technique, originally developed by Aspnes *et al.* [26] in the context of online machine scheduling and virtual circuit routing and later used by Awerbuch *et al.* [4] in online virtual circuit routing. Thus, our algorithm attains a provable guarantee which is only slightly worse as compared with the best available centralized performance guarantee till date, presented in [10]. We demonstrate via simulations that our scheme attains a significantly higher lifetime than several other existing schemes [10], [12], [20].

II. Preliminary

A. Network Model

We consider a *wireless sensor network* (WSN) consisting of a set S of n sensors that are also called *nodes*. Each node $u \in S$ can sense events of interest in its *sensing range* and communicate with nodes in its *transmission range*. We make the natural assumption that there are no two sensors at the same location. Also, each sensor $u \in S$ has a unique identification number, denoted by $ID(u)$. The sensors are distributed over a large 2-dimensional area. We refer to the region obtained by the union of the sensing ranges of all the sensors as the *distribution area* and it subsumes the region that needs to be monitored by the sensors, referred to as the *monitoring area*. The latter is typically significantly larger than the sensing range of a single sensor.

We assume that the sensing and transmission ranges of a node u are *open* discs, centered at u, with radii r_u and R_u respectively, where $R_u > r_u$. We refer to r_u and R_u as the *sensing radius* and *transmission radius* of node u respectively. Let $\hat{r} = \max_{u \in S} r_u$, and $\hat{R} = \min_{u \in S} R_u$. The boundary of the sensing range of any node u is a circle, which we refer to as the *sensing border* of node u. Let $d_{u,v}$ denote the Euclidean distance between nodes u and v. Nodes u and v are termed *adjacent* or *neighbors* if they are included in the transmission range of each other. Let N_u be the set of neighbors of u.

We assume that nodes only have *localized distance information*. Specifically, each node u knows (a) r_u, (b) $d_{u,v}$ and r_v for

each $v \in N_u$ and (c) $d_{v,w}$ for each pair $w, v \in N_u$ such that w and v are neighbors of each other. Thus, we assume that each node can estimate its sensing radius, and its distances from its neighbors without learning their orientations, and communicates this information to its neighbors. Note that recent studies [7], [8] have introduced accurate distance estimation techniques that are applicable to wireless sensors.

We define *periphery nodes* to be those whose sensing borders are not 1-covered at the beginning of the network operation (when all nodes have non-zero remaining energy) and the rest of the nodes to be *internal nodes*. Note that periphery nodes are located close to the boundary [1] of the distribution area in the sense that the minimum distance of every periphery node from a boundary point is at most $\hat{r}$ [2]. Although the sensors are not aware of their locations, every sensor knows whether it is a periphery or an internal node, for instance by using the mechanism in [3] or [30]. The algorithm in [30] requires a sensor to know some additional information, specifically, connectivity information of its two-hop neighbors.

Time is divided into *time slots* and we assume that the sensors have synchronized clocks, which notify them at the beginning of each time slot. Sensor $u \in S$ has an initial energy B_u and, as a normalization, we assume that each sensor consumes 1 unit of energy in each time slot in which it is *active*. For saving energy, a sensor may be in a *sleep* mode, in which it does not communicate with its neighbors nor sense its vicinity. A sensor in sleep mode consumes only negligible amount of energy, which we assume to be zero.

B. The Target Field

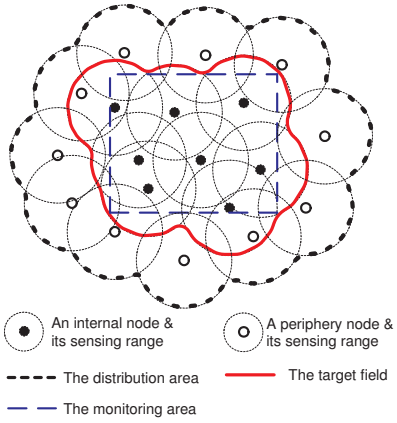

Fig. 1. An example of a small WSN and its target field.

Generally speaking, the *target field* is the area monitored by the system. This area is obviously subsumed in the distribution area and it should contain the monitoring area. Since the sensors are not aware of their locations, they are oblivious to their locations relative to each other and to the monitoring area. Addressing this difficulty, we next provide a precise definition of the target field that our scheme is committed to monitor.

Definition 1 (The Target Field): The *target field* is the area defined by the closure [3] of the union of the sensing ranges of all the internal sensors.

We assume that the target field subsumes the monitoring area. Fig. 1 illustrates a small WSN as well as its distribution area, monitoring area and target field.

Given a set $C \subseteq S$ of sensors and a positive integer k, we say that a point in the target field is *k-covered* by C if it is in the interior of the sensing ranges of at least k nodes in C. The target field is considered as *k-covered* by C if every point in the target field is k-covered by C.

Definition 2 (Sensor Cover): A set C of sensors that k-covers the target field is termed a *sensor cover*.

If there does not exist a sensor cover C such that all the nodes in C have non-zero energy, then the network is said to have a *coverage hole*.

Since the sensing ranges are open discs, no sensor covers its sensing border. Thus, any sensor cover must contain periphery sensors that cover the target-field boundary (see Fig. 1). Thus, sensor activation schemes must consider both internal and periphery nodes.

C. Problem Statement

We proceed to define the maximum network lifetime problem. Note that in this paper, we consider only the energy consumed in sensing, and not in other activities such as routing the sensed data. A similar approach has been used in several other papers on lifetime maximization *e.g.* [10] [12].

Definition 3 (The Network Lifetime): The *network lifetime* is the time interval from the activation of the network until the first time at which a coverage hole appears.

Definition 4: (The Maximum Network Lifetime Problem) An *activation schedule* is a sequence of sensor covers that are activated in successive slots, such that in every slot, each sensor in the activated sensor cover has non-zero energy. The *maximum network lifetime problem* seeks to find an activation schedule that maximizes the network lifetime.

In [12], the authors prove that the closely-related target coverage version of the maximum network lifetime problem is NP-hard. Moreover, in [9] it has been shown that for a given subset $C \subseteq S$, no coordinate-free algorithm can provably verify whether or not C covers the target field, if $\hat{R} < 2\hat{r}$. So henceforth, we assume that $\hat{R} \geq 2\hat{r}$ and we present a distributed coordinate-free algorithm for the maximum network lifetime problem with guarantee on the lifetime attained by the calculated schedule.

D. The Intersection Point Concept

We now present the intersection point concept that constitutes a cornerstone in our solution. Consider two sensors $v, z \in S$. The sensors are termed *intersecting* if their sensing borders intersect (but are not tangent to each other). In such case, we say that v intersects with z.

Property 1 (Intersection): The sensors $v, z \in S$ are intersecting if and only if $d_{v,z} < r_v + r_z$, $d_{v,z} + r_z > r_v$ and $d_{v,z} + r_v > r_z$.

[1] Recall that a boundary point of a set $A \subseteq R^2$ is a point $p \in R^2$ such that for every $\epsilon > 0$, its ϵ-neighborhood $B_\epsilon(p) = \{x \in R^2 : d(x,p) < \epsilon\}$ contains a point $x_1 \in A$ and a point $x_2 \notin A$, where $d(x,p)$ denotes the Euclidean distance between x and p [27].

[2] To prove this, consider a periphery node v. By definition, there exists a point p on v's sensing border that is not 1-covered. It is easy to see that p is a boundary point of the distribution area because every ϵ-neighborhood of p contains a point in v's sensing range and hence in the distribution area and the point p that is not in the distribution area. Also, $d(p,v) = r_v \leq \hat{r}$.

[3] Recall that the closure of a set A is the smallest closed set that contains A [27].

The first condition in Property 1 states that there is overlap between the sensing ranges of v and z and the second (respectively, third) condition states that the sensing border of z (respectively, v) is not subsumed in the sensing range of v (respectively, z).

Note that the sensing borders of any pair $v, z \in S$ of intersecting sensors have exactly two *intersection points* denoted by $IP(v, z, 1)$ and $IP(v, z, 2)$. Moreover, by Property 1, since the distance $d_{v,z} < r_v + r_z \leq 2 \cdot \hat{r}$ and we assume that $\hat{R} \geq 2 \cdot \hat{r}$, any two intersecting sensors v, z are adjacent.

We next show that for calculating a sensor cover we just need to consider sensors that have intersection points on their sensing borders.

Property 2: Consider a sensor cover $C \subset S$ and let $u \in C$ be a sensor without any intersection point on its sensing border. Then the set $C - \{u\}$ is also a sensor cover.

We omit the proof due to space constraints.

The next corollary directly follows from Property 2.

Corollary 1: Let $u \in S$ be a sensor without any intersection point on its sensing border and consider a schedule $\{C_1, C_2, \cdots, C_L\}$ of sensor covers with network lifetime of L in which node u is active in some slots. Then, the schedule $\{\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_L\}$, where $\hat{C}_j = C_j - \{u\}$, also defines a sequence of sensor covers with network lifetime of L.

From Corollary 1 it follows that the network lifetime is not affected by ignoring sensors without intersection points on their sensing borders. So henceforth, we will ignore such sensors.

Let P be the set of intersection points that are in the target field, referred to as the *IP set*. Recall that P contains every intersection point $IP(v, z, i)$, $i = 1, 2$, such that at least one of the nodes $v, z \in S$ is an internal node or $IP(v, z, i)$ is in the sensing range of an internal node.

Theorem 1: Consider a set $C \subset S$ of sensors. The set C is a sensor cover if and only if it k-covers every point in the IP set P.

We omit the proof of Theorem 1 due to space constraints– a similar result has been shown in [20].

Owing to Theorem 1, we henceforth consider as *sensor cover* any set of sensors that k-covers all the intersection points in P.

III. Algorithm Overview

We now describe the *Distributed Lifetime Maximization* (DLM) algorithm that we propose. In this section, we present a brief overview of the individual building blocks in DLM, and provide the details in Sections IV and V.

Our algorithm consists of an *initialization phase* and an *activation phase*. The initialization phase is executed once, at the beginning of the network operation, and informs the nodes of some network parameters. Every node executes the activation phase at the beginning of each subsequent time slot, and decides whether to activate itself in the slot based only on the state information in its neighborhood. We now describe the above phases, and introduce some new terminologies towards that end.

Consider a sensor cover C, and let sensor u have weight w_u, a positive real number. The weight of the sensor cover C is the sum of the weights of the sensors in C, i.e., $\sum_{u \in C} w_u$.

Definition 5 (A minimum weight sensor cover): A minimum weight sensor cover is a sensor cover that has the minimum weight among all sensor covers. An $\alpha-$approximate minimum weight sensor cover is one whose weight is at most α times that of the minimum weight sensor cover.

Let $P_u \subseteq P$ be the set of intersection points covered by sensor u, and T_u be the set of sensors v such that sensors u and v cover a common intersection point.

A. Initialization phase

An initialization phase is executed at the beginning of the network operation, i.e., at time $t = 0$. During the initialization phase, each sensor u acquires the following local information: (i) the set P_u of intersection points that it covers, (ii) the identities of the sensors in T_u and (iii) the intersection points in P_u that are covered by each sensor in T_u (i.e., the set $P_{u,v} = P_u \cap P_v$ for each $v \in T_u$). As we elaborate in Section V, each sensor u learns this information in a distributed manner by merely communicating with its neighbors and using only localized distance information. In addition, each sensor learns the following global network parameters: (i) n, the total number of sensors, and (ii) the maximum amount B of the initial energy of any sensor ($B = \max_{u \in S} B_u$). Using the above information, each sensor computes μ, where $\mu = 4nB$. The above constitutes the only global information each sensor needs to know throughout the execution of DLM, and can be communicated to each sensor using one network-wide broadcast.

B. Activation phase

The activation phase is executed at the beginning of each slot. We describe the computations in slot j.

Weight assignment: Let $b_u(j)$ be the amount of energy of sensor u that has been consumed in slots $1, \ldots j-1$. Then, at the beginning of slot j, sensor u has already consumed $l_u(j) = \frac{b_u(j)}{B_u}$ fraction of its energy. If $b_u(j) > B_u - 1$, i.e., sensor u does not have enough energy to monitor its sensing range throughout slot j, then it assigns itself a weight of ∞ at the beginning of slot j; otherwise it assigns itself a weight of $w_u(j) = \mu^{l_u(j)}/B_u$.

Sensor activation: Sensors that have infinite weights at the beginning of slot j do not activate themselves in slot j. Among the rest, sensors are activated (using the DSC algorithm described in Section IV) so that the subset of activated sensors, $S(j)$, constitutes an $O(\log n)$-approximate minimum weight sensor cover. The sensors that do not activate themselves in slot j, sleep in slot j. Refer to Fig. 2 for a pseudo-code of the activation phase of DLM.

Intuitively, DLM has been designed so that the sensors are activated so as to cover the target field whenever possible, and the sensors that have large residual energy are preferentially selected. We will later prove that the lifetime of DLM is at least $\frac{1}{O((\log n)(\log nB))}$ times that of the maximum lifetime of the network.

When there does not exist any more, a sensor cover such that each sensor in the cover has non-zero energy, the network lifetime is considered *terminated*. After the network lifetime termination, we can not provide any guarantee on the target field coverage, although the sensors with finite weights continue to execute the algorithm, and cover their sensing ranges.

Note that each sensor can determine its weight based only on local information. In the next section, we show how each sensor can execute the activation phase using distributed computations based only on local information obtained from its neighbors.

The DLM Activation phase of sensor u in slot j

begin

At the beginning of slot j:

if $b_u(j) > B_u - 1$ **then**
  $w_u(j) = \infty$.
  Enter sleep mode.
else
  Calculate $c_u(j) = \mu^{l_u(j)}$ and $w_u(j) = \frac{c_u(j)}{B_u}$.
  Use DSC in Fig. 3 to determine whether to stay active or enter sleep mode.
end if

end

Fig. 2. The DLM Algorithm

IV. Distributed Sensor Activation

We now describe an algorithm, which we call the Distributed Sensor Cover (DSC) algorithm, using which sensors can determine, using simple distributed computations, whether to activate themselves in each slot. Clearly, we need to design a sensor cover with guarantees on its weight using distributed computations. Note that a sensor cover is an instance of a *set cover*, and centralized algorithms that attain an $O(\log n)$-approximate set cover are well known [29]. We instead accomplish the same goal using distributed computations only, extending the design technique developed by Subhadrabandhu *et al.* [1] for the dominating set problem. We next describe our approach.

The sensor cover in each slot j is iteratively computed in an asynchronous manner[4]. At the beginning of the activation phase in each slot, all the sensors with finite weights are contending for staying active in the slot. At any time during the activation phase, each contending sensor u, determines the number of intersection points in P_u that have not yet been k-covered by the set of activated sensors, and computes its *activation preference ratio* (ar_u) as the ratio between its weight in slot j, $w_u(j)$, and the above number. We denote by *activation preference* (ap) of sensor u, the ordered pair $ap_u =< ar_u, ID(u) >$, where $ID(u)$ is sensor u's ID. We say that sensor u has lower ap than sensor v, i.e., $ap_u < ap_v$ if ap_u has lower lexicographic value than ap_v, that is, (i) $ar_u < ar_v$ or (ii) $ar_u = ar_v$ and $ID(u) < ID(v)$. Each contending sensor u communicates its activation preference to the sensors in T_u at the beginning of the activation phase and each time that its value changes. Note that the latter occurs only when one of u's neighbors in T_u becomes active. A contending sensor u activates itself once it detects that it has a lower activation preference than all contending sensors in T_u. Each sensor u that activates itself informs other sensors in T_u, accordingly. Once a sensor u detects that all the intersection points P_u in its sensing range are k-covered by the already active sensors in T_u, it updates its neighbors and enters a sleep mode. The activation process, in each slot, terminates after each sensor decides whether to stay active or enter a sleep mode. Refer to Fig. 3 for a pseudo-code.

[4]The sensors just need to know the beginning time of each time slot.

The Distributed Sensor Cover (DSC) algorithm of sensor u

Definitions:

- Let $UC_u \subseteq P_u$ be the set of intersection points that have not yet been k-covered by the set of activated sensors.
- Let $CT_u \subseteq T_u$ be the set of contending neighbors of sensor u.

Begin

if $w_u(j) = \infty$ or $P_u = \emptyset$ **then**
  $mode = sleep$
  Return $mode$
else
  $mode = contending$
  $UC_u = P_u$
  $CT_u = T_u$
  $ar_u = \frac{w_u(j)}{|UC_u|}$; $ap_u =< ar_u, ID(u) >$
  Send *My-Init-AP*(ap_u) message to every sensor $w \in T_u$
  Receive *My-Init-AP*(ap_w) message from every sensor $w \in T_u$
    // If *My-Init-AP* message not received from a sensor $w \in T_u$
    // within a given time period, then w is considered inactive
    // and it is removed from CT_u.
  if ($CT_u == \emptyset$ or $ap_u < ap_w$ for every $w \in CT_u$) **then**
    $mode = active$
    Send an *I-am-Active message* to every sensor $w \in CT_u$.
  end if

  while $mode == contending$ and upon reception of a message M from sensor $v \in CT_u$ **do**
    if the received message M is *I-Am-Active* **then**
      $CT_u = CT_u - \{v\}$
      // Let $NC_u \subseteq UC_u \cap P_{u,v}$ be the set of intersection
      // points that are k-covered (after v's activation).
      $UC_u = UC_u - NC_u$
      if ($UC_u == \emptyset$) **then**
        $mode = sleep$
        Send an *I-Am-Sleeping* message to every sensor $w \in CT_u$.
      else
        $old\_ap_u = ap_u$
        $ar_u = \frac{w_u(j)}{|UC_u|}$; $ap_u =< ar_u, ID(u) >$
        if ($CT_u == \emptyset$ or $ap_u < ap_w$ for every $w \in CT_u$) **then**
          $mode = active$
          Send an *I-Am-Active* message to each sensor $w \in CT_u$.
        else if ($old\_ap_u \neq ap_u$) **then**
          Send a *New-AP*(ap_u) message to each sensor $w \in CT_u$.
        end if
      end if
    else if the received message M is *New-AP*(ap_v) **then**
      Update ap_v
      if ($ap_u < ap_w$ for every $w \in CT_u$) **then**
        $mode = active$
        Send an *I-am-Active* message to each sensor $w \in CT_u$.
      end if
    else if the received message M is *I-Am-Sleeping* **then**
      $CT_u = CT_u - \{v\}$
      if ($CT_u == \emptyset$ or $ap_u < ap_w$ for every $w \in CT_u$) **then**
        $mode = active$
        Send an *I-am-Active* message to each sensor $w \in CT_u$.
      end if
    end if
  end while
  Return $mode$
end if

End

Fig. 3. The Distributed Sensor Cover (DSC) algorithm.

Clearly, each sensor can execute the above computations based only on locally available information, and the information it acquires in the Initialization phase (Subsection III-A). Recall that a sensor u enters a sleep mode only after all the intersection points P_u in its sensing range are already k-covered. Thus, according to Theorem 1, during the lifetime of the network (i.e., while there is no coverage hole) the subset of sensors activated at the end of the activation phase in each slot j, $S(j)$, induces a sensor cover for the network. Moreover, we will later prove that $S(j)$ constitutes an $O(\log n)$-approximate minimum weighted sensor cover.

As mentioned above, well-known centralized algorithms

such as the one in [29] can also be used to find an $O(\log n)$-approximate sensor cover. We now compare our DLM algorithm, which runs a distributed sensor cover selection algorithm in each slot, with two natural implementations of any given centralized sensor cover computation algorithm. In the first implementation, at the beginning of the network operation, a central controller such as a base station (i) collects the required information from each sensor, (ii) computes a complete schedule of the set of sensors to activate in each slot and (iii) distributes this schedule to the sensors. Note that in practice, sensors are prone to failure due to hardware malfunction or damage from the environment. Since the centralized algorithm computes a complete schedule only at the beginning of the network operation, sensor failure during network operation may cause coverage holes to form, which could persist for a long time. On the other hand, the DLM algorithm is more robust to sensor failure– it selects a sensor cover at the beginning of every slot from among the operating sensors at that point in time. So even if sensors fail in a slot, the resulting coverage holes will last only until the end of that slot.

The following alternative centralized implementation is more robust to sensor failure than the above centralized implementation: at the beginning of every slot, each operating sensor sends a message to the base station to notify the latter that it is operating. The base station then selects a sensor cover and informs each sensor whether to be active or not in that slot. However, in each slot, several network-wide message exchanges are required, which taxes the network resources. This overhead is not incurred under our DLM algorithm, since only a single network-wide broadcast is required at the beginning of the network operation and subsequently, messages only need to be exchanged locally in each slot.

V. The Initialization Phase

During the initialization phase (Subsection III-A), each sensor u gains the knowledge of, (i) the set P_u of intersection points that it covers, (ii) the identities of the sensors in T_u, which share intersection points with node u and (iii) the set $P_{u,v}$ of the intersection points in P_u that are covered by each sensor v in T_u (i.e., $P_{u,v} = P_u \cap P_v$ for each $v \in T_u$). We show that u can determine the above using localized computations based on simple geometric properties. In these computations, u only needs to know (a) r_u (b) N_u, and their ids, (c) $d_{u,v}$ and r_v for each $v \in N_u$ and (d) $d_{v,w}$ for each pair $v, w \in N_u$ such that v and w are neighbors of each other. We first provide a brief overview of the computations in Subsection V-A and subsequently present the details in Subsection V-B.

A. Overview

We assume that during the system activation every sensor u initially evaluates its distance to each one of its neighbors in N_u and it broadcasts these distances $d_{u,v}$, $v \in N_u$, as well as its sensing radius r_u to its neighbors. Next, u detects each neighbor $v \in N_u$ that intersects with u by using Property 1 in Subsection II-D and their joint intersection points. It also calculates the set Q_u of all the intersection points of u's sensing border with the sensing borders of its neighbors. For every intersection point $p \in Q_u$, u finds S_p, the set of sensors that cover p. Then, u communicates these sets S_p, $p \in Q_u$ to its neighbors. This process enables every neighbor $v \in N_u$ of u to know that a given intersection point $p \in Q_u$ is included in its sensing range and accordingly to add p to its set P_v of intersection points that it covers (for calculating (i) above). Moreover, the knowledge of each set S_p, $p \in P_v$, allows node v to identify its neighbors w that also cover each point $p \in P_v$ and update its set T_v accordingly (for calculating (ii) above). Node v can also calculate the sets $P_{v,w} = P_v \cap P_w$, for each $w \in T_v$ (for calculating (iii) above). Thus, to complete our description, we just need to present the process for detecting the set Q_u of any given sensor $u \in S$ and calculating the set S_p for every point $p \in Q_u$.

A major challenge in the initialization process is determining a unique identification for each intersection point. Since the sensors do not have any location information, the coordinates of the intersection points are unknown and cannot be used as identifiers. To overcome this difficulty, every intersection point of any pair u, v of intersecting sensors is identified by a triplet $IP(u, v, i)$, where u is the sensor with lower id, v is the sensor with higher id, and $i \in \{1, 2\}$ denotes the point index. Since every pair u, v of intersecting sensors have two common intersection points, the node with the lower id, say u, arbitrarily determines the index i of each point. In addition, u also calculates the set S_{p_i} for both points $p_i = IP(u, v, i)$, $i = 1, 2$ and communicates these sets to its neighbors, including node v. This ensures that each calculated set S_p corresponds to a single intersection point that is uniquely defined. We describe the calculation of such sets S_p in the next subsection.

B. Calculation of S_p

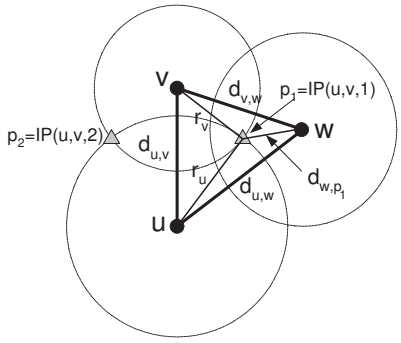

Fig. 4. A pair of intersecting nodes and their intersection points.

Consider a pair u, v of intersecting sensors and let $p_i = IP(u, v, i)$, $i = 1, 2$ denote their intersection points, as depicted in Fig. 4. We now describe a simple method for calculating the covering set $S_{p_i}, i = 1, 2$. First, note that a simple application of the triangle law on the distance metric and the fact that $\hat{R} \geq 2\hat{r}$ establishes that $S_{p_i} \subseteq N_u \cap N_v$. Now, our calculation proceeds in two steps. In *Step 1*, we partition $N_u \cap N_v$ in three sets: nodes that cover (i) none, (ii) only one (iii) both of p_1, p_2. In *Step 2*, we identify which nodes among the second set in the above partition cover p_1 (p_2, respectively). This completes the computation of both S_{p_1}, S_{p_2}. In absence of location information, we rely on the *Cosine Rule* throughout. Let $d_{a,b}$, $d_{a,c}$ and $d_{b,c}$ denote the distances between three points a, b and c accordingly, and let

$\angle a, b, c$ denote the angle [5] between the rays $[b, a]$ and $[b, c]$. The *Cosine Rule* states that

$$2 \cdot d_{a,b} \cdot d_{b,c} \cdot \cos \angle a, b, c = d_{a,b}^2 + d_{b,c}^2 - d_{a,c}^2 \qquad (1)$$

1) Step 1: For every node $w \in N_u \cap N_v$ that intersects with both u and v, we check if it covers one or both of the points p_i, $i = 1, 2$. Recall that the straight line (u, v) that traverses through the nodes u, v, partitions the plane into two halves, each one of which contains one of the points p_i, $i = 1, 2$. Without loss of generality, assume that w is located in the same half as p_1 and therefore, it is closer to p_1 than to p_2. We first find the angle $\angle w, u, p_1$ as follows. As shown in Figure 4, since p_1 and w are in the same half-plane, $\angle w, u, p_1 = |\angle v, u, w - \angle v, u, p_1|$ (note that unlike the case presented in Figure 4, $\angle w, u, p_1$ may also be equal to $\angle v, u, p_1 - \angle v, u, w$). Since the distances $d_{u,v}, d_{u,w}, d_{v,w}$, $d_{u,p_1} = r_u$ and $d_{v,p_1} = r_v$ are known, the cosine rule applied on the triangles $\triangle u, w, v$ and $\triangle u, v, p_1$ enable us to calculate the angles $\angle v, u, w$ and $\angle v, u, p_1$, and hence the angle $\angle w, u, p_1$. Now, in $\triangle u, w, p_1$, the distance $d_{u,w}$ is known and $d_{u,p_1} = r_u$. So d_{w,p_1} can be found using the Cosine rule. Thus, we can check whether w covers p_1 by checking whether $d_{w,p_1} < r_w$.

Similarly, we check if w covers p_2 by considering the triangle $\triangle u, w, p_2$. In this case the angle $\angle p_2, u, w = \angle v, u, w + \angle v, u, p_2$. From symmetry, $\angle v, u, p_2 = \angle v, u, p_1$. Thus, the angle $\angle p_2, u, w$ is known and d_{w,p_2} can be calculated by the cosine rule in $\triangle u, w, p_2$. Note that this process can also be used to calculate the distances d_{w,p_1} and d_{w,p_2} if w is located on the line (u, v). In such case, $d_{w,p_1} = d_{w,p_2}$; thus w covers both points or none of them.

2) Step 2: We now consider the set $Z \subseteq N_u \cap N_v$ of sensors that cover only one of the points p_1, p_2 and determine which sensors in Z cover p_1. Consider an arbitrary sensor $w \in Z$ and, without loss of generality, let p_1 be the point that it covers. Now, for each sensor $x \in Z$, $x \neq w$, we check, as described next, whether x covers p_1. As explained above, none of the nodes in Z is located on the line (u, v). Thus, all the sensors in Z that are located in the same half-plane as w cover p_1, while the others cover p_2. Thus, we just need to check if x and w are in the same half-plane. If w and x cover the same point, (and are therefore in the same half-plane), they must be neighbors. Thus, all the distances between every pair of nodes in $\{u, v, w, x\}$ are known and accordingly the three angles $\angle v, u, x, \angle v, u, w, \angle w, u, x$ can be calculated using the Cosine Rule. We use Property 3 to verify if x and w are in the same half-plane.

Property 3: Two sensors $w, x \in Z$ are located in the same half-plane, defined by the line (u, v), if and only if (1) $\angle w, u, x = |\angle v, u, w - \angle v, u, x|$ and (2) $\angle w, u, x + \angle v, u, w + \angle v, u, x < 360°$.

We omit the proof due to space constraints.

VI. Synchronization

We now discuss some synchronization related aspects of our algorithm. At the beginning of the network operation, the sensors synchronize their clocks using a distributed algorithm; see [33] for a survey of synchronization algorithms for wireless sensor networks. Then, each sensor exchanges distance information with its neighbors and carries out the initialization phase (see Section V). Subsequently, the sensors run the DSC algorithm in every slot.

Note that the only synchronization requirement in the DSC algorithm (Fig. 3) used for sensor cover computation in each slot is that sensors need to have synchronized clocks at the beginning of each slot. Thereafter, within the slot, the operation can be completely asynchronous. If the clocks of different sensors are accurately synchronized, then the DSC algorithm in Fig. 3 works correctly; otherwise the following problem occurs. Recall that at the beginning of the DSC algorithm, a sensor u sends its initial ap to each neighbor and then waits for the initial aps of its neighbors (see line 10 in Fig. 3). If u does not receive an initial-ap messsage from a neighbor w within a given time period, then it considers w to be inactive. Now, if the clock of a neighbor w lags behind u's clock, then w may send its initial ap after u decides that w is inactive. This may lead to an incorrect decision: u may activate itself even though w's ap is lower than u's.

We now describe a minor modification, with which the DSC algorithm works correctly even in the presence of discrepancies between the clocks of different sensors. Let the maximum time difference between the clocks of any two *neighboring* sensors be Δt, and let t_0 be a number slightly greater than Δt. When a sensor u finds that a slot has started according to its own clock, it listens to the channel for a duration t_0. Then, it sends out its initial ap and again listens to the channel for a duration t_0. If it does not receive an initial-ap message from a neighbor w in any of the two intervals, it assumes that w is inactive. Since $t_0 > \Delta t$, note that in the first interval, u receives aps of all neighbors whose slot starts before u's and in the second interval, it receives aps of all neighbors whose slot starts after u's. Thus, each node receives the initial ap of every other contending node at the end of the second interval [6].

Let the maximum time difference between the clocks of any two sensors in the network be ΔT, and in a given slot, let y be the sensor whose clock leads that of all other sensors. Then it is easy to check that a sensor activates itself within an interval $\Delta T + 2t_0 \approx \Delta T + 2\Delta t$ after the slot has started according to the clock of y. Now, ΔT is of the order of a few milliseconds because several efficient synchronization protocols, which have an accuracy of a few milliseconds, have been developed [33]; also, Δt is much smaller than ΔT. Also, the lifetimes of sensors are typically of the order of at least several days [34]; so a slot duration would be of the order of at least several minutes [7]. Hence, the increase in the convergence time of DSC (approximately $\Delta T + 2\Delta t$) compared to the case in which clocks of all sensors are accurately synchronized is negligible compared to a slot duration. Also, note that even if ΔT is much larger than Δt, DSC works correctly provided the above scheme is used with $t_0 > \Delta t$.

[5] Throughout, by $\angle a, b, c$ we mean the angle between rays $[b, a]$ and $[b, c]$ that is less than or equal to $180°$.

[6] For simplicity, in this discussion, we have assumed that the message propagation times between nodes are 0. The scheme can be easily generalized to handle non-zero propagation times.

[7] The smaller the slot duration, the larger the overhead due to sensor cover computations. So the slot duration must be as large as possible, while ensuring that Assumption 1 is satisfied. However, the condition in Assumption 1 is not stringent. For example, when $n = 1000$ and B_u is equal for all sensors u, it requires that the slot duration must be at least a factor 17 lower than the lifetime of a sensor. For smaller n, the condition is even more relaxed.

VII. Detection of Lifetime Termination

We now augment our scheme with a simple distributed mechanism for detecting the termination of the network lifetime. By definition the network lifetime terminates when there no longer exists a sensor cover such that every sensor in the cover has non-zero energy. Thus, from Theorem 1, the network lifetime ends once one of the intersection points in the IP set P (Subsection II-D) cannot be k-covered by the sensors that still have non-zero energy. Note that every point $p \in P$ is included in the closure of the sensing range of at least one internal node. Thus, once an internal node u detects that u itself and all its neighbors in N_u have already declared that they are either active or in sleep mode, it checks if each one of the intersection points in $P_u \cup Q_u$ is k-covered by the set of the active nodes in $N_u \cup \{u\}$. This is a simple test, as for every point $p \in (P_u \cup Q_u)$, the set S_p of sensors that cover p is already known (Section V). Node u informs the administrators about the coverage hole once this test fails.

VIII. Scheme Analysis

We now prove correctness and performance guarantees for the DLM algorithm. In Subsection VIII-A, we prove the guarantees for DSC which DLM invokes. Using the above, in Subsection VIII-B we prove the guarantees for DLM.

A. DSC Algorithm– Analysis

We prove that DSC computes an $O(\log n)$-approximate minimum weight sensor cover. Note that all the proofs allow for arbitrary, but finite transit times of status update messages transmitted by nodes to their neighbors.

Theorem 2: At every activation phase, (i) DSC computes a sensor cover if there is no coverage hole, (ii) DSC terminates in at most $2nV$ time if V is an upper bound on the transit delay of status update messages between the neighbors and (iii) DSC terminates in finite time, if the transit delays are finite but can not be upper-bounded.

We omit the proof due to space constraints.

Remark 1: Note that if, in a particular execution, DSC finds a sensor cover with $\tilde{n}$ sensors, then the bound in (ii) in Theorem 2 can be improved to show that DSC terminated in at most $2\tilde{n}V$ time. Also, note that the bound of $2\tilde{n}V$ is not tight because sensors do not activate themselves serially, but sets of sensors activate themselves in parallel. As shown by our simulations in Section IX-C, in practice, DSC converges in a time that is much lower than this bound.

Now, recall that finding a minimum weight sensor cover is an instance of the minimum weight set cover problem. We now briefly describe the well-known greedy Centralized Set Cover (CSC) algorithm that computes an $O(\log n)$-approximate minimum weight set cover [29]. At each iteration, it selects the sensor that has the lowest activation preference (ap) among all the sensors, where ap is defined in the same way as for DSC, and then updates the ap's of the unselected sensors. This process continues until the set of selected sensors constitutes a sensor cover.

Theorem 3: For a given setting and a set of weights to the sensors, DSC and CSC select the same set of sensors. Thus, DSC obtains an $O(\log n)$-approximate minimum weight sensor cover.

Proof: Let $Y^C = \{v_1, \cdots, v_{m^C}\}$ and $Y^D = \{u_1, \cdots, u_{m^D}\}$ be the sets of selected sensors by CSC and DSC, respectively, sorted in increasing order according to their ap values at the time that they were selected [8] (i.e., decided to stay active). Let v_j and u_j be the j-th sensors in Y^C and Y^D respectively, and let ${ap^C}_j$ and ${ap^D}_j$ be their ap values. Moreover, let ${Y^C}_j = \bigcup_{i=1}^{j} v_j$ and ${Y^D}_j = \bigcup_{i=1}^{j} u_j$ be the first j sensors in sets Y^C and Y^D respectively. Note that the sensors in Y^C are arranged in the order in which they were selected by CSC. However, the order on the sensors in Y^D is not necessarily the order in which they are activated by DSC.

Our proof utilizes the following properties:

(1) During the execution of DSC, the ap of each node is an increasing function of time.

(2) Consider any node $u \in Y^D$. Every sensor $w \in Y^D \cap T_u$ with lower ap value than u was selected before u by DSC. Similarly, any node $w \in Y^D \cap T_u$ with higher ap value than u was selected after node u by DSC.

This property follows from property (1) and from the fact that under DSC, a sensor u becomes active only when (and if) it has lower ap value than its unselected neighbors in T_u.

(3) The ap value of any node u during the execution of CSC and DSC is determined only by its already selected neighbors in T_u.

(4) Suppose $u \in Y^D$ becomes active at time t_1 under DSC. Then, for each $w \in Y^D \cap T_u$ that became active before t_1, u received an activation message from w before time t_1.

If this were not true for some w, then note that u would not have activated itself at t_1, since it would find its own ap to be higher than that of w.

We seek to prove that $Y^C = Y^D$. Let $Y^C \neq Y^D$ instead, and let j be the lowest index such that $v_j \neq u_j$. Initially, let us show by contradiction that $j \leq min(m^C, m^D)$. First, let $m^C > m^D$ and $j > m^D$. But then, the first m^D sensors selected by CSC constitute a sensor cover and therefore CSC terminates after selecting at most the first m^D sensors. Now, let $m^C < m^D$ and $j > m^C$ (in particular $j = m^C + 1$) and consider the vicinity of the node u_j. From Property (2), node u_j was selected by DSC after every node in ${Y^D}_{j-1} \cap T_{u_j} = {Y^D}_{m^C} \cap T_{u_j} = Y^C \cap T_{u_j}$. However, since Y^C is a sensor cover, all intersection points in u_j's sensing range are k-covered once DSC selects the nodes in $Y^C \cap T_{u_j}$. Thus, DSC does not select u_j after it has selected the sensors in ${Y^D}_{j-1} \cap T_{u_j}$, and thus it does not select u_j at all. Thus, $j \leq min(m^C, m^D)$.

We now show that ${ap^D}_j \geq {ap^C}_j$. If not, then ${ap^D}_j < {ap^C}_j$ and consider the j-th iteration of CSC. The algorithm selects as the j-th active sensor, the unselected sensor with minimum ap value. Recall that at this stage u_j has not been selected by CSC. Since ${Y^C}_{j-1} = {Y^D}_{j-1}$, from properties (2), (3) and (4) above, it follows that at the j-th iteration of CSC the ap value of node u_j is the same as ${ap^D}_j$ calculated by DSC. This is true since the ap value of node u_j depends only on its selected neighbors in ${Y^C}_{j-1} \cap T_{u_j} = {Y^D}_{j-1} \cap T_{u_j}$, which are the same sets [9] for both algorithms. Thus, CSC should select

[8] Here, by ap value of a node $u \in Y^D$, we mean the latest ap value calculated by u.

[9] Note that by property (4), just before u_j selected itself under DSC, it had updated its ap to account for the fact that all nodes in ${Y^D}_{j-1} \cap T_{u_j}$ had activated themselves.

node u_j rather than node v_j, which contradicts the assumption that ${ap^D}_j < {ap^C}_j$. Thus ${ap^D}_j \geq {ap^C}_j$.

We next show that ${ap^D}_j \leq {ap^C}_j$. If not, then ${ap^D}_j > {ap^C}_j$. Since v_j is in Y^C, it holds that nodes in ${Y^C}_{j-1}$ do not cover all the intersection points covered by the node v_j. Thus, there are some nodes, denoted by set W, in the vicinity of v_j, i.e., $W \subseteq T_{v_j} \cup \{v_j\}$, that are selected by DSC and are not in ${Y^D}_{j-1}$. First, assume that v_j is the first node in W selected by DSC. From Property (3) above, v_j's ap value is determined only by the selected sensors in ${Y^D}_{j-1} = {Y^C}_{j-1}$. Thus, by Property (4), v_j's ap value at the time it is selected by DSC, is the same value as that at the time it is selected by CSC, i.e., v_j's ap value is ${ap^C}_j$, which contradicts the assumption that ${ap^D}_j > {ap^C}_j$. Thus, v_j is not the first node in W selected by DSC. Let $x \in W$, $x \neq v_j$ be the first node in W selected by DSC and let ${ap^D}_x$ denote its ap value at the time it was selected by DSC, say time t_x. From our assumption, it follows that ${ap^D}_x \geq {ap^D}_j > {ap^C}_j$. Now, consider the ap value of node v_j as calculated by DSC just before time t_x when node x is selected. Since $x \neq v_j$ is the first node in W selected by DSC, just before time t_x, the neighbors of v_j selected by DSC must be from the set ${Y^D}_{j-1}$. (Note that all the neighbors of v_j in ${Y^D}_{j-1}$ may not necessarily have been selected). From Property (3), it holds that the ap value of v_j is determined only by its selected neighbors. Thus, the ap value of v_j just before time t_x as calculated by DSC, denoted by ${ap^D}_{v_j}$, is at most ${ap^C}_j$. Thus, ${ap^D}_{v_j} \leq {ap^C}_j < {ap^D}_j \leq {ap^D}_x$. But then, v_j should have been selected by DSC rather than node x and its ap value should have been ${ap^D}_{v_j}$, which contradicts the assumption that ${ap^D}_j > {ap^C}_j$.

Thus, ${ap^D}_j = {ap^C}_j$. Hence, $ID(u_j) = ID(v_j)$. Thus, $u_j = v_j$, which is a contradiction. The result follows. ∎

The following lemma gives the message complexity of DSC.

Lemma 1: The number of messages transmitted in a run of DSC is at most $n(\tilde{n}+2)$, where $\tilde{n}$ is the size of the sensor cover found. Hence, the average number of messages transmitted per sensor is at most $(\tilde{n}+2)$, which is upper bounded by $(n+2)$. We omit the proof due to space constraints.

Remark 2: Our simulations reveal that the actual number of messages per node is much lower than $\tilde{n}+2$ (see Section IX-C).

B. DLM Algorithm– Analysis

We now prove an approximation ratio for the lifetime attained by the Distributed Lifetime Maximization (DLM) algorithm in Fig. 2. Our analysis is similar to the ones used by Aspnes *et al.* [26] for online machine scheduling and virtual circuit routing problems, and Awerbuch *et. al* [4], [5] for the online virtual circuit routing problem.

Recall from Section II-A that a sensor that is active in a slot consumes 1 unit of energy and a sensor in sleep mode consumes no energy. Throughout this section, all logarithms are to the base 2. Finally, for proving the approximation ratio, we additionally assume that the initial energy of each sensor is large enough:

Assumption 1: $B_u \geq \log \mu,\ u \in S$.

For simplicity, in the proof, we assume that $B_u, u \in S$ are integers. The proof can be easily extended to the case when they are real numbers.

1) The DLM-T Algorithm: We describe in Fig. 5, DLM-T (Truncated DLM), a modified version of DLM, that will be used to prove an approximation ratio for DLM.

The DLM-T Algorithm

begin

1: Let $c_u(j) = \mu^{l_u(j)}$ and $w_u(j) = \frac{c_u(j)}{B_u}$ be the weight of sensor u at the beginning of slot j.
2: At the beginning of slot j:
3: Using DSC in Fig. 3, find an $O(\log n)$-approximate minimum weight sensor cover $S(j)$ with weight:

$$W(j) = \sum_{u \in S(j)} w_u(j)$$

4: If $W(j) \leq 2n$, then activate the sensor cover $S(j)$ in slot j, otherwise declare the network as dead.

end

Fig. 5. The DLM-T Algorithm

Note that DLM-T differs from DLM in the following: (i) the criterion it uses to declare the network as dead (step 4) (ii) it does not use a weight equal to ∞ for a sensor u with 0 remaining energy, but a weight of $\frac{\mu}{B_u}$ (iii) it considers all nodes in the sensor cover selection process whereas DLM considers only those that have at least one unit of energy remaining. It is therefore not clear whether DLM-T selects nodes that have at least one unit of energy left. The next lemma however shows that this is indeed the case.

Lemma 2: Under the DLM-T algorithm, if a sensor is activated at the beginning of slot j, it has at least one unit of energy remaining.

Proof: We need to show that for any $j \geq 1$, for any $u \in S(j)$, $l_u(j) \leq 1 - \frac{1}{B_u}$. Note that $W(j) \leq 2n$. Thus, for any such u,

$$w_u(j) \leq W(j) \leq 2n$$

Hence,

$$\mu^{l_u(j)} = B_u w_u(j) \leq 2nB_u \leq 2nB = \frac{\mu}{2} = \mu^{1-\frac{1}{\log \mu}}$$

where the last equality follows since the logarithms are to the base 2. So,

$$l_u(j) \leq 1 - \frac{1}{\log \mu} \leq 1 - \frac{1}{B_u}$$

by Assumption 1. The result follows. ∎

The next result establishes the relation between the lifetimes of the DLM and DLM-T algorithms.

Lemma 3: The lifetime of the network under the DLM algorithm is greater than or equal to that under DLM-T.
We omit the proof due to space constraints.

Note that unlike DLM, DLM-T requires not only the determination of an $O(\log n)$-approximate minimum weight sensor cover, but also the calculation of its weight. The latter requires network-wide coordination. Nevertheless, it follows from Lemma 3 that *any approximation ratio that holds for the lifetime of DLM-T, holds for DLM as well. We therefore prove an approximation ratio for DLM, by proving one for DLM-T next.*

2) Approximation Ratio: Now, we prove that the lifetime achieved by DLM-T is at most a factor $O((\log n)(\log \mu))$ lower than that achieved by an optimal algorithm OPT. Although DLM-T computes a sensor cover once every slot, we allow OPT to compute sensor covers M times every slot ($M \geq 1$), and obtain the above approximation ratio irrespective of M. Specifically, we divide each slot into M equal parts, which we call mini-slots, where $M \geq 1$ is an arbitrary integer, and allow the OPT algorithm to compute a sensor cover at the beginning of each mini-slot.

Let L be the network lifetime under the DLM-T algorithm and L^* be the network lifetime under OPT. Without loss of generality, assume that L^* is an integer [10]. Also, let $\mathcal{L} = \{1, \ldots, L\}$ be the set of slots when the network is alive under DLM-T and $\mathcal{L}^* = \{L+1, \ldots, L^*\}$ be the set of slots when the network is dead under DLM-T, but alive under OPT.

We can view the situation after the network dies under DLM-T as if at the beginning of every slot $j \in \mathcal{L}^*$, the network finds an approximate minimum weight sensor cover (it finds the same sensor cover for each $j \in \mathcal{L}^*$) and since the weight of this cover is greater than $2n$, it does not activate it. Under DLM-T, no sensor is activated after slot L and hence the weights of all sensors remain unchanged thereafter.

Similar to the definitions of $b_u(j)$, $l_u(j)$, $c_u(j)$ and $w_u(j)$, define $b_u(j,l)$ to be the amount of energy of sensor u that has been consumed at the beginning of the l'th mini-slot of the j'th slot, $l_u(j,l) = \frac{b_u(j,l)}{B_u}$, $c_u(j,l) = \mu^{l_u(j,l)}$ and the weight of sensor u to be $w_u(j,l) = \frac{c_u(j,l)}{B_u}$.

Let $S(j)$ be the sensor cover found by DLM-T in slot j and $W(j)$ be its weight. Let $S^*(j,l)$ be the sensor cover used by OPT in the l'th mini-slot of slot j, where $l \in \{1, \ldots, M\}$. Also, let $W^*(j,l)$ be the sum of the weights of the sensors in $S^*(j,l)$ at the beginning of the l'th mini-slot of the j'th slot, *when the network is running DLM-T*. We emphasize that the sensor cover $S^*(j,l)$ is the one used by OPT in the l'th mini-slot of slot j, but the weights of the sensors in $W^*(j,l)$ are those when the network is running DLM-T.

From the facts that (i) in each slot j, DLM-T finds an $O(\log n)$-approximate minimum weight sensor cover and (ii) the weight of each sensor is an increasing function of time, we get that there exists a constant α such that for each slot j:

$$W(j) \leq (\alpha \log n) W^*(j,l), \;\; l \in \{1, \ldots, M\}. \tag{2}$$

The following theorem proves the approximation ratio achieved by DLM-T.

Theorem 4: L^* is at most an $O((\log n)(\log \mu))$ factor greater than L.

Theorem 4 proves the surprising result that DLM achieves a non-trivial approximation ratio ($O((\log n)(\log \mu))$) even when compared to an optimal algorithm that computes sensor covers much more frequently than DLM and the approximation ratio is independent of M. Note that as M is increased, OPT's lifetime may increase (but is upper bounded by nB), and thus the ratio between the lifetimes attained by OPT and DLM may well be different for different M. However, our analysis reveals that this ratio is upper-bounded by $O((\log n)(\log \mu))$ for all M.

[10] If L^* is not an integer, then in the proof of Theorem 4, we can replace L^* by $\lfloor L^* \rfloor$ and then use the fact that $L^* \leq \lfloor L^* \rfloor + 1$.

The proof proceeds as follows. We first upper bound the amount by which the network lifetime under OPT can exceed that under the DLM-T algorithm (Lemma 4). Next, we lower bound the lifetime achieved by DLM-T (Lemma 5). Finally, we obtain an upper bound on the ratio $\frac{L^*}{L}$ by combining the above bounds.

Lemma 4:

$$L^* - L \leq \frac{\alpha \log n}{2n} \sum_{u \in S} c_u(L+1) \tag{3}$$

Proof: We define the indicator function:

$$I\{u \in S^*(j,l)\} = \begin{cases} 1 & \text{if } u \in S^*(j,l) \\ 0 & \text{else} \end{cases}$$

Since $W(j) > 2n$ for $j \in \mathcal{L}^*$, from (2) it follows that:

$$W^*(j,l) \geq \frac{2n}{\alpha \log n}, \;\; \forall j \in \mathcal{L}^*, l \in \{1, \ldots, M\}$$

Summing the above over $j \in \mathcal{L}^*$ and l:

$$\sum_{j \in \mathcal{L}^*} \sum_{l=1}^{M} W^*(j,l) \geq \frac{2n}{\alpha \log n} (L^* - L) M \tag{4}$$

Hence,

$$\begin{aligned}
&\frac{2nM}{\alpha \log n}(L^* - L) \\
\leq \;& \sum_{j \in \mathcal{L}^*} \sum_{l=1}^{M} \sum_{u \in S^*(j,l)} \frac{1}{B_u} c_u(j,l) \\
= \;& \sum_{j \in \mathcal{L}^*} \sum_{l=1}^{M} \sum_{u \in S^*(j,l)} \frac{1}{B_u} c_u(L+1) \qquad (5) \\
= \;& \sum_{j \in \mathcal{L}^*} \sum_{l=1}^{M} \sum_{u \in S} \frac{c_u(L+1)}{B_u} I\{u \in S^*(j,l)\} \\
= \;& \sum_{u \in S} c_u(L+1) \left[\frac{1}{B_u} \sum_{j \in \mathcal{L}^*} \sum_{l=1}^{M} I\{u \in S^*(j,l)\} \right] \\
\leq \;& M \sum_{u \in S} c_u(L+1) \qquad (6)
\end{aligned}$$

where in (5), we used the fact that since the network is dead under DLM-T at the beginning of slot $L+1$, the energy of each sensor remains same thereafter and hence $c_u(j,l) = c_u(L+1) \;\forall j \in \mathcal{L}^*, l \in \{1, \ldots, M\}$. Also, we get (6) from the inequality:

$$\sum_{j \in \mathcal{L}^*} \sum_{l=1}^{M} I\{u \in S^*(j,l)\} \leq M B_u,$$

which is true because its left hand side is the total number of mini-slots in slots $j \in \mathcal{L}^*$, in which sensor u is activated under OPT, and cannot exceed M times u's initial energy B_u.

The result follows from (6). ∎

Lemma 5:

$$\sum_{u \in S} c_u(L+1) \leq n(2L \log \mu + 1) \tag{7}$$

Proof: We begin by upper bounding the total growth in the functions $c_u(.)$ of sensors $u \in S(j)$ over slot j. For slot $j \in \mathcal{L}$, we have:

$$\begin{aligned}\sum_{u\in S(j)} (c_u(j+1) - c_u(j)) &= \sum_{u\in S(j)} (\mu^{l_u(j)+\frac{1}{B_u}} - \mu^{l_u(j)}) \\ &= \sum_{u\in S(j)} \mu^{l_u(j)}(2^{\frac{\log\mu}{B_u}} - 1) \\ &\leq \sum_{u\in S(j)} \mu^{l_u(j)} \left(\frac{\log\mu}{B_u}\right) \quad (8) \\ &= \log\mu \sum_{u\in S(j)} \frac{\mu^{l_u(j)}}{B_u} \\ &\leq 2n\log\mu \quad (9)\end{aligned}$$

where (8) results from the facts that $\frac{\log\mu}{B_u} \leq 1$ by Assumption 1 and $2^x - 1 \leq x \ \ \forall x \in [0,1]$. Inequality (9) follows from:

$$\sum_{u\in S(j)} \frac{\mu^{l_u(j)}}{B_u} = W(j) \leq 2n$$

which is true because the network is not declared dead by DLM-T at the beginning of slot j.

Now, in slot j, the energy of sensors $u \notin S(j)$ does not change and hence $c_u(j+1) = c_u(j) \ \ \forall u \notin S(j)$. So we get:

$$\begin{aligned}\sum_{u\in S}(c_u(j+1) - c_u(j)) &= \sum_{u\in S(j)} (c_u(j+1) - c_u(j)) \\ &\leq 2n\log\mu\end{aligned}$$

Summing this inequality over $j \in \mathcal{L}$:

$$\sum_{j=1}^{L}\sum_{u\in S}(c_u(j+1) - c_u(j)) \leq 2nL\log\mu$$

The left hand side is a telescoping sum. So we get:

$$\sum_{u\in S} c_u(L+1) \leq 2nL\log\mu + \sum_{u\in S} c_u(1)$$

But $c_u(1) = \mu^0 = 1 \ \ \forall u \in S$. Thus,

$$\sum_{u\in S} c_u(L+1) \leq n(2L\log\mu + 1)$$

■

Proof of Theorem 4: By Lemmas 4 and 5:

$$L^* \leq L(\alpha(\log n)(\log\mu) + 1) + \frac{\alpha\log n}{2}$$

The result follows since α is a constant. ■

Remark 3: For simplicity, we assumed that OPT computes a sensor cover every $\frac{1}{M}$ slots, for an integer M. It is easy to show that Theorem 4 continues to hold even when OPT computes a sensor cover every τ slots for an arbitrary real number $\tau > 0$.

IX. Simulations

We now evaluate the performance of DLM using simulations. We consider a WSN with n sensors, each with an initial energy of B units, sensing and transmission radii of 10 and 22 units respectively, deployed uniformly at random in a 50×50 units^2 target field. Each time slot was 1 unit long.

A. Lifetime Comparison with Other Schemes

We compared the lifetimes of the network under three algorithms: the DLM algorithm (Fig. 2), the Garg-Konemann (GK) algorithm [10] and a heuristic proposed in [12], [20] that we denote by Min-Num. At every slot, Min-Num finds a sensor cover with the minimum number of nodes (up to an $O(\log n)$ factor) and activates it. GK [10] generates a sequence of sets of weights to assign to the sensors and finds $O(\log n)$-approximate minimum weight sensor covers for each set of weights. When the initial energy of each sensor is the same, each sensor cover selected by GK is activated for an equal amount of time, which is a monotonically increasing function of an input parameter ϵ. Thus, the number of sensor cover computations per slot, and hence the computation time required for GK, increases as ϵ decreases. The lifetime approximation ratio guaranteed for the GK algorithm however worsens with increase in ϵ [11].

First, we plot in Fig. 6, lifetimes achieved by DLM, Min-Num, GK as a function of n, for $B = 15$. For GK, we select (i) ϵ such that it computes sensor covers at the same rate per unit time as DLM and Min-Num (i.e., approximately once every slot) (denoted by GK(1 slot)) and (ii) $\epsilon = 0.05$ (denoted by GK($\epsilon = 0.05$)). In (ii), GK computes sensor covers between 127.2 (for $n = 100$) and 146.1 (for $n = 200$) times per slot for the range of n we considered. Next, in Fig. 7, we plot the lifetimes of DLM, Min-Num, GK(1 slot) [12] and GK($\epsilon = 0.05$) as a function of B for $n = 150$. GK($\epsilon = 0.05$) computes sensor covers between 41.5 (for $B = 50$) and 138.3 (for $B = 15$) times per slot for the considered range of B. The figures show that the lifetimes achieved by GK($\epsilon = 0.05$) are very close to those by DLM, whereas those by GK(1 slot) are much lower. So GK and DLM perform similarly only when GK computes sensor covers much more frequently than DLM, and DLM outperforms GK otherwise. Thus, although GK guarantees a better approximation ratio (while using centralized computation and location information), in practice, DLM outperforms GK. DLM substantially out-performs Min-Num as well, which suggests that lifetime can be substantially enhanced by deciding which sensors to activate based on their residual energy.

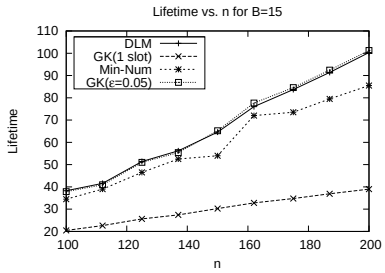

Fig. 6. Plot of lifetimes of DLM, GK (1 slot), Min-Num and GK ($\epsilon = 0.05$) vs. n for $B = 15$ units

[11] The network lifetime under the GK algorithm is guaranteed to be at most a factor $(1+\epsilon)f$ less than the optimal lifetime, where $f = O(\log n)$ is the approximation ratio of the algorithm used for finding minimum weight sensor covers [10].

[12] The values of ϵ used for GK(1 slot) lie in [0.67,0.72] for Fig. 6 and in [0.35,0.70] for Fig. 7.

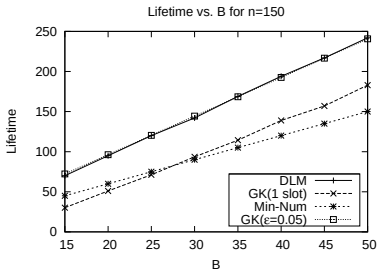

Fig. 7. Plot of lifetimes of DLM, GK(1 slot), Min-Num and GK ($\epsilon = 0.05$) vs. B for $n = 150$

B. Sensitivity of Lifetime to μ

Recall that DLM assigns weights to sensors using a parameter μ. In order to study the sensitivity of DLM to the value of μ, we fixed the values of n and B and plot in Fig. 8, the lifetime of DLM using values of μ between 2 and 40000. The plot shows that the lifetime achieved by DLM is roughly

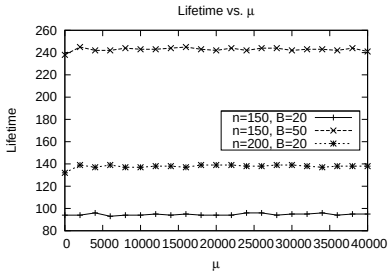

Fig. 8. Plot of lifetimes of the DLM algorithm vs. μ for ($n = 150, B = 20$), ($n = 150, B = 50$) and ($n = 200, B = 20$)

the same for different values of μ. This suggests that high lifetimes can be attained even without selecting $\mu = 4nB$, as required for proving the approximation guarantee. Thus, in practice, the sensors need not exchange any global information (specifically, n, B) even during the initialization phase.

C. Convergence Time and Message Complexity of DSC

Now, we study via simulations, the amount of time taken by DSC to converge. We consider the impact of different models for the message propagation times on the convergence time. Under each of these models, whenever a node transmits a status update message (of type "I-Am-Active", "Change-of-AP" or "I-Am-Sleeping"), the delay until reception by each contending neighbor is an independent, identically distributed (i.i.d.) random variable. We consider three different distributions for this delay: Constant, Uniform and Geometric. In the Constant model, the delay is constant and equals 1 nunit (nanounit, *i.e.*, 1×10^{-9} units) [13]. In the Uniform model, the delay is uniformly distributed in the range $\left[\frac{2}{3}, \frac{4}{3}\right]$ nunits. In the Geometric model, the delay equals $\sum_{i=1}^{\tilde{N}} \tilde{X}_i$, where $\tilde{N}$ has a geometric distribution [28] with success probability $p = 0.9$ and $\tilde{X}_1, \tilde{X}_2, \ldots$ are i.i.d. random variables uniformly distributed in the range $\left[\frac{20}{21}p, \frac{22}{21}p\right]$ nunits. Note that the Geometric model simulates the scenario in which a message may be received in error by the receiver with some probability and is repeatedly retransmitted until received successfully. For a fair comparison, the parameters of each model have been selected such that the expected delay is 1 nunit.

We plot in Fig. 9, the average convergence times of the DSC algorithm over the network lifetime for the above three delay models as a function of n. We also plot 2 times the average of the size ($\tilde{n}$) of the sensor cover found by DSC [14]. The figure shows that under each delay model, DSC converges in a time that is significantly lower than $2n$ times the expected delay of 1 nunit, *i.e.* $2n$ nunits, and also much lower than $2\times$ (the average $\tilde{n}$) nunits. Thus, the convergence times of DSC in practice are much lower than the pessimistic analytical bounds in Theorem 2 and Remark 1. Fig. 9 also shows that the convergence time in all three delay models is very small (less than 37 nunits) compared to the slot duration.

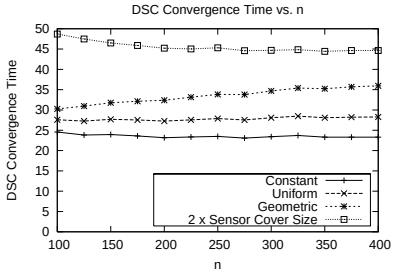

Fig. 9. The convergence times (in nunits) of the DSC algorithm versus n for the Constant, Uniform and Geometric delay models. Also shown is 2 times the average sensor cover size versus n.

Fig. 9 also shows that the convergence time decreases in n for the Constant model, and increases for the Uniform and Geometric models. It increases slightly for the Uniform model and somewhat faster for the Geometric model, but seems to saturate for the latter. We now explore the reasons behind these trends. Fig. 9 shows that the average size of the sensor cover decreases in n. Intuitively this is because, for higher n, there are more sensors to choose the sensor cover from and hence smaller sensor covers are more likely to exist. This explains the observed decrease in the convergence time with n for the Constant model– for higher n, DSC finds a sensor cover with fewer sensors and hence requires fewer status update messages. However, the convergence time for the Uniform and Geometric models increases in n because of the following effect. For greater n, the average number of neighbors of a node is higher. Hence, the average delay in communicating a status update message to *all* contending neighbors is higher. This leads to an increase in DSC's convergence time.

Now, we study the message complexity of DSC. Fig. 10 shows the average number of messages transmitted per node [15] in a sensor cover computation, averaged over the network lifetime, and the upper bound analytically established in Lemma 1. The figure shows that the average number of

[13]Recall, from Section VI, that a slot duration is of the order of at least several minutes. Also, the expected delay would be of the order of a few microseconds. Hence, the expected delay would be of the order of 10^{-9} of a slot duration, *e.g.*, the slot duration and the expected delay may be $1000s$ and $1\mu s$ respectively.

[14]Note that by Theorem 3, DSC finds the same sensor cover independent of the delay model.

[15]The number of messages in the plot is for the Constant delay model. The number of messages for the Uniform and Geometric delay models are very close to that for the Constant model.

messages transmitted per node in a sensor cover computation is quite low (less than 6.2). Hence, the energy consumption due to the transmission of messages is small. Fig. 10 also shows that the message complexity is much lower than the upper bound. This is because, when a node activates itself, the ap of a small number of nodes changes; in the calculation of the bound, this number was upper bounded by n.

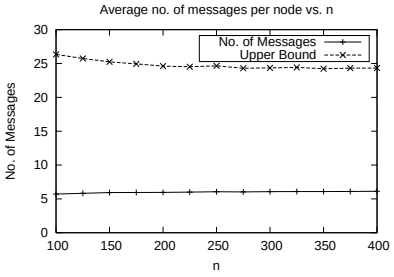

Fig. 10. The average number of messages transmitted per node during the DSC algorithm and the upper bound established in Lemma 1, versus n.

D. Inaccurate Distance Measurements

In our analysis, we assumed that each node accurately knows the distances between pairs of adjacent nodes in its vicinity. However, in practice, there may be some errors in the distance measurements. We now study the effect of inaccurate distance measurements on the performance of our algorithm. We assume that the measured distance between adjacent nodes u and v is given by:

$$Eval\_Dist_{u,v} = d_{u,v} \cdot (1 + X \cdot Error\_Index)$$

where, $d_{u,v}$ is the actual distance between the nodes, $X \sim N(0,1)$ is a normal random variable and $Error\_Index$ is a simulation parameter that controls the variance of the distance measurement errors. We varied $Error\_Index$ in the range $[0.5, 5\%]$. Note that the error in distance measurements has been experimentally found to be normally distributed [31], and values of its variance similar to the ones we use have been used in prior work [32].

Now, in the presence of random errors in the distance measurements, coverage cannot always be guaranteed while also maximizing lifetime. (Coverage can of course be guaranteed even without the knowledge of distances by trivial algorithms such as those that activate all sensors, but these algorithms will have low lifetimes). So instead, we seek a tradeoff between coverage and lifetime, which we achieve via a minor heuristic modification to our algorithm, which is described below.

Recall that under the DSC algorithm, a sensor goes to sleep when it finds that all intersection points in its sensing range are covered by active sensors. When there are errors in distance measurements, a sensor u may erroneously conclude (i) that an intersection point in its sensing range, which is actually covered by a sensor v, is not covered (false negative) or (ii) that an intersection point in its sensing range, which is not actually covered by an active sensor v, is covered (false positive). Our simulations revealed that false negative instances occur much more frequently than false positive instances. However, a simple heuristic modification, which we describe next, can be used to reduce the number of false negative instances and thereby attain the desired tradeoffs between coverage and lifetime.

While checking whether an intersection point in its sensing range is covered by a sensor v, a sensor u uses $r_v(1 + Margin * Error\_Index)$, instead of r_v, as the sensing radius of sensor v, where $Margin$ is a parameter. The larger the $Margin$, the more likely u is to conclude that v covers the intersection point, which reduces the number of false negatives. The top plot in Fig. 11 shows the lifetime, normalized with respect to the lifetime with no errors in distances, and the fraction of the target field covered versus $Margin$ for $Error\_Index = 5\%$. The plot shows that the lifetime increases significantly as $Margin$ is increased, but the fraction of the target field covered drops slightly below 1. Note that for large choices of $Margin$, the lifetime is more than the lifetime with no errors in distances, but the deterministic coverage guarantee is lost. The bottom plot in Fig. 11 shows the normalized lifetime versus $Error\_Index$ when $Margin$ is chosen so that, approximately, a fraction 0.95 of the target field is covered. It can be seen that the normalized lifetime is close to 1 for all values of $Error\_Index$. Thus, in the presence of errors in distance measurements, the desired tradeoff between lifetime and coverage can be achieved by using the above heuristic modification to the algorithm and selecting the margin appropriately.

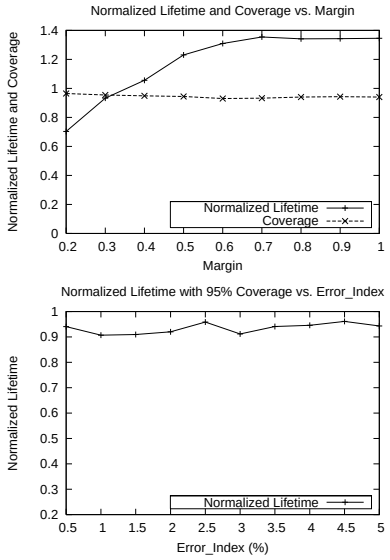

Fig. 11. The top plot shows the normalized lifetime and the fraction of the target field covered versus $Margin$ for $Error\_Index = 5\%$. The bottom plot shows the normalized lifetime versus $Error\_Index$ for $Margin$ corresponding to a coverage of approximately 0.95. For both plots, $n = 150$ and $B = 15$.

X. Conclusion

We designed a distributed, coordinate-free algorithm for attaining high lifetimes in sensor networks, subject to ensuring the k-coverage of the target field during the network lifetime. We proved that the lifetime attained by our algorithm approximates the maximum possible lifetime within a logarithmic approximation factor. Simulation results reveal that our algorithm substantially outperforms other schemes for lifetime maximization.

ACKNOWLEDGMENT

The contributions of G. Kasbekar and S. Sarkar have been supported by NSF grants 0621782, 0721308, 0914955, 0915203, 0915697.

REFERENCES

[1] D. Subhadrabandhu, F. Anjum, S. Kannan, S. Sarkar "Domination and Coverage Guarantees Through Distributed Computation" in *Proceedings of 43d Annual Allerton Conference on Communication, Control and Computing*, Allerton, Monticello, Illinois, September 28-30, 2005

[2] C-F. Huang and Y.-C Tseng, "The Coverage Problem in a Wireless Sensor Network", In Proc. of *ACM WSNA'03*, Sep. 2003.

[3] G. Kasbekar, Y. Bejerano and S. Sarkar, "Generic Coverage Verification without Location Information Using Dimension Reduction" in Proc. of *Wiopt'09*, Seoul, June 2009

[4] B. Awerbuch, Y. Azar, S. Plotkin "Throughput-Competitive On-Line Routing" in *Proceedings of IEEE Symposium on Foundations of Computer Science*, 1993.

[5] S. Leonardi "On-Line Network Routing" in *Online Algorithms- the State of the Art*, ed. A. Fiat and G. Woeginger, 1998.

[6] C. Zhang, Y. Zhang and Y. Fang "Detecting Coverage Boundary Nodes in Wireless Sensor Networks", In Proc. of *ICNSC '06*, April 2006.

[7] B. Alavi and K. Pahlavan "Modeling of the TOA-based Distance Measurement Error Using UWB Indoor Radio Measurements", In *IEEE Communications Letters*, Vol. 10, No. 4, April 2006, pp. 275-277.

[8] C. Y. Wen, R. D. Morris, and W. A. Sethares, "Distance Estimation Using Bidirectional Communications Without Synchronous Clocking", Accepted for publication in IEEE Trans. Signal Processing.

[9] Y. Bejerano, "Simple and Efficient k-Coverage Verification without Location Information". In Proc. of *Infocom'08*, Phoenix, Arizona, U.S.A., April 2008.

[10] P. Berman, G. Calinescu, C. Shah and A. Zelikovsky, "Efficient energy management in sensor networks". In Y. Xiao & Y. Pan (Eds.), *Ad hoc and sensor networks*, Nova Science, 2005.

[11] F. Zhao and L. Guibas, *"Wireless Sensor Networks: An Information Processing Approach"*. Morgan Kaufmann, 2004.

[12] M. Cardei, M.T. Thai, Y. Li, W. Wu "Energy-Efficient Target Coverage in Wireless Sensor Networks". In Proc. of *Infocom'05*, Miami, U.S.A., March 2005.

[13] M. Cardei and J. Wu. *"Coverage in Wireless Sensor Networks"*, Handbook of Sensor Networks. CRC Press 2004.

[14] L. Wang and Y. Xiao "A Survey of Energy-Efficient Scheduling Mechanisms in Sensor Networks" In *Mobile Networks and Applications,* Vol. 11, pp. 723-740, 2006.

[15] R. R. Choudhury and R. Kravets, "Location-Independent Coverage in Wireless Sensor Networks", Technical Report, UIUC, 2004

[16] R. Ghrist, A. Muhammad, "Coverage and hole-detection in sensor networks via homology". In Proc. of *IPSN 2005*, April 2005.

[17] A. Man-Cho So and Y. Ye. "On Solving Coverage Problems in a Wireless Sensor Network Using Voronoi Diagrams". In Proc. of *WINE 2005*, LNCS 3828, pp. 584-593, 2005.

[18] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks". In Proc. of *Infocom'01*, Anchorage, Alaska, U.S.A., April 2001.

[19] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks". *In International Journal of Wireless Ad Hoc and Sensor Networks*, vol. 1, num. 1-2, pp. 89-123, January 2005.

[20] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. "Integrated coverage and connectivity configuration in wireless sensor networks". In Proc. of *ACM SenSys'03*, Los Angeles, CA, Nov. 2003.

[21] Q. Zhao and M. Gurusamy, "Lifetime Maximization for Connected Target Coverage in Wireless Sensor Networks" In *IEEE/ACM Transactions on Networking,* Vol. 16, Issue 6, pp. 1378-1391, 2008.

[22] Y. Wu, S. Fahmy, N. Shroff, "On the Construction of a Maximum-Lifetime Data Gathering Tree in Sensor Networks: NP-Completeness and Approximation Algorithm" In Proc. of *Infocom'08*, Phoenix, Arizona, U.S.A., April 2008.

[23] D. Niculescu, "Positioning in ad hoc sensor networks", In *IEEE Network*, Volume 18, Issue 4, July-Aug. 2004, pp. 24-29

[24] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R.L. Moses and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks", In *IEEE Signal Processing Magazine*, Volume 22, Issue 4, July 2005 pp. 54-69

[25] Q. Shi, S. Kyperountas, N. S. Correal and F. Niu. "Performance Analysis of Relative Location Estimation for Multihop Wireless Sensor Networks". *IEEE Journal On Selected Areas In Communications (JSAC)*, Vol 23, No. 4, April 2005.

[26] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. "On-line load balancing with applications to machine scheduling and virtual circuit routing". In *Proc. ACM STOC*, pp. 623–631, 1993.

[27] W. Rudin, *"Principles of Mathematical Analysis"*, Mc-Graw Hill, Third Edition, 1976.

[28] A. Papoulis, S. Pillai, *"Probability, Random Variables and Stochastic Processes"*, Mc-Graw Hill Higher Education, Fourth Edition, 2002.

[29] V. Chvatal, "A Greedy Heuristic for the Set-Covering Problem", *Mathematics of Operations Research*, Vol. 4, No. 3 (Aug., 1979), pp. 233-235.

[30] X. Li, D. K. Hunter, K. Yang, "Distributed Coordinate-Free Hole Detection and Recovery", *IEEE GLOBECOM*, San Francisco, USA, 2006.

[31] N. Patwari, A. O. Hero, M. Perkins, N. S. Correal, R.J. O'Dea, "Relative Location Estimation in Wireless Sensor Networks", in *IEEE Trans. on Signal Processing*, Vol. 51, No. 8, pp. 2137–2148, Aug. 2003.

[32] C. Savarese, J. Rabaey, K. Langendoen "Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks", in *USENIX Technical Annual Conference*, Monterey, CA, June 2002.

[33] B. Sundararaman, U. Buy, A.D. Kshemkalyani, "Clock Synchronization for Wireless Sensor Networks: A Survey", *Ad Hoc Networks (Elsevier*, vol. 3, pp. 281–323, 2005.

[34] D. Culler, D. Estrin, M. Srivastava, "Overview of Sensor Networks", *IEEE Computer, Special Issue in Sensor Networks*, Aug. 2004.

[35] M. T. Thai, Y. Li, F. Wang, D.-Z. Du, "$O(\log n)$-Localized Algorithms on the Coverage Problem in Heterogeneous Sensor Networks", in *Proc. of IPCCC*, 2007.

Gaurav S. Kasbekar received B.Tech. in Electrical Engg. from Indian Institute of Technology, Bombay in 2004 and M.Tech. in Electronics Design and Technology (EDT) from Indian Institute of Science (IISc), Bangalore in 2006. He is currently a PhD student in the Electrical and Systems Engg. dept. at University of Pennsylvania. His research interests are in game theoretic and economic aspects of cognitive radio networks and algorithms for wireless sensor networks. He received the CEDT Design Medal for the best Masters student in EDT at IISc.

Yigal Bejerano received his B.Sc. in Computer Engineering in 1991 (summa cum laude), his M.Sc. in Computer Science in 1995, and his Ph.D. in Electrical Engineering in 2000, from the Technion - Israel Institute of Technology, Haifa, Israel. He is currently a member of the technical staff (MTS) at Bell Laboratories, Alcatel-Lucent. His research interests are mainly management aspects of high-speed and wireless networks. Dr. Bejerano has served as a technical program committee (TPC) member of numerous conferences of the networking research community.

Saswati Sarkar received the B.S. degree in Electronics and Telecommunications from Jadavpur University, Kolkata, India, in 1994; the M.S. degree in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1996; and the Ph.D. degree from the Electrical and Computer Engineering Department of the University of Maryland, College Park, in 2000. She is currently an Associate Professor with the Electrical and Systems Engineering Department, University of Pennsylvania, Philadelphia, which she joined as an Assistant Professor in 2000. Her research interests are in resource allocations, security and economics of wireless networks. Dr. Sarkar received the Motorola Gold Medal for being adjudged the best Masters student in the division of electrical sciences at the Indian Institute of Science and also received the NSF CAREER Award in 2003. She has served as an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2001 to 2006 and is currently in the Editorial Board of the IEEE/ACM TRANSACTIONS ON NETWORKING.