# Maximizing the Lifetime of a Barrier of Wireless Sensors

Santosh Kumar, *Member, IEEE,* Ten H. Lai, Marc E. Posner, and Prasun Sinha, *Member, IEEE*

***Abstract*—To make a network last beyond the lifetime of an individual sensor node, redundant nodes must be deployed. What sleep-wakeup schedule can then be used for individual nodes so that the redundancy is appropriately exploited to maximize the network lifetime?**

**We develop optimal solutions to both problems for the case when wireless sensor nodes are deployed to form an impenetrable barrier for detecting movements. In addition to being provably optimal, our algorithms work for non-disk sensing regions and heterogeneous sensing regions. Further, we provide an optimal solution for the more difficult case when the lifetimes of individual nodes are not equal.**

**Developing optimal algorithms for both homogeneous and heterogeneous lifetimes allows us to obtain, by simulation, several interesting results. We show that even when an optimal number of sensor nodes have been deployed randomly, statistical redundancy can be exploited to extend the network lifetime by up to seven times. We also use simulation to show that the assumption of homogeneous lifetime can result in severe loss (two-thirds) of the network lifetime. Although these results are specifically for barrier coverage, they provide an indication of behavior for other coverage models.**

***Index Terms*—Wireless sensor networks, sleep-wakeup, sensor deployment, barrier coverage, multi-route network flows.**

## I. Introduction

Wireless sensors are meant for outdoor deployments where they may remain unattended for long periods of time. Security applications such as intrusion detection, fire detection, and chemical leak detection require sufficient number of sensor nodes to be active at any time instant. Given that the sensor nodes operate with small batteries, individual nodes may not last a long time if continuously active. To make a network last beyond the lifetime of an individual node, redundant nodes must be deployed. What sleep-wakeup schedule can then be used for individual nodes so that redundancy is appropriately exploited to maximize the network lifetime?[1]

In this paper, we develop optimal solutions to this problem for the case when wireless sensor nodes are deployed to form an impenetrable barrier for detecting movements [1]. In addition to intrusion detection applications [2], [3], 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. As has been argued in [1], deploying sensor nodes as a barrier leads to an order of magnitude saving in the number of nodes over the full coverage model (where every point in the deployment region is covered), while guaranteeing that no movement will go undetected.

In addition to being provably optimal, our algorithms work for non-disk sensing regions and heterogeneous sensing regions. For our algorithms to be applicable, it is sufficient to determine whether the sensing regions of two sensors intersect. This can be determined even when a precise model for the sensing region is not known. For instance, if two sensors detect a target simultaneously, their sensing regions must intersect [4]. Our algorithms, therefore, allow for the sensing regions of various nodes to be different.

Further, we propose an optimal sleep-wakeup algorithm for the case when all sensors have equal lifetime (the *homogeneous lifetime* case) and also for the harder case when the sensors have distinct lifetimes (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 start with the same type of batteries, they are subject to different kinds of loads (due to routing structure, cluster structure, etc.).
- Different Recharging Rates: If sensors are using rechargeable batteries, then lifetime depends on the amount of energy (e.g., solar, wind) a sensor receives.
- Unanticipated Failures: When there are unanticipated sensor failures, a new schedule may be needed. Then, the remaining lifetimes of operational sensors may be distinct.
- Additional Deployment: When new sensors are deployed in an existing network to compensate for failed ones, at the time of deployment the remaining lifetimes are distinct.

Since our algorithm provides an optimal solution for heterogeneous lifetime case, when the network experiences an unanticipated sensor failure, new sleep-wakeup schedule can be computed that will again maximize the lifetime of the surviving network.

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 able 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.

[1]We consider a sensor node to be in *sleep* state if all of its components (e.g., radio, processor, sensors, voltage regulators, etc.) are put in the deep sleep state, and *awake* if it has at least one component (say, a sensor) active. Consequently, the average energy consumption in the sleep state is lower than that in the awake state.

However, for the heterogeneous lifetime case, we establish that finding the minimum number of sensor switches is NP-Hard.

It is not possible to design a provably optimal deterministic local sleep-wakeup algorithm because it can not be checked locally whether the network provides barrier coverage [1]. Consequently, our algorithms are centralized algorithms. Sleep-wakeup algorithms need to be executed rarely[2], and the optimal schedule can be distributed in the network using the same utility as used in network reprogramming [5], [6]).

We use our optimal algorithms to *study, by simulation, two interesting coverage/lifetime issues* that have not been addressed in the literature. First is the issue of statistical redundancy in a random sensor deployment. In a random deployment, sensors are deployed to achieve a target probability of coverage. For instance, if $n$ sensors are needed to achieve a target probability of barrier coverage of 0.001, then, on average, in 999 out of 1,000 instances of deployment, the region will be barrier covered with a random deployment of $n$ sensors. *For those instances where the $n$ sensors do provide barrier coverage, how much redundancy exists in the network?*

Second, lifetimes of individual sensors are rarely equal. However, for simplicity or tractability, the assumption of homogeneity is often used. As a result, network lifetime is lower than what could have been achieved if the lifetime optimization algorithm considered the heterogeneity of sensor lifetimes. *How much does one lose in expected network lifetime due to the assumption of homogeneity?*

For the first problem, using the density estimate from [7], we show that even when the minimum necessary number of sensors are deployed to achieve barrier coverage in a random deployment, the network may have enough redundancy to *last up to seven times longer* than planned. For the second problem, we show that the assumption of homogeneity in a heterogeneous network can *result in a two-thirds loss* of network lifetime, e.g., a year of expected lifetime may be reduced to four months.

We also observe in our simulations that if the energy *imbalance* in the network is sufficiently random (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).

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 simulation study. We conclude the paper in Section VII.

## II. Model, Definitions, and Background

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

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

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

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 [1], 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,** $\mathcal{G}(N)$ **[1]** 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, $V$ has two virtual nodes, $s$ and $t$ that correspond to the left and right boundaries, respectively. An edge exists 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 any model for which a coverage graph can be constructed. Further, the sensing range can be different for each sensor; the only requirement is that we can construct a coverage graph. Finally, the links of the coverage graph are virtual links and not used for actual communication between sensor nodes.

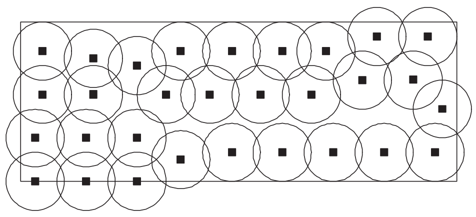
Fig. 1. A sensor network deployment that provides 3-barrier coverage.

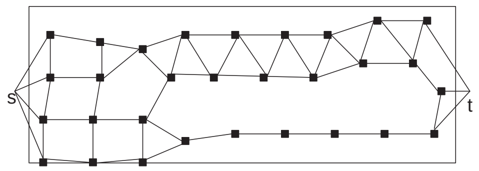

Fig. 2. The coverage graph of sensor network deployment of Figure 1.

*Theorem 2.1:* [1] 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.** A *sensor switch* occurs when a sensor is turned off and is later turned on. 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.** A *path switch* occurs when a group of sensors that together provide 1-barrier coverage is turned off and is later turned on as a group. 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 is equivalent to 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. NP-Hardness is established by [8]. Subsequently, several heuristic algorithms appeared [9]–[12]. However, no guarantee of performance is made by any of these works.

For the barrier coverage model, a randomized sleep-wakeup algorithm called Randomized Independent Sleeping (RIS) is proposed in [1]. In this algorithm, time is divided into intervals. In every 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 local (i.e. requires no central coordination and no message exchange with any neighbors). However, there are several shortcomings. First, the approach does not provide deterministic guarantee of barrier coverage. Second, if deployment is not determined by a uniform or Poisson distribution, then there is no guidance on how to choose a value for $p$. Third, if the lifetimes of the sensor nodes are not identical, then again there is no guidance on how to choose a value of $p$. Finally, there is no guarantee of performance.

A localized algorithm called Localized Barrier Coverage Protocol (LBCP) is proposed by [13] to increase the lifetime of a network deployed for barrier coverage. Although the localized version does not guarantee barrier coverage, the locality region size ($L$ in [13]) can be increased to match the length of the deployment region to deterministically guarantee barrier coverage. However, then the LBCP algorithm becomes a global algorithm since each node now needs to communicate with every other node in the network. Also, although the performance of LBCP is statistically close to optimal for some parameter settings, no performance guarantees are provided. Finally, LBCP does not address the heterogeneous lifetime case.

Several other works have addressed various aspects of barrier coverage. For example, in [14], a distributed algorithm is presented to construct barriers of wireless sensors. In [7], it is shown that the fact that percolation does not occur in thin long strips does not prevent one to derive critical conditions for barrier coverage (called *strong* barrier coverage in [1] and [14]. In fact, reliable estimates for sensor density, which are stronger results than critical conditions, are derived in [7] for barrier coverage in thin long strips. Also, in [15], a heuristic is proposed to select a subset of directional sensors to achieve $k$-barrier coverage. These works, however, do not address the problem of lifetime maximization.

## III. Homogeneous Lifetime

In this section, we begin by deriving an upper bound on the network lifetime when the sensor lifetimes are homogeneous. Then, we present algorithm *Stint* that determines an optimal sleep-wakeup schedule for individual sensors. Finally, we prove that *Stint* 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. 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 $m \geq k$.

The next lemma provides an upper bound on the maximum achievable network lifetime.

*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 [16], 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.

To provide $k$-barrier coverage, Theorem 2.1 states that 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 do not share any node. Therefore, each set of $k$ node-disjoint paths contains 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 a maximum lifetime of $m/k = 3/2$.

### B. Achieving the Upper Bound

In Section III-A, an upper bound on network lifetime for $k$-barrier coverage in the homogeneous lifetime case is derived. The *Stint* algorithm achieves this upper bound. While we now provide an informal description of this algorithm, the details appear in Figure 4.

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 found using a max flow algorithm as discussed in [1]. The Stint algorithm then determines whether $m$ is divisible by $k$. If it is, then $m$ disjoint paths are partitioned into $\ell = m/k$ groups of $k$ paths each. Then, $\ell$ groups of $k$ disjoint paths are activated in sequence. The first group provides $k$-barrier coverage until it runs out of energy. Then, the second group is activated. The process continues for $\ell$ iterations.

Alternatively, if $m$ is not divisible by $k$, then $\ell = \lfloor m/k \rfloor - 1$. Similar to the prior case, $\ell$ groups of $k$ disjoint paths 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, where each path is in $k$ sets. Each of these $f$ sets of paths is kept active for $\gcd(r, k)/k$ of the total sensor lifetime. 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 lifetime according to Lemma 3.1.

We use the coverage graph shown in Figure 3 to illustrate the operation of 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.

When $k = 3$, then $m$ is not divisible by $k$. Now, $\ell$ 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, where each path is in 3 sets. Five possible sets are $\{(4, 5, 6), (5, 6, 7), (6, 7, 8), (7, 8, 4), (8, 4, 5)\}$. 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.

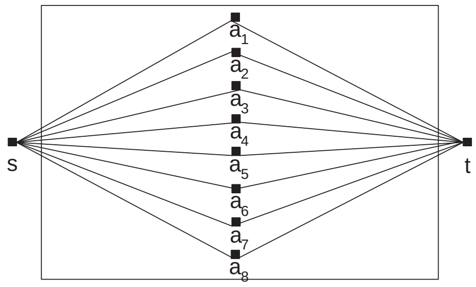

Fig. 3. The coverage graph of a sensor network used to illustrate the operation of the *Stint* algorithm.

We now prove that the *Stint* algorithm 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 provides $k$-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 show that this loop provides $k$-barrier coverage for $r/k$ units of time.

To prove the first part, observe that the sets $S_i, 0 \geq i \geq f - 1$ are disjoint. Whenever one of these sets $S_i$ is active, all 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. Therefore, the set is active in

**Input:** A sensor network $N$ deployed over an open belt region and the desired degree of coverage $k$. 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 $i^{\text{th}}$ 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:             $g \leftarrow \left(j + \frac{k}{x} - 1\right) \mod f$
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

exactly $f - (r - k)/x = k/x$ iterations. Since each iteration lasts for $x/k$ units of time, no set completely exhausts its energy before the end of the loop.

To prove the second part, observe that $k/x$ sets are active in each iteration of the loop. 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, 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 $\ell$ units of time. If $m \mod k = 0$, the proof is complete. Consequently, assume $m \mod k \neq 0$. This implies that $k < r < 2k$ (see Line 13 in Figure 4). 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, the *Stint* algorithm provides $k$-barrier coverage for $m/k$ units of time. ■

**Complexity:** 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 Edmonds-Karp algorithm can determine a max flow with complexity $O(VE^2)$ [17]. Some optimization procedures can reduce the complexity to $O(V^3/\log(V))$.

## C. Minimizing Path Switches

In this section, we consider a lexicographic two objective optimization problem where the first objective is to maximize the network lifetime, and the second objective is to minimize the number of path switches. 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 are *required* if 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. Then, we 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 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. In general, zero path switches are needed up to Line 13 in Figure 4, which is the minimum possible. Path switches are required only when $k < r < 2k$. The following lemma derives a lower bound on the total number of path switches for this case. We show that number of path switches is equivalent to the 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 to provide $k$-barrier coverage is at least $k - \gcd(m,k)$.

*Proof:* From Theorem 2.1, there must be $k$ node-disjoint paths active for $m/k$ time units. Further, no path can be active for more than $1 < m/k$ unit of time. Convert this problem to a machine scheduling problem [18]. 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 maximizing $k$-barrier coverage lifetime is 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. Moreover, the number of path switches is equivalent to 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. Because $1 < m/k < 2$, 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 $\sigma$ 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 3.3.

*Theorem 3.2:* Of all the optimal sleep-wakeup schedules for achieving $k$-barrier coverage, the *Stint* algorithm constructs a schedule that has 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 $k < r < 2k$. Lemma 3.3 establishes that any schedule which achieves $k$-barrier coverage requires at least $k - \gcd(m,k)$ path switches, where $k < m < 2k$. To establish the theorem, we show that the *Stint* algorithm constructs a maximum of $k - \gcd(m,k)$ path switches.

Notice that a 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 a sequence of $x$ 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 $x = \gcd(m,k)$. ■

## IV. Heterogeneous Lifetime

In this section, we derive an upper bound on the network lifetime when the sensor lifetimes are heterogeneous. Next, we present the *Prahari*[3] algorithm to determine an optimal sleep-wakeup 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? We provide a provably optimal solution to this problem by making use of multiroute network flows [19].

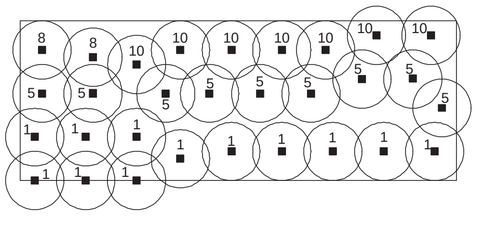

Fig. 5. The sensor network deployment of Figure 1, where the integer is the lifetime of the sensor located in the filled square.

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 [20] 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 [21]. Sensor lifetimes can have any real positive value.

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

*Definition 4.1:* **Coverage Graph with Lifetime, $\mathcal{G}_L(N)$.** A coverage graph with lifetime of a sensor network $N$, where each node $u \in V - \{s,t\}$ is assigned a capacity, $c(u)$, equal to the remaining lifetime. Each edge is assigned infinite capacity. 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 $\mathcal{G}_L(N) = (V,E)$ to a directed graph, replace each edge $\{u,v\}$ with a pair of directed edges $(u,v)$ and $(v,u)$. For the remainder of Section IV, we assume that $\mathcal{G}_L(N)$ is a directed graph.

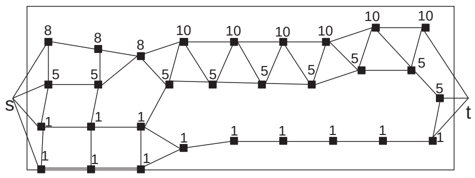

Fig. 6. The coverage graph with lifetime $\mathcal{G}_L(N)$ of the sensor network $N$ shown in Figure 5.

*Definition 4.2:* $s$-$t$ **Flow.** An *$s$-$t$ flow* in $\mathcal{G}_L(N)$ is a mapping $f : E \rightarrow \mathbb{R}^+$ such that
1) $\forall u \in V - \{s,t\}, \sum_{(u,v)\in E} f(u,v) = \sum_{(v,u)\in E} f(v,u)$,
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 from the coverage graph shown in Figure 6 are shown in Figure 7.

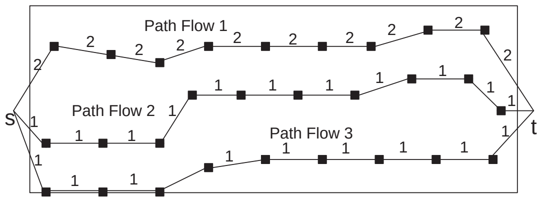

Fig. 7. A composite 2-flow of total value 4 for the sensor network $N$ shown in Figure 5.

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

The total value of the flow is $k*a$. In Figure 7, Path Flow 2 and Path Flow 3 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 $\mathcal{G}_L(N)$ that can be expressed as a sum of basic $k$-flows.
The total value of a composite $k$-flow is $\sum_{i=1}^{m} \lambda_i * k * a_i$, for appropriate weights $\lambda_i \in \mathbb{Z}^+$, if $m$ basic $k$-flows each with a value of $a_i$ make up the composite $k$-flow.

A composite 2-flow of total value 4 from the coverage graph shown in Figure 6 appears in Figure 7.

We now state the key result of this section, which enables us to find in polynomial time an upper bound on the network lifetime when the sensor lifetimes are heterogeneous. The basic idea of the *Prahari* algorithm comes from the proof of the 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 $\mathcal{G}_L(N)$denoted as $F$, we construct a sleep-wakeup schedule to achieve a lifetime of $T$ time units. 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 $i^{\text{th}}$ basic $k$-flow. In every basic $k$-flow $i$, there are $k$ node-disjoint flows each with value $a_i$. 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 $t$ with these sensors active. Denote this basic $k$-flow by $F_1$. Similarly, the total value of $F_i$ is $k * t_1$ for $i = 2, 3, \ldots, m$, where $m$ is the number of 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 $\mathcal{G}_L(N)$, then we can derive an upper bound on the network lifetime achievable by $N$. For this purpose, we make use of the MEM algorithm [19].

As an example, applying this algorithm to the coverage graph shown in Figure 6, we determine that the maximum value of a composite 2-flow, $\hat{f}/k$, 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*

We present the *Prahari* algorithm that achieves the upper bound derived in Section IV-A on the network Lifetime that any sleep-wakeup algorithm can achieve for $k$-barrier coverage in the heterogeneous lifetime case. 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 [19] 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 into $m > k$ node-disjoint path flows. Then, the *Prahari* algorithm uses a machine scheduling algorithm proposed by [22] to schedule the $m$ paths to achieve a lifetime of $\hat{f}/k$ time units, or equivalently to schedule $m$ jobs on $k$ machines to achieve a makespan of $\hat{f}/k$. Thus, we achieve the maximum lifetime in this case.

Alternatively, 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 [19] to decompose the flow network into $\alpha' > k$ basic $k$-flows. SEM then merges identical basic $k$-flows into a single aggregate basic $k$-flow. Let $\alpha$ 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 $\alpha$ 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, $\hat{f} = 4$. Because 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 are used for scheduling. Also, the minimum makespan, which is equivalent to the maximum network lifetime, 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 generates a schedule for the three paths. Path Flow 1 is active for 2 time units continuously. Path Flow 2 is active for 1 time unit starting at time $t = 0$. At $t = 1$, Path Flow 2 runs out of energy and Path Flow 3 is activated. Thus, we achieve a lifetime of 2 time units, which is the maximum possible.

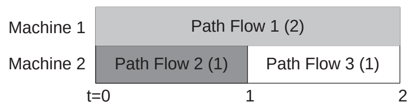

Fig. 8. The machine scheduling approach followed by the *Prahari* algorithm for the flow network shown in Figure 7. The numbers in the parentheses denote the lifetime of the individual paths.

We now formally establish the optimality of the *Prahari* algorithm.

*Theorem 4.2:* The *Prahari* algorithm determines a sleep-wakeup schedule that provides $k$-barrier coverage for the maximum network lifetime.

*Proof:* Consider a sensor network $N$ and the coverage graph $\mathcal{G}_L(N) = (V, E)$. Let $f_k(N)$ denote the maximum value of the composite $k$-flow. Corollary 4.1 establishes that an upper bound on the lifetime to provide $k$-barrier coverage for $N$ is $f_k(N)/k$. Therefore, to establish that the *Prahari* Algorithm finds the maximum network lifetime, we only need to prove that the algorithm finds a schedule where the network $N$ provides $k$-barrier coverage for $f_k(N)/k$ units of time.

The *MEM* algorithm computes the value of $f_k(N)$ ( [19]). If the flows in the network that result from applying the *MEM* algorithm are node-disjoint (besides $s$ and $t$), then Lines 3

**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 $c(u)$.

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

**The *Prahari* Algorithm**

```
1: Invoke the MEM algorithm for $\mathcal{G}_L(N)$. Let $\hat{f}$ be the value
   of the 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 $t + f_i \le \hat{f}/k$ 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
      $\alpha'$ basic $k$-flows, $N_1, N_2, \ldots, N_{\alpha'}$.
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 $\alpha$.
18:   $t \leftarrow 0$.
19:   for $i \leftarrow 1$ to $\alpha$ 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 maximizing network lifetime.

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 interval $[0, f_k(N)/k]$, $k$ node-disjoint paths are active. Each of these paths provides 1-barrier coverage. Further, since the value of any individual flow 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, then the *SEM* algorithm decomposes 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 [23]. Since each basic flow $N_i$ provides $k$-barrier coverage for $f(N_i)/k$ time units and $\sum_{i=1}^{\alpha} f(N_i) = f_k(N)$, 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* algorithm [19], whose complexity is $O(kV^3/\log(V))$.

## *C. Minimizing Path/Sensor Switches*

In this section, we consider a lexicographic two objective optimization problem where the first objective is to maximize the network lifetime, and the second objective is to minimize the number of sensor switches. We show that this two-objective optimization problem is NP-Hard. We prove that the decision version of this problem is *strongly* NP-Complete. This problem remains *NP-Hard* even if all paths between $s$ and $t$ in the associated coverage graph are node-disjoint. Then, we can minimize the number of path switches instead of sensor switches.

We now prove *NP-Hardness* for the decision version of the problem of minimizing the number of sensor switches when the threshold is zero. *Barrier Coverage Lifetime With Zero Sensor switches:*
INSTANCE: Integers $L, k \in \mathbb{Z}^+$, location of $n$ sensors each with a sensing radius of $r$, and $\ell_i \in \mathbb{Z}^+$, the lifetime of sensor $i = 1, 2, \ldots, n$.
QUESTION: Can the network provide $k$-barrier coverage for $L$ units of time with 0 sensor switches?

We use a reduction from the 3-partition problem ( [24]). *3-Partition*
INSTANCE: Set $A$ with $3m$ 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 $\sum_{a \in A} s(a) = mB$.
QUESTION: Can $A$ be partitioned into $m$ disjoint sets $A_1, A_2, \ldots, A_m$ such that $\sum_{a \in A_i} s(a) = B$ for $1 \le i \le m$?

*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 invoke the algorithm in [1] to verify for each interval $I_i$ whether the sensors active in this interval provide $k$-barrier coverage. Notice that it is sufficient to check $n$ intervals, since at least one sensor must be exhausted to cause a schedule switch and there are at most $n$ sensors that are 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 Sensor Switches problem is *strongly NP-Complete*, we Provide a reduction from *3-Partition*. Given an instance of the 3-Partition problem, we construct a coverage graph as follows: Set $L = B$ and $k = m$. Create two disjoint sets of $k$ nodes each, called $S$ and $T$, such that $n = 3m + 2k = 5k$. Let $\ell_i = B = L$ for $i \in S \cup T$, and $\ell_i = s(i)$ for $i \in A$. Connect the $k$ nodes in set $S$ to the virtual node $s$ in the coverage graph and to the $3k$ nodes in set $A$. Similarly, connect the $k$ nodes in set $T$ to the virtual node $t$ and to the $3k$ nodes in set $A$. (See Figure 10 for an example.) Now, we show that the network provides $k$-barrier coverage for $L$ units of time with zero sensor switches iff set $A$ can be partitioned into $m$

disjoint sets $A_1, A_2, \ldots, A_m$ such that $\sum_{a \in A_i} s(a) = B$ for $1 \leq i \leq m$.

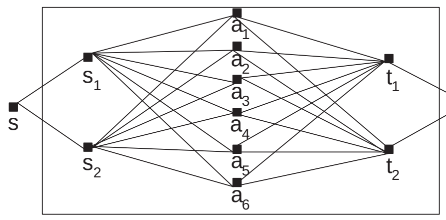

Fig. 10. The coverage graph constructed from an instance of 3-partition when $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$.

$\Rightarrow$. Assume set $A$ can be partitioned in $m$ disjoint sets such that $\sum_{a \in A_i} s(a) = B$ for $1 \leq i \leq m$. Since $B/4 < s(a) < B/2$ for $a \in A$, each set $A_i$ consists of exactly three elements. Label the $k$ nodes in set $S$ as $s_1, s_2, \ldots, s_k$ and label the $k$ nodes in $T$ as $t_1, t_2, \ldots, t_k$. Now, form sets of three paths $P_i$, $1 \leq i \leq k$, that consist of nodes $s_i$, $t_i$, 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.

$\Leftarrow$. 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 $t$ includes one or more nodes from the set $A$, $\sum_{a \in A} s(a) = kL$. Because the network provides $k$ barrier coverage for $L$ units of time, every node in set $A$ fully exhausts its lifetime in $L$ time units.

Suppose 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, switch all the nodes that were used by $s_1$ and $s_2$ prior to this instant. This switch has no effect on the system lifetime. Now, there is no node sharing up to this time instant. Other node sharing can be eliminated in a similar fashion. Hence, we assume that no node in set $A$ is shared by two nodes in set $S$ or by two nodes in set $T$.

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 $\sum_{a \in A_i} \ell_a = L = B$ for $1 \leq i \leq m = k$. Let $A_i$ be the set of nodes in $A$ that are used by node $i \in S$. This generates a 3-partition of set $A$, which completes the proof. ■

We now show that minimizing the number of path switches is NP-Complete even if all paths between the virtual nodes $s$ and $t$ in the underlying coverage graph are node-disjoint. The following is the decision version of this problem when the threshold is zero.

*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 $\ell_i \in \mathbb{Z}^+$, the lifetime of sensor $i = 1, 2, \ldots, n$.
QUESTION: Can the network provide $k$-barrier coverage for $L$ units of time with 0 path switches?

We provide a reduction from the *Partition* problem. *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:* It can be verified that the *Node-Disjoint Barrier Coverage Lifetime With Zero Path Switches* problem is in NP in a way that is similar to 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 $n$ 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 $\{1, 2, \ldots, n\} - S$, where each set provides 1-barrier coverage for $L$ units of time.

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

## 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 the base stations are located at the two ends of the network and can directly reach all sensors located on the respective ends. Therefore, if the sensors providing barrier coverage form a path in the *Communication Graph*[4] between the two ends of the network, then all detection events are communicated to the base stations.

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

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 ends of the network. Each of these disjoint paths provide 1-barrier coverage, implying that the network provides $k$-barrier coverage. 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 use our optimal algorithms to study three interesting issues that may have implication in real-life deployments highlighted in Section I: 1.) On average, how much statistical redundancy exists in optimal random deployments?, 2.) On average, how much loss in potential network lifetime is incurred if homogeneous sensor lifetime is assumed when they are not?, and 3.) What is the impact of imbalance in the lifetimes of individual sensors?

We use the following parameters in the simulations. Sensors are deployed in a rectangular region of dimension $1\text{km}\times200\text{m}$, and each sensor has a sensing range of 50m. These parameters are in same ratio as in Figure 9 in [7]. As a result, the density needed in a random deployment can be readily determined from this figure.

### A. Statistical Redundancy in Random Deployments

Random deployment is often used in simulations. In real-life deployments also, random deployment can provide a reliable estimate of the density of sensors needed to achieve a desired quality of monitoring. This is true even if sensors are deployed deterministically, because randomness is introduced due to unanticipated failures after deployment and due to errors in placement [25]. To compensate for these sources of randomness, several additional sensors need to be deployed as compared to an optimal deterministic deployment. The sensor density in such deterministic deployments is close to that needed in a pure random deployment model (see [25] for details). Randomness, whether due to the randomness in deployment or due to unanticipated failures and errors in placement, can be compensated by increasing the sensor density.

The appropriate density that is needed to guarantee a desired probability of coverage is known [7], [25], [26]. As can be seen in Figure 9 of [7] where density estimates for barrier coverage are presented, these estimates are reliable; they closely match the behavior observed in experiments. The density needed to achieve barrier coverage with probability 0.9 is 4.1, which translates to 82 sensors. With these many sensors there is 1 out of 10 chance of not having barrier coverage. To achieve barrier coverage with probability 0.99, the density is 5.6, and 112 sensors are needed. For barrier coverage with probability 0.999, 140 sensors are sufficient (with a density of 7). With these many sensors there is only 1 out of 1000 chance of not having barrier coverage. Each of these densities are optimal in the sense that they provide the desired probability of coverage with a minimal number of sensors.

Figure 11 shows the lifetime enhancement achieved. With 82 sensors, it may be possible to achieve 3 units of lifetime when only a unit of lifetime is planned. On average, the network has a lifetime of 1.21 units (an enhancement by 21%). For 112 sensors, up to 5 units of lifetime can be achieved, with an average of 2.68 (an enhancement by 168%). For 140 sensors, up to 7 units of lifetime can be achieved, with an average of 4.19 (an enhancement by 319%). Thus, in instances where barrier coverage is provided, there exist sufficient redundancies to get several hundred percentage enhancements in lifetime. These numbers could not be obtained previously because optimal algorithms for lifetime enhancement were not available.

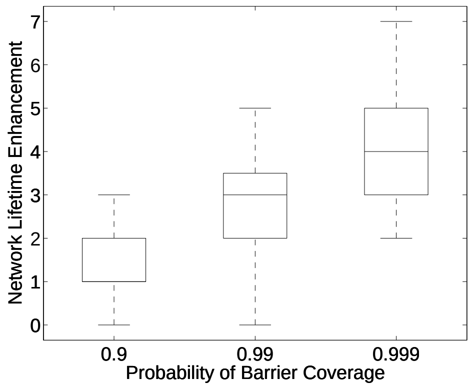

Fig. 11. Sensors are deployed randomly but non-redundantly to provide 1-barrier coverage with probabilities of 0.9, 0.99, and 0.999. The network is planned to last for 1 unit time. For each case, lifetime increase by using the *Stint* algorithm is shown for 100 instances of deployment.

### B. The Cost of Homogeneous Assumption

As stated in Section I, the lifetime of individual sensors are rarely equal. Even if they start with the same lifetime, the remaining lifetime of sensors cease to be homogeneous due to load imbalance, unanticipated failures, etc. However, for simplicity and/or tractability, sensor lifetimes are often assumed homogeneous. We now study the potential loss in network lifetime due to the assumption of homogeneity. We consider individual sensor lifetimes distributed between $5 - i$ and $5 + i$ for $i = 0, 1, 2, 3, 4, 5$. For each of these six cases, we use Stint algorithm to maximize the network lifetime, assuming the lifetimes to be homogeneous. In this case, a path of sensors provides barrier coverage only as long as all the sensors have positive energy remaining. We then use Prahari algorithm to again maximize the network lifetime, but this time taking into account actual heterogeneous lifetimes of individual nodes. The results appear in Figure 12. We notice that lifetime obtained with the assumption of homogeneous lifetime decreases as the degree of heterogeneity in sensor lifetimes increases. When the lifetimes are distributed between

0 and 10, the lifetime reduces by 65%. As a result, the network lasts three times longer if the sleep-wakeup scheduling algorithm takes heterogeneity of sensor lifetimes into account.

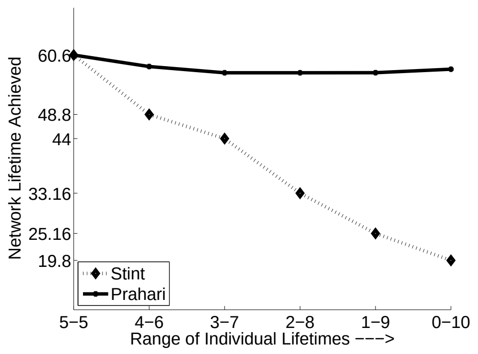

Fig. 12. Network lifetimes obtained using the Stint (that assumes homogeneous lifetimes) and Prahari algorithms.

## C. Effect of Energy Imbalance

Load on individual sensors are rarely identical. Consequently, the lifetime of individual sensors, even if they have the same battery pack to begin with, is unlikely to be the same. One method pursued in the literature to optimize the network lifetime is to balance the load in the network. However, we observe that imbalance in individual sensor lifetimes does not necessarily translate into loss in network lifetime. In fact, as we see from the network lifetime achieved for the Prahari algorithm in Figure 12, the overall network lifetimes remain approximately the same even when the degree of imbalance is high (when sensor lifetimes are uniformly distributed between 0 and 10). This leads us to conclude that if the load imbalance is sufficiently random, it does not cause significant loss in network lifetime. Hence, balancing the load to maximize network lifetime may not be necessary.

## 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 these algorithms generate solutions where the network lasts up to seven times longer even if a minimal number of sensors have been deployed in a random deployment. We also show that the loss in potential network lifetime is severe (reduced by two-thirds) if sensor lifetimes are assumed to be homogeneous when they are not. Finally, we show that imbalance in load in a network does not cause loss in network lifetime, as previously assumed, provided the imbalance is sufficiently random.

Prior to this work, the problem of sleep-wakeup was considered to be NP-Hard. Now that the sleep-wakeup problem has been solved in polynomial time for the barrier coverage model, new research is likely to investigate the tractability of this problem for other coverage models.

## Acknowledgment

This work was partly sponsored by National Science Foundation Grants CCF-0728928, CNS-0721983, CNS-0721817, CNS-CNS-0721434, and National Institutes of Health (NIH) Grant U01DA023812 from National Institute for Drug Abuse (NIDA). The content is solely the responsibility of the authors and does not necessarily represent the official views of the sponsors.

## References

[1] 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.

[2] 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.

[3] 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.

[4] Joengmin Hwang, Tian He, and Yongdae Kim, "Exploring in-situ sensing irregularity in wireless sensor networks," in *Fifth ACM Conference on Embedded Networked Sensor Systems (SenSys)*, New York, NY, 2007.

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

[6] Jonathan W. Hui and David Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," in *ACM Conference on Ebmedded Networked Sensor Systems (Sensys)*, 2004.

[7] Paul Balister, Béla Bollobás, Amites Sarkar, and Santosh Kumar, "Reliable density estimates for coverage and connectivity in thin strips of finite length," in *13th Annual ACM international conference on Mobile computing and networking (MobiCom)*, Montreal, Canada, 2007, pp. 75–86.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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 Wireless, and Peer-to-Peer Networks*, 2004.

[13] Ai Chen, Santosh Kumar, and Ten H. Lai, "Designing localized algorithms for barrier coverage," in *13th annual ACM international conference on Mobile computing and networking (MobiCom)*, Montreal, Canada, 2007, pp. 63–74.

[14] B. Liu, O. Dousse, J. Wang, and A. Saipulla, "Strong barrier coverage of wireless sensor networks," in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2008, pp. 411–420.

[15] K.-F. Ssu, W.-T. Wang, F.-K. Wu, and T.-T. Wu, "$k$-barrier coverage with a directional sensing model," *International Journal on Smart Sensing and Intelligent Systems*, vol. 2, no. 1, 2009.

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

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

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

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

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] Paul Balister and Santosh Kumar, "Random vs. Deterministic Deployment of Sensors in the Presence of Failures and Placement Errors," in *IEEE INFOCOM Miniconference*, Rio De Janeiro, Brazil, 2009.

[26] Paul Balister, Zizhan Zheng, Santosh Kumar, and Prasun Sinha, "Trap Coverage: Allowing Coverage Holed of Bounded Diameter in Wireless Sensor Networks," in *IEEE INFOCOM*, Rio De Janeiro, Brazil, 2009.

**Santosh Kumar** is an Assistant Professor of Computer Science at the University of Memphis, where he received an Early Career Research Award from the College of Arts and Sciences in 2008. He received his Ph.D. in Computer Science and Engineering from the Ohio State University in 2006, where his dissertation work won him SBC Presidential Fellowship. He is leading several multidisciplinary research projects in wireless sensor networks involving more than twenty faculty members spread across eight universities, whose expertise span nine disciplines. More information about his current and past projects is available at his homepage.

**Ten H. Lai** is a Professor of Computer Science and Engineering at the Ohio State University. He is interested in applying Zen to teaching and research. He served as program chair of ICPP'98, general chair of ICPP'00, program co-chair of ICDCS'04, general chair of ICDCS'05, and recently, general co-chair of ICPP'07. He is/was an editor of IEEE Transactions on Parallel and Distributed Systems, ACM/Springer Wireless Networks, Academia Sinica's Journal of Information Science and Engineering, International Journal of Sensor Networks, and International Journal of Ad Hoc and Ubiquitous Computing.

**Marc E Posner** is a Professor of Operations Research in the Integrated Systems Engineering Department at The Ohio State University. He received a B.A. in Mathematics from Brandeis University, an M.S. and Ph.D. in Operations Research from the University of Pennsylvania. He has published in most of the major industrial engineering and operations research journals on a variety of topics ranging from the construction of statistical decision rules to decomposing nonlinear programming problems. His research is primarily in the field of deterministic optimization with an emphasis on integer programming. He is interested both in heuristic and exact methods. An area of focus is on scheduling and production problems. Currently, he is the Area Editor in Scheduling and Logistics for IIE Transactions and is an Associate Editor for Naval Research Logistics.

**Prasun Sinha** received his PhD from University of Illinois, Urbana-Champaign in 2001, MS from Michigan State University in 1997, and B. Tech. from IIT Delhi in 1995. Currently he is an Associate Professor in the Department of Computer Science and Engineering at Ohio State University. From 2001 to 2003, he was a Member of Technical Staff at Bell Labs, in Holmdel, New Jersey. His research focuses on ubiquitous networking. He has served as the TPC chair for ICST QShine 2009 and is the TPC co-chair for BROADNETS 2010. He has won several awards including Lumley Research Award (OSU, 2009), CAREER award (NSF, 2006) Ray Ozzie Fellowship (UIUC, 2000), Mavis Memorial Scholarship (UIUC, 1999), and Distinguished Academic Achievement Award (MSU, 1997).