Optimal Sleep-Wakeup Algorithms for Barriers of Wireless Sensors

Santosh Kumar, Ten H. Lai, Marc E Posner, and Prasun Sinha
The Ohio State University
Email: {kumar.74,lai.1,posner.1,sinha.43}@osu.edu

Abstract— **The problem of sleep-wakeup is to determine a sleeping schedule for sensors such that the lifetime of the network is maximized while maintaining the desired quality of monitoring. This problem has received considerable attention in the literature. However, given the NP-Hardness results even for the homogeneous lifetime case, it is now widely accepted that the sleep-wakeup problem does not have a polynomial-time solution for wireless sensor networks, in general. Therefore, in practice, when a wireless sensor network is prototyped or deployed, either heuristic algorithms are used, or sleep-wakeup is not used at all. Both of these choices lead to a wastage of valuable energy, which in turn, requires over-deployment of sensors and/or causes degradation in the quality of monitoring.**

In this paper, for the first time, we propose polynomial-time algorithms and prove its optimality to solve the sleep-wakeup problem for a specific class of applications, where a wireless sensor network is deployed as a smart barrier or a tripwire for detecting moving objects as in intrusion detection. We not only optimally solve the sleep-wakeup problem for the homogeneous lifetime case but also for the more practical case of heterogeneous lifetimes by using the concept of multiroute network flows. Since intrusion detection is an important application of wireless sensor networks (as it has alreday been successfully demonstrated on large scale sensor networks and is now in production for real life deployments), we believe our work will have a significant practical impact. Finally, our work can be applied to other important applications as well, such as fire detection.

I. Introduction

Energy efficiency is one of the most fundamental issues in Wireless Sensor Networks. Although achieving a long life for a sensor network deployed outdoors is a major challenge for most applications, it is more so for those applications that require continuous monitoring such as intrusion detection or fire detection.

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 calculated such that at any given time only a subset of sensors are active and the remaining are put to sleep. The challenge in this scheme is to design a sleeping schedule that *maximizes* the network lifetime while maintaining the desired quality of monitoring. This preceding problem is referred to as the problem of *sleep-wakeup*.

The problem of sleep-wakeup has been proved NP-Hard in general, even when the lifetime of each sensor is assumed to be the same [1], [2]. As a result, it has become widely accepted that the problem of sleep-wakeup is not polynomially solvable, leading to a proliferation of different heuristic algorithms [1], [2], [3], [4], [5], [6], [7], [8], [9], [10]. In practice, therefore, when a wireless sensor network is prototyped or deployed, it is implicitly assumed that the the problem of sleep-wakeup is not polynomially solvable and then either smart heuristic algorithms are used for sleep-wakeup as in [5] or no sleep wakeup is used as in [11]. Both of these options lead to either a wastage of valuable battery energy or an over-deployment of sensors, or both.

We are not aware of any work that explores the tractability of the sleep-wakeup problem, i.e., for what class of applications the sleep wakeup problem is polynomially solvable. Such an exploration, we believe, will be a valuable contribution to the advancement of wireless sensor networks, given the importance of energy efficiency in this domain.

In this paper, for the first time, we propose polynomial time algorithms to optimally solve the problem of sleep wakeup for a specific class of applications, where a wireless sensor network is deployed as a smart barrier or a tripwire for detecting moving objects as in intrusion detection. The intrusion detection application has been successfully demonstrated on large scale sensor networks as part of at least two projects, the VigilNet project [5] and the Extreme Scaling project [11]. It is already under production for real-life deployments. As such, it is an important application for wireless sensor networks.

The NP-Hardness results [1], [2] are for a specific model of coverage, called *full-coverage* or *blanket coverage*, where every point in the deployment region needs to be covered by one or more sensors. Since full-coverage is a very general concept of coverage, it is usually the first model that comes to mind when considering any application. Even if it is realized later, when developing heuristics for sleep-wakeup, that not every point needs to be covered at all times, as in [8], [3], it is still assumed that the applicable model of coverage is that of full coverage, and, therefore, the sleep-wakeup problem is NP-Hard. In fact, for several applications, every point does not need to be covered at all times, i.e., full-coverage is not the most appropriate model of coverage. For example, for intrusion detection applications, *barrier coverage* is a

more appropriate notion of coverage. This is because a sensor network providing k-barrier coverage guarantees that every moving object of interest will be detected by at least k distinct sensors before it crosses the barrier of wireless sensors, and this guarantee is sufficient for intrusion detection. This guarantee is sufficient for several other applications as well such as fire detection. If fire detecting sensors are deployed as a smart barrier between a forest and an adjoining city, any fire originating from the forest will be detected as soon as it tries to cross the barrier of wireless sensors, which can be well before the city limits begin.

Our algorithms maximize the network lifetime while guaranteeing that the network provides k-barrier coverage at all times. As such, our algorithms can be used to maximize the network lifetime for all applications for which barrier coverage is an appropriate model of coverage.

By optimally solving the sleep-wakeup problem for the heterogeneous lifetime case, we have made the sleep-wakeup scheme even more practical. This is because sensor lifetimes are rarely the same, except after the first deployment. For example, upon sensor failures, which is a frequent event in an outdoor deployment [12], [13], a new sleeping schedule may need to be computed to maintain the desired quality of monitoring and to maintain optimality. This time, however, the sensor lifetimes will not all be the same. Also, when enough sensors have failed so as to warrant additional sensor deployment, the lifetimes of the newly deployed sensors will not be the same as that of previously deployed sensors. Hence, heterogeneous lifetime case is a more pracical one.

As is the case with general version of many problems, solving the sleep-wakeup problem for the heterogeneous case is harder than its homogeneous counterpart. We make use of the results in the area of multiroute network flows [14] to derive an optimal solution to the heterogeneous lifetime case. Prior to our work, multiroute network flows were applied only to the routing problems.

Another difference between the two versions of the sleep-wakeup problem is in minimizing the number of sensor switches. Minimizing the number of such switches simplifies network operation. This is because each time a sensor or a group of sensors in turned off, neighbor discovery, route computation, time synchronization, and other such initialization activities have to be performed afresh. Minimizing the number of times such initialization tasks are performed not only reduces the energy consumption in the network, it makes the network more available to perform the monitoring task, which is the primary reason for deploying the network.

Our sleep-wakeup algorithm for the homogeneous case minimizes the total number of times that a group of sensors is turned off and on during its entire lifetime, while for the heterogeneous lifetime case, we prove that minimizing the number of sensor switches in NP-Hard.

Both of our algorithms (for the homogeneous and heterogeneous lifetime cases) are global algorithms. In Section V, we will describe why it is not possible to design local algorithms for the sleep-wakeup problem for the model of barrier coverage. In that Section, we will also discuss how our algorithms may be implemented and the complexity issue.

In summary, we make three contributions in this paper:

1) We point out that the sleep-wakeup problem is not NP-Hard for all applications and thus open this research area for further exploration.
2) We propose a polynomial-time algorithm to optimally solve the sleep-wakeup problem for barrier coverage for homogeneous sensor lifetime case. In this case, our algorithm also minimizes the total number of (groups of) sensor switches.
3) We propose a polynomial-time algorithm to optimally solve the sleep-wakeup problem for barrier coverage for heterogeneous lifetime case. In this case, we prove that minimizing the number of sensor switches is NP-Hard.

The rest of the paper is organized as follows: In Section II, we present some definitions and summarize some earlier known results. In Section III, we present our algorithm for the homogeneous sensor lifetime case. In Section IV, we present our algorithm for the heterogeneous lifetime case. In Section V, we discuss implementation issues. We conclude the paper in Section VI.

II. Model, Definitions, and Some Earlier Results

In this section, we introduce some definitions and state some known related results.

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

For the purpose of this paper, we assume that a sensor network is deployed over a belt region. Intrusion is assumed to happen from top to bottom. As in [15], a path is said to be a *crossing path* if it crosses from top to bottom. Further, a crossing path is said to be *k-covered* if it intersects the sensing region of at least k distinct sensors. Finally, a sensor network N is said to provide *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,** $\mathcal{G}(N)$ **[15]** A coverage graph of a sensor network N is derived 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 between two nodes if their sensing disks 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.

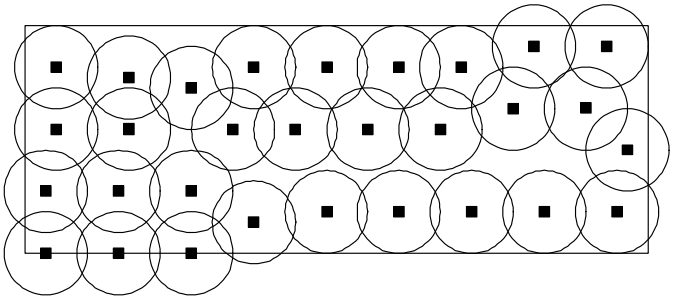

Fig. 1. A sensor network deployment that provides 3-barrier coverage.

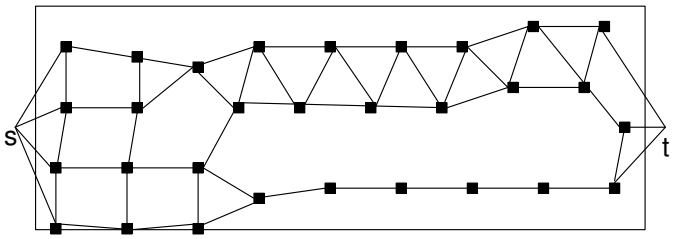

Fig. 2. The coverage graph of sensor network deployment of Figure 1.

Theorem 2.1: [15] A network N provides k-barrier coverage iff there exist k node-disjoint paths between the two virtual nodes s and t in $\mathcal{G}(N)$.

We assume that it is possible to estimate the remaining lifetime of a sensor node. A profile of expected energy consumption of every node may be built using analytical models or using simulators such as PowerTOSSIM [16].

Definition 2.3: **Sensor Switch.** Any time a sensor is turned off before it exhausts its lifetime and is turned on later, it is referred to as a *sensor switch*. If a sensor is allowed to fully exhaust its lifetime when it is turned on for the first time, it is counted as zero sensor switches.

Definition 2.4: **Path Switch.** Any time a group of sensors that together provide 1-barrier coverage is turned off before any of the sensors in the group fully exhausts its lifetime, it is referred to as a *path switch*. If this group of sensors is allowed to fully exhaust its lifetime when it is turned on for the first time, it is counted as zero path switches.

Minimizing the number of path switches has a similar effect as minimizing the number of sensor switches in terms of reducing the frequency of network initialization and maintenance operations. Also, one path of sensors may form a communication path from the one end of the network to the other end if the communication range is twice the sensing range, and used for communication with the base station. In this case, no route recomputation may be necessary when all sensors in a path are turned off and on together.

III. Homogeneous Lifetime

In this Section, we present our *Stint* algorithm to determine an optimal sleeping schedule for individual sensors when sensor lifetimes are homogeneous. The *Stint* algorithm optimizes two objectives simultaneously — 1) the network lifetime for providing k-barrier coverage, and 2) the number of path switches.

A. Upper Bound on the Network Lifetime

We begin by deriving an upper bound on the network lifetime when the sensor lifetimes are homogeneous. 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 1 unit? The following 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 can not provide k-barrier coverage even if all sensors are turned on, and 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 [17], there exists a set of m nodes (corresponding to m sensors) removing which will make virtual nodes s and t disconnected in the coverage graph. Let these m sensors be called *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 can provide 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, we arrive at 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 in this case.

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 now present our *Stint* algorithm that achieves this upper bound.

We first provide an informal description of the *Stint* algorithm. We will use the coverage graph shown in Figure 3 as a running example. The Stint algorithm first computes m, the maximum number of node disjoint paths between s and t. The value of m is 8 in Figure 3. It then checks if m is divisible by k. If it is, then it divides the m disjoint paths in ℓ groups of k paths each. If $k = 2$ in the example of Figure 3, then, $\ell = 4$. These four groups of two disjoint paths each are activated in sequence. The first group provides 2-barrier coverage till it fully exhausts its energy. The second group is activated next, and so on.

If, on the other hand, m is not completely divisible by k, e.g., $k = 3$ in Figure 3, then ℓ is set to $\lfloor m/k \rfloor - 1$, which is 1 for Figure 3. Now, $\ell(= 1)$ group of $k(= 3)$ disjoint paths is allowed to exhaust its lifetimes at a stretch. Let this group be the set of paths $(1, 2, 3)$. Next, the remaining $r = m - \ell * k(= 5)$ disjoint paths are arranged in $f = r/\gcd(r, k)(= 5)$ 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 $\gcd(r, k)/k = (1/3)$ 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. The detailed *Stint* algorithm appears in Figure 4.

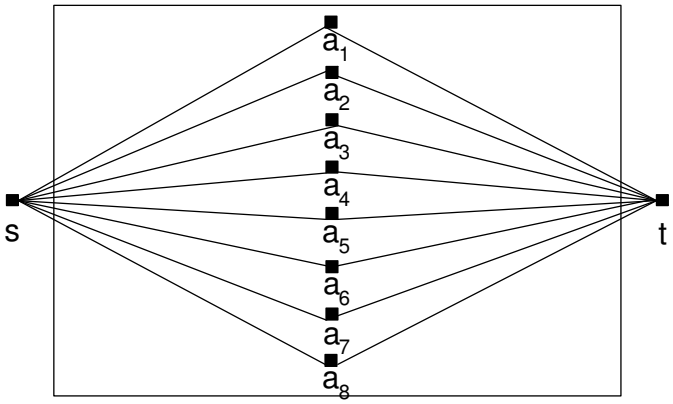

Fig. 3. The coverage graph of a sensor network used to illustrate the operation of the *Stint* algorithm.

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: Partition the sequence of sensors $S_{m-r+1}, \ldots, S_m$ into f sets of sequences, each consisting of x sequences. Call these sets $X_0, X_1, \ldots, X_{(f-1)}$.
18: **for** $j \leftarrow 0$ to $f - 1$ **do**
19: $g \leftarrow \left(j + \frac{k}{x} - 1\right) \mod f$
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

We now prove that the *Stint* algorithm always maximizes the network lifetime for k-barrier coverage. We first consider the case when $k < m < 2k$.

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 *Stint* algorithm allows the network N to provide k-barrier coverage for m/k units of time.

Proof: We first prove that no sensor in the sequences S_{m-r+1} through S_m completely exhausts its energy before the end of the *for loop* in Line 18 through Line 21. Then, we prove that this loop provides k-barrier coverage for r/k units of time, to complete the proof.

To prove the first part, we proceed as follows. First, notice that the sets $S_i, 0 \geq i \geq f-1$ are disjoint. Whenever one of these sets S_i is active all the sensors in this set are active. Since each sensor has a lifetime of unity, the lifetime of each set is unity. In the *for loop* (Line 18 through Line 21), which runs $f = r/x$ times, each set is inactive in $(r-k)/x$ iterations and, therefore, active in exactly $f-(r-k)/x = k/x$ iterations. Since each iteration lasts for x/k units of time, no set will completely exhaust its energy before the end of the loop.

To prove the second part, we observe that k/x sets are active in each iteration of the loop and each of these node-disjoint sets provides x-barrier coverage. Hence, each iteration provides k-barrier coverage. Also, since each iteration lasts for x/k units of time and there are a total of

$f = r/x$ iterations, the *for loop* (Line 18 through Line 21) provides k-barrier coverage for m/k units of time. ■

We now prove the optimality of the *Stint* algorithm for all values of m.

Theorem 3.1: The *Stint* algorithm is an optimal sleep-wakeup algorithm for k-barrier coverage.

Proof: 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. From Lemma 3.1, we know that any sleep-wakeup algorithm for k-barrier coverage can achieve a lifetime of at most m/k. To prove the theorem, we only need to prove that the *Stint* algorithm achieves a network lifetime of m/k.

Lines 10 through 12 in the *Stint* algorithm provide k-barrier coverage for ℓ units of time. If $m \mod k = 0$, the proof is complete. So, assume $m \mod k \neq 0$ so that $k < r < 2k$ (see Line 13 in Figure 4), we apply Lemma 3.2 to conclude that Line 14 through 22 in Figure 4 provide k-barrier coverage for r/k units of time. Since $r = m - \ell * k$, $\ell + r/k = m/k$. Hence, we conclude that the *Stint* algorithm provides k-barrier coverage for m/k units of time and is therefore an optimal sleep-wakeup algorithm for achieving k-barrier coverage. ■

C. Minimizing Path Switches

As discussed in Section I, minimizing the number of times that sensors or a group of sensors is turned on and off reduces the number of times that network initialization tasks such as neighbor discovery and route computation, and makes the network more available for the monitoring task. 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, on the other hand, is only 2. This is because, as described in the operation of the *Stint* algorithm in Section III-B, only paths 4 and 5 are turned off once each before they exhaust their full energy. All other paths are allowed to exhaust their full energy once they are turned on. Also, notice that achieving zero number of 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, which is equivalent to the total number of preemptions in the domain of machine (or processor) scheduling.

Lemma 3.3: 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: For the entire duration of m/k, there must be k node-disjoint paths active for the network to provide k-barrier coverage (using Theorem 2.1). Out of the total m node disjoint paths, no path can be active for m/k units of time since $m/k > 1$.

Let the k disjoint paths that are required for k-barrier coverage be like k machines which have to 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 then becomes equivalent to minimizing the makespan on k machines. The minimum value of makespan is m/k and it will 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 in terms of machine scheduling 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_a 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 start with another new machine and construct a new sequence of q_2 machines, which finish some number of jobs without any leftovers 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 thus encountered 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 proves that the *Stint* algorithm achieves the lower bound of Lemma 3.3.

Theorem 3.2: Of all the optimal sleep-wakeup algorithms for achieving k-barrier coverage, the *Stint* algorithm involves

the minimum number of path switches.

Proof: Observe that zero path switches are involved upto Line 13 in Figure 4, which is the minimum possible. One or more path switches are involved only when $k < r < 2k$. Therefore, we only need to prove that in this case, the Stint algorithm involves the minimum number of path switches.

Lemma 3.3 establishes that any optimal sleep-wakeup algorithm for achieving k-barrier coverage will require at least $k - \gcd(m, k)$ path switches, where m is the maximum number of node disjoint paths available between the virtual nodes u and v in the coverage graph $\mathcal{G}(N)$ of the given sensor network N, if $k < m < 2k$. So, to prove the theorem, we only need to prove that the *Stint* algorithm involves a maximum of $k-\gcd(m, k)$ path switches, which we prove in the following.

Notice that any path switch is performed only in the *for loop* in Line 18 through Line 21. So, we focus our proof 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 $j < \frac{k}{x} - 1$. This is because every group S_i completely exhausts its lifetime if it is active continuously for k/x iterations. Except for the first $\frac{k}{x} - 1$ sets $S_i, 0 \leq i < \frac{k}{x} - 1$, each of which is turned off once it is continuously active for $i + 1$ iterations before it is turned off, every other set of sensors $S_i, \frac{k}{x} - 1 \geq i \geq f - 1$ is active continuously for k/x iterations. Further, the first $\frac{k}{x} - 1$ sets which are turned off before completely exhausting their lifetime, 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 involves exactly $x * \left(\frac{k}{x} - 1\right) = k - x$ path switches. Since $x = \gcd(m, k)$, the claim of the theorem is proved. ■

IV. Heterogeneous Lifetime

In this section, we present the *Prahari*[1] algorithm for maximizing the network lifetime when the lifetimes of sensors are not equal. We then consider the problem of minimizing the number of path switches. For some special cases, the *Prahari* algorithm minimizes the number of path switches. But, in general, the problem of dual optimization of the network lifetime and the number of path switches is *NP-Hard*, in contrast with the homogeneous case. This dual optimization problem continues to be NP-Hard even if it is known that the coverage graph associated with the given sensor network is node-disjoint.

A. Upper Bound on Network Lifetime

The problem of determining the maximum lifetime achievable is made possible by Lemma 3.1 when the sensor lifetimes are all same. When the sensor lifetimes are not all equal, the problem of determining the maximum achievable lifetime becomes significantly more challenging. For example, consider the same network as shown in Figure 1, but with sensors having distinct lifetimes, as shown in Figure 5. What is the maximum time for which this network can provide 2-barrier coverage? This problem may appear NP-Hard at first. However, it is polynomially solvable using the concept of multiroute network flows [14].

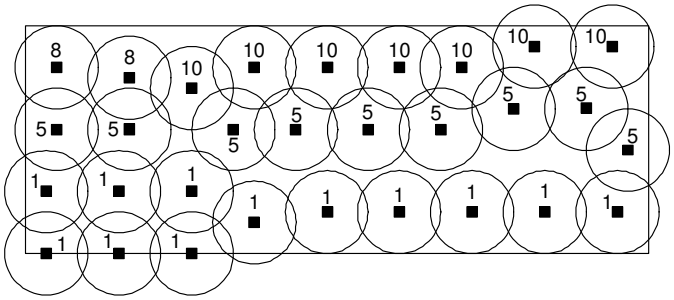

Fig. 5. The sensor network deployment of Figure 1, but with heterogeneous sensor lifetimes. The integers next to the filled squares denote the lifetime of the sensors located there. What is the maximum time for which this network can provide 2-barrier coverage?

We begin by stating some assumptions and relevant definitions. We assume the lifetimes of the sensors can be represented as integers.

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 will regard a $\mathcal{G}_L(N)$ as a directed graph.

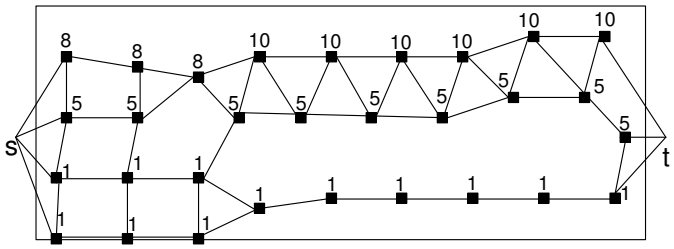

Fig. 6. The coverage graph with lifetime $\mathcal{G}_L(N)$ of the sensor network N shown in Figure 5. The integers next to the filled squares denote the lifetime of the sensors located there.

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

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

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

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

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.

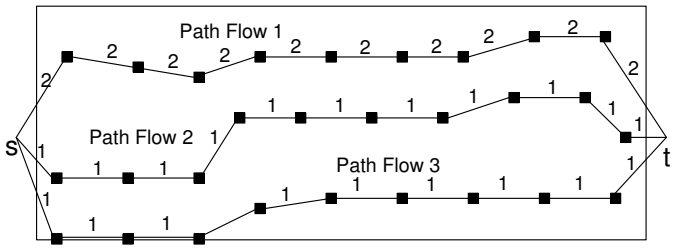

Fig. 7. A composite 2-flow of total value 4 for the sensor network N shown in Figure 5. The edges are labeled with the flow value passing through them. This composite 2-flow is of the maximum value possible for the sensor network N.

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, together make 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 is of value 2, can be decomposed in two path flows of value 1 each.

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 our *Prahari* algorithm emerges from the proof of the following theorem.

Theorem 4.1: Given a sensor network N, there exists a sleep-wakeup schedule to achieve 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: We first prove the "if" part. Given a composite k-flow of T units in $\mathcal{G}_L(N)$, we will construct a sleep-wakeup schedule to achieve a lifetime of T time units. Let F be a composite k-flow of value T. By its 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 $\sum_{i=1}^{m} \lambda_i * a_i = T$.

We now prove the "only if" part. Given a sleep-wakeup schedule that allows N to provide k-barrier coverage for T units of time, we will 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 equal to the maximum value of composite k-flow in $\mathcal{G}_L(N)$ divided by k.

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 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 now present our *Prahari* algorithm that achieves this upper bound.

We first provide an informal description of the *Prahari* algorithm. We will use the coverage graph shown in Figure 6 as a running example.

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. 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 in this case.

If the flow network $F_{MEM}(N)$ is such that the indegree and outdegree of every node other than s and t is 1, as is the case in Figure 6, then the flow network can be decomposed in m number of node-disjoint path flows for some $m > k$. The value of m is 3 in our current example. In this case, the *Prahari* algorithm uses a machine scheduling approach 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 $\hat{f}/k$. Since $k = 2$ in our current example, 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 exhaust its energy and Path Flow 3 will be activated. Thus, we achieve a lifetime of 2 time units, which is the maximum possible in this case.

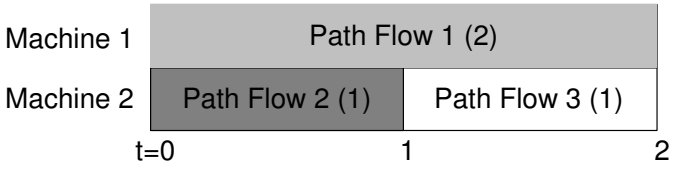

Fig. 8. The machine scheduling approach followed by the *Prahari* algorithm is illustrated for the flow network shown in Figure 7. The numbers in the parantheses denote the lifetime of the individual paths.

If the flow network $F_{MEM}(N)$ is such that some node in $V - \{s, t\}$ has a degree 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 optimal network lifetime of $\hat{f}/k$ is achieved in this manner. The details of the *Prahari* algorithm appears in Figure 9.

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, which we prove in the following.

It has been proved in [14] that the *MEM [14]* 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 Lines 3 through 14 are executed. We will prove that this scheduling 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.

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 $\left(t_w^{(i)}(v), t_s^{(i)}(v)\right)$, 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.

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. ■

C. Minimizing Path/Sensor Switches

The problem of lifetime maximization becomes NP-Hard if we additionally require the minimization of sensor switches. We prove the decision version of this problem to be *strongly* NP-Complete by reducing the 3-Partition [19] problem to it. The problem remains *NP-Hard* even if we are given a network where all paths between s and t in the associated coverage graph are node-disjoint so that we try to minimize the number of path switches instead of sensor switches.

We first make few observations about the *Prahari* algorithm in terms of the number of path switches, if the path flows resulting from applying the *MEM* algorithm are node-disjoint:

1) The number of path switches involved is at most m' (for $m' \leq m$), where m' is the number of flows with value strictly less than f_k/k.
2) Let $A = \{f_i, 1 \leq i \leq m\}$ be the set of flow values. Let P be the following predicate

$$\exists B \subset A : \sum f_i \in B f_i/k' = \sum f_i \in A f_i/k,$$

for some $k' < k$. If P does not hold, then the number of path switches involved in the *Prahari* algorithm is optimal.

It can be verified that the above observations hold.

We now prove the *NP-Hardness* of the dual optimization problem. We first define the decision version of the optimization problem. *Barrier Coverage Lifetime With Zero Sensor switches (BCLZSS):*
INSTANCE: Integers $L, k \in \mathbb{Z}^+$, location of n sensors each with a sensing radius of r, and the lifetime of sensors i is $\ell_i \in \mathbb{Z}^+$ units.
QUESTION: Can the network provide k-barrier coverage for L units of time with 0 sensor switches?

We use the 3-partition problem, defined below, in the proof. *3-Partition*
INSTANCE: Set A of $3m$ elements, a bound $B \in \mathbb{Z}^+$, and a size $s(a) \in \mathbb{Z}^+$ for each $a \in A$ such that $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$.
QUESTION: Can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that, for $1 \leq i \leq m, \sum_{a \in A_i} s(a) = B$?

Theorem 4.3: The *Barrier Coverage Lifetime With Zero Sensor switches* problem is NP-Complete.

Proof: We first show that the Barrier Coverage Lifetime With Zero Sensor switches is in *NP.* Assume we are provided with a schedule, i.e. a sequence of intervals I_j and the set of sensors that are active in that interval. It can be checked in polynomial time whether the active time for any sensor exceeds its available lifetime. Next, we can invoke the algorithm described in [15] to verify for each interval I_i whether the sensors active in this interval provide k-barrier coverage. Notice that n intervals are enough since at least one sensor must be exhausted to cause a schedule switch, and there are a total of at most n sensors that will be used to provide k-barrier coverage. Finally, we can check for each sensor if it involves an on/off.

To prove that the Barrier Coverage Lifetime With Zero On/off problem is *strongly NP-Complete*, we reduce *3-Partition* to it. Given an instance of the 3-Partition problem (i.e. a set A of $3k$ nodes each with a size of $s(a)$ for each $a \in A$, and a number B), we construct a coverage graph as follows: Set $L = B$ and $k = m$. Create two disjoint sets of k nodes each, called S and T, so that $n = 3m + 2k = 5k$. For each $i \in S \cup T$, $\ell_i = B = L$ units, and $\forall i \in A$, $\ell_i = s(i)$. Connect the k nodes in set S to the virtual node s in the coverage graph and also connect it to all the $3k$ nodes in set A. Connect the k nodes in T with the virtual node t and also to all the $3k$ nodes in the set A. (See Figure 10 for an example.) Now, we claim that the network provides k-barrier coverage for L units of time with zero sensor switches iff the set A can be partitioned in m disjoint sets $A_1, A_2, \ldots, A_m$ such that, for $1 \leq i \leq m, \sum_{a \in A_i} s(a) = B$.

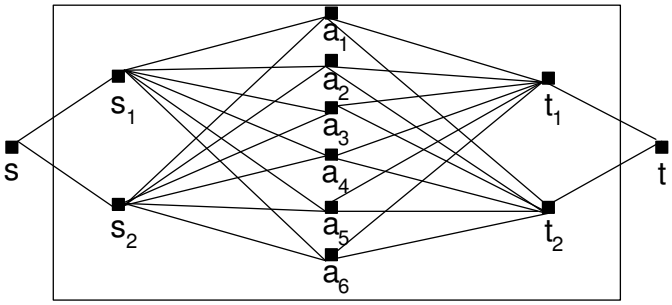

Fig. 10. The coverage graph constructed from an instance of 3-partition when $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$.

Let us first prove the *if* part. Assume the set A can be partitioned in m disjoint sets such that $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$. Since for each $a \in A$, $B/4 < s(a) < B/2$, each set A_i consists of exactly three elements. Number the k nodes in set S as $1, 2, \ldots, k$. Do the same numbering for the k nodes in T. Now, form a set of three paths $P_i, 1 \leq i \leq k$, that consist of node i from set S, node i from set T, and nodes in the set A_i. Since $\sum_{a \in A_i} \ell_a = L$, each path set P_i provides 1-barrier coverage for $B = L$ units of time. Since the set of paths P_i are node-disjoint, the network provides k-barrier coverage for L units of time.

We now prove the *only if* part of the claim. Assume that the network provides k-barrier coverage for L units of time with zero sensor switches. Since every path between the virtual nodes s and t has to include one or more nodes from the set A, $\sum_{a \in A} s(a) = kL$, and the network has to provide k barrier coverage for L units of time, every node in set A will fully exhaust its lifetime in L time units. Without loss of generality, we may assume that no node in set A will be shared by two nodes in set S, or by any two nodes in set T. Suppose a node $a \in A$ is shared by two nodes $s_1, s_2 \in S$. Also, assume that a is first used by

s_1 and then by s_2. Further assume that this is the first time instant that a node is transferred between two nodes of S and that the node in A that was in use by s_2 prior to this instant was fully exhausted. Then, we can switch all the nodes that had been used by s_1 and s_2 prior to this instant and have no effect on the system lifetime. Now, there is no sharing upto this time instant. All other sharing can be eliminated in a similar fashion.

Since the network provides k-barrier coverage for L units of time, it must be the case that there are m disjoint sets of nodes in set A such that for $1 \leq i \leq m = k$, $\sum_{a \in A_i} \ell_a = L = B$. Let A_i be the set of nodes in A that are used by node $i \in S$. This gives us the desired 3-partition of set A. This completes the proof. ■

We now prove that minimizing the number of path switches is NP-Complete even if it is known that all the paths between the virtual nodes s and t in the underlying coverage graph are node-disjoint. We reduce the *Partition* problem to it. We first define the two problems.

Node Disjoint Barrier Coverage Lifetime With Zero Path Switches:
INSTANCE: Integers $L, k \in \mathbb{Z}^+$, location of n sensors each such that all the paths between the virtual nodes s and t in the associated coverage graph are node-disjoint, and the lifetime of sensors i is $\ell_i \in \mathbb{Z}^+$ units.
QUESTION: Can the network provide k-barrier coverage for L units of time with 0 path switches?

3-Partition
INSTANCE: Set A of integers $c_1, c_2, \ldots, c_n$.
QUESTION: Does there exist a set $S \subset \{1, 2, \ldots, n\}$ such that $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$?

Theorem 4.4: The *Node-Disjoint Barrier Coverage Lifetime With Zero Path Switches* problem is NP-Complete.

Proof: The fact that the *Node-Disjoint Barrier Coverage Lifetime With Zero Path Switches* problem is in NP can be verified in the same way as was done for the *Barrier Coverage Lifetime With Zero Sensor Switches* in Theorem 4.3.

We reduce the Partition problem to the *Node-Disjoint Barrier Coverage Lifetime With Zero Path Switches* problem. Given an instance of the partition problem we construct a sensor network as follows: Let the deployment region be rectangular with the left bottom corner at the origin, i.e. with coordinate $(0, 0)$. Let the right bottom corner be at the coordinate $(2r, 0)$. Let $\epsilon > 0$. For every integer $c_j \in A$ we place a sensor at coordinate $(r, (j-1) * (2r + \epsilon))$. Set $k = 2$ and $L = \sum_{j=1}^{n} c_j/2$. Notice that in the coverage graph of this sensor network, all the n paths between the two virtual nodes s and t are node-disjoint.

If the answer to the partition problem is "yes," then $\exists S \subset \{1, 2, \ldots, n\}$ such that $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$. Now, the sensor network can achieve k barrier coverage for L units of time since the set of sensors can be partitioned into two sets corresponding to S and $\{1, 2, \ldots, n\} - S$ and each of these sets will provide 1-barrier coverage for L units of time.

Conversely, if the sensor network can provide 2-barrier coverage for L units of time, the sensors can be partitioned into two disjoint sets such that each set provides 1-barrier coverage for L units of time. This is because any sensors that is turned on remains on until it exhausts its lifetime (c_j for some j), and every sensor has to completely exhaust its lifetime if the network is to provide 2-barrier coverage for L units of time. ■

V. Implementation Issues

In this Section, we discuss some implementation issues. We first describe why local sleep-wakeup algorithms are not possible for barrier coverage, then describe how our two sleep-wakeup algorithms may be implemented in a typical sensor network. Finally, we discuss the complexity of our two algorithms.

While it is possible to locally determine whether a region is not k-fully covered [20], it is not possible to determine locally whether a region is not k-barrier covered, as is shown in [15]. It is the local check for violation of k-full coverage that allows the development of local heuristic algorithms for sleep wakeup for the full coverage model. Non-locality of such a check for barrier coverage implies nonexistence of not only optimal sleep-wakeup algorithms but also nonexistence of heuristic sleep-wakeup algorithms for barrier coverage.

Although non-local distributed versions of our sleep-wakeup algorithms are possible, there are several reasons why a global algorithm may be practical and sometimes more desirable than a distributed algorithm in wireless sensor networks. First, sensor networks are often optimized for sensor-to-base station communication, and hence the implementation of a global algorithm is well supported by the sensor network architecture as we discuss in the next paragraph. Second, base stations are typically wall powered and have much higher computation capability as compared to the sensors. Third, performing one global computation, if it saves computation and communication among the sensors, may sometimes be prefered over distributed computation at the sensors.

We now discuss how our algorithms may be implemented. Both of our algorithms need two basic information about the network — the location information of all sensors, and a list of healthy sensors. The location information is available as a result of localization operation which is usually done at the time of initial deployment. A list of healthy sensors is also usually available at the base station because regular health monitoring is an integral part of most deployed sensor networks [21]. With these information, our algorithms can be executed at a base station to determine an optimal sleeping schedule. This sleeping schedule, as well as any subsequent updates to it, can be communicated to the sensors using SNMS [21] like protocols. Note that there is no need for any extra sensor-to-sensor communication in our algorithms.

In terms of complexity, both of our algorithms (*Stint* and *Prahari*) have the best comlexity possible in terms of traditional measure of comlexity. This is because the complexity in both of these algorithms is dominated by the standard graph theoretic computations. In the *Stint* algorithm, the dominant part is the computation of node-disjoint paths, whose best known complexity is higher than that for other steps in the *Stint* algorithm. Similarly, in the *Prahari* algorithm, the dominant part is the computation performed by the *MEM* algorithm and the *SEM* algorithm. Again, the best known complexity of these algorithms or their alternatives is higher than that for the remaining steps of the *Prahari* algorithm.

VI. Conclusions

The problem of optimal sleep-wakeup is a fundamental problem for wireless sensor networks given the criticality of energy efficiency. It has been widely accepted that the sleep-wakeup problem is not polynomially solvable. However, the NP-Hardness results proved for sleep-wakeup are for a specific model of coverage called full coverage. It is an open issue to explore whether sleep-wakeup is polynomially solvable for other models of coverage.

In this paper, we showed that the sleep-wakeup problem can be optimally solved in polynomial-time for the barrier coverage model, thus enabling applications such as intrusion detection to use sleep-wakeup in practical situations to maximize the network lifetime. We not only solved the case of homogeneous sensor lifetimes but also the more general case of heterogeneous lifetime.

Our results open the area of sleep-wakeup for further exploration. It will be interesting and useful to determine the class of applications and the class of coverage models for which the sleep-wakeup problem is polynomially solvable. Such an exploration will enable the use of sleep-wakeup in several practical applications with tangible benefits in terms of network lifetime maximization.

References

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

[2] 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.

[3] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare event detection," in *IPSN*, Nashville, TN, 2006.

[4] 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.

[5] T. He, P. Vicaire, T. Yan, Q. Cao, L. Gu, G. Zhou, J. A. Stankovic, and T. Abdelzaher, "Achieving long term surveillance in vigilnet," in *IEEE INFOCOM*, Bercelona, Spain, 2006.

[6] R. Iyer and L. Kleinrock, "Qos control for sensor networks," in *IEEE International Communications Conference (ICC)*, 2003.

[7] J. Hui, Z. Ren, and B. H Krogh, "Sentry-based power management in wireless sensor networks," in *IPSN*, Palo Alto, CA, 2003, pp. 448–47.

[8] Chao Gui and Prasant Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *International Conference on Mobile Computing and Networking (ACM MobiCom)*, Philadelphia, PA, 2004, pp. 129–143.

[9] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *ACM Conference on Ebmedded Networked Sensor Systems (SenSys)*, Los Angeles, CA, 2003, pp. 28–39.

[10] 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.

[11] "Extreme scale wireless sensor networking," Tech. Rep., http://www.cse.ohio-state.edu/exscal, 2004.

[12] Sandip Bapat, V. Kulathumani, and A. Arora, "Analyzing the yield of exscal, a large scale wireless sensor network experiment," in *IEEE International Conference on Network Protocols (ICNP)*, Boston, MA, 2005.

[13] R. Szewczyk, J. Polastre, A. M. Mainwaring, and D. E. Culler, "Lessons from a sensor network expedition," in *EWSN*, Berlin, Germany, 2004.

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

[15] 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.

[16] V. Shnayder, M. Hempstead, B. Chen, B. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *ACM Sensys*, Baltimore, MD, 2004.

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

[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] C. Huang and Y. Tseng, "The coverage problem in a wireless sensor network," in *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, San Diego, CA, 2003, pp. 115–121.

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