Chapter 2

Maximum Lifetime Scheduling in Wireless Sensor Networks

Akshaye Dhawan

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/48575

1. Introduction

Wireless sensor networks (WSNs) have attracted a lot of recent research interest due to their applicability in security, monitoring, disaster relief and environmental applications. WSNs consist of a number of low-cost sensors scattered in a geographical area of interest and connected by a wireless RF interface. Sensors gather information about the monitored area and send this information to gateway nodes. The radio on board these sensor nodes has limited range and allows the node to transmit over short distances. In most deployment scenarios, it is not possible for each node to communicate directly to the sink and hence, the model of communication is to transmit over short distances to other peers in the direction of the sink nodes.

In order to keep their cost low, the sensors are equipped with limited energy and computational resources. The energy supply is typically in the form of a battery and once the battery is exhausted, the sensor is considered to be dead. The nodes also have limited memory and processing capabilities. Hence, harnessing the potential of these networks involves tackling a myriad of different issues from algorithms for network operation, programming models, architecture and hardware to more traditional networking issues. For a more detailed survey on the various computational research aspects of Wireless Sensor Networks, see the survey papers [2, 13, 24, 37, 39], or the more recent books [23, 28] and a special issue of the CACM [14].

This section focuses on the algorithmic aspects of Wireless Sensor Networks. Specifically, we look at the problem of covering a set of targets or an area for the longest duration possible. The next section focuses on a more detailed discussion of the problem and provides a formal statement for it. It is worth mentioning that there is an abundance of algorithmic research related to WSNs. A lot of this focuses on traditional distributed computing issues like localization, fault tolerance, robustness. This naturally raises the interesting question of how different are WSNs as a computational model than more traditional distributed computing environments or even ad-hoc networks? This question has been explored briefly in [43].

2. Coverage problems

Many intended applications of Wireless Sensor Networks involve having the network monitor a region or a set of targets. To ensure that the area or targets of interest can be covered, sensors are usually deployed in large numbers by randomly dropping them in this region. Deployment is usually done by flying an aircraft over the region and air dropping the sensors. Since the cost of deployment far exceeds the cost of individual sensors, many more sensors are dropped than needed to minimally cover the region. The leads to a very dense network and gives rise to an overlap in the monitoring regions of individual sensors.

A simplistic approach to meet the coverage objective would be to turn on all sensors after deployment. But this needlessly reduces the *lifetime* of the network since the overlap between monitoring regions implies that not all sensors need to be on at the same time. This can also lead to a very lossy network with several collisions happening in the medium access control (MAC) layer due to the density of nodes. In order to extend the lifetime of a sensor network while maintaining coverage, a minimal subset of the deployed sensors are kept active while the other sensors can sleep. Through some form of scheduling, this active subset changes over time until there are no more such subsets available to satisfy the coverage goal. In using such a scheme to extend the lifetime, the problem is two fold. First, we need to select these minimal subsets of sensors. Then there is the problem of *scheduling* them wherein, we need to determine how long to use a given set and which set to use next. For an arbitrarily large network, there are exponential number of possible subsets making the problem intractable and it has been shown to be NP-complete in [6, 20].

Centralized solutions like those in [6, 41] are based on assuming that the entire network structure is known at one node (typically the gateway node), which then computes the schedule for the network. The schedule is computed using *linear programming* based algorithms. Like any centralized scheme, it suffers from the problems of scalability, single point of failure and lack of robustness. The latter is particularly relevant in the context of sensor networks since sensor nodes are deployed in hostile environments and are prone to frequent failures.

Existing distributed solutions in [4, 5, 42] work by having a sensor exchange information with its neighbors (limited to k-hops). These algorithms use information like targets covered and battery available at each sensor to greedily decide which sensors remain on. Distributed algorithms are organized into rounds so that the set of active sensors is periodically reshuffled at the beginning of each round. The problem with these algorithms is that they use simple greedy criteria to make their decision on which sensors become active at each round and thus, do not efficiently take into account the problem structure.

3. Problem statement

The lifetime problem can be stated as follows. Given a monitored region R, a set of sensors S and a set of targets T, find a monitoring schedule for these sensors such that

- the total time of the schedule is maximized,
- all targets are constantly monitored, and
- no sensor is in the schedule for longer than its initially battery.

A related problem is that of monitoring an area of interest. In general, the area and target coverage problems have been shown to be equivalent. [3, 10, 41] provide ways to map an area to a set of points (targets) . In the work presented in the remainder of this dissertation, we focus on the target coverage problem with the implicit understanding that the algorithms and techniques presented can be translated to the area coverage problem by mapping the area to a set of points (virtial targets) with an appropriate granularity.

There are also several other variations of this basic problem. For example the p% coverage problem [30] requires only a certain percentage of all targets to be covered. The fault tolerant k-coverage version of this problem requires each target to be covered by at least k sensors [27, 47]. Also, the basic problem has been modified to include sensors that have adjustable sensing ranges, non uniform sensing shapes and othe heterogeneous sensor network models.

4. Related work

In this section, we briefly survey existing approaches to maximizing the lifetime of sensor networks, while meeting certain coverage objectives. [9] gives a more detailed survey on the various coverage problems and the scheduling mechanisms they use. [38] also surveys the coverage problem along with other algorithmic problems relevant to sensor networks. We end this section by focusing on two algorithms, LBP [4] and DEEPS [5], since we use them for comparisons against our algorithms.

A key application of wireless sensor networks is the collection of data for reporting. There are two types of data reporting scenarios: event-driven and on-demand [10]. Event-driven reporting occurs when one or more sensor nodes detect an event and report it to the sink. In on-demand reporting, the sink initiates a query and the nodes respond with data to this query.

Coverage problems essentially state how well a network observes its physical space. As pointed out in [32], coverage is a measure of the quality of service (QoS) for the WSN. The goal is to have each point of interest monitored by at least one sensor at all times. In some applications, it may be a requirement to have more than one sensor monitor a target for achieving fault tolerance. Typically, nodes are randomly deployed in the region of interest because sensor placement is infeasible. This means that more sensors are deployed than needed to compensate for the lack of exact positioning and to improve fault tolerance in harsh environments. The question of placing an optimal number of sensors in a deterministic deployment has been looked at in [17, 26, 34]. However, in this dissertation we focus on networks with very dense deployment of sensors so that there is significant overlap in the targets each sensor monitors. This overlap will be exploited to schedule sensors into a low power sleep state so as to improve the lifetime of these networks. Note that this definition of the network lifetime is different from some other definitions which measure this in terms of number of operations the network can perform [22].

The reason for wanting to schedule sensors into sense-sleep cycles that we talked about in Section 1, stems from the fact that sensor nodes have four states - *transmit, receive, idle and sleep*. As shown in [36] for the WINS Rockwell sensor, the transmit, receive and idle states all consume much more power than the sleep state - hence, it is more desirable for a sensor to enter a sleep state to conserve its energy. The goal behind sensor scheduling algorithms is to select the activity state of each sensor so as to allow the network as a whole to monitor its points of interest for as long as possible. For a more detailed look at power consumption

Name	Area/Target	Disjoint	Main Idea
Abrams, Goel [1]	Area	Yes	Greedy: Max uncovered area
Meguerdichian [33]	Area	No	Integer Linear Program
Cardei [6]	Target	Yes	Mixed Integer Programming
Shah [3]	Area	Yes	LP, Garg Könemann
Cardei [8]	Target	No	Integer Linear Program

Table 1. Centralized Algorithms

models for ad-hoc and sensor networks we refer the reader to [18, 19, 25]. We now look at coverage problems in more detail.

The maximum lifetime coverage problem has been shown to be NP-complete in [1, 6]. Initial approaches to the problem in [1, 6, 41] considered the problem of finding the maximum number of *disjoint* cover sets of sensors. This allowed each cover to be used independently of others. However, [3, 8] and others showed that using non-disjoint covers allows the lifetime to be extended further and this approach has been adopted since.

Broadly speaking, the existing work in this category can be classified into two parts - Centralized Algorithms and Distributed Algorithms. For centralized approaches, the assumption is that a single node (usually the base station) has access to the entire network information and can use this to compute a schedule that is then uploaded to individual nodes. Distributed Algorithms work on the premise that a sensor can exchange information with its neighbors within a fixed number of hops and use this to make scheduling decisions. We now look at the individual algorithms in both these areas.

A common approach taken with centralized algorithms is that of formulating the problem as an optimization problem and using linear programming (LP) to solve it [3, 6, 16, 33]. In [41], the authors develop a most-constrained least-constraining heuristic and demonstrated its effectiveness on variety of simulated scenarios. In this heuristic, the main idea is to minimize the coverage of sparsely covered areas within one cover. Such areas are identified using the notion of the critical element, defined as the element which is a member of the smallest number of sets. Their heuristic favors sets that cover a high number of uncovered elements, that cover more sparsely covered elements, that do not cover the area redundantly and that redundantly cover the elements that do not belong to sparsely covered areas. [33] is a followup work by the same authors in which they formulate the area coverage problem using a Integer LP and relax it to obtain a solution. They also presented several ILP based formulations and strategies to reduce overall energy consumption while maintaining guaranteed sensor coverage levels. Additionally, their work demonstrated the practicality and effectiveness of these formulations on a variety of examples and provided comparisons with several alternative strategies. They also show that the ILP based technique can scale to large and dense networks with hundreds of sensor nodes.

In order to solve the target coverage problem, [6] considers the disjoint cover set approach. Modeling their solution as a Mixed Integer Program shows an improvement over [41]. The authors define the disjoint set covers (DSC) problem and prove its NP-completeness. They also prove that any polynomial-time approximation algorithm for DSC problem has a lower bound of 2. They first transform DSC into a maximum-flow problem (MFP), which is then formulated as a mixed integer programming. Based on the solution of the MIP, the authors

design a heuristic to compute the number of covers. They evaluate the performance by simulation, against the most constrainedï£¡minimally constraining heuristic proposed in [41] and found that their heuristics has a larger number of covers (larger lifetime) at the cost of a greater running time.

[3] formulates a packing LP for the coverage problem. Using the $(1+\epsilon)$ Garg-Könemann approximation algorithm [21], they provide a $(1+\epsilon)(1+2lnn)$ approximation of the problem. They also present an efficient data structureto represent the monitored area with at most n^2 points guaranteeing the full coverage which is superior to the previously used approach based on grid points in [41]. They also present distributed algorithms that tradeoff between monitoring and power consumption but these are improved upon by the authors in LBP and DEEPS.

A similar problem is solved by us for sensors with *adjustable* ranges in [16]. We present a linear programming based formulation that also uses the $(1+\epsilon)$ Garg-Könemann approximation algorithm [21]. The main difference is the introduction of an adjustable range model that allows sensors to vary their sensing and communication ranges smoothly. This was the first model that allows sensors to vary their range to any value upto a maximum. The model is an accurate representation of physical sensors and allows significant power savings over the discreetly varying adjustable model.

A different algorithm to work with disjoint sets is given in [7]. Disjoint cover sets are constructed using a graph coloring based algorithm that has area coverage lapses of about 5%. The goal of their heuristic is to achieve energy savings by organizing the network into a maximum number of disjoint dominating sets that are activated successively. The heuristic to compute the disjoint dominating sets is based on graph coloring. Simulation studies are carried out for networks of large sizes.

[1] also gives a centralized greedy algorithm that picks sensors based the largest uncovered area. They have designed three approximation algorithms for a variation of the SET K-COVER problem, where the objective is to partition the sensors into covers such that the number of covers that include an area, summed over all areas, is maximized. The first algorithm is randomized and partitions the sensors within a fraction of the optimum. The other two algorithms are a distributed greedy algorithm and a centralized greedy algorithm. The approximation ratios are presented for each of these algorithms.

[8] also deal with the target coverage problems. Like similar algorithms, they also extend the sensor network life time by organizing the sensors into a maximal number of set covers that are activated successively. But they allow non-disjoint set covers. The authors model the solution as the maximum set covers problem and design two heuristics that efficiently compute the sets, using linear programming and a greedy approach. The greedy algorithm selects a critical target at each step. This is the least covered target. For the greedy selection step, the sensor with the greatest contribution to the critical target is selected.

The distributed algorithms in the literature can be further classified into greedy, randomized and other techniques. The greedy algorithms [1, 4, 5, 8, 29, 41] all share the common property of picking the set of active sensors greedily based on some criteria. [41] considers the area coverage problem and introduces the notion of a *field* as the set of points that are covered by the same set of sensors. The basic approach behind the picking of a sensor is to first pick the one that covers that largest number of previously uncovered fields and to then avoid including

Name	Area/Target	Disjoint	Main Idea
Sliepcivic [41]	Area	Yes	Greedy: Max uncovered fields
Tian [42]	Area	No	Geometric calculation of sponsored area
PEAS [45]	Area	No	Probing based determination of sponsored area
CCP [44]	Area	No	Random timers to evaluate coverage requirements
OGDC [46]	Area	No	Random back off node volunteering
Lu [29]	Area	No	Highest overall denomination sensor picks
Abrams [1]	Area	Yes	Randomized, Greedy picks max uncovered area
Cardei et al. [8]	Target	No	Sensor with highest contribution to bottleneck
LBP [4]	Target	No	Targets are covered by higher energy nodes
DEEPS [5]	Target	No	Minimize energy consumption for bottleneck target

Table 2. Distributed Algorithms

more than one sensor that covers a sparsely covered field. [1] builds on this work and presents three algorithms that solve variations of the set k-cover problem. The greedy heuristic they propose works by selecting the sensor that covers the largest uncovered area. [29] defines the sensing denomination (SD) of a sensor as its contribution, i.e., the area left uncovered when the sensor is removed. The authors assume that each sensor can probabilistically detect a nearby event, and build a probabilistic model of network coverage by considering the data correlation among neighboring sensors. The more the contribution of a sensor to the network coverage, the higher the sensorï£¡s SD is. Based on the location information of neighboring sensors, each sensor can calculate its SD value in a distributed manner. Sensors with higher sensing denomination have a higher probability of remaining active.

[3] gives a distributed algorithm based on using the faces of the graph. If all the faces that a sensor covers are covered by other sensors with higher battery that are in an active or deciding state, then a sensor can switch off (sleep). Their work has been extended to target coverage in the load balancing protocol (LBP).

Some distributed algorithms use randomized techniques. Both OGDC [46] and CCP [44] deal with the problem of integrating coverage and connectivity. They show that if the communication range is at least twice the sensing range, a covered network is also connected. [46] uses a random back off for each node to make nodes volunteer to be the start node. OGDC addresses the issues of maintaining sensing coverage and connectivity by keeping a minimum number of sensor nodes in the active mode in wireless sensor networks. They investigate the relationship between coverage and connectivity. They also derive, under the ideal case in which node density is sufficiently high, a set of optimality conditions under which a subset of working sensor nodes can be chosen for complete coverage. OGDC algorithm is fully localized and can maintain coverage as well as connectivity, regardless of the relationship

between the radio range and the sensing range. OGDC achieves similar coverage with an upto 50% improvement in the lifetime of the network. A drawback of OGDC is that it requires that each node knows its own location.

In [31] the authors combine computational geometry with graph theoretic techniques. The use Voronoi diagrams with graph search to design a polynomial time worst and average case algorithm for coverage calculation in homogeneous isotropic sensors. The also analyze and experiment with using these techniques as heuristics to improve coverage.

[44] sets a random timer for each node following which a node evaluates its current state based on the coverage by its neighbors. The authors present a Coverage Configuration Protocol (CCP) that can provide different degrees of coverage requested by applications. This flexibility allows the network to self-configure for a wide range of applications. They also integrate CCP to SPAN [11, 12] to provide both coverage and connectivity guarantees. [1] also present a randomized algorithm that assigns a sensor to a cover chosen uniformly at random.

A different approach has been taken in PEAS [42, 45]. PEAS is a distributed algorithm with a probing based off-duty rule is given in [45]. PEAS is localized and has a high resilience to node failure and topology changes. Here, every sensor broadcasts a probe PRB packet with a probing range γ. Any working node that hears this probe packet responds. If a sensor receives at least one reply, it can go to sleep. The range can be chosen based on several criteria. Note that this algorithm does not preserve coverage over the original area. The results for PEAS showed an increase in the network lifetime in linear proportion to the number of deployed nodes. In [42] the authors give a distributed and localized algorithm. Every sensor has an off-duty eligibility rule. They give an algorithm for a node to compute its sponsored area. To prevent the occurrence of blind-points by having two sensors switch off at the same time, a random back off is used. They show improved performance over PEAS.

To our knowledge, [44] was the first work to consider the k-coverage problem. [27] also addresses the k-coverage problem from the perspective of choosing enough sensors to ensure coverage. Authors consider different deployments with sensors given a probability of being active and obtain bounds for deployment. [47] solves the problem of picking minimum size connected k-covers. The authors state this as an optimization problem and design a centralized approximation algorithm that delivers a near-optimal solution. They also present a communication-efficient localized distributed algorithm for this problem.

Now, we look at the two protocols that we compare our heuristics against. The load balancing protocol (LBP) [4] is a simple 1-hop protocol which works by attempting to balance the load between sensors. Sensors can be in one of three states sense/on, sleep/off or vulnerable/undecided. Initially all sensors are vulnerable and broadcast their battery levels along with information on which targets they cover. Based on this, a sensor decides to switch to off state if its targets are covered by a higher energy sensor in either on or vulnerable state. On the other hand, it remains on if it is the sole sensor covering a target. This is an extension of the work in [3]. LBP is simplistic and attempts to share the load evenly between sensors instead of balancing the energy for sensors covering a specific target.

The other protocol we consider is DEEPS [5]. The maximum duration that a target can be covered is the sum of the batteries of all its nearby sensors that can cover it and is known as the life of a target. The main intuition behind DEEPS is to try to minimize the energy consumption rate around those targets with smaller lives. A sensor thus has several targets with varying

lives. A target is defined as a *sink* if it is the shortest-life target for at least one sensor covering that target. Otherwise, it is a *hill*. To guard against leaving a target uncovered during a shuffle, each target is assigned an in-charge sensor. For each sink, its in-charge sensor is the one with the largest battery for which this is the shortest-life target. For a hill target, its in-charge is that neighboring sensor whose shortest-life target has the longest life. An in-charge sensor does not switch off unless its targets are covered by someone. Apart from this, the rules are identical as those in LBP protocol. DEEPS relies on two-hop information to make these decisions.

5. The lifetime dependency graph model

In this section, we introduce the Lifetime Dependency (LD) Graph as a model for the maximum lifetime coverage problem defined in Section 3. This model is a key contribution of this dissertation since the heuristics and algorithms that follow in subsequent sections rely heavily on the LD Graph.

Recall from Section 3 that given a sensor network and a set of static targets, the maximum lifetime sensor scheduling problem is to select a subset of sensors that covers all targets and then periodically shuffle the members of this subset so as to maximize the total time for which the network can cover all targets.

Since these sensors are powered by batteries, energy is a key constraint for these networks. Once the battery has been exhausted, the sensor is considered to be dead. The lifetime of the network is defined as the amount of time that the network can satisfy its coverage objective, i.e., the amount of time that the network can cover its *area* or *targets* of interest. Having all the sensors remain "on" would ensure coverage but this would also significantly reduce the lifetime of the network as the nodes would discharge quickly. A standard approach taken to maximizing the lifetime is to make use of the overlap in the sensing regions of individual sensors caused by the high densit y of deployment. Hence, only a subset of all sensors need to be in the "on" or "sense" state, while the other sensors can enter a low power "sleep" or "off" state. The members of this active set, also known as a *cover* set, are then periodically updated so as to keep the network alive for longer duration. In using such a scheduling scheme, there are two problems that need to be addressed. First, we need to determine how long to use a given cover set and then we need to decide which set to use next. This problem has been shown to be NP-complete [1, 6].

A key problem here is that since a sensor can be a part of multiple covers, these covers have an impact on each other, as using one cover set reduces the lifetime of another set that has sensors common with it. By making greedy choices, the impact of this dependency is not being considered, since none of the heuristics in the literature study this reduction in the lifetime of other cover sets caused by using a sensor that is a member of several such sets. The earlier disjoint formulations mentioned in the previous section, entirely avoided this problem by preventing it.

We capture this dependency between covers by introducing the concept of a local Lifetime Dependency (LD) Graph. This consists of the cover sets as nodes with any two nodes connected if the corresponding covers intersect. The graph is an example of an intersection graph since it represents the sensors common to different cover sets. By looking at the graph locally (fixed 1-2 hop neighbors), we are able to construct all the local covers for the local targets and then model their dependencies. Based on these dependencies, a sensor can then

prioritize its covers and negotiate these with its neighbors. We also present some simple heuristics based on the graph. The material presented in this section was published in [35].

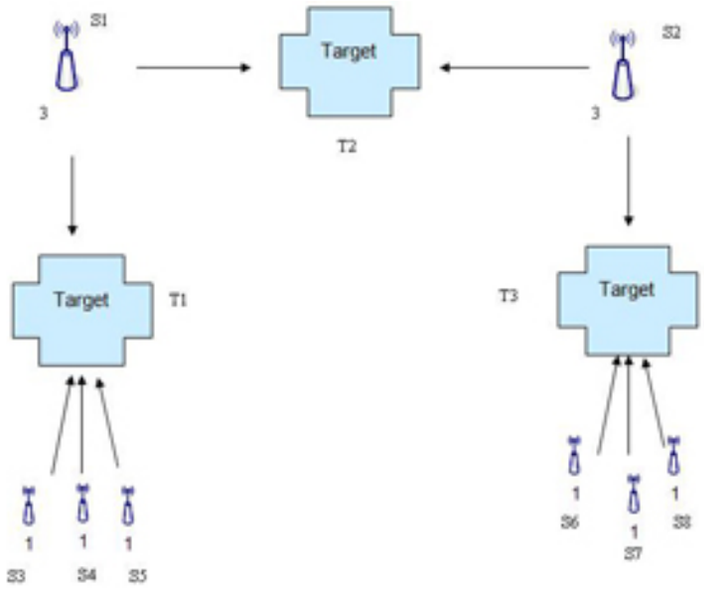

Figure 1. A sensor network

6. Symbols and definitions

Let us begin with a few basic conventions and definitions that will be used in the rest of this dissertation. Individual chapters will introduce additional notation as and when necessary. The notation presented here applies to the basic LD grapg model and will be utilized for all the chapters that follow.

We will use s_1, s_2, etc., to represent sensors, t_1, t_2, etc., to represent targets, and C, C', etc., to denote covers.

Let us assume we have n sensors and m targets, both stationary.

Consider the sensor network in Figure 1 with $n = 8, s = \{s_1, s_2, ..., s_8\}$ and $m = 3$ targets, t_1, t_2, and t_3.

We will employ the following definitions, illustrated using this network.

- $b(s)$: strength of the battery of sensor s; for example, $b(s_1) = 3$ while $b(s_3) = 1$.
- $T(s)$: set of targets that sensor s can sense; e.g., $T(s_1) = \{t_1, t_2\}$;
- $N(s, k)$: closed set of neighbors of sensor s at no more than k hops (i.e, those neighbors that s can communicate with using $\leq k$ hops) - this contains s itself; thus, $N(s_1, 1) = \{s_1, s_2, s_3, s_4, s_5\}$.
- Cover: C is a cover for targets in set T if
  (i) for each target $t \in T$ there is at least one sensor in C which can sense t and
  (ii) C is minimal. For example, the possible (minimal) covers for the two targets of s_1 are $\{s_1\}, \{s_2, s_3\}, \{s_2, s_4\}$ and $\{s_2, s_5\}$. There are other non-minimal covers as well such as s_1, s_2

which need to be avoided. Likewise, the possible covers for the only target of sensor s_3 are $\{s_1\}, \{s_3\}, \{s_4\}$ and $\{s_5\}$.

- $lt(C) = min_{s \in C} b(s)$, the maximum lifetime of a cover. The bottleneck sensor of the cover $\{s_2, s_3\}$ is s_3 with the weakest battery of 1. Therefore, $lt(\{s_2, s_3\}) = 1$.

An optimal lifetime schedule of length 6 for this network is $(\{s_1, s_6\}, 1)$, $(\{s_1, s_7\}, 1)$, $(\{s_1, s_8\}, 1)$, $(\{s_2, s_3\}, 1)$, $(\{s_2, s_4\}, 1)$, $(\{s_2, s_5\}, 1))$ where each tuple is a cover for the entire network followed by its duration.

7. Lifetime dependency (LD) graph

Let the local lifetime dependency graph be $G = (V, E)$ where nodes in V denote the local covers and edges in E exist between those pairs of nodes whose corresponding covers share one or more common sensors. For simplicity of reference, we will not distinguish between a cover C and the node representing it, and an edge e between two intersecting covers C and C' and the intersection set $C \bigcap C'$. Each sensor constructs its local LD graph considering its one- or two-hop neighbors and the corresponding targets. Figure 2 shows the local lifetime dependency graph of sensor s_1 in the example network of Figure 1, considering its one-hop neighbors $N(s_1, 1)$ and its targets $T(s_1)$.

In the LD graph, we will use the following two definitions:

- $w(e) = min_{s \in e} b(s)$, the weight of an edge e (if e does not exist, i.e., if e is empty, then $w(e)$ is zero).
- $d(C) = \sum_{e \in E \text{ and incident to } C} w(e)$, the degree of a cover C.

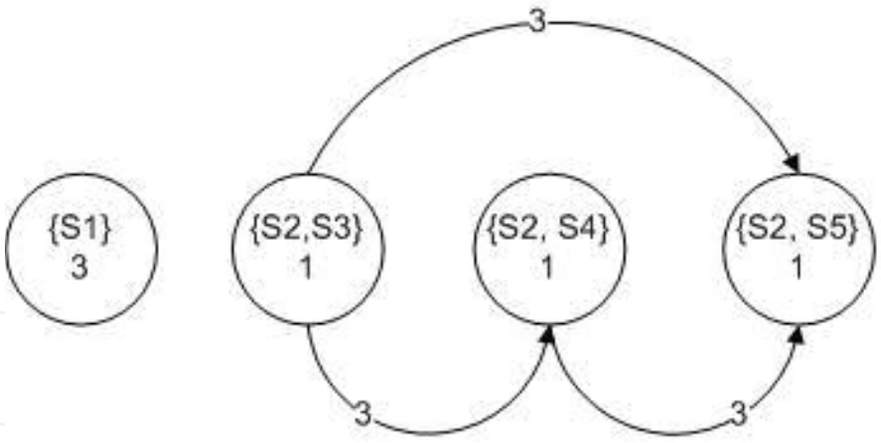

Figure 2. The local lifetime dependency graph of sensor s_1

In Figure 2, the two local covers $\{s_2, s_3\}$ and $\{s_2, s_4\}$ for the targets of sensor s_1 have s_2 in common, therefore the edge between the two covers is $\{s_2\}$ and $w(\{s_2\}) = 3$. Therefore,s_2's battery of 3 is an upper bound on the lifetime of the two covers collectively. It just so happens that the individual lifetimes of these covers are each 1 due to their bottleneck sensors and, therefore, a tighter upper bound on their total life is 2. In general, given two covers C and C', a tight upper bound on the life of two covers is $min(lt(C) + lt(C'), w(C \bigcap C'))$.

8. The basic algorithm

For the purpose of this explanation, without loss of generality, let us assume that the covers are constructed over one-hop neighbors. The algorithm consists of two phases. During the initial setup phase, each sensor calculates and prioritizes the covers. Then, for each reshuffle round of predetermined duration, each sensor decides its on/off status at the beginning, and then those chosen remain on for the rest of the duration.

Initial setup: Each sensor s communicates with each of its neighbor $s' \in N(s,1)$ exchanging mutual locations, battery levels $b(s)$ and $b(s')$, and the targets covered $T(s)$ and $T(s')$. Then it finds all the local covers using the sensors in $N(s,1)$ for the target set being considered. The latter can be solely $T(s)$ or could also include $T(s')$ for all $s' \in N(s,1)$. It then constructs the local LD graph $G = (V,E)$ over those covers, and calculates the degree $d(C)$ of each cover $C \in V$ in the graph G.

The "priority function" of a cover is based on its degree (lower the better). Ties among covers with same degree are broken first by preferring (i) those with longer lifetimes, then (ii) those which have fewer remaining sensors to be turned on, and finally (iii) by choosing the cover containing the smaller sensor id. A cover which has a sensor turned off becomes infeasible and falls out of contention. Also, a cover whose lifetime falls below the duration of a round is taken out of contention, unless it is the only cover remaining.

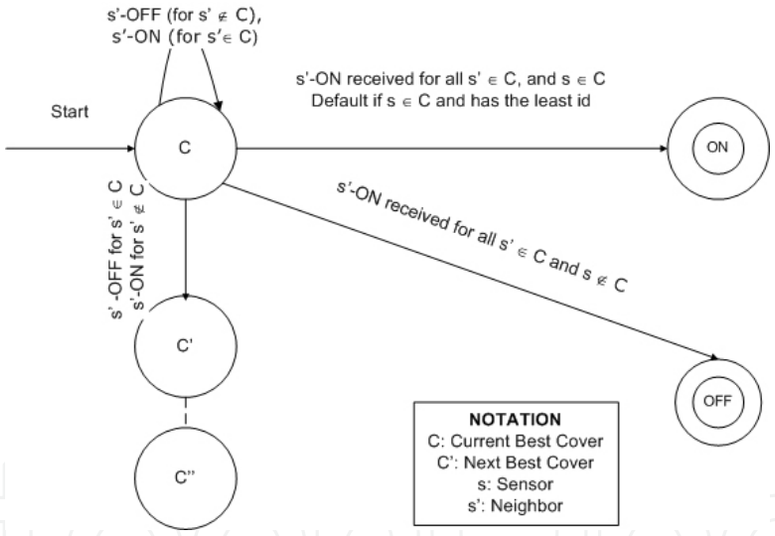

Figure 3. The state transitions to decide the On-Off Status

Reshuffle rounds: The automaton in Figure 3 captures the algorithm for this phase. A sensor s starts with its highest priority cover C as its most desirable configuration for its neighborhood. If successful, the end result would be switching on all the sensors in C, while others can sleep. Else, it transitions to the next best priority cover C', C'', etc., until a cover gets satisfied. The transitions are as follows.

- Continue with the best cover C: Sensor s continues with its current best cover C if its neighbor $s' \notin C$ goes off (thus not impacting the chances of ultimately satisfying C) or if neighbor $s' \in C$ becomes on (thus improving chances for C).

- To on/sense status: If all the neighboring sensors in cover C except s become on, s switches itself on satisfying the cover C for its neighborhood, and sends its on-status to its neighbors.

- To off/sleep status: If all the neighboring sensors in cover C become on thus satisfying C, and s itself is not in cover C, s switches itself off, and sends its off-status to its neighbors.

- Transition to the next best cover C': Sensor s transitions to the next best priority cover C', if (i) C becomes infeasible because a neighboring sensor $s' \in C$ has turned off, or (ii) priority of C is now lower because a sensor $s' \notin C$ has turned on causing another cover C', with same degree and lifetime as C, with fewer sensors remaining to be turned on.

The transitions from C' are analogous to that from C, with the possibility of even going back to C.

Correctness: We sketch a proof here that this algorithm ensures that, in each reshuffle round, all the targets are covered and the algorithm itself terminates enabling each sensor to decide and reach on/off status.

For contradiction, let us assume that in a given round a target t remains uncovered. This implies that either this target has no neighboring sensor within sensing range and thus network itself is dead, or else all the neighboring sensors which could have covered t have turned off. In the latter case, each of the sensor s whose $T(s)$ contains t has made the transition from its current best cover C to off status. However, s only does that if C covers all its targets in $T(s)$ and $s \notin C$. The last such sensor s to have turned off ensures that C is satisfied, which implies that all targets in $T(s)$ including t are covered, a contradiction. Next, for contradiction, let us assume that the algorithm does not terminate. This implies that there exists at least one sensor s which is unable to decide, i.e., make a transition to either on or off status. There are three possibilities: (i) all the covers of s have become infeasible, or

(ii) s is continually transitioning to the next best cover and none of them are getting satisfied, or

(iii) s is stuck at a cover C.

For case (i), for each cover C, at least one of its sensor $s' \in C$ has turned off. But the set of targets considered by sensor s is no larger than $T' = \bigcup_{s' \in N(s,1)} T(s')$. Since s itself can cover $T(s)$, there exist a target $t \in T' - T(s)$, from $T(s')$, that none of the cover sets at s are able to cover. This implies that s' is off, else $\{s, s'\}$ would have formed part of a cover at s covering t (given that s constructs all possible covers). This leads to the contradiction, as before turning off, s' ensures that $t \in T(s')$ is covered.

For case (ii), each transition implies that a neighbor sensor has decided its on/off status, thereby making some of the covers at s infeasible and increasingly satisfying portions of some other covers, thus reducing the choices from the finite number of its covers. Eventually, when the last neighbor decides, s will be able to decide as well becoming on if any target in $T(s)$ is still uncovered, else going off.

For case (iii), the possibility that all sensors are stuck at their best initial covers is conventionally broken by a sensor $s \in C$ with least id in its current best cover C pro actively becoming on, even though C may not be completely satisfied. This is similar to the start-up problem faced by others distributed algorithms such as DEEPS with similar deadlock breaking solutions. At a later stage, if s is stuck at C, it means that either all its neighbors have decided or one or more neighbors are all stuck. In the former case, there exists a cover C at s which will be satisfied with s becoming on (case i). The latter case is again resolved by the start-up deadlock breaking rule by either s or s' pro actively becoming on.

Message and time complexities: Let us assume that each sensor s constructs the covers over its one-hop neighbors to cover its targets in $T(s)$ only. Let $S = \{s_1, s_2, ..., s_n\}$ $\Delta = _{maxs \in S} |N(s,1)|$, the maximum number of neighbors a sensor can communicate with. The communication complexity of the initial setup phase is $O(\Delta)$, assuming that there are constant number of neighboring targets that each sensor can sense. Also, for each reshuffle round, a sensor receives $O(\Delta)$ status messages and sends out one. Assuming Δ is a constant practically implies that message complexity is also a constant. Let maximum number of targets a sensor considers is $\tau = _{maxs \in S} |T(s)|$, a constant. The maximum number of covers constructed by sensor s during its setup phase is $O(\Delta^\tau)$, as each sensor in N(s,1) can potentially cover all its targets considered. Hence the time complexity of setup phase is $O((\Delta^\tau)2)$ to construct the LD graph over all covers and calculate the priorities. For example, if $\tau = 3$, the time complexity of the setup phase would be $O(\Delta^6)$. The reshuffle rounds transition through potentially all the covers, hence their time complexity is $O(\Delta^\tau)$.

9. Variants of the basic algorithm

We briefly discussed some of the properties of the LD graph earlier. For example, an edge e connecting two covers C and C' yields an upper bound on the cumulative lifetime of both the covers. However, if $w(e)$, which equals $b(s)$ for weakest sensor $s \in e$, is larger than the sum of the lifetimes of C and C', then the edge e no longer constrains the usage of C and C'. Therefore, even though C and C' are connected, they do not influence each other's lifetimes. This leads to our first variant algorithm.

Variant 1: Redefine the edge weight e as follows:

If $min_{s \in e} b(s) < lt(C) + lt(C')$, then $w(e) = min_{s \in e} b(s)$, else $w(e) = 0$.

Thus, when calculating the degree of a cover, this edge would not be counted when not constraining, thus elevating the cover's priority. Next, the basic framework is exploiting the degree of a cover to heuristically estimate how much it impacts other covers, and the overall intent is to minimize its impact. Therefore, we sum the edge weights emanating from a cover for its degree. However, if a cover C is connected to two covers C' and C'' such that both C' and C'' have the same bottleneck sensor s, s is depleted by burning either C' or C''. That is, in a sense, only one of C' and C'' can really be burned completely, and then the other is rendered unusable because s is completely depleted. Therefore, for all practical purposes, C' and C'' can be collectively seen as one cover. As such, the two edges connecting C to C' and C'' can be thought of as one as well. This yields our second variant algorithm.

Variant 2: Redefine the degree of a cover C in the LD graph as follows. Let a cover C be connected to a set of covers $V' = C_1, C_2, , C_q$ in graph G. If there are two covers C_i and C_j in V' sharing a bottleneck sensor s, then if $w(C, C_i) < w(C, C_j)$ then $V' = V' - C_j$ else $V' = V' - C_i$. With this reduced set of neighboring covers V', the degree of cover C is

$d(C) = \sum_{C' \in V'} w(C, C')$

In the basic algorithm, each sensors constructs cover sets using its one-hop neighbors to cover its direct targets $T(s)$. However, with the same message overheads and slightly increased time complexity, a sensor can also consider its neighbors' targets. This will enable it to explore the constraint space of its neighbors as well.

Variant 3: In this variant, each sensor s constructs LD graph over one-hop neighbors $N(s,1)$ and targets in $\bigcup_{s' \in N(s,1)} T(s')$.

Variant 4: In the basic two-hop algorithm, each sensor s constructs LD graph over two-hop neighbors $N(s,2)$ and targets in $\bigcup_{s' \in N(s,1)} T(s')$. In this variant, each sensor s constructs LD graph over two-hop neighbors $N(s,2)$ and targets in $\bigcup_{s' \in N(s,2)} T(s')$.

10. Taming the exponential state space of the maximum lifetime sensor cover problem

If we consider the LD graph, it is quickly obvious that even creating this graph will take exponential time since there are 2^n cover sets to consider where, n is the number of sensors. However, the target coverage problem has a useful property - if the local targets for every sensor are covered, then globally, all targets are also covered. In [35], we make use of this property to look at the LD graph locally (fixed 1-2 hop neighbors), and are able to construct all the local covers for the local targets and then model their dependencies. Based on these dependencies, a sensor can then prioritize its covers and negotiate these with its neighbors. Simple heuristics based on properties of this graph were presented in [35] and showed a 10-15% improvement over comparable algorithms in the literature. [15] built on this work by examining how an optimal sequence would pick covers in the LD graph and designing heuristics that behave in a similar fashion. Though the proposed heuristics are efficient in practice, the running time is a function of the number of neighbors and the number of local targets. Both of these are relatively small for most graphs but theoretically are exponential in the number of targets and sensors.

A key issue that remains unresolved is the question of how to deal with this exponential space of cover sets. In this paper we present a reduction of this exponential space to a linear one based on grouping cover sets into *equivalence classes*. We use $[C_i]$ to denote the equivalence class of a cover C_i. The partition defined by the equivalence relation on the set of all sensor covers Given a set C and an equivalence relation $\Re$, the equivalence class of an element $C_i \in C$ is the subset of all elements in C which are equivalent to C_i. The notation used to represent the equivalence class of C_i is $[C_i]$. In the context of the problem being studied, C is the set of all sensor covers and for any single cover C_i, $[C_i]$ represents all other covers which are *equivalent* to C_i as given by the definition of some equivalence relation $\Re$. Our approach stems from the understanding that from the possible exponential number of sensor covers, several covers are very similar, being only minor variations of each other. In Section 11, we present the definition of the relation $\Re$, based on a grouping that considers cover sets equivalent if their lifetime is bounded by the same sensor. We then show the use of this relation to collapse the exponential LD Graph into an *Equivalence Class* (EC) Graph with linear number of nodes. This theoretical insight allows us to design a sampling scheme that selects a subset of all local covers based on their equivalence class properties and presents this as an input to our simple LD graph degree-based heuristic. Simulation results show that class based sampling cuts the running time of these heuristics by nearly half, while only resulting in a less than 10% loss in quality.

11. Dealing with the exponential space

In this section, we present our approach of dealing with the exponential solution space of possible cover sets. The next section utilizes these ideas to develop heuristics for maximizing

the lifetime of the network. Even though the total number of cover sets for the network may be exponential in the number of sensors, for any given cover set, there are several other sets that are very *similar* to this set. We begin by attempting to define this notion of similarity by expressing it as an equivalence relation.

Definition 1: Let $\Re$ be an equivalence relation defined on the set of all sensor covers such that $C_i \Re C_j$ if and only if C_i and C_j share the same bottleneck sensor s_{bot}.

Theorem: $\Re$ is an equivalence relation
Proof: $\Re$ is reflexive, since $C_i \Re C_i$. $\Re$ is symmetric, since if $C_i \Re C_j$ then, $C_j \Re C_i$ since both covers C_i and C_j share the same bottleneck sensor. Finally, if $C_i \Re C_j$ and $C_j \Re C_k$, then $C_i \Re C_k$ and $\Re$ is transitive since if C_i shares the same bottleneck sensor with C_j and C_j shares the same bottleneck sensor with C_k then, clearly both C_i and C_k have the same bottleneck sensor in common. Therefore, $\Re$ is and equivalence relation. □

Every equivalence relation defined on a set, specifies how to partition the set into subsets such that every element of the larger set is in exactly one of the subsets. Elements that are related to each other are by definition in the same partition. Each such partition is called an *equivalence class*. Hence, the relation $\Re$ partitions the set of all possible sensor covers into a number of *disjoint* equivalence classes.

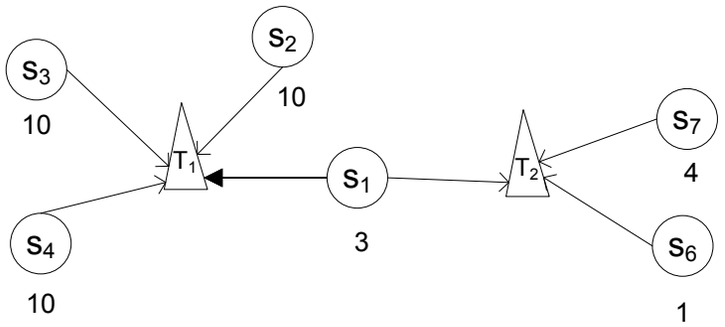

Figure 4. Example Sensor Network

Notation: Henceforth, we represent the equivalence class of covers sharing a bottleneck sensor s_i by $[s_i]$. Note that this is a slight abuse of notation since s_i is not a member of this class, but is instead the property that is common to all members of this class. Hence, $[s_i]$ can be read as the equivalence class for all covers having sensor s_i as their bottleneck sensor.

We now define what we would call the Equivalence Class (EC) Graph. Each node of this graph represents an equivalence class. Just as the LD graph models the dependency between sensor covers, the EC Graph models the dependency between classes of covers.

Definition 2: Equivalence Class Graph (EC). The Equivalence Class graph $EC = (V', E')$ where, V' is the set of all possible equivalence classes defined by $\Re$ and two classes $[s_i]$ and $[s_j]$ are joined by an edge for every cover in each class that share some sensor in common. Hence, the graph EC is a multi-edge graph.

The cardinality of the vertex set of the Equivalence Class Graph is at most n. This result follows from the observation that for any network of n sensors, there can be at most one equivalence class corresponding to each sensor, since every cover can have only one of the n sensors as its bottleneck (in case two or more sensors all have the same battery and are the bottleneck, sensor id's can be used to break ties).

To better understand these definitions, let us consider an example. Consider the sensor network shown in Figure 4. The network comprises of seven sensors, $s_1, ..., s_7$ and two targets, T_1, T_2. Observe that T_2 is the bottleneck target for the network since it is the least covered target (8 units of total coverage compared to 33 for T_1). Also note that only one sensor, s_1 can cover both targets.

For the given network, the set of all possible minimal sensor covers, S is,

$S = \{\{s_2, \mathbf{s_6}\}, \{s_2, \mathbf{s_7}\}, \{s_3, \mathbf{s_6}\}, \{s_3, \mathbf{s_7}\}, \{s_4, \mathbf{s_6}\}, \{s_4, \mathbf{s_7}\}, \{\mathbf{s_1}\}\}$

For each individual cover in this set, the bottleneck sensor is the sensor shown in bold face.

Figure 5 shows the Lifetime Dependency graph for these covers. As defined, an edge exists between any two covers that share at least one sensor in common and the weight of this edge is given by the lifetime of the common sensor having the smallest battery (the bottleneck). For example, an edge of weight 4 exists between C_1 and C_2 because they share the sensor s_2 having a battery of 4.

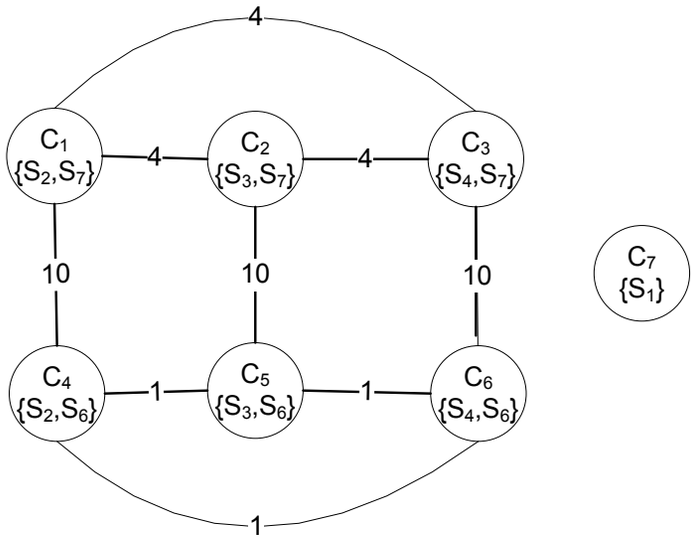

Figure 5. LD Graph for the example network

To obtain the EC Graph from this LD Graph, we add a node to represent the equivalence class for each sensor that is a bottleneck sensor for any cover. For the above example, given all sensor covers in the set S, there are three sensors s_1, s_6, s_7 that are each the bottleneck for one or more covers in S. Hence, the EC Graph is a three node graph. Figure 6 shows the complete EC Graph for the covers in S. There is a node corresponding to the equivalence class for each of the three sensors s_1, s_6, s_7 and for each cover in the class we retain edges to the class corresponding to the bottleneck sensor of the cover on which the edge terminated in the LD graph. Hence, we have three edges between the nodes s_6 and s_7.

It is key to realize that the EC graph is essentially an encapsulation of the LD Graph that can have at most n nodes. This view is presented in Figure 7, where we show the LD Graph that is embedded into the EC Graph. Each rectangular box shows the nodes in the LD graph that are in the same equivalence class. This figure also illustrates our next theorem.

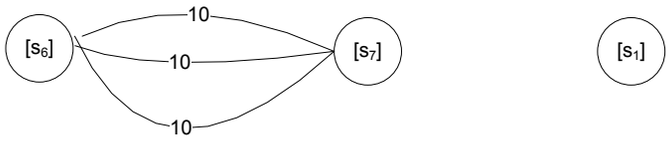

Figure 6. EC Graph for the example network

Theorem: For sensor covers in the same equivalence class, the induced subgraph on the LD Graph is a clique
Proof: This theorem states that for the nodes in the LD graph that belong to the same class, the induced subgraph is a clique. Since by definition, all sensor covers in a class $[s]$ share the sensor s as their bottleneck sensor, the induced subgraph will be a complete graph between these nodes. □

Also, a subtle distinction has been made between inter-class edges and intra-class edges in going from the LD graph to the EC graph.

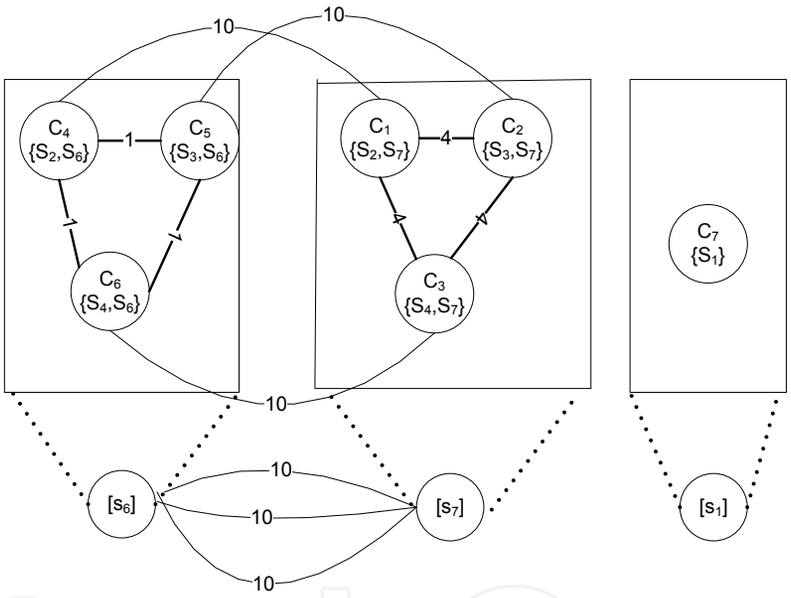

Figure 7. EC Graph for the example network along with the LD Graph embedded in it

12. Sampling based on the equivalence class graph

The previous section defined the concepts behind reducing the exponential space of covers in the LD Graph to the linear space of the EC Graph. In this section, we build on these concepts to discuss techniques for generating a limited number of covers for the LD Graph. Specifically, our goal is to improve the timing performance of the distributed algorithms we presented in [15, 35]. As presented, the EC Graph is not very useful since it still requires the exponential LD graph to be populated, before it can be constructed. However, by realizing that the exponential space of cover sets can be expressed in this linear space of equivalence classes, we can generate only a subset of the set of all covers.

Recall that even though the number of global sensor covers is exponential in the number of sensors, our heuristics presented in [15, 35] worked by constructing *local* covers. After exchanging one or two hop coverage information with neighboring sensors, a sensor can exhaustively construct all possible local covers. A local cover here is a sensor cover that covers all the local targets. The number of local covers is also exponential but is determined by the maximum degree of the graph and the number of local targets, typically much smaller values than the number of all sensors or targets. The heuristics then construct the LD graph over these local covers. The choice of which cover to use is determined by looking at properties of the LD graph such as the degree of each cover in the LD graph.

By making use of the idea of related covers in the same equivalence class, our goal is to use our existing heuristics from [15, 35] but to modify them to run over a subset of the local covers as opposed to all local covers. This should give considerable speedup and if the subset is selected carefully, it may only result in a slight reduction of the overall lifetime. We present such a local cover sampling scheme in Section 12.1 and then present the modified basic algorithm of [15, 35] to operate on this sample in Section 8. Finally, we evaluate the effectiveness of sampling in Section 13.

12.1. Local bottleneck target based generation of local cover sets

Understanding the underlying equivalence class structure, we now present one possible way of generating a subset of the local cover sets. Our approach is centered around the bottleneck target. For any target t_i, the total amount of time this target can be monitored by any schedule is given by:

$$lt(t_i) = \sum_{\{s \mid t_i \in T(s)\}} b(s)$$

Clearly there is one such target with the smallest $lt(t_i)$ value, and is hence a bottleneck for the entire network [40]. Without global information it is not possible for any sensor to determine if the global bottleneck is a target in its vicinity. However, for any sensor s, there is a least covered target in $T(s)$ that is the *local* bottleneck. A key thing to realize is the fact that the global bottleneck target is also the local bottleneck target for the sensors in its neighborhood. Hence, if every sensor optimizes for its local bottleneck target, then one of these local optimizations is also optimizing the global bottleneck target. We use t_{bot} to denote this local bottleneck target. Let C_{bot} be the set of sensors that can cover this local bottleneck target.That is,

$$C_{bot} = \{s \mid t_{bot} \in T(s)\}$$

Implementation: This understanding of bottleneck targets, along with our definition of equivalence classes, now gives us a simple means to generate local covers. Since no coverage schedule can do any better than the total amount of time that the global bottleneck can be covered, instead of trying to generate all local covers, what we really need are covers in the equivalence classes corresponding to each sensor $s_i \in C_{bot}$, such that each class can be completely exhausted. Also, to only select covers that conserve the battery of the sensors in C_{bot}, we want to ensure that the covers we generate are disjoint in C_{bot}. In terms of equivalence classes, for any two classes $[s_i]$ and $[s_j]$ such that $s_i, s_j \in C_{bot}$, we want to generate cover sets that are in these classes but do not include *both* s_i and s_j.

To generate such cover sets, we can start by picking only one sensor s'_{bot} in C_{bot}. This ensures that the local bottleneck target is covered. For each target t_i in the one/two-hop neighborhood being considered, we can then randomly pick a sensor s, giving preference to any $s \notin C_{bot}$. Note that this does not necessarily create a sensor cover in the class $[s'_{bot}]$, since any one of our randomly picked sensors could be the bottleneck for the cover generated. However, replacing that sensor with another randomly picked sensor that covers the same target ensures that the we finish by using a cover in $[s'_{bot}]$. Such a selection essentially ensures that we burn the entire battery of this sensor s'_{bot} in C_{bot} through different covers, while trying to avoid using other sensors in C_{bot}. This process is then repeated for every sensor in C_{bot}. Hence, instead of generating all local covers, we only generate a small sample (constant number) of these corresponding to the equivalence class for each sensor covering the bottleneck target and some related randomly picked covers. We already showed that there can be at most n equivalence classes for the network. Thus, the sampled graph generated has O(n) nodes. If we consider the maximum number of sensors covering any target as a constant for the network, sampling only takes cumulative time of $O(n\tau)$, where $\tau = max_{s \in S}|T(s)|$, since we do this for n sensors, each of which has a maximum of τ targets to cover, which are in turn covered by a constant number of sensors (as per our assumption). Even if this assumption is removed, in the worst case, all n sensors could be covering the same target making the time complexity $O(n^2\tau)$. Next, we run our basic heuristic from [35] on this sampled LD graph.

13. Performance evaluation

In this section, we evaluate the performance of the proposed sampling scheme and evaluate it against our degree based heuristics of [35]. By not constructing all local covers and instead constructing a few covers for key equivalence classes, we should achieve considerable speedup. But the effectiveness of sampling can only be evaluated by analyzing its tradeoff between faster running time for possible reduced performance. The objective of our simulations was to study this tradeoff. For completeness, we create both one-hop and two-hop versions of our sampling heuristic and also compare its performance to two other algorithms in the literature, the 1-hop algorithm LBP [4] and the 2-hop algorithm DEEPS [5].

In order to compare the equivalence class based sampling against our previous degree based heuristics, LBP, and DEEPS, we use the same experimental setup and parameters as employed in [4]. We carry out all the simulations using C++. For the simulation environment, a static wireless network of sensors and targets scattered randomly in $100m \times 100m$ area is considered. We conduct the simulation with 25 targets randomly deployed, and vary the number of sensors between 40 and 120 with an increment of 20 and each sensor with a fixed sensing range of $60m$. The communication range of each sensor assumed to be two times the sensing range [44, 46]. For these simulations, we use the linear energy model wherein the power required to sense a target at distance d is proportional to d. We also experimented with the quadratic energy model (power proportional to d^2). The results showed similar trends to those obtained for the linear model.

Figure 8 shows the Network Lifetime for the different algorithms. As can be seen from the figure, the sampling heuristics is only between 7-9% worse than the degree based heuristic. Sampling also outperforms the 1-hop LBP algorithm by about 10%. It is interesting to observe that for smaller network sizes, sampling is actually much closer to the degree-based heuristics in terms of performance.

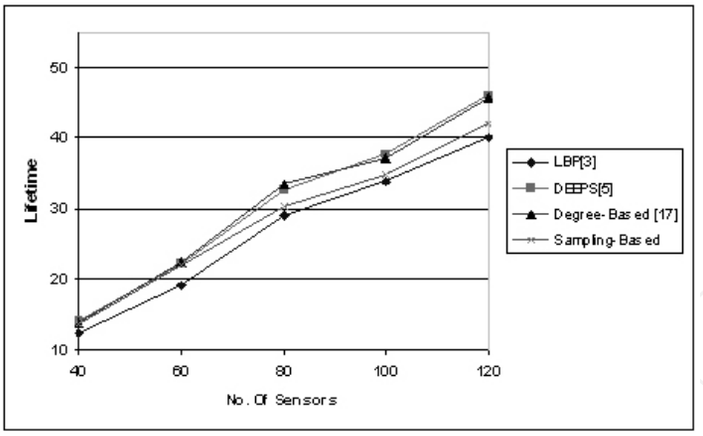

Figure 8. Comparison of Network Lifetime with 25 Targets

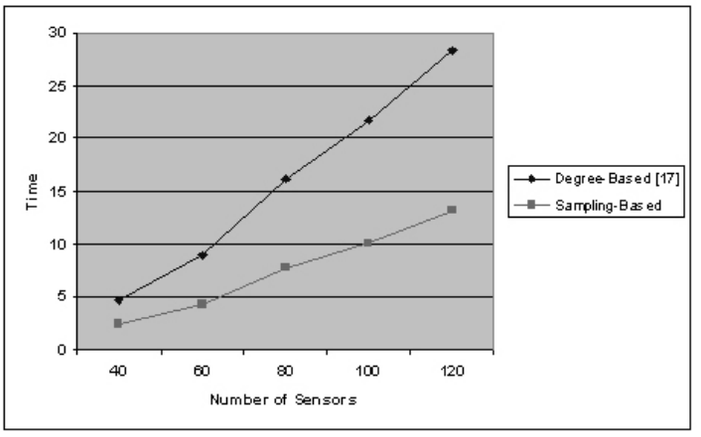

Figure 9. Comparison of Running Time with 25 Targets

Algorithm	n=40	n=80	n=120
LBP [4]	12.4	29.1	40.1
Degree-Based [35]	13.8	33.4	45.6
Sampling-Based	13.7	30.3	42.1
Randomized-Sampling	10.1	17.6	30.1

Table 3. Comparison of Network Lifetime for 1-hop algorithms

Now that we have seen that sampling works well when compared to the degree based heuristic, the question that remains to be answered is how much faster is the sampling algorithm? Figure 9 compares head-to-head the running time for the degree based heuristic (potentially exponential in m) and the linear time sampling algorithm. As can be seen from the figure the running time for the sampling algorithm is about half of the running time for the degree-based heuristic.

Finally, we individually study the 1-hop (Table 3) and 2-hop (Table 4) sampling heuristics with comparable algorithms. For the 1-hop algorithms, we also include a randomized-sampling algorithm that makes completely random picks for each target, without considering properties of the equivalence classes. The intention is to ensure that the performance of our sampling-heuristic can be attributed to the selection algorithm. For the 2-hop versions of

Algorithm	n=40	n=80	n=120
DEEPS [5]	14.1	32.7	46.1
Degree-Based (2-hop) [35]	15.2	36.2	49.6
Sampling-Based (2-hop)	14.4	33.4	47.5

Table 4. Comparison of Network Lifetime of 2-hop algorithms

our proposed sampling heuristic, the target set $T(s)$ of each sensor is expanded to include $\cup_{s' \in N(s,1)} T(s')$ and the neighbor set is expanded to all 2-hop neighbors, i.e., $N(s, 2)$. Covers are now constructed over this set using the same process as before. As can be seen from the table, both the 1-hop and 2-hop version are under 10% worse than the comparable degree-based heuristics. Also, the 2-hop sampling slightly outperforms the DEEPS by a 5% improvement in network lifetime.

14. Conclusion

Despite a lot of recent research effort, creating real-world deployable sensor networks remains a difficult task. A key bottleneck is the limited battery life of sensor motes. Hence, energy conservation at every layer of the network stack is critical. Creating realistic theoretical models for problems in this domain that take this into account remains a challenge. Our work addresses energy efficiency at only point in the network stack. However, a holistic approach to energy efficiency design should not only account for energy concerns in each layer of the network stack for problems like routing, medium access etc., but also consider cross-layer issues and interactions.

In this chapter, we present innovative models and heuristics to address the coverage problem in Wireless Sensor Networks. Our work points to the potential of lifetime dependency graphs while serving to highlight the shortcomings of using standard distributed algorithms to this problem. In order to successfully bridge the gap between the theory and practice of wireless sensor networks, there is a clear need for algorithms that are designed keeping the unique constraints of these networks in mind. The improvements in network lifetime obtained by our approach using the dependency graph and heuristics that stem serve to underscore this point.

Acknowledgements

The author would like to thank his wife Kelli for her support while working on this chapter.

Author details

Akshaye Dhawan
Department of Mathematics and Computer Science, Ursinus College, USA.

15. References

[1] Abrams, Z., Goel, A. & Plotkin, S. [2004]. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks, *Third International Symposium on Information Processing in Sensor Networks* pp. 424–432.

[2] Akyildiz, I., Su, W., Sankarasubramaniam, Y. & Cayirci, E. [2002]. A survey on sensor networks, *IEEE Commun. Mag.* pp. 102–114.
[3] Berman, P., Calinescu, G., Shah, C. & Zelikovsky, A. [2004]. Power efficient monitoring management in sensor networks, *Wireless Communications and Networking Conference (WCNC)* 4: 2329–2334 Vol.4.
[4] Berman, P., Calinescu, G., Shah, C. & Zelikovsky, A. [2005]. Efficient energy management in sensor networks, *In Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing* .
[5] Brinza, D. & Zelikovsky, A. [2006]. Deeps: Deterministic energy-efficient protocol for sensor networks, *Proceedings of the International Workshop on Self-Assembling Wireless Networks (SAWN)* pp. 261–266.
[6] Cardei, M. & Du, D.-Z. [2005]. Improving wireless sensor network lifetime through power aware organization, *Wireless Networks* 11: 333–340(8).
[7] Cardei, M., MacCallum, D., Cheng, M. X., Min, M., Jia, X., Li, D. & Du, D.-Z. [2002]. Wireless sensor networks with energy efficient organization, *Journal of Interconnection Networks* 3(3-4): 213–229.
[8] Cardei, M., Thai, M., Li, Y. & Wu, W. [2005]. Energy-efficient target coverage in wireless sensor networks, *INFOCOM 2005* 3.
[9] Cardei, M. & Wu, J. [2006]. Energy-efficient coverage problems in wireless ad hoc sensor networks, *Computer Communications* 29(4): 413–420.
[10] Carle, J. & Simplot-Ryl, D. [2004]. Energy-efficient area monitoring for sensor networks, *Computer* 37(2): 40–46.
[11] Chen, B., Jamieson, K., Balakrishnan, H. & Morris, R. [2001]. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, Rome, Italy, pp. 85–96.
[12] Chen, B., Jamieson, K., Balakrishnan, H. & Morris, R. [2002]. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, *ACM Wireless Networks* 8(5).
[13] Chong, C.-Y. & Kumar, S. [2003]. Sensor networks: evolution, opportunities, and challenges, *Proceedings of the IEEE* 91(8): 1247–1256.
[14] Culler, D. & Hong, W. [2004]. Wireless sensor networks, *Special Issue, CACM* .
[15] Dhawan, A. & Prasad, S. K. [2008]. Energy efficient distributed algorithms for sensor target coverage based on properties of an optimal schedule, *HiPC: 15th International Conference on High Performance Computing, LNCS 5374*.
[16] Dhawan, A., Vu, C. T., Zelikovsky, A., Li, Y. & Prasad, S. K. [2006]. Maximum lifetime of sensor networks with adjustable sensing range, *Proceedings of the International Workshop on Self-Assembling Wireless Networks (SAWN)* pp. 285–289.
[17] Dhillon, S. S. & Chakrabarty, K. [2003]. Sensor placement for effective coverage and surveillance in distributed sensor networks, Vol. 3, pp. 1609–1614 vol.3.
[18] Feeney, L. M. [2001]. An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks, *Mob. Netw. Appl.* 6(3): 239–249.
[19] Feeney, L. & Nilsson, M. [2001]. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, Vol. 3, pp. 1548–1557 vol.3.

[20] Garey, M. R. & Johnson, D. S. [1979]. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA.
[21] Garg, N. & Koenemann, J. [1998]. Faster and simpler algorithms for multicommodity flow and other fractional packing problems., *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, Washington, DC, USA, p. 300.
[22] Giridhar, A. & Kumar, P. [2005]. Maximizing the functional lifetime of sensor networks, *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pp. 5–12.
[23] Iyengar, S. & Brooks, R. [2005]. *Handbook of Distributed Sensor Networks*, Chapman and Hall/CRC.
[24] Iyengar, S. S. & Brooks, R. [2004]. Computing and communications in distributed sensor networks, *Special Issue, Jr. of Parallel and Distributed Computing* 64(7).
[25] Jung, E.-S. & Vaidya, N. H. [2005]. Power aware routing using power control in ad hoc networks, *SIGMOBILE Mob. Comput. Commun. Rev.* 9(3): 7–18.
[26] Kar, K. & Banerjee, S. [2003]. Node placement for connected coverage in sensor networks, *In Proc. of WiOpt 2003: Modeling and Optimization in Mobile, Ad-Hoc and Wirless Networks*.
[27] Kumar, S., Lai, T. H. & Balogh, J. [2004]. On k-coverage in a mostly sleeping sensor network, *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, ACM, New York, NY, USA, pp. 144–158.
[28] Li, Y., Thai, M. T. & Wu, W. [2008]. *Wireless Sensor Networks and Applications*, Springer.
[29] Lu, J. & Suda, T. [2003]. Coverage-aware self-scheduling in sensor networks, *18th Annual Workshop on Computer Communications (CCW)* pp. 117–123.
[30] Mao, Y., Wang, Z. & Liang, Y. [21-25 Sept. 2007]. Energy aware partial coverage protocol in wireless sensor networks, *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on* pp. 2535–2538.
[31] Megerian, S., Koushanfar, F. & Potkonjak, M. [2005]. Worst and best-case coverage in sensor networks, *IEEE Transactions on Mobile Computing* 4(1): 84–92. Senior Member-Srivastava, Mani B.
[32] Meguerdichian, S., Koushanfar, F., Potkonjak, M. & Srivastava, M. B. [2001]. Coverage problems in wireless ad-hoc sensor networks, *in IEEE INFOCOM*, pp. 1380–1387.
[33] Meguerdichian, S. & Potkonjak, M. [2003]. Low power 0/1 coverage and scheduling techniques in sensor networks, *UCLA Technical Reports 030001* .
[34] Patel, M., Chandrasekaran, R. & Venkatesan, S. [2005]. Energy efficient sensor, relay and base station placements for coverage, connectivity and routing, *Performance, Computing, and Communications Conference, 2005. IPCCC 2005. 24th IEEE International*, pp. 581–586.
[35] Prasad, S. K. & Dhawan, A. [2007]. Distributed algorithms for lifetime of wireless sensor networks based on dependencies among cover sets, *HiPC: 14th International Conference on High Performance Computing, LNCS 4873*, pp. 381–392.
[36] Raghunathan, V., Schurgers, C., Park, S., Srivastava, M. & Shaw, B. [2002]. Energy-aware wireless microsensor networks, *IEEE Signal Processing Magazine*, pp. 40–50.
[37] Römer, K. & Mattern, F. [2004]. The design space of wireless sensor networks, *IEEE Wireless Communications* 11(6): 54–61.
[38] Sahni, S. & Xu, X. [2004]. Algorithms for wireless sensor networks, *Intl. Jr. on Distr. Sensor Networks* 1.
[39] Schmid, S. & Wattenhofer, R. [2006]. Algorithmic models for sensor networks, *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pp. 11 pp.–.

[40] Schmid, S. & Wattenhoffer, R. [n.d.]. Maximizing the lifetime of dominating sets.
[41] Slijepcevic, S. & Potkonjak, M. [2001]. Power efficient organization of wireless sensor networks, *IEEE International Conference on Communications (ICC)* pp. 472–476 vol.2.
[42] Tian, D. & Georganas, N. D. [2002]. A coverage-preserving node scheduling scheme for large wireless sensor networks, *WSNA: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, ACM, New York, NY, USA, pp. 32–41.
[43] Wattenhoffer, R. [n.d.]. Sensor networks: Distributed algorithms reloaded - or revolution?
[44] Xing, G., Wang, X., Zhang, Y., Lu, C., Pless, R. & Gill, C. [2005]. Integrated coverage and connectivity configuration for energy conservation in sensor networks, *ACM Trans. Sen. Netw.* 1(1): 36–72.
[45] Ye, F., Zhong, G., Lu, S. & Zhang, L. [2002]. Peas: A robust energy conserving protocol for long-lived sensor networks, *IEEE International Conference on Network Protocols (ICNP)* 00: 200.
[46] Zhang, H. & Hou, J. [2005]. Maintaining sensing coverage and connectivity in large sensor networks, *Ad Hoc and Sensor Wireless Networks (AHSWN)* .
[47] Zongheng Zhou; Das, S.; Gupta, H. [11-13 Oct. 2004]. Connected k-coverage problem in sensor networks, *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on* pp. 373–378.