Optimal Sleep-Wakeup Algorithms for Barriers of Wireless Sensors

Santosh Kumar † Ten H. Lai ‡ Marc E Posner ‡ Prasun Sinha ‡
† University of Memphis ‡ The Ohio State University
santosh.kumar@memphis.edu {lai.1,posner.1,sinha.43}@osu.edu

***Abstract*— The problem of sleep wakeup has been extensively studied for the full coverage model, where every point in the deployment region is covered by some sensor. Since the sleep-wakeup problem is NP-Hard for this model, several heuristics exist. For the model of barrier coverage, however, where sensors are deployed to form an impenetrable barrier for detecting moving objects (a flagship application of wireless sensor networks), design of an optimal sleep-wakeup algorithm is open. In this paper, we solve this open problem by proposing optimal algorithms not only for the often-used case of equal lifetime but also for the much harder case when sensor lifetimes are different. We prove the optimality of both algorithms. Our algorithms can be used to maintain not just barrier coverage but fault tolerant connectivity, as well, while maximizing the network lifetime.**

We use simulation to show that for random deployments, even when a minimal number of sensors have been deployed, our optimal algorithms can increase the network lifetime by 500% (from 10 weeks to more than a year). Finally, we show that using our optimal algorithms increases the network lifetime six times longer than that achievable using an existing sleep wake-up algorithm called Randomized Independent Sleeping (RIS).

I. Introduction

Intrusion detection is an important application of wireless sensor networks [1], [2]. Although prototype demonstrations have so far been targeted at military deployments, as prices of motes continue to fall, it will soon become attractive to deploy sensors for intrusion detection in urban regions, as well (e.g., for securing private premises, corporate establishments, and government buildings). When wireless sensors are deployed as a barrier to detect moving objects of interest the sensor network is said to provide barrier coverage [3]. In such cases, the sensor network acts as a smart trip wire. In addition to intrusion detection applications, barriers of sensors can also be deployed around forests to detect the spread of fire or around chemical factories to detect leakage of harmful chemicals.

When sensors are deployed outdoors where access to wall power is unavailable or infeasible, ensuring long-term unattended operation becomes quite challenging. Although this is a major challenge for most applications, it is even more so for security applications that require monitoring at every instant. All applications where sensors are deployed as smart trip wires fall into this category. Loss of monitoring even for an instant could prove disastrous in some situations, such as when sensors have been deployed to detect the spread of a fire or the spread of lethal pollutants from a chemical plant.

A widely proposed technique to extend the lifetime of a sensor network deployed for continuous monitoring applications is to use sleep-wakeup. Under this technique, a sleeping schedule for sensors is computed such that at any given time only a subset of sensors are active. The remaining sensors are put to sleep. The challenge is to design a sleeping schedule that *maximizes* the network lifetime while maintaining the desired quality of monitoring. This problem is referred to as the *sleep-wakeup* problem.

For the barrier coverage model, a randomized sleep-wakeup algorithm called Randomized Independent Sleeping (RIS) is proposed in [3]. In this algorithm, time is divided into intervals and in each interval each sensor independently decides whether to sleep or stay active using a predetermined probability value p. The value of p is chosen such that the network is guaranteed to provide weak barrier coverage (a weaker version of barrier coverage) with high probability. An advantage of this algorithm is that it is purely local (i.e. requires no central coordination and no message exchange with any neighbors). However, there are several shortcomings. First, it does not provide deterministic guarantee of barrier coverage. Second, if the deployment does not follow random uniform or Poisson distribution, there is no guidance on how to choose a value for p. Third, if the lifetimes of the sensor nodes are not the same (as happens in real networks since different sensors have different loads and hence different lifetimes even on the same batteries), there is no guidance on how to choose a value of p. Last, but not least, there is no guarantee of performance; it is not even known how worse it performs as compared with an optimal schedule. It may be possible (as we show in simulations) that even when a network has sufficient resources to provide a lifetime of more than one year by using an optimal schedule, by using the RIS algorithm, the network will last just 10 weeks. When sensors are deployed in remote and unaccessible locations, such a loss in network lifetime can be problematic.

In this paper, we propose polynomial time algorithms to optimally solve the sleep-wakeup problem for the barrier coverage model. We show that when sensors are deployed randomly, the network lifetime can be increased by up to six times (from 10 weeks to 60 weeks). This is true even when the number of sensors deployed is the minimum necessary to provide barrier coverage with high probability in randomized deployments. The enhancement in lifetime comes by deterministically exploiting the inherent statistical redundancy.

We not only propose an optimal sleep-wakeup algorithm for the case when all sensors have equal lifetime (referred to as

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.

the *homogeneous lifetime* case) but also for the harder case when the sensors have distinct lifetimes (referred to as the *heterogeneous lifetime* case). Solving the heterogeneous case makes the sleep-wakeup method of extending network lifetime more practical. We list below some reasons why sensors may have different lifetimes:

- Uneven Load: Even when all sensors have the same batteries to begin with, they are subject to different kinds of loads (due to routing structure, cluster structure, etc.) and hence will have different lifetimes.
- Different Recharging Rates: If sensors are using rechargeable batteries, depending on the amount of light each sensor receives over the same period of time, the lifetime of each sensor may be different.
- Unanticipated Failures: When there are unanticipated sensor failures, a new schedule may need to be computed. At this time, the remaining lifetimes of operational sensors may be distinct.
- Additional Deployment: When new sensors are deployed in an existing network to compensate for the failed ones, sensors may have distinct lifetimes.

We show that if the energy imbalance in the network is evenly distributed, so that sensor lifetimes can be modeled as a random uniform distribution, then network lifetime is approximately the same as when the energy consumption is perfectly balanced in the network (so that all sensors have equal lifetime). According to our results, there is minimal or no loss in the network lifetime if the energy imbalance is sufficiently random.

We also show that our algorithms can be used to not only maintain barrier coverage at all times while maximizing the network lifetime but also to *maintain fault-tolerant connectivity with base station(s)*. Further, our algorithms are applicable to not only omnidirectional sensors but also to *directional sensors* such as cameras. This is because our algorithms rely only on a virtual graph structure where a virtual link exists between two sensors iff their sensing regions overlap.

Given a solution that maximizes network lifetime, a secondary criteria of interest is the minimization of the number of times that sensors are turned on/off, called *sensor switches*. Each time a sensor is turned on, neighbor discovery, route computation, time synchronization, and other such activities have to be performed. Minimizing the number of times such tasks are executed reduces the energy consumption in the network. It also makes the network more available to perform the monitoring task, which is the primary reason for deploying a sensor network.

Our sleep-wakeup algorithm for the homogeneous case minimizes the total number of sensor switches. However, for the heterogeneous lifetime case, we prove that minimizing the number of sensor switches is NP-Hard.

Although our algorithms (for the homogeneous and heterogeneous lifetime cases) are centralized algorithms, they have at least two advantages over a localized algorithm. First, since it is not possible to design a deterministic local sleep-wakeup algorithm because it can not be checked locally whether the network provides barrier coverage [3], only heuristic localized algorithms may be designed. Our optimal centralized algorithms can provide a significant improvement in network lifetime over such heuristics as we show is the case for the RIS algorithm. Second, since sleep-wakeup algorithms need to be executed only once or very rarely[1], the total number of messages exchanged to distribute the optimal schedule in the network (which can be done using the same utility as used in network reprogramming [4]) may be less than that used in a distributed heuristic algorithm that involves significant periodic message exchanges.

In summary, we make the following **key contributions** in this paper:

1) Propose the *Stint* algorithm to optimally solve the sleep-wakeup problem for barrier coverage when sensor lifetimes are same. The *Stint* algorithm also minimizes the number of sensor switches.
2) Propose the *Prahari* algorithm to optimally solve the sleep-wakeup problem for barrier coverage when sensor lifetimes are different.
3) Show that our algorithms can be used to maintain both barrier coverage and fault-tolerant connectivity while maximizing the network lifetime.
4) Use simulations to determine the extent of lifetime improvement achievable when sensors are deployed randomly but non-redundantly.

The rest of the paper is organized as follows. In Section II, we present some definitions, background, and related work. In Sections III and IV, we present our algorithms for the homogeneous and heterogeneous lifetime cases. In Section V, we describe how to use our algorithms to maintain fault-tolerant connectivity. In Section VI, we present the results of our simulations. We conclude the paper in Section VII.

II. Model, Definitions, and Background

In this section, we introduce some definitions and mention some background work including related work.

Definition 2.1: **Sensor Network,** N. A sensor network, N, is a collection of sensors with the locations of sensor deployments.

We assume that a sensor network is deployed over a belt region (see Figure 1). Intrusion is assumed to occur from top to bottom. As in [3], a path is a *crossing path* if it crosses from top to bottom. Further, a crossing path is k*-covered* if it intersects the sensing region of at least k distinct sensors. Finally, a sensor network N provides k*-barrier coverage* over a deployment region R if all crossing paths through region R are k-covered by sensors in N.

Definition 2.2: **Coverage Graph,** $\mathcal{G}(N)$ **[3]** A coverage graph of a sensor network N is constructed as follows. Let $\mathcal{G}(N) = (V, E)$. The set V consists of a vertex corresponding to each sensor. In addition, it has two virtual nodes, s and t to correspond to the left and right boundaries. An edge exists

[1]Once a schedule has been distributed to the sensors, there is no need for any further communication unless critical sensors fail.

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.

between two nodes if their sensing regions overlap in the deployment region R. An edge exists between u and s (or t) if the sensing region of u overlaps with the left boundary (or right boundary) of the region.
The coverage graph for the sensor network deployment in Figure 1 is shown in Figure 2.

Remark: Although we use a disk model for the sensing region, our results hold for all other models for which a coverage graph can be constructed. Further, the sensing range need not be the same for all sensors; all we require is that we be able to construct a coverage graph. Finally, the links of coverage graph are virtual links and not used for actual communication between sensor nodes.

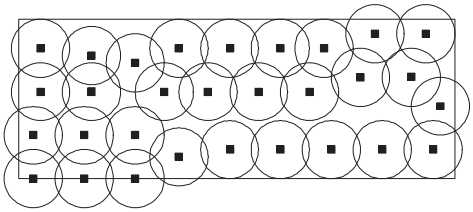

Fig. 1. A sensor network deployment that provides 3-barrier coverage.

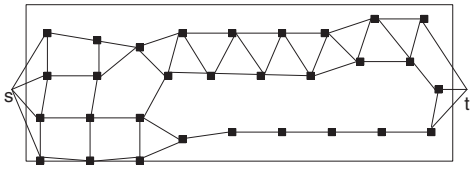

Fig. 2. The coverage graph of sensor network deployment of Figure 1.

Theorem 2.1: [3] A network N provides k-barrier coverage iff there exist k node-disjoint paths between the two virtual nodes s and t in $\mathcal{G}(N)$.

Definition 2.3: **Sensor Switch.** Any time a sensor is turned off and is turned on later, it is considered one *sensor switch.* If a sensor is turned on once and is allowed to exhaust its lifetime, then this sensor has no sensor switches.

Definition 2.4: **Path Switch.** Any time a group of sensors that together provide 1-barrier coverage is turned off and is later turned on as a group, it is considered a *path switch.* If this group of sensors exhausts its lifetime once it is turned on, then this group of sensors has no path switches.
Minimizing the number of path switches has the same effect as minimizing the number of sensor switches in terms of reducing the frequency of initialization operations such as neighbor discovery, route computation, and time synchronization.

Some Related Work: The problem of sleep wakeup for the full coverage model has been studied extensively. After the NP-Hardness was proved in [5] for this model, several heuristic algorithms appeared [6], [7], [8], [9]. No guarantee on performance is, however, made in any of these work.

The Randomized Independent Sleeping (RIS) algorithm of [8] was proposed to be used for computing a sleeping schedule for barrier coverage, as well in [3]. However, as argued in Section I, this algorithm neither guarantees barrier coverage nor provides any guarantee on performance.

The *Stint* and *Prahari* algorithms presented in this paper, on the other hand, guarantee optimal performance. Given the importance of energy-efficiency in ensuring long-term unattended operation of wireless sensor networks, optimal usage of available energy in the network is critical.

III. Homogeneous Lifetime

In this Section, we begin by deriving an upper bound on the network lifetime when the sensor lifetimes are homogeneous. Next, we present our *Stint* algorithm to determine an optimal sleeping schedule for individual sensors. Finally, we prove that the *Stint* algorithm minimizes the number of path switches in addition to maximizing the network lifetime.

A. Upper Bound on the Network Lifetime

Consider the sensor network shown in Figure 1. What is the maximum time for which this network can provide 2-barrier coverage, if all sensors have a lifetime of unity?

The next lemma enables us to compute the maximum achievable network lifetime. If the maximum number of node disjoint paths between s and t, m, is less than k, then the sensor network cannot provide k-barrier coverage even if all sensors are turned on. Therefore, the maximum lifetime of the network is 0. In the following, we only consider the case when $m \geq k$.

Lemma 3.1: Consider a sensor network N. Let $m \geq k$ be the maximum number of node-disjoint paths between the virtual nodes s and t in the coverage graph $\mathcal{G}(N)$. Also, let the lifetime of an individual sensor node be unity. Then, the maximum time for which the network N can provide k-barrier coverage is at most m/k.

Proof: By assumption, there exist m node-disjoint paths in the coverage graph of N. From Menger's Theorem [10], there exists a set of m nodes (corresponding to m sensors), which when removed disconnects virtual nodes s and t in the coverage graph. Call these m sensors *critical sensors.* Every path from s to t must contain at least one of these critical sensors.

From Theorem 2.1, in order to provide k-barrier coverage, a set of sensors must be activated such that they form k node-disjoint paths between the two virtual nodes s and t in the coverage graph. Each of these k paths must contain at least one of the m critical nodes. Further, since these k paths are node-disjoint, they can not share any node. Therefore, each set of k node-disjoint paths must contain at least k of the m critical nodes. Since at any time instant at least k of the m critical nodes need to be active, the maximum time that these m nodes can remain active is at most m/k. Once these m critical nodes run out of energy, the network can no longer provide k-barrier coverage. Hence, the network provides k-barrier coverage for at most m/k units of time. ■

Applying Lemma 3.1 to the sensor network shown in Figure 1, whose coverage graph appears in Figure 2, gives

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.

a maximum lifetime of 3/2. This is because the value of m, the maximum number of node disjoint paths between s and t, is 3 and $k = 2$..

B. Achieving the Upper Bound

Having derived an upper bound on the network lifetime that any sleep-wakeup algorithm can achieve for k-barrier coverage in the homogeneous lifetime case in Section III-A, we present the *Stint* algorithm that achieves this upper bound. The detailed *Stint* algorithm appears in Figure 4. We now provide an informal description of this algorithm.

The *Stint* algorithm first computes m, the maximum number of node disjoint paths between s and t. The maximum number of node disjoint paths can be computed using a max flow algorithm as discussed in [3]. The Stint algorithm then checks if m is divisible by k. If it is, then m disjoint paths are partitioned in ℓ groups of k paths each. These ℓ groups of k disjoint paths each are activated in sequence. The first group provides k-barrier coverage until it runs out of energy. The second group is activated next. The process continues in a similar fashion.

Conversely, if m is not divisible by k,, then ℓ is set to $\lfloor m/k \rfloor - 1$,. Now, ℓ groups of k disjoint paths each exhaust their lifetimes in sequence. Next, the remaining $r = m - \ell * k$ disjoint paths are arranged in $f = r/\gcd(r, k)$ sets of k disjoint paths each. Each of these f sets of paths is kept active for $\gcd(r, k)/k$ of the total lifetime of a sensor. In this way, the network provides k-barrier coverage for $\ell + r/k = m/k$ units of time, if each sensor has a lifetime of one unit. This is the maximum possible according to Lemma 3.1.

We use the coverage graph shown in Figure 3 to illustrate the *Stint* algorithm. For the coverage graph in Figure 3, the value of m is 8. If $k = 2$, then k divides m. Therefore, $\ell = 4$. The eight disjoint paths are partitioned into four sets of two paths each. These four sets are activated in sequence to provide a lifetime of four units.

Conversely, if $k = 3$, then m is not divisible by k. Now, ℓ is set to $\lfloor 8/3 \rfloor - 1 = 1$. Let this one group be the set of paths $(1, 2, 3)$. These three paths are kept active for their entire lifetime.

Next, the remaining $r = 8 - 1 * 3 = 5$ disjoint paths are arranged in $f = 5/\gcd(5, 3) = 5$ sets of 3 disjoint paths each. The five sets in this case will be $\{(4, 5, 6),\ (5, 6, 7),\ (6, 7, 8),\ (7, 8, 4),\ (8, 4, 5)\}$. Each of these five sets of paths is kept active for $\gcd(5, 3)/3 = (1/3)$ of the total lifetime of a sensor. In this way, the network provides 3-barrier coverage for $1+5/3 = 8/3$ units of time, if each sensor has a lifetime of one unit. This is the maximum possible according to Lemma 3.1.

Optimality and Complexity: The example considered above gives an evidence of why the *Stint* algorithm will always maximize the network lifetime. We omit the proof for brevity and refer the readers to [11]. The complexity of the *Stint* algorithm is dominated by the computation of maximum number of disjoint paths. The maximum disjoint paths can be computed with the max flow algorithm using the standard transformation of replacing each vertex with a set of *in* and *out* vertices, connecting all incoming arcs to the *in* vertex, connecting all outgoing arcs to the *out* vertex and connecting the *in* and *out* vertices with a directed arc of same capacity as the node's lifetime. The complexity of the Edmonds-Karp algorithm that can determine a max flow independent of the value of max flow and does not suffer from the count to infinity problem, is $O(VE^2)$ [12]. Some optimizations can improve the complexity to $O(V^3/\log(V))$.

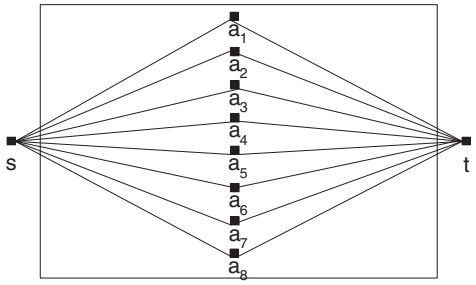

Fig. 3. The coverage graph of a sensor network used to illustrate the operation of the *Stint* algorithm.

C. Minimizing Path Switches

In this Section, we first illustrate why minimizing the number of path switches is non-trivial. We then derive a lower bound on the total number of path switches that *has to be* performed in a network if the network lifetime is to be maximized for k-barrier coverage. Finally, we prove that the *Stint* algorithm achieves this lower bound.

Consider again the network whose coverage graph appears in Figure 3. Let $k = 3$. Form 8 groups of 3 disjoint paths each, e.g., $\{(1, 2, 3),\ (4, 5, 6),\ (7, 8, 1),\ (2, 3, 4),\ (5, 6, 7),\ (8, 1, 2),\ (3, 4, 5),\ (6, 7, 8)\}$. Let each set of 3 disjoint paths be active for 1/3 units of time. This way we can achieve a network lifetime of 8/3, while providing 3-barrier coverage. Notice that the total number of path switches in this case is 16 since each path is turned off twice before it exhausts its full energy.

The total number of path switches in the schedule computed by the *Stint* algorithm is only 2. This is because, as described in the operation of the *Stint* algorithm only paths 4 and 5 are turned off once each before they run out of energy. All other paths run out of energy once they are turned on. Notice that achieving zero path switches is not possible in this case since 8 is not divisible by 3. The following lemma derives a lower bound on the total number of path switches. This is equivalent to the total number of preemptions in the domain of machine (or processor) scheduling.

Lemma 3.2: Consider a sensor network N. Let m be the maximum number of node-disjoint paths between the virtual nodes s and t in the coverage graph $\mathcal{G}(N)$. Also, let the lifetime of an individual sensor node be unity. If $k < m < 2k$, then the total number of path switches needed by any optimal sleep-wakeup algorithm for providing k-barrier coverage is at least $k - \gcd(m, k)$.

Proof: From Theorem 2.1, for the entire duration of m/k, there must be k node-disjoint paths active for the network to

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.

Input: A sensor network N deployed over an open belt region and the desired degree of coverage k. Assume that each sensor has the same lifetime, which is one unit of time.

Output: An optimal sleep-wakeup schedule for k-barrier coverage.

The *Stint* Algorithm

1: Compute the Coverage Graph $\mathcal{G}(N)$.
2: Compute the maximum number of node-disjoint paths between the two virtual nodes s and t in $\mathcal{G}(N)$. Denote the number of paths by m.
3: **if** $m > k$ **then**
4: Let $S_i, 1 \leq i \leq m$ be the sequence of sensors forming the i^{th} node-disjoint path.
5: **if** $m \mod k = 0$ **then**
6: $\ell \leftarrow m/k$
7: **else**
8: $\ell \leftarrow \lfloor m/k \rfloor - 1$
9: **end if**
10: **for** $j \leftarrow 0$ to $\ell - 1$ **do**
11: Activate all the sensors in sequence $S_{k*j+1}, \ldots, S_{k*(j+1)}$ for one unit of time.
12: **end for**
13: $r \leftarrow m - \ell * k$
14: **if** $r \neq 0$ **then**
15: $x \leftarrow \gcd(r, k)$
16: $f \leftarrow r/x$
17: From the sequence of sensors $S_{m-r+1}, \ldots, S_m$ form f sets of sequences, $X_0, X_1, \ldots, X_{(f-1)}$, such that set X_i consists of sequences $S_{m-r+1+x*i}$ through $S_{m-r+x*(i+1)}$.
18: **for** $j \leftarrow 0$ to $f - 1$ **do**
19: $g \leftarrow \ j + \frac{k}{x} - 1 \mod f$
20: Activate all the sensors in sets $X_j, \ldots, X_g$ for x/k units of time. Put all other sensors to sleep.
21: **end for**
22: **end if**
23: **else**
24: No schedule can achieve k-barrier coverage.
25: **end if**

Fig. 4. The *Stint* sleep-wakeup schedule assignment algorithm

provide k-barrier coverage. Out of the total m node disjoint paths, no path can be active for m/k units of time since $m/k > 1$.

Convert this problem to a machine scheduling problem [13]. The k disjoint paths that are required for k-barrier coverage correspond to k machines that process m jobs. Each job has a processing time of 1 unit on any machine. Let the k machines be numbered $1, 2, \ldots, k$. The objective of achieving a lifetime of m/k becomes equivalent to minimizing the makespan on k machines. The minimum value of the makespan is m/k, which is achieved only when all machines are busy for m/k units of time. Also, the number of path switches is the same as the number of job preemptions. Hence, the claim to be proved is that the minimum number of preemptions needed to achieve a makespan of m/k is at least $k - \gcd(m, k)$.

For any given optimal schedule, let a_1 be an arbitrary machine. At least one job is not finished on a_1. Let the set of unfinished jobs be J. Since $J \neq \phi$, pick an arbitrary unfinished job $j_1 \in J$. Let a_2 be the machine on which j_1 is resumed. We thus have one preemption (for job j_1). Add all the unfinished jobs from machine a_2 into set J. Again pick an unfinished job from J and let a_3 be the machine on which it is resumed. We have another preemption. We continue in this fashion until the set J of unfinished jobs becomes empty. This way we construct a sequence of machines $a_1, a_2, \ldots, a_{q_1}$ such that each machine $a_i, i \neq 1$ has at least one preemption. Also, these q_1 machines together finish some number of jobs completely within m/k time units with at least $(q_1 - 1)$ preemptions. This implies that $q_1 * m/k$ is an integer, which is possible only when q_1 is a multiple of $k' = k/\gcd(m, k)$.

If $q_1 \neq k$, we select a new machine and construct a new sequence of q_2 machines, which together finish some number of jobs with at least $(q_2 - 1)$ preemptions, where q_2 is a multiple of k'. Let there be σ such q_i's such that $\sum_{i=1}^{\sigma} q_i = k$, where $\sigma \leq \gcd(m, k)$. The total number of preemptions is at least $\sum_{i=1}^{\sigma} (q_i - 1) = k - \sigma \geq k - \gcd(m, k)$. Since this holds for any optimal schedule, the claim is proved. ■

The next theorem establishes that the *Stint* algorithm achieves the lower bound of Lemma 3.2.

Theorem 3.1: Of all the optimal sleep-wakeup algorithms for achieving k-barrier coverage, the *Stint* algorithm needs the minimum number of path switches.

Proof: Observe that no path switches are needed up to Line 13 in Figure 4, which is the minimum possible. Path switches are required only when $k < r < 2k$. Therefore, we only need to prove that for this case, the *Stint* algorithm involves the minimum number of path switches.

Lemma 3.2 establishes that any optimal sleep-wakeup algorithm for achieving k-barrier coverage requires at least $k - \gcd(m, k)$ path switches, where $k < m < 2k$. To prove the theorem, we only need to show that the *Stint* algorithm involves a maximum of $k - \gcd(m, k)$ path switches.

Notice that any path switch is performed only in the *for loop* in Line 18 through Line 21. Consequently, we focus on these lines only. Each time a sensor is turned off, every sensor in its group is turned off. Since a group consists of x sequence of sensors, each of which provides 1-barrier coverage, every on/off involves $x$path switches.

A set $S_i, 0 \geq i \geq f - 1$ of sensors is turned off before it exhausts its lifetime only when the index of the loop $j < k/x - 1$. This is because every group S_i runs out of energy if it is active continuously for k/x iterations. Except for the first $k/x - 1$ sets $S_i, 0 \leq i < k/x - 1$, each of which is turned off once it is continuously active for $i + 1$ iterations, every other set of sensors $S_i, k/x - 1 \geq i \geq f - 1$ is active continuously for k/x iterations. Further, the first $k/x - 1$ sets which are turned off before running out of energy, are not turned off again when they are turned on later. Since each of these sets involves x path switches and they are turned off and on exactly once, the *Stint* algorithm needs exactly $x * (k/x - 1) = k - x$ path switches, where $x = \gcd(m, k)$. ■

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.

IV. Heterogeneous Lifetime

In this section, we begin by deriving an upper bound on the network lifetime when the sensor lifetimes are heterogeneous. Next, we present the *Prahari*[2] algorithm to determine an optimal sleeping schedule for individual sensors. Finally, we consider the problem of minimizing the number of path switches.

A. Upper Bound on Network Lifetime

The maximum lifetime can be determined using Lemma 3.1 when the sensor lifetimes are identical. When the sensor lifetimes are not identical, the problem of determining the maximum achievable lifetime becomes significantly more challenging. For example, consider the network in Figure 5. This is the same network as Figure 1, except that sensors have distinct lifetimes. What is the maximum time for which this network can provide 2-barrier coverage? This problem appears to be NP-Hard. However, it is polynomially solvable using multiroute network flows [14].

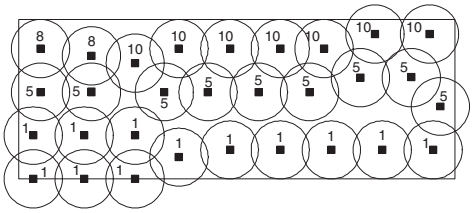

Fig. 5. The sensor network deployment of Figure 1, with heterogeneous sensor lifetimes. The integers next to the filled squares denote the lifetime of the sensors located there. What is the maximum time for which this network can provide 2-barrier coverage?

We begin with some assumptions and definitions. We assume that it is possible to estimate the remaining lifetime of a sensor node. With new mote hardware, it is possible to measure the remaining battery level [15] and based on the load observed so far, the remaining lifetime can be estimated. Also, a profile of expected energy consumption of every node may be built using analytical models or using simulators such as PowerTOSSIM [16]. Sensor lifetimes do not have to be integral; they can assume any real value.

Definition 4.1: **Coverage Graph with Lifetime,** $\mathcal{G}_L(N)$. A coverage graph with lifetime of a sensor network N, denoted by $\mathcal{G}_L(N)$, is a coverage graph where all nodes $u \in V - \{s,t\}$ are assigned a capacity, $c(u)$, equal to their remaining lifetimes. All the edges are assigned infinite capacities. The vertex s is the source and t the sink.

The $\mathcal{G}_L(N)$, corresponding to the network shown in Figure 5 appears in Figure 6. To convert a $\mathcal{G}_L(N) = (V, E)$ to a directed graph, we replace all the edges $\{u, v\}$ with a pair of directed edges (u, v) and (v, u). For the rest of Section IV, we assume that $\mathcal{G}_L(N)$ is a directed graph.

Definition 4.2: s-t **Flow.** An s-t *flow* in $\mathcal{G}_L(N)$ is defined as $f : E \rightarrow \mathbb{R}^+$ such that

[2]The word "Prahari" is a Sanskrit word for securityman who guards a region for a fixed time interval.

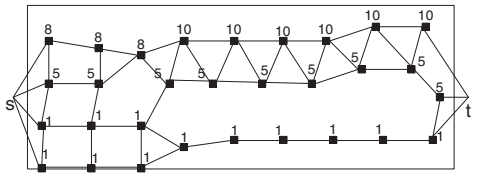

Fig. 6. The coverage graph with lifetime $\mathcal{G}_L(N)$ of the sensor network N shown in Figure 5. The integers next to the filled squares denote the lifetime of the sensors located there.

1) $\forall u \in V - \{s,t\}, \sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$,
2) $\sum_{(s,v)\in E} f(s,v) = \sum_{(v,t)\in E} f(v,t)$, and
3) $\forall u \in V - \{s,t\}, \sum_{(u,v)\in E} f(u,v) \leq c(u)$.

Definition 4.3: s-t **Path Flow.** A s-t *path flow* in $\mathcal{G}_L(N)$ is a s-t flow with the property that the flow network is a single path from s to t.

Three path flows (Path Flow 1, Path Flow 2, and Path Flow 3) from the coverage graph shown in Figure 6 are shown in Figure 7.

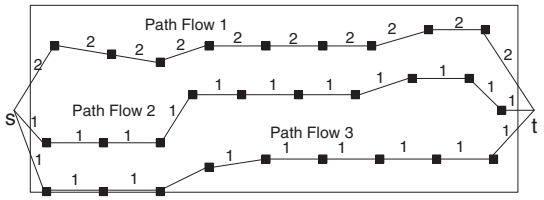

Fig. 7. A composite 2-flow of total value 4 for the sensor network N shown in Figure 5. The edges are labeled with the flow value passing through them. This composite 2-flow is of the maximum value possible for N.

Definition 4.4: **Basic k-Flow of Value a.** A *basic k-flow of value a* in $\mathcal{G}_L(N)$ is a set of k node-disjoint s-t path flows, each of which has a value of a. The total value of the flow is $k * a$.

In Figure 7, Path Flow 1 and Path Flow 2 comprise a basic 2-flow of value 1. The total value of this basic 2-flow is 2.

Definition 4.5: **Composite k-Flow.** A set of flows in $\mathcal{G}_L(N)$ is called a *composite k-flow* if it can be expressed as a sum of basic k-flows. The total value of this composite k-flow is $\sum_{i=1}^{m} \lambda_i * k * a_i$, for $\lambda_i \in \mathbb{Z}^+$, if m basic k-flows each with a value of a_i make up this composite k-flow.

A composite 2-flow of total value 4 from the coverage graph shown in Figure 6 appears in Figure 7. Notice that Path Flow 1, which has value 2, can be decomposed in two path flows, each with value 1.

We now state the key result of this Section, which makes deriving an upper bound on the network lifetime tractable when the sensor lifetimes are heterogeneous. The basic idea of the *Prahari* algorithm comes from the proof of the following theorem.

Theorem 4.1: Given a sensor network N, there exists a sleep-wakeup schedule that achieves a lifetime of T time units for k-barrier coverage iff there is a composite k-flow of value $k * T$ in $\mathcal{G}_L(N)$.

Proof: $\Rightarrow$. Given a composite k-flow of T units in $\mathcal{G}_L(N)$, we construct a sleep-wakeup schedule to achieve a

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.

lifetime of T time units. Let F be a composite k-flow of value T. By definition, F can be decomposed into a set of m basic k-flows (for some $m > 0$) such that $\sum_{i=1}^{m} \lambda_i * k * a_i = k * T$ for $\lambda_i \in \mathbb{Z}^+$, where a_i is the value of i^{th} basic k-flow. In every basic k-flow i, there are k node-disjoint flows each with value a_i (by the definition of basic k-flow). Consider the m basic k-flows in order. Turn on the nodes in the basic k-flow i at $\sum_{j=1}^{i-1} \lambda_j * k * a_j$ time units from the start of sleep-wakeup schedule and keep them continuously active for a duration of $\lambda_i * a_i$ time units. With this schedule, the network N provides k-barrier coverage for T time units since each basic k-flow i provides k-barrier coverage for $\lambda_i * a_i$ units of time and $\sum_{i=1}^{m} \lambda_i * a_i = T$.

$\Leftarrow$. Given a sleep-wakeup schedule that allows N to provide k-barrier coverage for T units of time, we construct a k-flow of value $k * T$ in $\mathcal{G}_L(N)$. Let t_1 be the first time instant when some sensor changes its state from off to on or vice versa. The set of sensors that are on in the interval $[0, t_1]$ form a basic k-flow of value t_1 in $\mathcal{G}_L(N)$ since by Theorem 2.1 there exist k node-disjoint paths between s and t in $\mathcal{G}_L(N)$ with these sensors active. Denote this basic k-flow by F_1. The total value of F_i is $k * t_1$. Similarly, define F_i. Let there be m such time instants when the sensors change state. Since N provides k-barrier coverage for T units of time, $\sum_{i=1}^{m} k * t_i = k * T$. Hence, the set of basic k-flows together define a composite k-flow of value $k * T$. ■

Corollary 4.1: The maximum time for which the sensor network N can provide k-barrier coverage is $\hat{f}/k$, where $\hat{f}$ is the maximum value of composite k-flow in $\mathcal{G}_L(N)$.

Proof: The proof follows from Theorem 4.1. ■

If we can devise a method to determine the maximum value of a composite k-flow in a $\mathcal{G}_L(N)$, we can derive an upper bound on the network lifetime achievable by N. For this purpose, we make use of the MEM algorithm from [14]. Applying this algorithm, we determine that the maximum value of composite 2-flow, $\hat{f}/k$, for the coverage graph shown in Figure 6 is 4. Hence, the maximum time for which this network can provide 2-barrier coverage is 4/2=2 (from Corollary 4.1).

B. Achieving the Upper Bound

Having derived an upper bound on the network lifetime that any sleep-wakeup algorithm can achieve for k-barrier coverage in the heterogeneous lifetime case in Section IV-A, we present our *Prahari* algorithm that achieves this upper bound.

The detailed *Prahari* algorithm appears in Figure 9. We now provide an informal description of this algorithm.

The *Prahari* algorithm first invokes the *MEM* algorithm from [14] to determine $\hat{f}$, the maximum value of composite k-flow in $\mathcal{G}_L(N)$. Let $F_{MEM}(N)$ be the flow network resulting from this step.

If the flow network $F_{MEM}(N)$ is such that the indegree and outdegree of every node other than s and t is 1, then the flow network can be decomposed in m number of node-disjoint path flows for some $m > k$. In this case, the *Prahari* algorithm uses a machine scheduling algorithm proposed in [17] to schedule the m paths to achieve a lifetime of $\hat{f}/k$ time units, as is done in the scheduling of m jobs on k machines to achieve a makespan of $\hat{f}/k$. Thus, we achieve the maximum lifetime in this case.

Conversely, if the flow network $F_{MEM}(N)$ is such that some node in $V - \{s, t\}$ has an indegree or outdegree of more than 2, then the *Prahari* algorithm invokes the *SEM* algorithm from [14] to decompose the flow network into α' number of basic k-flows for some $\alpha' > k$. It then merges identical basic k-flows into a single aggregate basic k-flow. Let α be the number of distinct basic k-flows resulting from the preceding step. Since the set of nodes in each basic k flow provides k-barrier coverage, the *Prahari* algorithm schedules these α basic k-flows one by one. Since the sum of total flow values of all basic k-flows is precisely $\hat{f}$, the maximum network lifetime of $\hat{f}/k$ is achieved.

We use the coverage graph shown in Figure 6 to illustrate the operation of the *Prahari* algorithm. Let $k = 2$. Figure 6 shows $F_{MEM}(N)$ for the network N shown in Figure 5. As can be seen from this figure, the value of $\hat{f}$ is 4. Since the indegree and outdegree of every node other than s and t is 1 in the flow network $F_{MEM}(N)$, the flow network is decomposed in $m = 3$ node-disjoint path flows. Since $k = 2$, two machines will be used for scheduling. Also, the minimum makespan, which is equivalent to the maximum network lifetime, for the jobs is 4/2=2. As shown in Figure 8, Path Flow 1 is scheduled on Machine 1 for 2 time units, Path Flow 2 and 3 are scheduled on Machine 2 for 1 time unit each. This way we have a schedule for the three paths. Path Flow 1 will be active for 2 time units continuously. Path Flow 2 will be active for 1 time unit starting at time $t = 0$. At $t = 1$, Path Flow 2 will run out of energy and Path Flow 3 will be activated. Thus, we achieve a lifetime of 2 time units, which is the maximum possible. We now formally prove the optimality of the *Prahari* algorithm.

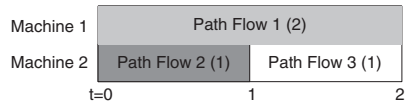

Fig. 8. The machine scheduling approach followed by the *Prahari* algorithm is illustrated for the flow network shown in Figure 7. The numbers in the parantheses denote the lifetime of the individual paths.

Theorem 4.2: The *Prahari* algorithm is an optimal sleep-wakeup algorithm for providing k-barrier coverage.

Proof: Consider a sensor network N. Let $\mathcal{G}_L(N) = (V, E)$ be its coverage graph with lifetime. Let $f_k(N)$ denote the maximum value of its composite k-flow. Corollary 4.1 established that the maximum lifetime of N for providing k-barrier coverage is $f_k(N)/k$. Therefore, to prove the optimality of the *Prahari* algorithm, we only need to prove that the algorithm allows the network N to provide k-barrier coverage for $f_k(N)/k$ units of time.

[14] shows that the *MEM* algorithm computes the value of $f_k(N)$. If the flows in the network resulting from applying the *MEM* algorithm are node-disjoint (besides s and t), then

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.

Input: A coverage graph $\mathcal{G}_L(N) = (V, E)$ for a sensor network N, and $k \in \mathbb{Z}^+$. The capacity of a node u is denoted by $c(u)$.

Output: A sequence $t_w^{(i)}(v), t_s^{(i)}(v)$, the wakeup time and sleep time for each node $v \in V$.

The *Prahari* Algorithm

1: Invoke the *MEM* algorithm on $\mathcal{G}_L(N)$. Let $\hat{f}$ be the value of maximum composite k-flow. Each vertex $v \in V - \{s, t\}$ is assigned a flow, $f(v)$.
2: Delete all vertices and associated edges from $\mathcal{G}_L(N)$ with $f(v) = 0$.
3: **if** $\forall v \in V - \{s, t\}$, the indegree and outdegree of v is 1 **then**
4: Decompose $\mathcal{G}_L(N)$ into m disjoint path flows.
5: Sort these path flows in descending order of flow value. Let the sequence of flows be $F_1, F_2, \ldots, F_m$ with flow values $f_1, f_2, \ldots, f_m$.
6: $t \leftarrow 0$.
7: **for** $i \leftarrow 1$ to m **do**
8: **if** $t + f_i \leq \hat{f}/k$ **then**
9: $\forall v \in F_i$, set $t_w^{(1)}(v) \leftarrow t$ and $t_s^{(1)}(v) \leftarrow t + f_i$.
10: **else**
11: $\forall v \in F_i$, set $t_w^{(1)}(v) \leftarrow 0$, $t_s^{(1)}(v) \leftarrow t + f_i - \hat{f}/k$, $t_w^{(2)}(v) \leftarrow t$, and $t_s^{(2)}(v) \leftarrow \hat{f}/k$.
12: **end if**
13: $t \leftarrow t_s^{(1)}(v)$.
14: **end for**
15: **else**
16: Invoke the *SEM* algorithm to decompose the k-flow into α' basic k-flows. Denote them by $N_1, N_2, \ldots, N_{\alpha'}$.
17: Merge all the basic k-flows that have the same set of vertices with positive flow into a single basic k-flow. Let the distinct number of basic k-flows be α.
18: $t \leftarrow 0$.
19: **for** $i \leftarrow 1$ to α **do**
20: $\forall v \in V - \{s, t\}$ such that $f(v) > 0$ in N_i, $t_w^{(i)}(v) \leftarrow t$ and $t_s^{(i)}(v) \leftarrow f(N_i)/k$.
21: $t \leftarrow f(N_i)/k$
22: **end for**
23: **end if**

Fig. 9. The *Prahari* algorithm to determine sleep-wakeup schedule for optimizing the network lifetime.

Lines 3 through 14 are executed. We establish that this schedule achieves a lifetime of $f_k(N)/k$.

Since $\sum_{i=1}^{m} f_i = f_k(N)$, at every time instant in the duration $[0, f_k(N)/k]$, k node-disjoint paths are active. Each of these paths provides 1-barrier coverage. Further, as the value of any individual flow (out of m flows) is at most $f_k(N)/k$, there is no schedule conflict for any node, i.e. no node is assigned to provide more than 1-barrier coverage at any time instant.

If the flows are not node-disjoint, the *SEM* algorithm is invoked to decompose the k-flow computed by the *MEM* algorithm in component basic k-flows. For a proof of the correctness of the *SEM* algorithm, we refer the reader to [18]. Since each basic flow N_i provides k-barrier coverage for $f(N_i)/k$ time units and the sum of basic k-flows, $\sum_{i=1}^{\alpha} f(N_i) = f_k(N)$, the network N provides k-barrier coverage for $f_k(N)/k$ time units. ■

Complexity: The complexity of the *Prahari* algorithm is dominated by the *SEM* [14] algorithm, whose complexity is $O(kV^3/\log(V))$

C. Minimizing Path/Sensor Switches

For some special cases, the *Prahari* algorithm minimizes the number of path switches. However, in general, the problem of sensor switch minimization is NP-Hard. We prove the decision version of this problem to be *strongly* NP-Complete by reducing the 3-Partition [19] problem to it. The problem continues to be *NP-Hard* even if the coverage graph of the given network has only node disjoint paths between s and t, and the objective is to minimize the number of path switches. For details and for the proofs of NP-Completeness, we refer the reader to [11].

V. Maintaining Coverage and Connectivity

In this section, we briefly discuss how our algorithms can be used to maximize the network lifetime not only for maintaining k-barrier coverage but also for maintaining k node-disjoint paths.

We first observe that when sensors are deployed for barrier coverage, the sensor network does not need to have every sensor connected to each other. It is sufficient if all sensors that participate in providing barrier coverage can communicate with base station(s) via multi-hop routes. Without loss of generality, we assume that base station(s) are located at the two extreme ends of the network, and can directly reach all sensors located on the respective extreme ends. Therefore, if the sensors providing barrier coverage form a path in the *Communication Graph*[3] between the two extreme ends of the network, then all detection events will be communicated to the base station(s).

Now, if the communication range is twice the sensing range, then k-barrier coverage implies that all sensors that form k-disjoint paths between the two virtual nodes s and t in the coverage graph, also form k-node disjoint paths in the communication graph between the two extreme ends of the network. If, on the other hand, the communication range is less than twice the sensing range, then our algorithms can be applied to the communication graph (instead of the coverage graph) to find k-node disjoint paths across the two extreme ends of the network. Each of these disjoint paths provide 1-barrier coverage, implying that the network provides k-barrier coverage, as well. In both cases, our algorithms can be used to provide both barrier coverage and fault-tolerant connectivity with base station(s) while maximizing the network lifetime.

VI. Simulations

In this section, we present results of our simulations (carried out in Matlab). We consider a deployment scenario where a

[3]Two sensors are neighbors in the *Communication Graph* if they can communicate with each other directly.

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.

rectangular belt region of dimension $2\text{km} \times 100\text{m}$ is to be barrier-covered by sensors, each of which has a sensing range of 30m. For the homogeneous case, we assign a lifetime of 10 weeks for each sensor. For the heterogeneous case, we assign a lifetime of between 5 and 15 weeks (chosen randomly and uniformly) to each sensor. Notice that the average lifetime is still 10 weeks in the latter case. Our goal in this section is to investigate four key issues.

1) If sensors are deployed randomly but not redundantly (i.e., deploying these many sensors is needed to achieve k-barrier coverage with high probability), then what level of lifetime increase can we expect in practice?

By using (44) in [20], we determine that 465 sensors are needed to achieve 1-barrier coverage with high probability [4]. The probabilistic conditions such as (44) in [20] are designed to protect against the rare worst cases when sensor locations may be such that the network may have no redundancy. However, in most instances of deployment, there may exist redundancy if we examine each instance deterministically. By executing the *Stint* algorithm on 10 instances[5] of random deployment of 465 sensors, where in each instance different locations for each node is selected (using random uniform distribution), we observe that in some instances, the network lifetime may be increased by 6 times, from 10 weeks to 60 weeks (see Figure 10).

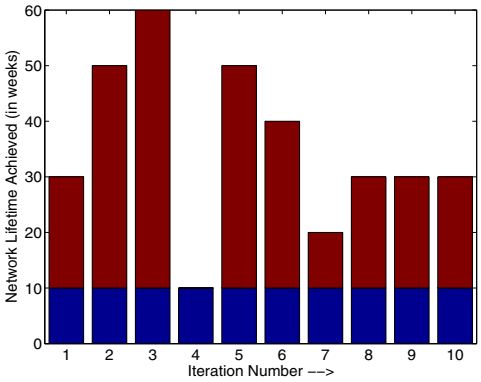

Fig. 10. Sensors are deployed randomly but non-redundantly to provide 1-barrier coverage with high probability. The network is planned to last for 10 weeks. Lifetime increase by using the *Stint* algorithm is shown for 10 iterations of deployment.

2) If sensors are deployed randomly but not redundantly, then what level of lifetime increase can we expect in practice as the value of k is increased?

We again use (44) in [20] to determine the number of sensors needed to achieve k-barrier coverage for $k = 1, 2, 3, 4$, which are 465, 529, 596, and 663 respectively. As evident from these numbers, statistical redundancy in the network decreases as the value of k increases (e.g., 465 sensors are required to provide 1-barrier coverage, but only 64 additional sensors are needed to provide 2-barrier coverage, and so on). Consequently, the enhancement in network lifetime by using an optimal sleep-wakeup algorithm decreases as the value of k increases. In Figure 11, we show the median network lifetime[6] achieved by the optimal algorithms. We see that, on average, the network can be made to last 30-40 weeks when $k = 1$, 20-22 weeks when $k = 2$, 18 weeks when $k = 3$ and 16-17 weeks when $k = 4$.

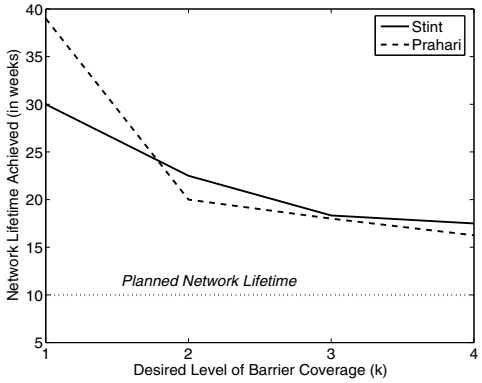

Fig. 11. Sensors are deployed randomly but non-redundantly to provide k-barrier coverage with high probability for values of k from 1 to 4. For the *Stint* and *Prahari* algorithms, median lifetimes (over 10 iterations) are plotted.

3) How is the optimal network lifetime affected if the lifetimes of nodes are not all same?

By comparing the lifetimes achieved by the *Stint* and *Prahari* algorithms in Figure 11, we conclude that the median network lifetime enhancement achieved in both cases are comparable. The key observation is that uneven distribution of sensor lifetimes is not always harmful, provided their distribution is uniform and provided that a heterogeneous sleep-wakeup algorithm such as *Prahari* is used to compute the sleeping schedule.

4) How much increase in network lifetime is possible using an optimal sleep-wakeup algorithm as compared to that achievable using a randomized algorithms such as RIS?

In Figure 12, we observe that the RIS algorithm provides a network lifetime of close to 20 and 30 weeks when the number of nodes deployed are 930 and 1395 respectively, as expected from analysis in [20]. We provide two trajectories of lifetime for RIS. In the basic case, the network lifetime is defined as the number of weeks that the network provides 1-barrier coverage continuously. In the $RIS_{Extended}$ case, we relax the requirement of continuity in counting the total number of weeks that the network provides 1-barrier coverage for.

The lifetime enhancement provided by the *Stint* algorithm is between 5-6 times better than that provided by using RIS. We can predict analytically that the lifetime enhancement provided by the optimal sleep-wakeup algorithms continues to be similar as the number of sensors is increased. The extension in the network lifetime using RIS requires a linear increase in

[4]Although using (44) in [20] only guarantees a weaker version of barrier coverage, it is, nevertheless, a necessary condition for achieving barrier coverage with high probability.

[5]Computation of node-disjoint paths is a compute intensive procedure and therefore we limit our simulations to 10 instances.

[6]We use median instead of mean to reduce the effect of extreme cases.

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.

the number of nodes [20]. Extension in the network lifetime using the optimal algorithms requires an increase in the value of k of k-barrier coverage. The number of sensors needed to achieve a desired value of k in k-barrier coverage can be derived by using (44) in [20]. For 1-barrier coverage, we plot the lifetime enhancement offered by the *Stint* algorithm versus that offered by the RIS algorithm in Figure 13, which matches the behavior observed in Figure 12.

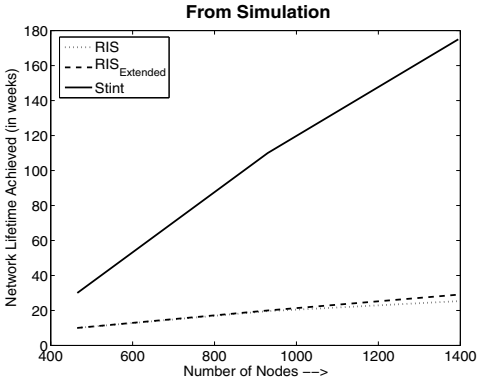

Fig. 12. Sensors are deployed randomly to provide 1-barrier coverage with high probability. Median lifetimes achieved are shown for all sleep-wakeup algorithms.

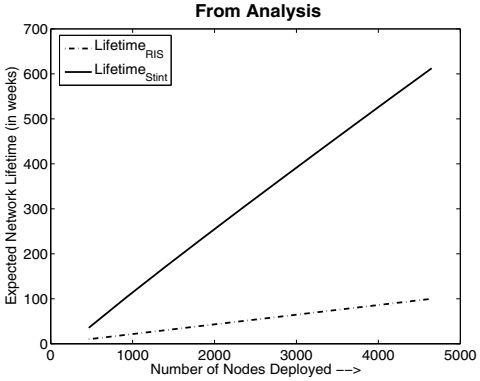

Fig. 13. Analytical prediction of how the network lifetime achieved using the *Stint* algorithm will compare against that achieved using the RIS algorithm.

VII. Conclusions

In this paper, we propose optimal solutions to the sleep-wakeup problems for the model of barrier coverage for both the homogeneous and heterogeneous lifetime cases. We show that using these algorithms can enable the network to last upto six times longer even if a minimal number of sensors have been deployed in a random deployment. We also show that the network lifetime achieved in both homogeneous and heterogeneous cases are comparable if the unevenness in individual sensor lifetimes is distributed uniformly in the network. Finally, we show that using our optimal algorithms increases the network lifetime six times longer than that achievable by using a previously known randomized algorithm.

Prior to our work, the problem of sleep-wakeup was usually considered to be NP-Hard. Now that the sleep-wakeup problem has been solved optimally for the barrier coverage model, new research is expected to investigate the tractability of this problem for other coverage models. It will also be interesting to design smart distributed heuristic algorithms for sleep-wakeup in barrier coverage that perform close to the optimal most of the time.

References

[1] A. Arora and et. al., "Line in the sand: A wireless sensor network for target detection, classification, and tracking," *Computer Networks*, vol. 46, no. 5, pp. 605–634, 2004.

[2] A. Arora and et.al., "Exscal: Elements of an extreme scale wireless sensor network," in *Eleventh IEEE International Conference on Real-Time Computing Systems and Applications (IEEE RTCSA)*, Hong Kong, 2005.

[3] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," in *International Conference on Mobile Computing and Networking (ACM MobiCom)*, Cologne, Germany, 2005, pp. 284–298.

[4] Gilman Tolle and D. E. Culler, "Design of an application-cooperative management system for wireless sensor networks," in *EWSN*, Istanbul, Turkey, 2005.

[5] S. slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *IEEE International Conference on Communications*, Helsinki, Finland, 2001, vol. 2, pp. 472–476.

[6] M. Cardei, M. Thai, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *IEEE INFOCOM*, Miami, FL, 2005.

[7] T. He and et al, "Energy-efficient surveillance system using wireless sensor networks," in *International Conference on Mobile Systems, Applications, and Services (ACM Mobisys)*, Boston, MA, 2004, pp. 270–283.

[8] S. Kumar, T. H. Lai, and J. Balogh, "On k-coverage in a mostly sleeping sensor network," in *International Conference on Mobile Computing and Networking (ACM MobiCom)*, Philadelphia, PA, 2004, pp. 144–158.

[9] H. Zhang and J. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," in *NSF International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wirelsss, and Peer-to-Peer Networks*, 2004.

[10] Douglas B. West, *Introduction to Graph Theory*, Prentice Hall, 2001.

[11] S. Kumar, T. H. Lai, M. E. Posner, and P. Sinha, "Optimal sleep wakeup algorothms for barrier of wireless sensors," Tech. Rep., OSU-CISRC-8/06-TR69, The Ohio State University, Available at: ftp://ftp.cse.ohio-state.edu/pub/tech-report/2006/TR69.pdf, 2006.

[12] Alexander Schrijver, *Combinatorial Optimization*, Springer, 2003.

[13] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, 1995.

[14] Wataru Kishimoto, "A method for obtaining the maxmimum multiroute flows in a network," *Networks*, vol. 27, no. 4, pp. 279–291, 1996.

[15] Ben Kuris and Terry Dishongh, "Shimmer mote:hardware guide," *Online at http://www.eecs.harvard.edu/~konrad/projects/shimmer/ references/SHIMMER_HWGuide_REV1P3.pdf*, 2006.

[16] V. Shnayder, M. Hempstead, B. Chen, B. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD, 2004.

[17] R. McNaughton, "Scheduling with deadlines and loss functions," *Management Science*, vol. 6, pp. 1–12, 1959.

[18] Wataru Kishimoto and M. Takeuchi, "On m route flows in a network," *IEICE Transactions (in Japanese)*, vol. J-76-A(8), pp. 1185–1200, 1993.

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

[20] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," *Wireless Networks (WINET)*, to appear (availavle online at http://www.cs.memphis.edu/~santosh/Papers/ BarrierCoverage_WINET.pdf).

Authorized licensed use limited to: MONTANA STATE UNIV BOZEMAN. Downloaded on March 26, 2009 at 11:50 from IEEE Xplore. Restrictions apply.