EFFICIENT K-COVERAGE ALGORITHMS FOR WIRELESS SENSOR NETWORKS AND THEIR APPLICATIONS TO EARLY DETECTION OF FOREST FIRES

by

Majid Bagheri

B.Sc., Sharif University of Technology, Tehran, Iran, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Summer 2007

APPROVAL

Name:	Majid Bagheri
Degree:	Master of Science
Title of thesis:	Efficient k-Coverage Algorithms for Wireless Sensor Networks and Their Applications to Early Detection of Forest Fires
Examining Committee:	Dr. Dirk Beyer Chair
	Dr. Mohamed Hefeeda, Senior Supervisor
	Dr. Joseph Peters, Supervisor
	Dr. Funda Ergun, Examiner

Date Approved: April 18, 2007

Abstract

Achieving k-coverage in wireless sensor networks has been shown before to be NP-hard. We propose an efficient approximation algorithm which achieves a solution of size within a logarithmic factor of the optimal. A key feature of our algorithm is that it can be implemented in a distributed manner with local information and low message complexity. We design and implement a fully distributed version of our algorithm. Simulation results show that our distributed algorithm converges faster and consumes much less energy than previous algorithms. We use our algorithms in designing a wireless sensor network for early detection of forest fires. Our design is based on the Fire Weather Index (FWI) System developed by the Canadian Forest Service. Our experimental results show the efficiency and accuracy of the proposed system.

To the wandering soul of the desert

"We feel free because we lack the very language to articulate our unfreedom."

— *Slavoj Zizek*

Acknowledgments

I am deeply indebted to my senior supervisor, Dr. Mohamed Hefeeda, for his continuous support, encouragement and guidance through my research. Mohamed provided valuable insights, and a lot of help during my graduate career. This thesis would not have been possible without him.

I would like to thank my supervisor Dr. Joseph Peters and my thesis examiner Dr. Funda Ergun for being on my committee and reviewing this thesis. I would like to thank Dr. Dirk Beyer for taking the time to chair my thesis defense. I would also like to extend my gratitude to the faculty and staff in the school of computing science at SFU.

Last but certainly not least, I would like to thank my family and friends for their love, care and support. I would not have accomplished anything without them.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

In this chapter, we briefly introduce the coverage problem in wireless sensor networks and describe the related work. We also look into applications of wireless sensor networks to early detection of forest fires. Finally, we summarize the contributions of this thesis.

1.1 Introduction

Mass production of sensor devices with low cost enables one to deploy large-scale sensor networks for real-life applications such as forest fire detection and vehicle traffic monitoring. A fundamental issue in such applications is the quality of monitoring provided by the network. This quality is usually measured by how well deployed sensors *cover* a set of target points. In its simplest form, coverage means that every point is monitored by, i.e., within the sensing range of, at least one sensor. This is called 1-coverage. In this paper, we consider the more general k-coverage ($k \geq 1$) problem, where each point should be within the sensing range of k or more sensors.

Covering each point by multiple sensors is desired for many applications, because it provides redundancy and fault tolerance. Furthermore, k-coverage is necessary for the proper functioning of many other applications, such as intrusion detection [3, 40], data gathering [32, 60], and object tracking [20]. To illustrate, consider an intrusion detection system in military applications, where k-coverage is essential to identify intruding objects of different sizes. A tank, for instance, is detected by many sensors, while a soldier is detected by only a few. A high degree of coverage makes the classification more precise [3], because of errors in the measurement and the susceptibility of sensors to failure and power shortage.

When deployed sensors are dense, point coverage approximates area coverage. That is, if all sensor locations are covered by the set of activated sensors, the entire area is covered. In this paper, we address the problem of selecting the minimum set of sensors to activate from an already deployed set of sensors such that all locations are k-covered. Achieving a minimal set of sensors is critical, because it reduces total energy consumption, and thus prolongs the lifetime of the whole network.

The problem of selecting the minimum number of sensors, however, is NP-hard [54].[1] We propose an efficient approximation algorithm for it, which achieves a solution of size within a logarithmic factor of the optimal and terminates quickly (in the order of seconds in most cases). Although the approximation factor is logarithmic, it is only a worst-case upper bound and our simulation results show a better performance. We take a novel approach in solving the k-coverage problem. In particular, we model the problem as a set system for which an optimal *hitting set* corresponds to an optimal solution for coverage. Finding the optimal hitting set is NP-hard [18], but there is an efficient approximation algorithm for it [9]. Our k-coverage algorithm is inspired by the approximation algorithm for the optimal hitting set problem. We prove that our algorithm is correct and analyze its complexity. We implement our algorithm and compare it against other centralized algorithms in the literature. Our comparison reveals that our algorithm is about four orders of magnitude faster than the currently-known k-coverage algorithms.

A key feature of our centralized k-algorithm is that it can be implemented in a distributed manner with local information and low message complexity. We design and implement a fully distributed version of our algorithm. Our distributed algorithm does not require sensors to know their locations. Comparison with two other distributed algorithms in the literature indicates that our algorithm: (i) converges much faster than the others, (ii) activates near-optimal number of sensors, and (iii) significantly prolongs the network lifetime because it consumes much less energy than the other algorithms. We also extend our distributed algorithm to provide hot spot coverage which is required for some applications where more important areas such as human neighborhood need a higher coverage degree. Simulation results verify that our algorithm indeed achieves the required coverage degree for different regions inside the field.

We also address the coverage problem in relation with forest fire detection and show how

[1] Note that this problem is different from the problem of *placing* sensors in an area to cover it, which can be solved efficiently [29].

to apply our distributed k-coverage algorithm to such applications. We provide the design of a wireless sensor network for early detection of forest fires based on the Fire Weather Index (FWI) System. The FWI System designed by Canadian Forest Service [12] is backed by several decades of forestry research. It uses weather observations to produce a set of fire weather indexes which indicate the likelihood of the current weather condition to cause a fire. We present a new aggregation paradigm to efficiently calculate the fire weather indexes in a wireless sensor network. More importantly, we establish a relationship between the desired accuracy of the system and the required coverage degree.

1.2 Previous Work

1.2.1 K-Coverage Algorithms

The problem of *verifying* k-coverage is studied in [28]. Each sensor is modeled as a disk and it is proved that the area is properly k-covered if the perimeter of all disks are k-covered. The running time of the algorithm is $O(n^2 \log n)$ in the worst case for a set of n sensors. An improved modeling is presented in [48], where the authors use the concept of order-k Voronoi diagrams [44] to build a verifier algorithm. They show that if all vertices of a bounded Voronoi diagram is sufficiently covered then the whole area is covered. The running time of the algorithm is bounded by the construction time of the Voronoi diagram which is $O(n \log n + nk^2)$ [34]. These works do not address the problem of ensuring k-coverage and they do not propose distributed algorithms.

Meguerdichian et al. [39] consider a slightly different definition of coverage: finding maximal support and maximal breach paths for which the observability is maximum and minimum, respectively. Authors in [40] improve the work in [39] and present a more efficient algorithm. These algorithms are not applicable to our k-coverage problem.

The authors of [53] propose a distributed Coverage Configuration Protocol (CCP), which provides different degrees of coverage requested by applications. CCP is based on a proof that if the intersection points between all sensors are k-covered, the whole area is k-covered. Unlike our distributed algorithm, CCP assumes that nodes know their locations.

In [13], the authors formulate the k-coverage problem of a set of n grid points as an integer linear programming. They assume two types of sensors with different sensing ranges and costs. The problem is to determine the minimum cost of sensors to cover all grid points. The authors show that the problem is NP-hard since it is a generalized version

of the minimum-cost satisfiability problem [18]. Small instances of the problem are solved using the branch and bound method, which takes exponential time in the worst case. For large instances, a divide and conquer scheme is provided.

The closest work to ours are [54] and [61]. In [54], the authors address the problem of selecting the minimum number of sensors to activate from a set of already deployed sensors for k-coverage. They prove that the problem is NP-hard since it is an extension of the dominating set problem [18]. They formulate the problem and provide a centralized approximation solution based on integer linear programming. The algorithm works by relaxing the problem to ordinary linear programming, where the variables may take real values. They also design a distributed algorithms, PKA, which uses pruning to reduce the number of active sensors. The work in [61] presents a centralized algorithm that works by iteratively adding a set of nodes which maximizes a measure called *k-benefit* to an initially empty set of nodes. The authors also present a distributed algorithms, DPA, that works by pruning unnecessary nodes and putting them into sleep. We compare our algorithms against the algorithms in [54, 61].

1.2.2 Applications of Wireless Sensor Networks

Sensor networks have several appealing characteristics to be used in environmental monitoring applications such as habitat monitoring [37], and forest fire detection systems [15, 21, 49, 57].

For example, in [37], the authors apply wireless sensor networks to habitat monitoring. A set of system requirements are developed and a system architecture is proposed to address these requirements. They discuss different issues such as deployment, data collection, and communication protocols and provide design guidelines. The system architecture is comprised of patches of sensor nodes connected to a base station through gateway nodes. Each sensor reports its readings to the base station. The base station is connected to the internet and exposes the collected data to a set of web based applications. They present experimental results from a habitat monitoring system consisting of 32 nodes deployed on a small island off the coast of Maine. The sensors were placed in burrows to collect temperature data which are used to detect the presence of nesting birds. In this work, we target a different application: forest fire detection.

The authors of [15] show the feasibility of wireless sensor networks for forest fire monitoring. Experimental results are reported from two controlled fires in San Francisco, California.

The system is composed of 10 GPS-enabled MICA motes [45] collecting temperature, humidity, and barometric pressure data. The data is communicated to a base station which records it in a database and provides services for different applications. The experiments show that most of the motes in the burned area were capable of reporting the passage of the flame before being burned. The system in [15], collects and reports raw weather metrics. In contrast, our system processes weather condition based on the Fire Weather Index System [11] and reports more useful, summarized, fire indexes.

A forest fire detection system based on neural networks is designed in [57]. In this system, sensor nodes organize themselves into clusters. Each sensor measures weather data such as temperature and relative humidity and reports its readings to the cluster head. Cluster heads use the data as input to a neural network which produces a weather indexe based on United States National Fire Danger Rating System (NFDRS) [42]. These indexes indicate the likelihood of the weather condition to cause a fire. The cluster heads report their indexes to a sink which is connected to a manager node. The main role of the manager node is to facilitate forest fire detection by analyzing the indexes. The manager node is also involved in the neural network learning process by calculating new values of neural network parameters using the collected data. These parameters are then propagated to the cluster heads through the sink. Simulation results show the effectiveness of the neural network approach in reducing the communication overhead. Our system design is similar to the one in [57], however we do not use a neural network based approach. In addition, we take advantage of two fire indexes which better describe the fire potential as well as intensity.

In [21], the authors address the problem of studying fire behavior rather than fire detection. They present FireWxNet, a portable fire sensor network to study the weather conditions surrounding active fires. The system is comprised of sensor nodes, webcams, and long range communication base stations. FireWxNet is deployed at the fire site to study the fire behavior using weather data measurements and visual images. Temperature, relative humidity, wind speed and direction are reported every half an hour while cameras provide a continuous view of the current fire condition. The experimental results indicate that the system is capable of providing useful data for fire behavior analysis. Our system is designed for a different application which is forest fire detection.

A Forest fire Surveillance System is designed in [49]. The authors provide a general structure for sensor networks and provide details for a forest fire detection application. The sensor types, operating system and routing protocol are discussed. In their design, sensor

nodes use a minimum cost path forwarding to send their packets to a sink node which is connected to the internet. The data is reported to a middleware which calculates the forest fire risk level according to formula defined by forestry service. The calculation is depending on the values of relative humidity, precipitation, and solar radiation of the day. The results are recorded in a database that can be accessed by web applications over the internet. We propose using a clustered network to calculate fire indexes at cluster heads where the data is more likely to be correlated. Thus, instead of communicating all sensor readings to the sink, we only report a few packets to save more energy.

1.3 Thesis Contributions

In this thesis we present efficient solutions for the problem of selecting the minimum number of sensor nodes from a set of already deployed ones to achieve k-coverage. We also apply our algorithms to design a wireless sensor network for early detection of forest fires. Specifically, our contributions [23–26] can be summarized as follows.

- We develop a centralized k-coverage algorithm, RKC, which achieves a solution of size $O(\widehat{N} \log \widehat{N})$, where $\widehat{N}$ is the minimum number of sensors required for k-coverage [25]. We implement our algorithm and compare it against two others in the literature. The logarithmic factor is only a worst-case upper bound and our simulation results show a better performance. Moreover, comparison with other k-coverage algorithms in the literature shows that our algorithm runs four orders of magnitude faster.

- A key feature of our algorithm is that it can be implemented in a distributed manner with local information and low message complexity. We design and implement a fully distributed version of our algorithm [24]. Our distributed algorithm, DRKC, does not require that sensors know their locations. Comparison with two other distributed algorithms in the literature indicates that our algorithm: (i) converges much faster than the others, (ii) activates near-optimal number of sensors, and (iii) significantly prolongs (almost doubles) the network lifetime because it consumes much less energy than the other algorithms. We extend our distributed algorithm to provide hot spot coverage. This is a highly desired feature for wireless sensor network applications that need to emphasize certain areas over others. For example, in a forest fire detection system, area near residential neighborhood may need a higher coverage degree that

others. Simulation results verify that our algorithm indeed achieves the requested coverage degrees at different regions of the monitored area.

- We design a wireless sensor network for early detection of forest fires [26]. Our system is based on the Fire Weather Index (FWI) System developed by the Canadian Forest Service [12]. The FWI System uses weather observations to produce a set of indexes which indicate the likelihood of the current weather conditions to cause a fire. We present a new aggregation paradigm to efficiently calculate the fire weather indexes in a wireless sensor network. In addition, we establish a relationship between the desired accuracy of the system and the required coverage degree.

1.4 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 describes the k-coverage problem and our solution approach. We show the correctness of our algorithm, evaluate its performance, and compare it against other works in the literature. In Chapter 3, we provide the design of a wireless sensor network for early detection of forest fires and show how our algorithms can be applied to such applications. Chapter 4 summarizes the conclusions of this thesis and outlines future directions for this research. Appendix A lists the equations used in the Fire Weather Index System.

Chapter 2

K-Coverage Algorithms and Evaluation

In this chapter, we describe the k-coverage problem and present our solution. Then we show the correctness of our algorithm, evaluate its performance, and compare it against other works in the literature.

2.1 The K-Coverage Problem and Our Solution Approach

In this section, we state the k-coverage problem in wireless sensor networks, and provide an overview of our solution. Our problem is to select a minimal subset of nodes for activation to ensure that all sensor locations are k-covered by the set of activated nodes. The k-coverage problem can formally be stated as follows.

Problem 1 (k-Coverage Problem) *Given n already-deployed sensors in a target area, and a desired coverage degree $k \geq 1$, select a minimal subset of sensors to cover all sensor locations such that every location is within the sensing range of at leas k different sensors. It is assumed that the sensing range of each sensor is a disk with radius r, and sensor deployment can follow any distribution.*

The above k-coverage problem is proved to be NP-hard by reduction from the minimum dominating set problem in [54]. The proof idea is to model the network as a graph where there is an edge between any two nodes if they are within the sensing range of each other.

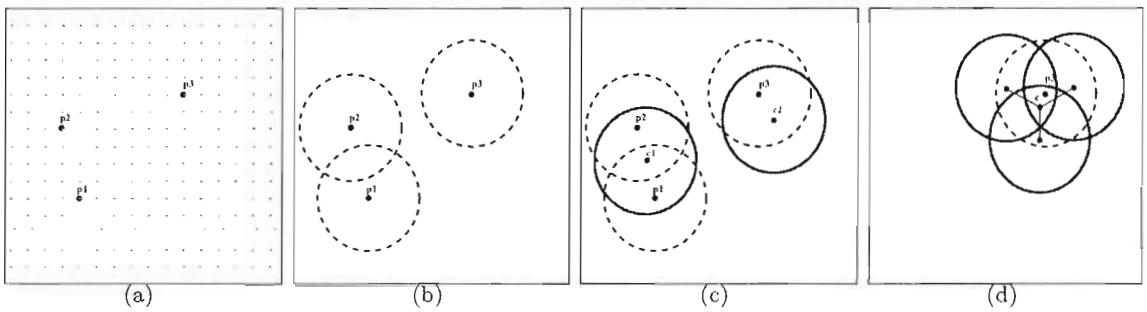


Figure 2.1: Modelling the k-coverage problem as a set system $(X, \mathcal{R})$. (a) shows the set of points which constitute X. (b) shows only three subsets of $\mathcal{R}$ that are associated with the three highlighted points in (a). (c) shows a hitting set $\{c_1, c_2\}$ that 1-covers the three subsets in (b). (d) shows one 3-flower that 3-covers only one subset in $\mathcal{R}$.

Finding the minimum number of nodes to 1-cover the set of all sensor locations is equivalent to finding the minimum dominating set for the graph. Since Problem 1 is a generalization of this problem, it is also NP-hard. We propose an efficient approximation algorithm for solving the k-coverage problem. We start describing our solution approach with the following definition of set systems and hitting sets [9].

Definition 1 (Set System and Hitting Set) *A set system $(X, \mathcal{R})$ is composed of a set X and a collection $\mathcal{R}$ of subsets of X. We say that $H \subseteq X$ is a hitting set if H has a non-empty intersection with every element of $\mathcal{R}$, that is, $\forall R \in \mathcal{R}$ we have $R \cap H \neq \emptyset$.*

Our solution does not require a grid deployment, and any node deployment such as uniform or Poisson distribution can be used. We define X to be the set of all sensor locations. Thus, we have $|X| = n$. We define the collection $\mathcal{R}$ as follows. For each point p in X, we draw a circle of radius r centered at p. All points in X that fall within that circle constitute one set in $\mathcal{R}$. Fig. 2.1(b) shows only three elements of $\mathcal{R}$ that correspond to the three highlighted points p_1, p_2, p_3 in Fig. 2.1(a). Now the minimum hitting set problem on $(X, \mathcal{R})$ is to find the minimum set of points in X that hit (intersect) all elements (disks) of $\mathcal{R}$. Fig. 2.1(c) shows a possible hitting set for the three disks of $\mathcal{R}$ shown in Fig. 2.1(b). The hitting set has two points c_1 and c_2. If we consider c_1 and c_2 to be locations of sensors, we will ensure that points p_1, p_2 and p_3 are 1-covered, because each of them is within the sensing range of at least one of the sensors located at c_1 and c_2, as shown in Fig. 2.1(c).

For k-coverage, elements in the hitting set are not locations for individual sensors. Rather, each element in the hitting set is a center of what we call a *k-flower*, which is a set of k sensors that all intersect at that center point. Fig. 2.1(d) shows one 3-flower centered at point c that 3-covers point p_3. Details of constructing k-flowers are discussed in Section 2.2. The k-coverage problem now reduces to finding a minimum hitting set where elements in that set are the centers of k-flowers. Finding the minimum hitting set is NP-hard [18], thus we try to find a near optimal hitting set. We propose an approximation algorithm that uses the concept of ϵ-nets [22], which is defined as follows.

Definition 2 (ϵ-Net) *Let $0 < \epsilon \leq 1$ be a constant. The set $N \subseteq X$ is called an ϵ-net for the set system $(X, \mathcal{R})$ if N has a non-empty intersection with every element of $\mathcal{R}$ of size greater than or equal to $\epsilon|X|$, that is, $\forall R \in \mathcal{R}$ such that $|R| \geq \epsilon|X|$ we have $R \cap N \neq \emptyset$.*

The definition of ϵ-net is similar to that of the hitting set, except that the ϵ-net is required to hit only *large* elements of $\mathcal{R}$ (ones that are greater than or equal to $\epsilon|X|$), while the hitting set must hit every element of $\mathcal{R}$. This similarity is exploited by our approximation algorithm to find a near optimal hitting by finding ϵ-nets of increasing sizes (i.e., decreasing ϵ) until one of them hits all elements of $\mathcal{R}$. For this to work, we clearly need to efficiently: (i) compute ϵ-nets, and (ii) verify coverage. We use a simple verifier that checks all points in $O(n)$ steps. Computing ϵ-nets can be done efficiently for set systems with finite VC-dimensions (defined below). Specifically, Haussler and Welzl [22] show that for any set system $(X, \mathcal{R})$ with a finite VC-dimension d, randomly sampling $m \geq \max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8d}{\epsilon}\log\frac{8d}{\epsilon}\right)$ points of X constitutes an ϵ-net with a probability at least $1-\delta$, where $0 < \delta < 1$. Notice that m does not depend on the size of X, which allows X to be arbitrarily large with no effect on the size of the ϵ-net. Brönnimann and Goodrich [9] further extend the concept of ϵ-net by assigning *weights* to elements of X. Weights accelerate the process of finding a near optimal hitting set, and help in establishing an upper bound on its size, as we discuss in Section 2.2.

We now present the definition of the VC (Vapnik and Červonenkis) dimension of a set system and the associated concept of *set shattering* [9, 22].

Definition 3 (Set Shattering and VC-dimension) *Consider a set system $(X, \mathcal{R})$ and a set $Y \subseteq X$. Y is said to be shattered by $\mathcal{R}$ if for any $A \subseteq Y$ there exists a set $B \in \mathcal{R}$ such that $Y \cap B = A$. Furthermore, the cardinality of the largest set that can be shattered by $\mathcal{R}$ is called the VC-dimension of the set system $(X, \mathcal{R})$.*

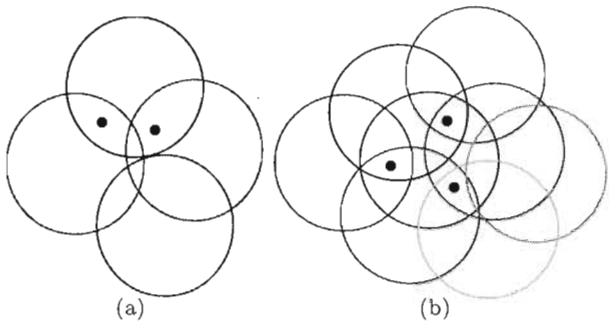


Figure 2.2: The concept of set shattering: (a) Two points shattered by four disks, and (b) three points shattered by eight disks.

Informally, Y is shattered by $\mathcal{R}$ if all subsets of Y can be constructed by intersecting Y with some set in $\mathcal{R}$. For instance, Fig. 2.2 (left) shows two points shattered by four disks, each disk contains only one subset of the four possible subsets that can be created by the two points.

The VC-dimension of our set system is proved to be 3 by the following lemma [19].

Lemma 1 *Consider the set system $(X, \mathcal{R})$, where X is the set of points, and $\mathcal{R}$ contains a disk of radius r for each point in X. This set system has a VC-dimension of 3 [19].*

To summarize, we model the k-coverage problem as a set system $(X, \mathcal{R})$ where X is the set of sensor locations and $\mathcal{R}$ is the collection of subsets of X created by intersecting disks of radius r with points of X. This set system has a VC-dimension of 3, therefore, we can efficiently implement a *net-finder* algorithm to find ϵ-nets of various sizes. Our approximation algorithm for the k-coverage problem employs the net-finder to compute ϵ-nets of increasing sizes, and for each ϵ-net it verifies the coverage until all points are sufficiently covered. We assign weights to points of X to guarantee termination and to bound the approximation factor of the output solution. Finally, each element in the output represents the center of what we call a k-flower, which is a set of k sensors that all intersect at that center point and should be activated for k-coverage.

2.2 Centralized K-Coverage Algorithm

In the previous section, we presented the theoretical foundations of our k-coverage algorithm. In this section, we present the details of the algorithm and analyze its complexity. A very

important feature of the algorithm is that it can be implemented in a distributed manner with local information and low message complexity. The distributed version of the algorithm is presented in the next section.

The pseudo code of the k-coverage algorithm, which we call RKC (Randomized k-Coverage algorithm), is given in Fig. 2.3. The algorithm takes as input the set of sensor locations X, sensing range of sensors r, and required degree of coverage k. If the algorithm succeeds, it will return a subset of nodes to activate in order to ensure k-coverage. The algorithm may only fail if activating all sensors is not enough for k-coverage because of low density. The minimum required density can be calculated as follows. If every point is to be k-covered, it has to be in the sensing range of at least k sensors. Thus, for each node p, there should be at least k other nodes inside a disk of radius r centered at p.

In every single iteration of the while loop, the algorithm tries up to $4c\log(n/c)$ $\frac{1}{c}$–nets one at a time (the for loop in lines 5–11). Each $\frac{1}{c}$–net is computed by the net-finder (Section 2.2.1), and hits all disks with weight greater than or equal to $\frac{1}{c}|X|$. For each net, the verifier checks whether this net is a hitting set, i.e., it completely k-covers all points. We use a simple verifier that checks all points in $O(n)$ steps.[1] If a net is a hitting set, the algorithm returns it and terminates. Otherwise, the algorithm doubles the weight of a point that was under covered by that net. Then, the algorithm chooses another $\frac{1}{c}$–net. Points with increased weights will have higher probability of being included in the new net. The size of each returned $\frac{1}{c}$–net is $O(c\log c)$ (see the description of the net-finder algorithm for details). The reason behind trying up to $4c\log(n/c)$ nets is that a result (Lemma 3.4) in [9] states that if there is a hitting set of size c, the weight doubling process cannot iterate more than $4c\log(n/c)$ times. This also means if we iterate beyond $4c\log(n/c)$ without finding a hitting set, it is guaranteed that there is no hitting set of size c [9]. This helps us to establish the following bound on the number of sensors required to achieve k-coverage.

Lemma 2 *The solution returned by the k-coverage algorithm is of size $O(\widehat{N}\log\widehat{N})$, where $\widehat{N}$ is the optimal number of sensors required to k-cover all sensor locations.*

Proof: Suppose that the algorithm terminates with c and the optimal number of nodes required for k-coverage is $\widehat{N} \leq c$. This means that the algorithm has failed to find a

[1] Asymptotically more efficient verifiers are possible to design using order-k Voronoi diagrams [48]. However, these verifiers are complex to implement in practice, and the performance gain is not significant due to the large constants in the time complexity.

Randomized K-Coverage: RKC(X, r, k)

```
1.   c = 1; // sets the initial size of ε-net
2.   while (net-size(1/c) ≤ n) do
3.      set weights of all points to 1;
4.      ε = 1/c;
5.      for i = 1 to 4c log n/c
6.         N = net-finder(X, k, ε, r);
7.         u = verifier (X, N, k, r);
8.         if (u == null)
9.            return N;
10.        else
11.           double weight of u;
12.     c = 2 × c;
13.  return ∅;
```

Figure 2.3: A centralized approximation algorithm for the k-coverage problem.

solution for $c/2$. Since the algorithm iterated $4(c/2)\log(n/(c/2))$ times, doubling weights of uncovered points in each iteration, then by Lemma 3.4 in [9], there is no hitting set of size $c/2$. That is, we must have $\widehat{N} > c/2$. Therefore, we have $c < 2\widehat{N}$. Since the size of the $\frac{1}{c}$-net is $O(c \log c)$, the size of the solution is $O(\widehat{N} \log \widehat{N})$.

Our simulation results (Section 2.4) show that the upper bound in this lemma is indeed very conservative, and usually our algorithm produces better results.

Next, we prove the time complexity of the algorithm in the following lemma.

Lemma 3 *The k-coverage algorithm terminates in time $O(n \log n(T_F + T_V))$, where T_F and T_V are the running times of the net-finder and verifier algorithms, respectively.*

Proof: Since c is doubled at each step, the number of iterations of the while loop (lines 2–12) is at most $\log n$, and the algorithm indeed terminates. Furthermore, the algorithm iterates for $4c\log(n/c)$ steps for each c (lines 5–11). Since $c < n$, the number of iterations is bounded by $O(n)$. Thus the algorithm running time is $\log n\,[T_i + O(n)\,(T_F + T_V + T_d)]$, where T_i and T_d are the cost of initializing and doubling weights, respectively. T_i for all points takes no more than n steps and T_d takes $O(1)$ steps. Substituting, we get the running time as $O(n \log n(T_F + T_V))$.

Net-finder(X, k, ϵ, r)

```
1.  δ = 10^{-6};
2.  net = ∅; // net of k-flowers
3.  for i = 1 to net-size(ε)
4.    q = getRandomPoint(X); //biased
5.    f = createFlower(X, q, k, r);
6.    add f to net;
7.  return net;
```

Figure 2.4: The net finder algorithm.

2.2.1 The Net-Finder Algorithm

The idea of the net-finder algorithm is based on Corollary 3.8 in [22], which states that randomly selecting at least $\max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{24}{\epsilon}\log\frac{24}{\epsilon}\right)$ points of the set X yields an ϵ-net with a probability at least $1-\delta$, where $0 < \delta < 1$. Selecting a small $\delta = 10^{-6}$ will yield an ϵ-net with probability almost 1.

The pseudo code of the net-finder algorithm is given in Fig. 2.4. Note that the term *net-size(ϵ)* denotes the number of k-flowers in the ϵ-net, which is equal to $\frac{24}{\epsilon}\log\frac{24}{\epsilon}$ as shown in Lemma 4 below. The algorithm iterates for *net-size(ϵ)* steps, and in every iteration, selects a random point q biased based on the weights. Then it finds a k-flower centered at q and adds it to the *net*. Any point q is selected with probability $w(q)/w(X)$, where w is a function which assigns weights to points. The weight of a set is the summation of weights of all points in that set. After the center point q of the k-flower is selected, k other points $p_1, \ldots, p_k$ are selected uniformly inside a disk of radius r centered at q. The location of each of these points is given by: $p_i = (x_q + d_i\cos\theta_i,\ y_q + d_i\sin\theta_i)$, where x_q and y_q are coordinates of q, and θ_i and d_i are selected at random from $[0, 2\pi]$ and $[0, r]$, respectively. We explore different ways of choosing d_i, and θ_i in Section 2.4.

The following lemma provides the time complexity of the net-finder algorithm, and the size of the net returned.

Lemma 4 *The algorithm net-finder terminates in $O(n\log n)$ steps, and returns an ϵ-net of size $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$.*

Proof: The algorithm iterates for *net-size(ϵ)* steps. In each iteration, one point is selected based on weights as follows. Weights are maintained in a cumulative list. Each

entry in the list contains the sum of all weighs from the head of the list up to and including the current entry. We generate a uniform random number between 1 and the sum of weights of all points. Then, we perform a binary search on the cumulative field, choosing the closest point that has cumulative weight greater than or equal to the random number. This selection process takes $O(\log n)$ steps. After selecting the center point, k other points (k-flower) are selected in constant time. Since *net-size*(ϵ) can be at most n, the net-finder iterates up to $O(n)$ times. Therefore, the total running time of the net-finder is $O(n \log n)$.

From Corollary 3.8 in [22], randomly selecting $m \geq \max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{24}{\epsilon}\log\frac{24}{\epsilon}\right)$ points yields an ϵ-net with a probability at least $1-\delta$. If we set δ to a very small value 10^{-6}, the probability of finding an ϵ-net will be almost 1, and the second term in max(.) will dominate the first term. Thus, the number of k-flowers added to the *net* variable during the for-loop in Fig 2.4 will be at most $\frac{24}{\epsilon}\log\frac{24}{\epsilon}$. Since each k-flower contributes at most k sensors, the ϵ-net will have no more than $k(\frac{24}{\epsilon}\log\frac{24}{\epsilon}) = O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ sensors.

Remark: A more efficient net-finder algorithm, i.e., one that returns an ϵ-net of size $O(\frac{1}{\epsilon})$, is possible to design [38]. However, the constant in this linear bound is quite high. Moreover, the algorithm involves triangulation which requires sensors to be aware of their locations, and more importantly, it is not clear how the algorithm can be implemented in a distributed manner. Therefore, although the efficient net-finder in [38] would make our RKC algorithm produce a solution that is a constant factor from the optimal, we opt to use the simpler net-finder algorithm because it can be implemented in a distributed manner, and it produces near-optimal results on the average, as shown by our simulations in Section 2.4.

2.2.2 Algorithm Correctness and Complexity

The following theorem proves that our algorithm is correct, provides its time complexity, and proves the upper bound on the solution.

Theorem 1 *The k-coverage algorithm (RKC) in Fig. 2.3 guarantees that every point in the area is k-covered, terminates in $O(n^2 \log^2 n)$ steps, and returns a solution of size at most $O(\widehat{N} \log \widehat{N})$, where $\widehat{N}$ is the minimum number of sensors required for k-coverage.*

Proof: Suppose that the algorithm terminates by providing a set S of sensor locations. By construction, this set of points is guaranteed to hit every disk of radius r. Since for our set system $(X, \mathcal{R})$, we put a disk in $\mathcal{R}$ for each point $p \in X$, there should be at least one

element (i.e., a k-flower) in S that hits the disk centered at p. In addition, the center of each sensor in the k-flower is within a distance r from p (see Section 2.2.1 for details on constructing k-flowers). Therefore, p is k-covered by sensors of this k-flower. Hence, all points are k-covered by sensors in S.

The time complexity follows from Lemmas 3, and 4, and by using a simple verifier that checks all n points in $O(n)$ steps. The bound on the solutions size follows from Lemma 2.

2.3 DRKC: Distributed Randomized K-Coverage Algorithm

In the previous section, we presented a centralized algorithm for the k-coverage problem. A key feature of this algorithm is that it does not rely heavily on global information. Therefore, it can be implemented in a distributed manner. In this section, we present a decentralized version of our k-coverage algorithm. We start with an overview describing how the decentralized algorithm emulates the centralized one. Next, we present the details and the pseudo code of the distributed algorithm. Then, we analyze the communication complexity of the distributed algorithm.

2.3.1 Overview of DRKC

Our centralized k-coverage algorithm (shown in Fig. 2.3) maintains two global variables: the size of the current ϵ-net, and weights of all points. At every iteration of the outer loop, the size of the ϵ-net is doubled, and at every iteration of the inner loop, the weight of one under-covered node is doubled. The basic idea of our distributed algorithm, which we call DRKC (Distributed Randomized k-Coverage algorithm), is to emulate the centralized algorithm by keeping *local* estimates for these two global variables. To simplify the presentation, we first assume that all nodes are time-synchronized. In Section 2.4, we show through simulations that only coarse-grained time synchronization is sufficient for our algorithm.

The ϵ-net size is estimated as follows. All nodes keep track of the desired ϵ-net size using the local variable $netSize$, which is initially set to 1. Since nodes execute the same loop iteration at about the same time, they can get an accurate estimate of the desired size of the ϵ-net for the current round. This is because the ϵ-net size is simply doubled in every iteration. Knowing the desired ϵ-net size enables nodes to independently contribute to the current ϵ-net in a way when all contributions are added up, the desired global ϵ-net is produced.

Distributed K-Coverage Algorithm (DRKC)

DRKC Sender

```
1. while (true) {
2.     /* initialize parameters */
3.     weight = 1, totalWeight = n, netSize = 1;
4.     curCoverage = 0, state = TEMP;
5.     while (netSize ≤ n) {
6.         /* activate neighbors to achieve k coverage */
7.         if (netSize × (weight/totalWeight) > rand()) {
8.             state = ACTIVE;
9.             reqCoverage = k − curCoverage;
10.            P_a = reqCoverage/(neighborSize − curCoverage);
11.            broadcast an ACTIVATE message containing P_a and reqCoverage to neighbors;
12.        }
13.        wait for NOTIFY messages;
14.        /* verify k-Covergae */
15.        if (curCoverage ≥ k) { break; }
16.        /* update variables for next iteration */
17.        if (1/(n − netSize) > rand()) { weight = weight × 2; }
18.        netSize = netSize × 2;
19.        totalWeight = totalWeight + totalWeight/n;
20.    }
21.    if (state ≠ ACTIVE ) { state = SLEEP; }
22.    wait until end of round;
23.}
```

DRKC Receiver

```
/* upon receiving a message msg */
1. if (msg.type == ACTIVATE and msg.P_a > rand()) } /* chosen to be active */
2.     /* wait random time to reduce collision */
3.     send a NOTIFY message to msg.source after int_rand(0, msg.reqCoverage) × T_m
sec;
4.     state = ACTIVE;
5. }
4. update (neighborSize, curCoverage); /* based on msg.source */
```

Figure 2.5: A distributed algorithm for the k-coverage problem.

Node weights are used in the net-finder algorithm (Fig. 2.4) to add nodes to the current ϵ-net biased on their weights. That is, a node becomes part of the ϵ-net with a probability proportional to its weight relative to total weight of all nodes. To make this decision locally, a node does not need to know the weight of every other node. Rather, a node needs only the aggregate weight of all nodes in the network. A node uses the variable $totalWeight$ to estimate this aggregate. $totalWeight$ is initialized to the number of nodes in the network n. In the centralized algorithm, in every iteration of the inner loop, the weight of only one under-covered node is doubled. In the distributed algorithm, an under-covered node doubles its weight with probability $1/n_u$, where n_u is the number of under-covered nodes in the network. n_u is approximated locally as $(n - netSize)$. Thus, the expected number of nodes that double their weights is equal to 1 in each loop iteration, which is the same as in the centralized case. Now since the total weight is increased by the weight of a single under-covered node in each iteration, a node can estimate the total weight by adding the average weight of nodes $(totalWeight/n)$ to its own current value of $totalWeight$.

Estimating the current ϵ-net size and the total weight in the network allows a node to decide (locally) whether it should be a member of the ϵ-net. A node decides to be part of the ϵ-net with a probability $p = (weight/totalWeight) \times netSize$. If a node is chosen, it will activate k other nodes to create a k-flower as in the centralized algorithm.

Finally, k-coverage verification in the centralized algorithm is done by checking all nodes one by one. In the distributed algorithm, each node independently checks its own coverage by listening to messages exchanged in its neighborhood, and counting number of active nodes. A node terminates the algorithm if it is sufficiently covered. Otherwise, it doubles its weight with probability $1/n_u$, and starts another loop iteration. A node may also terminate the algorithm if it has been looping for $\log n$ steps without getting sufficiently covered, which can occur because of low node density.

2.3.2 Description of DRKC

DRKC works in rounds of equal length. The round length measured in real time is chosen to be much smaller than the average lifetime of sensors. In the beginning of each round, every node runs DRKC independent of other nodes. A number of messages will be exchanged between nodes to determine which nodes will be active during the current round, and which will sleep until the beginning of the next round. We denote the time it takes the DRKC protocol to determine active/sleep nodes as the *convergence time*. After convergence, no

node changes its state and no protocol messages are exchanged until the beginning of the next round.

The pseudo code of DRCK is given in Fig. 2.5. A node can be in one of three states: ACTIVE, SLEEP, and TEMP. In ACTIVE state, all modules (transmission, receiving, and sensing) are turned on, while all modules are turned off in SLEEP state. The TEMP state is a transient state in which transmission and receiving modules are on. The sensing module is set to its state in the immediate preceding round. This is done to avoid any coverage outage during round transitions. A node starts a round in the TEMP state, where it initializes parameters such as $netSize$, $weight$, and $totalWeight$. The $neighborSize$ variable is updated during each round, and its value is retained across rounds.

After initialization, the algorithm iterates up to $\log n$ times in the while loop, or until it achieves k coverage. In each iteration, a node decides to be a member of the current ϵ-net with a probability proportional to its weight, as described in the previous subsection. If a node is chosen, it broadcasts an ACTIVATE message to its neighbors in order to increase its own coverage to be k. The ACTIVATE message includes two parameters: $reqCoverage$ and P_a. The first parameter determines the number of additional nodes needed to achieve k coverage at the sender of the ACTIVATE message. P_a is calculated as $reqCoverage/(neighborSize - curCoverage)$, where $curCoverage$ is the current degree of coverage at the node. When a neighbor receives an ACTIVATE message, it becomes active with probability P_a. P_a is so chosen to make the expected number of newly activated nodes equal to $reqCoverage$. If a node decides to be active, it broadcasts a NOTIFY message informing all its neighbors that it has become active. To reduce collisions between NOTIFY messages, a node waits a random period between 0 and $reqCoverage \times T_m$, where T_m is the average transmission time of a message. After waiting enough time to receive NOTIFY messages, a node verifies its own coverage. If the node is sufficiently covered, it terminates the algorithm in the current round and waits until the beginning of the next round. Otherwise, another loop iteration is needed. Before starting the new iteration, a node doubles its weight with probability $1/(n - netSize)$, and updates the $netSize$ and $totalWeight$ variables.

2.3.3 Communication Complexity of DRKC

In the following theorem, we provide the average- and worst-case communication complexities of the DRKC protocol. We note that the two types of messages exchanged in the

protocol (ACTIVATE and NOTIFY) have fixed sizes. Therefore, we analyze the communication complexity in terms of number of messages exchanged.

Theorem 2 *The number of messages sent by a node in any round of the DRKC protocol is* $O(\log n)$ *in the worst case,* $O(1)$ *on average (over all nodes).*

Proof: In the worst case, a node iterates up to $\log n$ times in the while loop (lines 5–20 in Fig 2.5), and it may send one ACTIVATE message in each iteration. Thus, the maximum number of ACTIVATE messages sent by a node in a round is $O(\log n)$. A node can also reply by a NOTIFY message to ACTIVATE messages sent by its immediate neighbors. A node broadcasts a NOTIFY message if it decides to become active which may only occur once in each round. Therefore, the worst-case message complexity is $O(\log n)$.

For the average-case analysis, we consider the whole network, and assume that nodes are time synchronized. In each iteration of the while loop, the algorithm chooses *netSize* number of nodes to send ACTIVATE messages. Since *netSize* starts at 1 and is doubled in every iteration, the total number of ACTIVATE messages is at most $\sum_{j=0}^{\log n} 2^j = 2n - 1$. Each ACTIVATE message generates up to k NOTIFY messages in response. Thus, the total number of messages sent in the whole network in a round is $O(n)$. Furthermore, because nodes in an ϵ-net are chosen randomly, ϵ-nets in different rounds are expected to contain different nodes. Therefore, across rounds, the total message load is distributed over all n nodes. Thus, the average number of messages sent by a node in a round is $O(1)$.

2.4 Performance Evaluation

In this section, we evaluate various aspects of our k-coverage algorithms. We start by showing that our centralized algorithm ensures that the requested coverage degrees are satisfied, produces close to optimal results, and runs much faster than other centralized algorithms in the literature. Then, we analyze the performance of our decentralized algorithm, and show that its performance is close to the centralized one, converges in a short period of time, does not require fine-grained timing synchronization , and outperforms all other distributed k-coverage algorithms. Before presenting our results, we discuss our experimental setup and the algorithms implemented.

2.4.1 Algorithms Implemented and Experimental Setup

The closest works to ours are [61] and [54], which we choose for comparison. Thus, we implemented six algorithms in total: three centralized and three distributed. We implemented our RKC and DRKC algorithms in Figs. 2.3 and 2.5. The centralized algorithm in [61] works by iteratively adding nodes to an initially empty set based on a measure called *k-benefit*. We refer to this algorithm as CKC. The implementation of CKC is provided by its authors. For CKC, we set the communication range of sensors as twice the sensing range to allow the algorithm to find any solution for k-coverage without incurring the overhead of ensuring connectivity. The decentralized algorithm in [61], called DPA, employs a localized pruning technique: All nodes start marked active and try to unmark themselves by checking the connectivity and coverage in their neighborhood. We implemented DPA without the connectivity condition. We validated our implementation by running the same experiments as in [61] and obtained the same results. The centralized algorithm in [54], called, LPA, solves the k-coverage problem by modeling it as an integer linear program and then relaxing it to a linear program. We use LPSOLVE [36] to solve the linear program. LPSOLVE is an open source tool for solving mixed integer linear programming problems. The distributed algorithm in [54], called, PKA, uses a similar pruning idea as DPA. We implemented PKA and validated our implementation by comparing the results with those in [54].

To conduct fair comparisons, we use the same experimental set up as in [61]: We randomly deploy $5,000$ sensors in an area of $40m \times 40m$ with a uniform distribution. We vary coverage degree k between 1 and 8 in our simulation. Sensing range of sensors is fixed at 4 meters. We set the round length for our DRKC algorithm at 50 seconds. All running times are measured on a Xeon (P4) machine with 3.6 GHz CPU and 4 GB of RAM. The operating system is Linux Suse 10.0.

We employ the energy model in [55] and [59], which is based on the Berkeley Mote hardware specifications. In this model, the node power consumption in transmission, reception, idle and sleep modes are 60, 12, 12, and 0.03 mW, respectively. As in [59], we express the energy model as ratios: $20 : 4 : 4 : 0.01$. The initial energy of a node is assumed to be 60 Jules which allows a node to operate for about $5,000$ seconds in reception/idle modes.

We assume that the size of a packet containing one integer value (e.g., node's ID) is 40 bytes, and each integer value added to the packet increases its size by 4 bytes. The wireless channel capacity is assumed to be 32 Kbps, therefore the transmission time is 10 ms for a

message of size 40 bytes. For DPA and PKA, we ignore the overhead of the *priority* field included in messages exchanged between sensors. Therefore, when a node broadcasts its neighborhood information, the size of the message is assumed to be $40 + 4l$ bytes, where l is the number of its neighbors. DRKC does not broadcast neighborhood information and therefore uses fixed size messages of 40 bytes each.

Unless otherwise specified, the above parameters are used, and each experiment is repeated 10 times with different seeds and the average over all of them is reported.

2.4.2 Evaluation of our Centralized K-Coverage Algorithm (RKC)

Our first set of experiments studies the coverage degrees achieved by our centralized algorithm. We vary the requested coverage degree k between 1 and 8 and observe the achieved coverage at every single point in the area. Some of the results are shown in Fig. 2.6, where the x-axis shows the observed coverage degree and the y-axis shows the percentage of points that achieve that degree. The figure shows that while the RKC algorithm achieves the goal: 100% of the points are sufficiently covered, the percentage of points with observed coverage degree higher than k decreases fast. This is important because, while coverage redundancy might be desirable, it should be controlled in order to reduce interference.

In the next set of experiments, we compare our centralized RKC algorithm against two other centralized k-coverage algorithms which are the closest to our work: CKC [61] and LPA [54]. As shown in Fig. 2.7(a), RKC runs several order of magnitudes faster than LPA and CKC. For instance, for $k = 4$, our algorithm terminates in less than 1 second, while LPA takes 3 minutes and CKC takes 2 hours. Notice that y-axis is shown in logarithmic scale. Moreover, as shown in Fig. 2.7(b), the percentage of active sensors resulted by our algorithm is consistently lower than that of LPA and is very close to that of CKC for all values of k.

In the above experiment we could not use large network sizes in the comparison, because CKC took very long time in some cases and it did not even terminate in others. Our algorithm is designed for large-scale sensor networks, thus we need to study its efficiency in these networks. By efficiency we mean how close the number of active sensors computed by our RKC algorithm is to the optimal number of sensors. Since the optimal number of active sensors are prohibitively expensive to compute (NP-hard problem), we compare the results of our RKC algorithm against the asymptotic necessary and sufficient conditions for k-coverage proved in [32] for uniformly deployed sensors. These conditions are obtained for

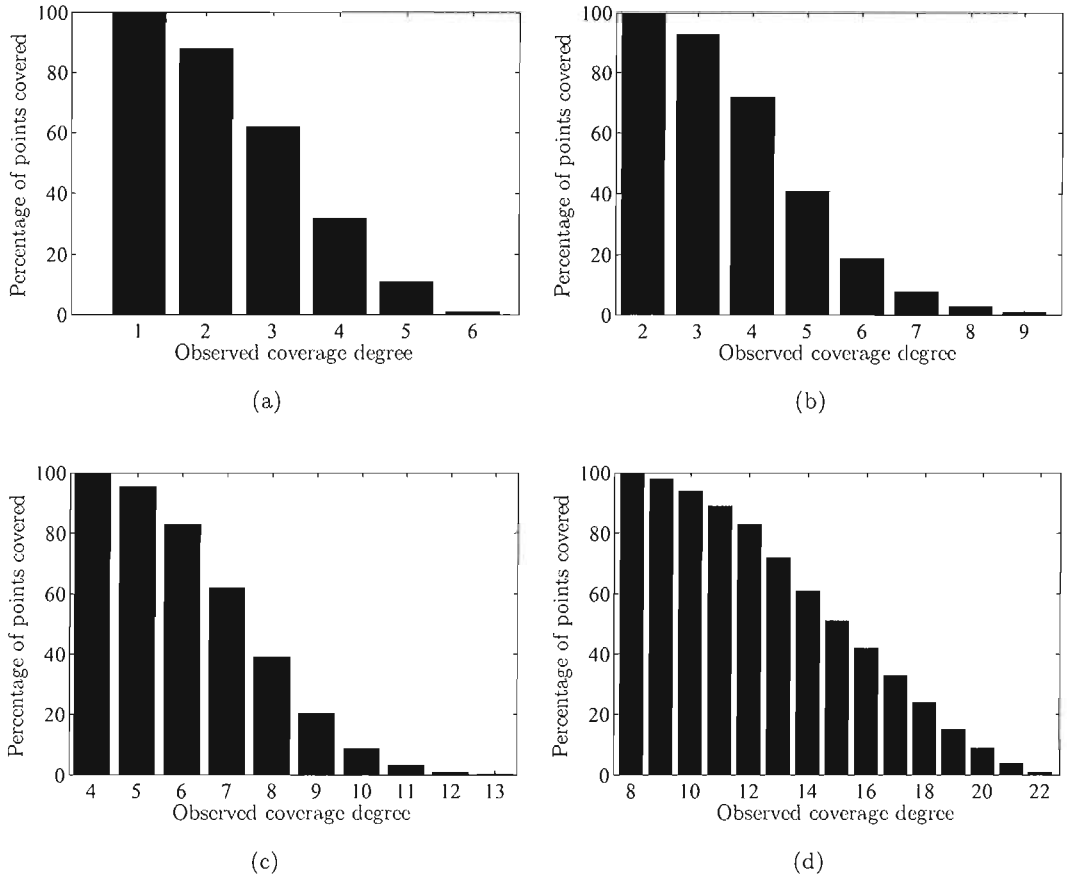


Figure 2.6: Coverage achieved by our centralized k-coverage algorithm (RKC). The figure shows the achieved coverage distribution for various requested coverage degrees: (a) $k = 1$, (b) $k = 2$, (c) $k = 4$, and (d) $k = 8$.

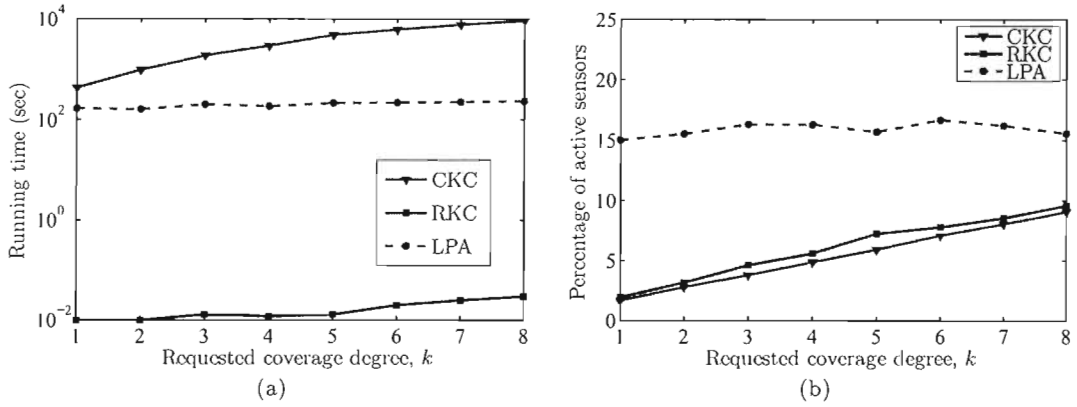


Figure 2.7: Comparing the performance of our centralized k-coverage algorithm (RKC) versus two others: (a) Running time, and (b) Percentage of active sensors.

networks with sensors that can fail (or sleep) with a probability $1-p$, and they require the existence of a slowly growing function $\phi(np)$. For our comparison, we set $p = 1$, i.e., sensors are always on. Then we compute the minimum number of sensors that are necessary to achieve k-coverage, and the minimum number of sensors that are sufficient to achieve k-coverage. We set $\phi(np) = \sqrt{\log\log(np)}$, which is the function used by the authors of [32] in their simulations. We use a large area of size $1000m \times 1000m$ with $30,000$ nodes and vary the sensing range r. The results for $k = 4$ are shown in Fig. 2.8, where Nec_cond and Suff_cond denote the necessary and sufficient conditions, respectively. The figure shows that our algorithm does not unnecessarily activate too many sensors, because its output is very close to the necessary condition. The results of this experiment show that the worst-case logarithmic factor proved in Theorem 1 is very conservative, and on average our centralized algorithm produces near-optimal number of active sensors.

Our last experiment studies the effect of k-flower selection strategy on the performance of the RKC algorithm. Two different strategies can be used in the net-finder algorithm. In the first strategy which we call *uniform*, all k center points of sensors are selected randomly inside a circle of radius r centered at the reference point. The other strategy, called *min-overlap*, is to choose the k furthest points which are placed regularly around the reference point as shown in Fig. 2.9. This is done by selecting k center points at distance r at angles $i\frac{2\pi}{k}$ for $0 \leq i \leq k$. This strategy minimizes the overlap between sensors of a k-flower. As shown in Fig. 2.10, min-overlap strategy outperforms uniform strategy in term of percentage

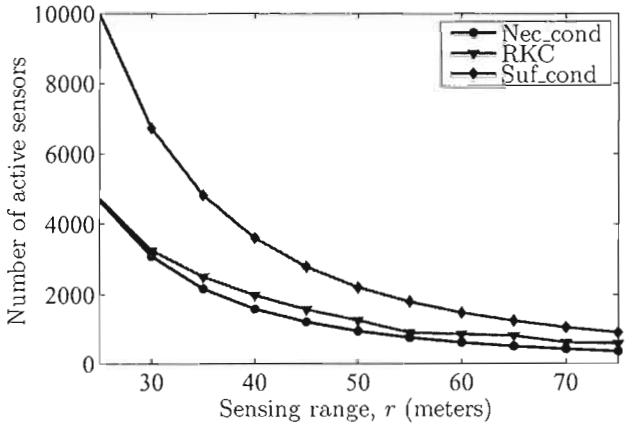


Figure 2.8: Efficiency of our centralized k-coverage algorithm (RKC). The figure compares the number of active sensors produced by our RKC algorithm versus the necessary (Nec_cond) and sufficient (Suf_cond) conditions proved in [32].

of active sensors which verifies the intuition that decreasing intra-flower overlap will reduce the number of active sensors.

2.4.3 Evaluation of our Distributed K-Coverage Algorithm (DRKC)

We start by verifying that the DRKC algorithm achieves the requested coverage degree. As in the case of the centralized algorithm, we vary the requested coverage degree k between 1 and 8 and observe the achieved coverage at every point in the area. The results for $k = 1, 4$

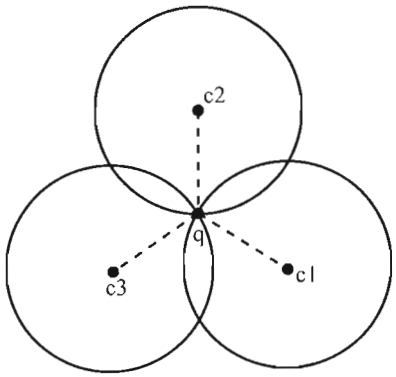


Figure 2.9: A 3-flower with minimum overlap between sensors.

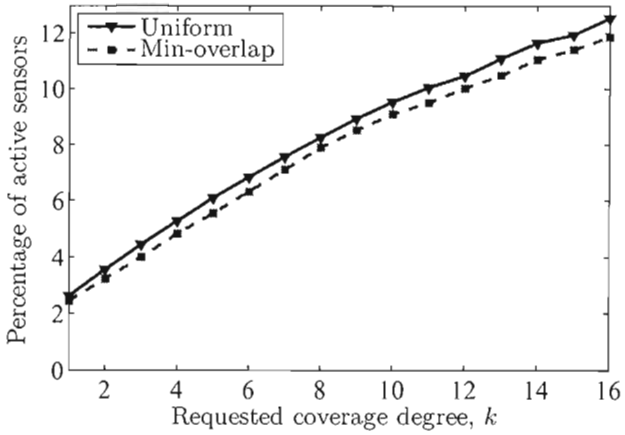


Figure 2.10: The impact of using different k-flower selection strategies on the performance of the RKC algorithm.

and 8 are shown in Fig. 2.11. The figure confirms that 100% of the points meet the coverage requirements.

Next we compare the number of active sensors resulted from the distributed DRKC algorithm versus the number of sensors resulted from the centralized RKC algorithm, which was shown to be close to the optimal number in the previous section. In Fig. 2.12(a), we vary the requested coverage degree k and fix all other parameters for both the centralized and distributed algorithms. And in Fig. 2.12(b), we vary the number of deployed sensors (i.e., sensor density) while fixing everything else ($k = 4$ in this case). Both figures show that the performance of the distributed algorithm is very close to that of the centralized one, especially for high-density networks. This means that the distributed algorithm is expected to activate close-to-optimal number of sensors to achieve k coverage.

In Section 2.3, we assumed that sensors start a round of the DRKC algorithm at the same time, i.e., they are time-synchronized. In this experiment, we examine the effect of clock drift on the performance of DRKC. By clock drift we mean that sensors may start a round at different points in time. We add a random offset to the clock of each sensor. This offset is uniformly distributed between 0 and 500 ms. The requested coverage degree is fixed at $k = 4$ for this experiment. Fig. 2.13 summarizes the impact of clock drift on the performance on the DRKC algorithm. As Fig. 2.13(a) shows, the k-coverage of the area was not seriously impacted: The percentage of points that achieved less than the requested coverage degree is

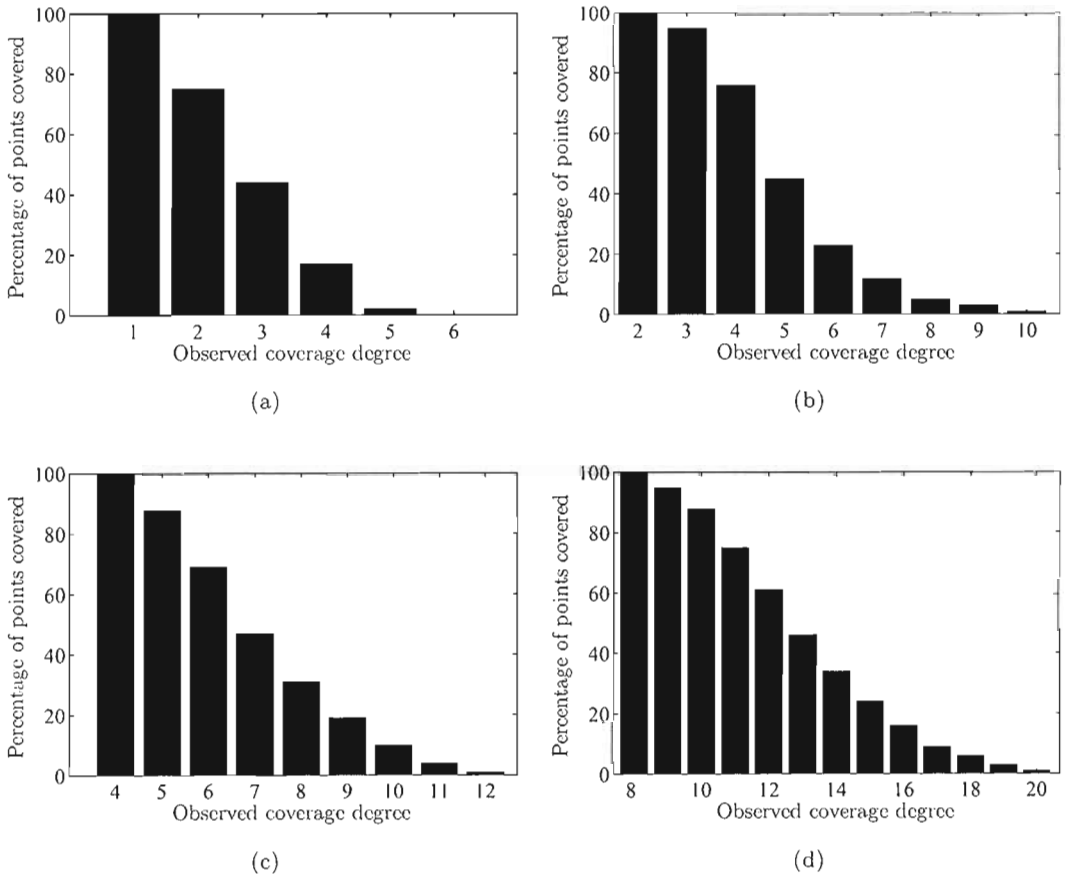


Figure 2.11: Correctness of our distributed k-coverage algorithm (DRKC). The figure shows the achieved coverage distribution for various requested coverage degrees: (a) $k = 1$, (b) $k = 2$, (c) $k = 4$, and (d) $k = 8$.

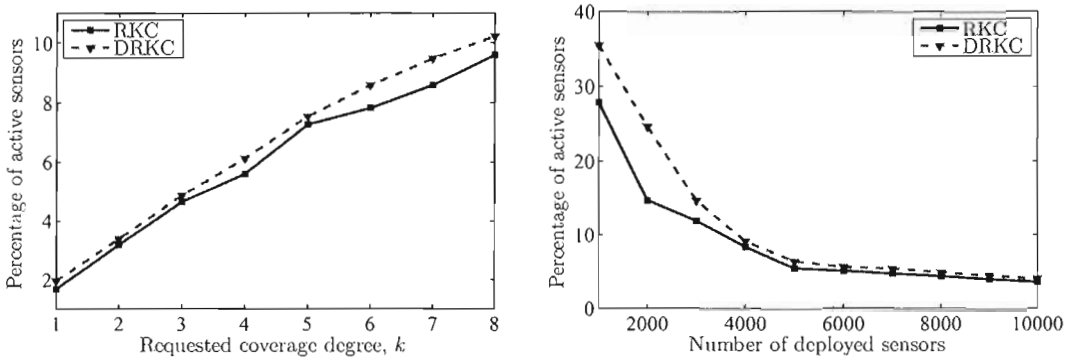


Figure 2.12: Comparing the number of sensors activated by the distributed DRKC algorithm versus that of the centralized RKC algorithm for different: (a) coverage degrees, and (b) sensor densities.

less than 2% for clock drifts as large as 400 ms. Furthermore, by investigation the coverage distribution of all points in the area, we found that these under covered nodes are near the borders of the area. Fig. 2.13(b) indicates that only a minor increase in the average number of messages exchanged between sensors may result from large clock drifts. In addition, as shown in 2.13(c), the convergence time of the DRKC algorithm did not suffer much. Convergence time is defined as the time it takes the algorithm to decide the final state for each and every sensor (either active or sleep) in a round, and it is desired to be small. Even with large clock drifts, the convergence time of our algorithm is less than 1.5 seconds. The results of this experiment confirm that our distributed k-coverage algorithm does not require fine-grained synchronization mechanisms, which may be costly to implement in large-scale networks.

Now we compare our distributed DRKC algorithm against two other distributed k-coverage algorithms: DPA [61] and PKA [54] along various performance metrics. We first vary the coverage degree k and compute the number of sensors activated by each algorithm to achieve the requested coverage degree. We normalize the number of active sensors by the total number of deployed sensors. We also determine the convergence time of each algorithm. The results are given in Fig. 2.14. Fig. 2.14(a) indicates that the DRKC algorithm converges much faster than the other two algorithms: It converges in less than 1 second compared to 25 seconds for the others. Short convergence times are desirable because the network

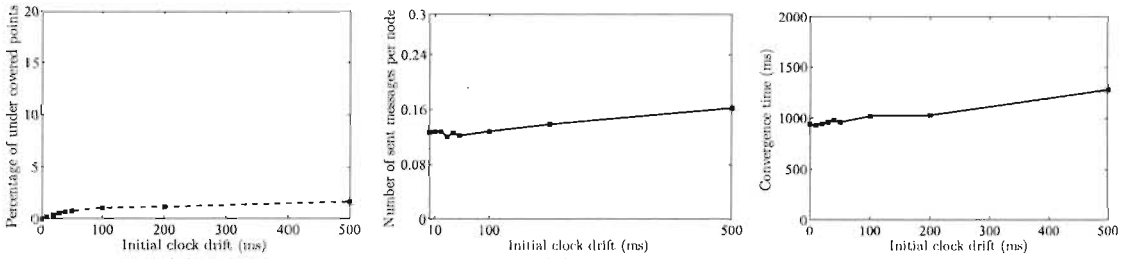


Figure 2.13: The impact of clock drift on the performance of our distributed k-coverage algorithm: (a) Percentage of under covered points, (b) Average number of messages sent per node, and (c) Convergence time.

reaches steady state faster. This reduces the energy consumed by sensors as we will show later in this section. Fig. 2.14(b) shows that DRKC always results in much smaller numbers of activated sensors. For a coverage degree of 4, for instance, DRKC activates about 5% of the deployed sensors while the other two algorithms activate at least double that number (more than 12%).

In the next set of experiments, we compare the energy consumption of the DRKC, DPA, and PKA distributed k-coverage algorithms. In Fig. 2.15(a), we plot the total remaining energy in all sensors in the network as the time progresses. The figure clearly shows that DRKC consumes energy at a much lower rate than the other two algorithms. This can be explained by the results in Fig 2.14, which show that the DRKC algorithm puts a larger fraction of sensors in sleep mode and converges much faster than the other two algorithms. The results in Fig. 2.15(a) were obtained at a specific sensor density. In the next experiment, we vary the number of deployed sensors and measure the average per-node energy consumption. As shown in Fig. 2.15(b), the amount of energy consumed per node is much lower (about one-fifth) in DRKC than the other two algorithms. Moreover, the difference between the per-node energy consumption in the DRKC algorithm and the other two algorithms grows larger as the network density increases. This is because messages exchanged between sensors in both DPA and PKA algorithms grow in size and number as the average number of neighbors per node grows. Our algorithm uses fixed-size messages. Larger messages require longer transmission times, increases chances of collision, and ultimately consum more energy.

In the last experiment, we look at the lifetime of the sensor network under different

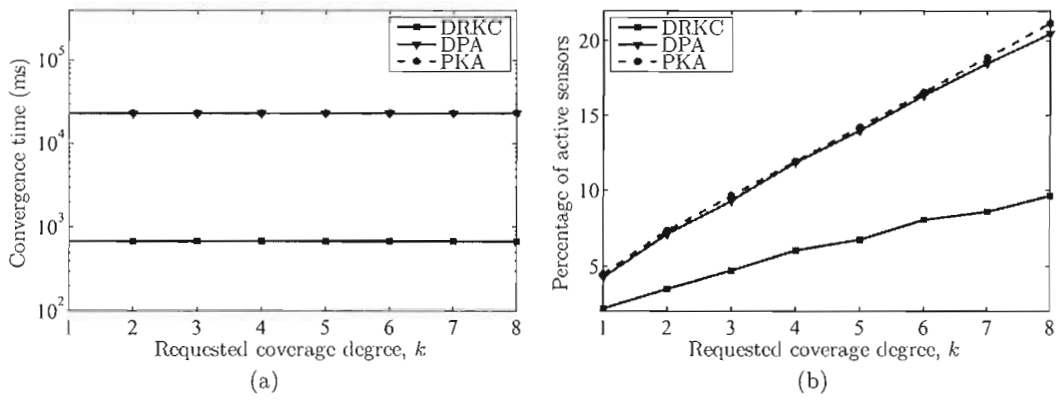


Figure 2.14: Comparing the performance of our DRKC algorithm versus two other distributed k-coverage algorithms for various coverage degrees: (a) Convergence time, and (b) Percentage of active of sensors.

distributed algorithms. Fig. 2.16 compares the percentage of alive sensors as the time progresses for the three algorithms. The figure clearly demonstrates that our algorithm prolongs (almost doubles) the lifetime of the network, because it consumes much smaller amount of energy than the other two algorithms, as shown in the previous experiment.

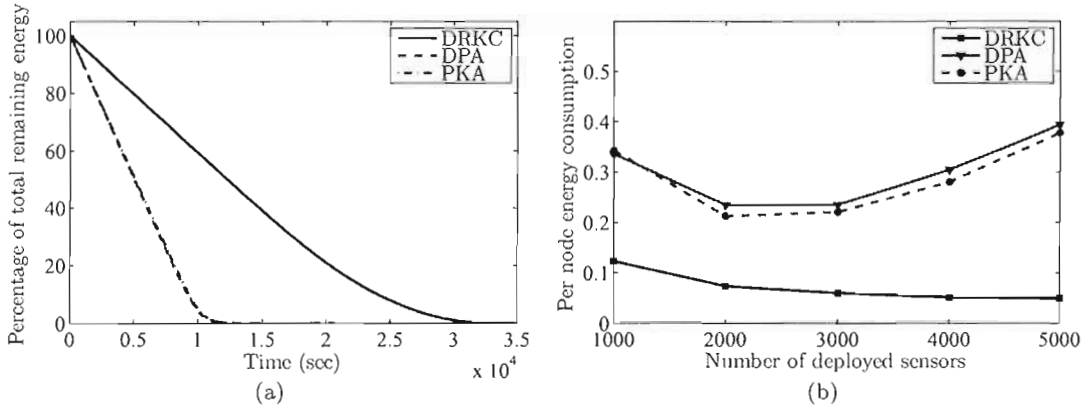


Figure 2.15: Comparing the energy consumption of our DRKC algorithm versus two other distributed k-coverage algorithms: (a) Total remaining energy, and (b) Average per-node energy consumption.

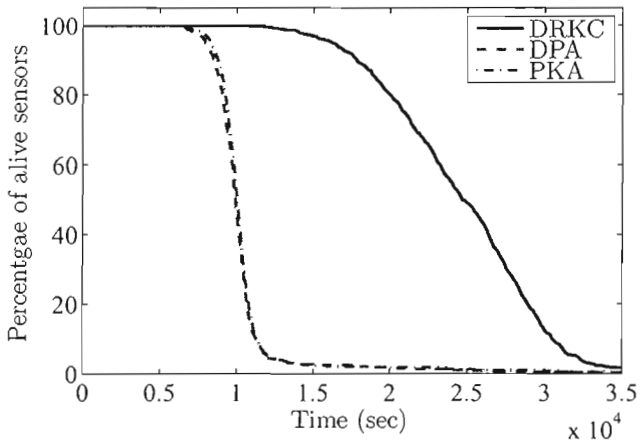


Figure 2.16: Comparing the network lifetime under different distributed k-coverage algorithms.

Chapter 3

Applying K-Coverage to Forest Fire Detection

In the previous chapter, we presented our k-coverage algorithms and evaluated its performance. In this chapter, we show how our algorithms can be used for real life applications such as forest fire detection. We describe the design of a wireless sensor network for early detection of forest fires.

3.1 Introduction

Forest fires, also known as wild fires, are uncontrolled fires occurring in wild areas and cause significant damage to natural and human resources. Wild fires eradicate forests, burn the infrastructure, speed up the extinction of species, aggravate the greenhouse effect, harm the ozone layer, and may result in high human death toll near urban areas. Common causes of forest fire include lightning, human carelessness, and exposure of fuel to extreme heat and aridity. It is known that in some cases fires are part of the natural ecosystem and they are important to the natural life cycle of indigenous habitats. However, in most cases, the damage caused by fires to public health and safety and natural resources is intolerable and early detection and suppression of fires deem crucial. For example, in August 2003, a wild fire was started by a lightning strike in the Okanagan Mountain Park in the province of British Columbia, Canada. The fire was spread by the strong wind and within a few days it had turned into a firestorm. The fire forced the evacuation of 45,000 residents and

Table 3.1: Wild fires in the province of British Columbia, Canada since 1995 [7].

Year	Total Fires	Total Hectares	Total Cost (millions)	Average Hectares per Fire	Average Cost per Hectare
2006	2,590	131,086	$156	50.6	$1,190
2005	976	34,588	$47.2	35.4	$1,365
2004	2394	220,516	$164.6	92.1	$746
2003	2473	265,050	$371.9	107.2	$1,403
2002	1783	8,539	$37.5	4.8	$4,392
2001	1266	9,677	$53.8	7.6	$5,560
2000	1539	17,673	$52.7	11.5	$2,982
1999	1208	11,581	$21.1	9.6	$1,819
1998	2665	76,574	$153.9	28.7	$2,009
1997	1175	2,960	$19.0	2.5	$6,412
1996	1358	20,669	$37.1	15.2	$1,794
1995	1474	48,080	$38.5	32.6	$801

burned 239 homes. Most of the trees in the Okanagan Mountain Park were burned, and the park was closed. Although 60 fire departments, 1,400 armed forces troops and 1,000 fire fighters took part in the fire fighting operation, they were largely unsuccessful in stopping the disaster. The official reports estimate the burned area as 25,912 hectares and the total cost as $33.8 million [6].

In the province of British Columbia alone, there have been 2,590 wild fires during 2006 [7]. These burned 131,086 hectares and costed about $156 million. Table 3.1 summarizes the extent and cost of wild fires in BC in previous years. Regretfully, other reports from Ministry of Forests and Range show that on average 45% of fires are caused by human activities, hence about half of unwanted wild fires were preventable.

Apart from preventive measures, early detection and suppression of fires is the only way to minimize the damage and casualties. Different methods for early detection of wild fires have been developed based on local conditions and advances in related technologies. Such systems include fire lookout towers, automatic video surveillance systems, firewatch aeroplanes, satellite imagery, and wireless sensor networks. In this chapter we present the design of a wireless sensor networks for early detection of forest fires. Our design is based on the Fire Weather Index (FWI) System designed by the Canadian Forest Service [12].

Figure 3.1: A fire lookout tower near Nelson, BC.

3.2 Evolution of Forest Fire Detection Systems

Traditionally, forest fires have been detected using fire lookout towers usually located at high points. A fire lookout tower houses a person whose duty is to look for fires using special devices such as Osborne fire finder [17]. Osborne fire finder is comprised of a topographic map printed on a disk with graduated rim. A pointer aimed at the fire determines the location and the direction of the fire. Once the fire location is determined, the fire lookout alerts fire fighting crew. Fire lookout towers are still in use in many countries around the world including USA, Australia, and Canada [5]. Fig. 3.1 shows a fire lookout tower near Nelson, BC, Canada.

Unreliability of human observations in addition to the difficult life conditions for fire lookout personnel have led to the development of automatic video surveillance systems [8,16,31]. Fig. 3.2 shows the typical structure of an automatic surveillance system. Most of the systems use Charge-Coupled Device (CCD) cameras and Infrared (IR) detectors installed on top of towers. CCD cameras use image sensors called CCD which contain an array of light sensitive capacitors or photodiodes. In case of fire observation or smoke activity, the system alerts local fire departments, residents, and industries. Current automatic video surveillance systems used in Germany, Canada, and Russia are capable of scanning a circular range of 10 km in less than 8 minutes [16]. The accuracy of these systems is largely affected by weather conditions such as clouds, light reflection, and smoke from agricultural or industrial activities.

Automatic video surveillance systems cannot be applied to large forest fields easily and cost effectively, thus for large forest areas either aeroplanes or Unmanned Aerial Vehicles

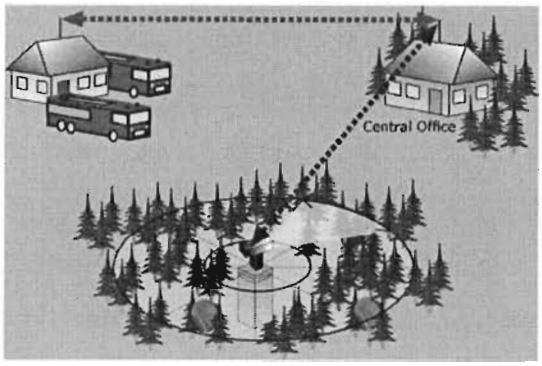


Figure 3.2: An automatic fire surveillance system.

(UAV) are used to monitor forests for fires. Aeroplanes fly over forests and the pilot alerts the base station in case of fire or smoke activity. UAVs, on the other hand, carry both video and infrared cameras and transmit the collected data to a base station on the ground that could be up to 50 km away. UAVs can stay atop for several hours and are commanded by programming or joystick controls [1].

More advanced forest fire detection systems are based on satellite imagery. Advanced Very High Resolution Radiometer (AVHRR) [4] was launched by (National Oceanic and Atmospheric Administration) NOAA in 1998 to monitor clouds and thermal emission of the Earth. Moderate Resolution Imaging Spectroradiometer (MODIS) [41] was launched by NASA in 1999 on board of the Aqua satellite to capture cloud dynamics and surface radiation from the Earth. MODIS uses its 36 spectral bands to provide a complete image of the Earth every 1 to 2 days.

Current satellite-based forest fire detection systems use data from these instruments for forest fire surveillance. The minimum detectable fire size is 0.1 hectare, and the fire location accuracy is 1 km [33, 35]. The accuracy and reliability of satellite based systems are largely impacted by weather conditions. Clouds and rain absorb parts of the frequency spectrum and reduce spectral resolution of satellite imagery which consequently affects the detection accuracy. Although satellite based systems can monitor a large area, relatively low resolution of satellite imagery means a fire can be detected only after it has grown large. More importantly, long scan period–which can be as long as 2 days–indicates that such systems cannot provide timely detection.

To summarize, the most critical issue in a forest fire detection system is immediate

response in order to minimize the scale of the disaster. This requires constant surveillance of the forest area. Current medium and large-scale fire surveillance systems do not accomplish timely detection due to low resolution and long period of scan. Therefore, there is a need for a scalable solution that can provide real time fire detection with high accuracy. We believe that wireless sensor networks (WSN) can potentially provide such solution.

Recent advances in Wireless Sensor Networks support our belief that they make a promising framework for building near real time forest fire detection systems. A wireless sensor network, is a network of small devices usually referred to as *motes*. Motes are tiny computers, with processors, RAMs, and run an operating system (e.g. TinyOS [51]). Each mote is also equipped with two modules: a radio communication module, and a sensing module. The sensing module can sense a variety of phenomena including relative humidity, temperature, and smoke which are all helpful for fire detection system [11,45]. The collected data are then exchanged between motes and/or relayed to a processing facility using the communication module. Sensor nodes can operate for months on a pair of AA batteries to provide constant monitoring during the fire season. Moreover, sensor nodes are capable of organizing themselves into a self-configuring network. Therefore, large-scale surveillance systems can be easily deployed using aeroplanes at a low cost compared to the damages and loss of properties caused by forest fires. In this work we present the design of a wireless sensor network for early detection of forest fires.

3.3 Understanding and Modelling Forest Fires

Forests cover large areas of the earth and are often home to many animal and plant species. They function as soil conserver and play an important role in the carbon dioxide cycle. To assess the possibility of fires starting in forests and rate by which they spread, we adopt one of the most comprehensive forest fire danger rating systems in North America. We use the Fire Weather Index (FWI) System developed by the Canadian Forest Service (CFS) [12], which is based on several decades of forestry research [47].

The FWI System estimates the moisture content of three different fuel classes using weather observations. These estimates are then used to generate a set of indicators showing fire ignition potential, fire intensity, and fuel consumption. The daily observations include temperature, relative humidity, wind speed, and 24-hour accumulated precipitation, all recorded at noon Local Standard Time (LST). The system predicts the peak fire danger

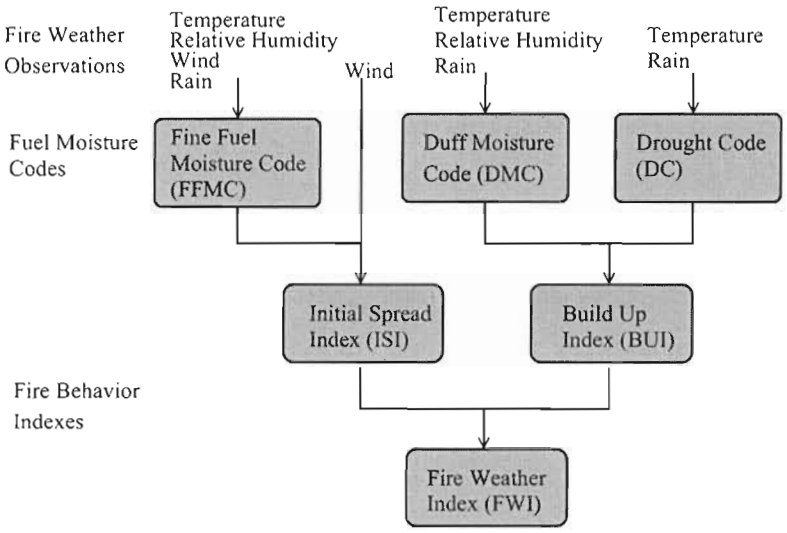


Figure 3.3: Structure of the Fire Weather Index (FWI) System.

potential at 4:00 pm LST. Air temperature influences the drying of fuels and thus affects the heating of fuels to ignition temperature. Relative humidity shows the amount of moisture in the air. Effectively, a higher value means slower drying of fuels since fuels will absorb moisture from the air. Wind speed is an important factor in determining fire spread for two main reasons: (a) it controls combustion by affecting the rate of oxygen supply to the burning fuel, and (b) it tilts the flames forward, causing the unburned fuel to be heated [46]. The last factor, precipitation, plays an important role in wetting fuels.

As shown in Fig. 3.3, the FWI System is comprised of six components: three fuel codes and three fire indexes. The three fuel codes represent the moisture content of the organic soil layers of forest floor, whereas the three fire indexes describe the behavior of fire. In the following two sections, we briefly describe these codes and indexes. In Section 3.3.3, we present how these codes and indexes can be interpreted and utilized in designing a wireless sensor network for early forest fire detection.

3.3.1 Fuel Codes of the FWI System

The forest soil can divided into five different layers [12, 14] as shown in Fig. 3.4. Each layer has specific characteristics and provides different types of *fules* for forest fires. These characteristics are reflected in fuel codes of the FWI System. Related to each fuel type, there

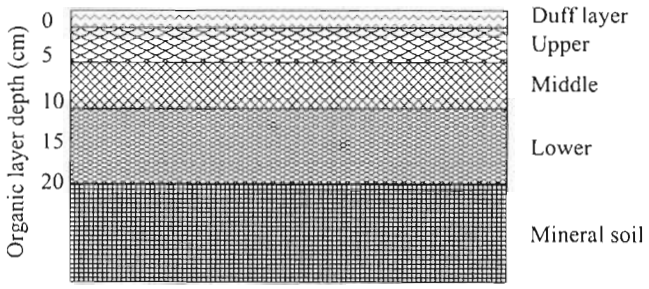


Figure 3.4: Forest soil layers.

is a drying rate at which the fuel loses moisture. This drying rate, called timelag, is the time required for the fuel to lose two-thirds of its free moisture content with a noon temperature reading of 21°C, relative humidity of 45%, and a wind speed of 13 km/h [14]. Also, each fuel type has a fuel loading metric, which describes the average amount (in tonnes) of that fuel that exists per hectare.

There are three fuel codes in the FWI System: Fine Fuel Moisture Code (FFMC), Duff Moisture Code (DMC), and Drought Code (DC). FFMC represents the moisture content of litter and fine fuels, 1–2 cm deep, with a typical fuel loading of about 5 tonnes per hectare. The timelag for FFMC fuels is 16 hours. Since fires usually start and spread in fine fuels [14], FFMC can be used to indicate ease of ignition, or ignition probability.

The Duff Moisture Code (DMC) represents the moisture content of loosely compacted, decomposing organic matter, 5–10 cm deep, with a fuel loading of about 50 tonnes per hectare. DMC is affected by precipitation, temperature and relative humidity. Because these fuels are below the forest floor surface, wind speed does not affect the fuel moisture content. DMC fuels have a slower drying rate than FFMC fuels, with a timelag of 12 days. Although the DMC has an open-ended scale, the highest probable value is about 150 [14]. The DMC determines the probability of fire ignition due to lightning and also shows the rate of fuel consumption in moderate depth organic layers. The last fuel moisture code, the Drought Code (DC), is an indicator of the moisture content of the deep layer of compacted organic matter, 10–20 cm deep, with a fuel loading of about 440 tonnes per hectare. Temperature and precipitation affect the DC, but wind speed and relative humidity do not have any effect on it due to the depth of this fuel layer. DC fuels have a very slow drying rate, with a timelag of 52 days. The DC is indicative of long-term moisture conditions, determines fire's resistance to extinguishing, and indicates fuel consumption in

Table 3.2: Summary of Fuel Codes in the FWI System [14].

Item	FFMC	DMC	DC
Fuel Association	Litter and other organic fine fuels	Loosely compacted layers of moderate depth	Deep compacted organic layers
Fire Potential Indicator	Ease of ignition	Probability of lightning fires, Fuel consumption in moderate layers	Resistance to extinguishing, Consumption in deep layers
Depth (cm)	1–2	5–10	10–20
Fuel Loading	5 t/ha	50 t/ha	440 t/ha
Timelag	16 hours	12 days	52 days
Value Range	0–99	0–150	0–800
Spring Start Value	85	6	15

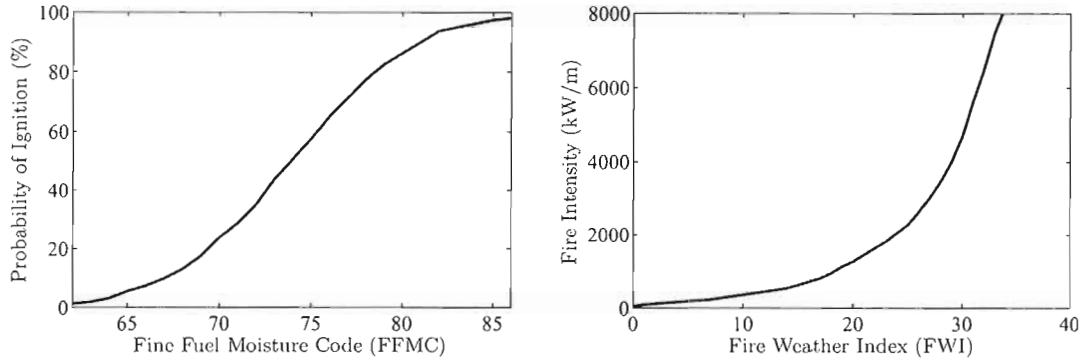
deep organic material. The DC scale is also open-ended, although the maximum probable value is about 800 [14].

Table 3.2 summarizes the features of the three fuel codes in the FWI System.

3.3.2 Fire Indexes of the FWI System

Fire indexes of the FWI System describes the spread and intensity of fires. There are three fire indexes: Initial Spread Index (ISI), Buildup Index (BUI), and Fire Weather Index (FWI). As indicated by Fig. 3.3, ISI and BUI are intermediate indexes and are used to compute the FWI index. The ISI index indicates the rate of fire spread immediately after ignition. It combines the FFMC and wind speed to predict the expected rate of fire spread. Generally, a 13 km/h increase in wind speed will double the ISI value. The BUI index is a weighted combination of the DMC and DC codes, and it indicates the total amount of fuel available for combustion. The DMC code has the most influence on the BUI value. For example, a DMC value of zero always results in a BUI value of zero regardless of what the DC value is. DC has its strongest influence on the BUI at high DMC values, and the greatest effect that the DC can have is to make the BUI value equal to twice the DMC value.

The Fire Weather Index (FWI) is calculated from the ISI and BUI to provide an estimate of the intensity of a spreading fire. In effect, FWI indicates fire intensity by combining the rate of fire spread with the amount of fuel being consumed. Fire intensity is defined as



(a) Probability of ignition as a function of the FFMC code

(b) Fire intensity as a function of the FWI index

Figure 3.5: Using two main components of the Fire Weather Index System in designing a wireless sensor network to detect and combat forest fires. Figures are produced by interpolating data from [14].

the energy output measured in kilowatts per meter of flame length at the head of a fire. The head of a fire is the portion of a fire edge showing the greatest rate of spread and fire intensity. The FWI index is useful for determining fire suppression requirements as well as being used for general public information about fire danger conditions. Although FWI is not directly calculated from weather data, it depends on those factors through ISI and BUI.

3.3.3 Interpreting and Using the FWI System

There are two goals of the proposed wireless sensor network for forest fires: (i) provide early warning of a potential forest fire and, (ii) estimate the scale and intensity of the fire if it materializes. Both goals are needed to decide on required measures to combat a forest fire. To achieve these goals we design our sensor network based on the two main components of the FWI System: (i) the Fine Fuel Moisture Code (FFMC), and (ii) the Fire Weather Index (FWI). The FFMC code is used to achieve the first goal and the FWI index is used to achieve the second. We justify the choice of these two components in the following.

The FFMC indicates the relative ease of ignition and flammability of fine fuels due to exposure to extreme heat. To show this, we interpolate data from [14] to plot the probability of ignition as a function of FFMC. The results are shown in Fig. 3.5(a). The FFMC scale ranges from 0–99 and is the only component of the FWI System without an open-ended

Table 3.3: Ignition Potential Based on the FFMC code.

Ignition Potential	FFMC Value Range
Low	0–63
Moderate	63–84
High	84–88
Very High	88–91
Extreme	91+

(a) Moderate surface fire (FWI = 14)

(b) Very intense surface fire (FWI = 24)

(c) Developing active fire (FWI = 34)

Figure 3.6: Experimental validation of the FWI index. Pictures shown from experiments conducted by Alberta Forest Service [2].

scale. Generally, fires begin to ignite at FFMC values near 70, and the maximum probable value that will ever be achieved is 96 [14]. Table 3.3 summarizes the potential of fire ignition based on FFMC value ranges. Low values of FFMC are not likely to be fires and can be simply ignored, while larger values indicate more alarming situations.

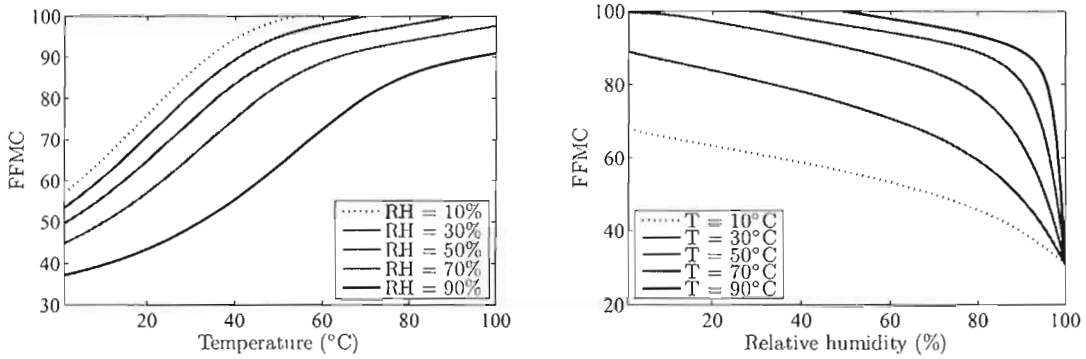
The FWI index indicates the fire intensity by combining the rate of fire spread (from the Initial Spread Index, ISI) with the amount of fuel being consumed (from the Buildup Index, BUI). A high value of the FWI index indicates that in case of fire ignition, the fire would be difficult to control. This intuition is backed up by several experimental studies. For example, in 1974, the Alberta Forest Service performed a short term study of experimental burning in the jack pine forests of north eastern Alberta. Snapshots of the resulting fires and the computed FWI indexes are shown in Fig. 3.6 for three fires with different FWI values [2]. Furthermore, the study [14] relates the fire intensity with the FWI index. We plot this relationship in Fig. 3.5(b) by interpolating data from [14]. Finally, Table 3.4 provides a classification of fire danger based on the values of FWI [12].

Table 3.4: Potential Fire Danger Based on the FWI index.

FWI Class	Range	Type of Fire	Potential Danger
Low	0–5	Creeping surface fire	Fire will be self extinguishing
Moderate	5–10	Low vigor surface fire	Easily suppressed with hand tools
High	10–20	Moderately vigorous surface fire	Power pumps and hoses are needed
Very High	20–30	Very intense surface fire	Difficult to control
Extreme	30+	Developing active fire	Immediate and strong action is critical

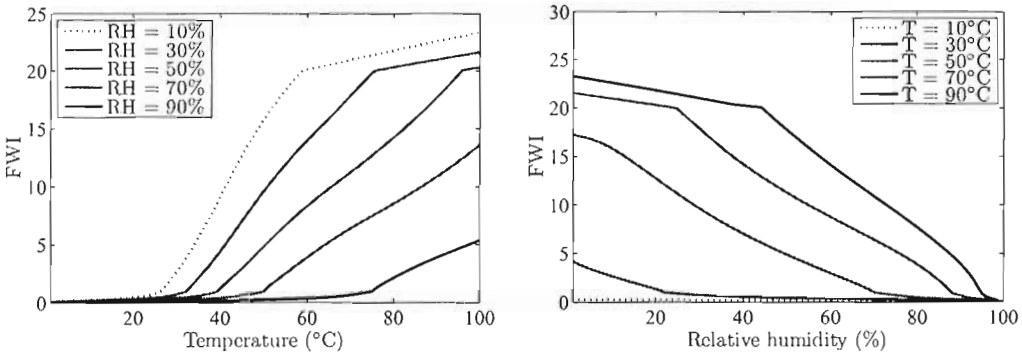
Both of the FFMC code and the FWI index are computed from four basic weather conditions: temperature, relative humidity, precipitation, and wind speed. These weather conditions can be measured by sensors deployed in the forest. The accuracy and the distribution of the sensors impact the accuracy of the FFMC code and the FWI index. Therefore, we need to quantify the impact of these weather conditions on FFMC and FWI. Using this quantification, we can design our wireless sensor network to produce the desired accuracy in FFMC and FWI. To do this, we contacted the Canadian Forest Service to obtain the closed-form equations that describe the dependence of the FFMC and FWI on the weather conditions. We were given access to these equations as well as a program that computes them [52]. We used this program to study the sensitivity of FFMC and FWI to air temperature and relative humidity. Sample of our results are shown in Fig. 3.7 and Fig. 3.8. Fig. 3.7 shows the sensitivity of FFMC to temperature and relative humidity for fixed wind speed at 5 km/h and precipitation level of 5 mm. And, Fig. 3.8 shows the sensitivity of FWI to temperature and relative humidity under similar conditions. We will use these figures to bound the errors in estimating FFMC and FWI in the next section.

In summary, the FFMC code and FWI index provide quantifiable means to detect and respond to forest fires. Low values of FFMC are not likely to be fires and may be ignored. In case of higher FFMC values, where a fire is possible, based on the values of FWI, some fires might be left to burn, some should be contained and others need to be extinguished immediately. We design our wireless sensor network for forest fire detection based on the FFMC code and FWI index. Our system uses weather data collected by sensor nodes to calculate these indexes.



(a) FFMC versus temperature (b) FFMC versus humidity

Figure 3.7: Sensitivity of the FFMC code to basic weather conditions.



(a) FWI vesurs temperature (b) FWI versus humidity

Figure 3.8: Sensitivity of the FFMC code to basic weather conditions.

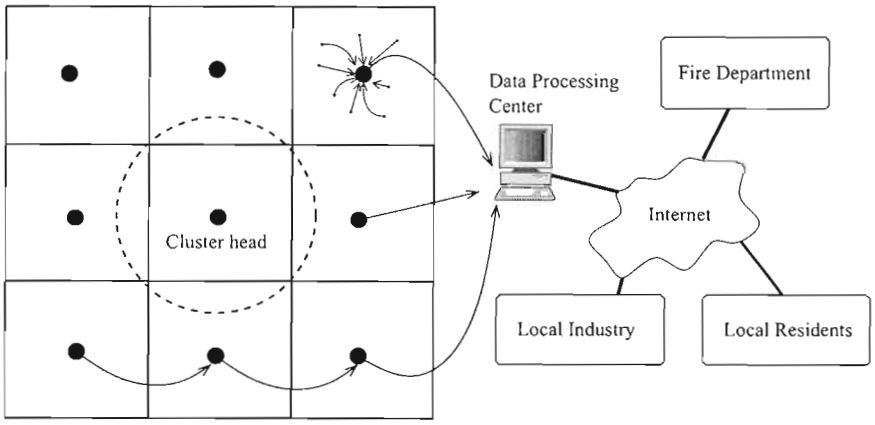


Figure 3.9: Forest fire detection system architecture.

3.4 Wireless Sensor Networks for Forest Fire Detection

3.4.1 Overview

Our design for forest fire detection system is based on the FWI system which uses weather measurements to produce a set of fire weather indexes that indicate the ease of fire ignition, initial rate of fire spread, and fire intensity. In particular we are interested in two of the indexes namely FFMC which indicates the probability of ignition and FWI which shows the fire intensity. Weather measurements are used to produce these indexes according to formulas provided by Canadian Forest Service [52] which are obtained by private communication.

The general architecture of our system is shown in Fig. 3.9. A large number of sensor nodes are densely deployed in the forest field to measure weather data such as temperature. The sensor nodes are assumed to know their location through GPS or localization protocols [10]. The nodes organize themselves into clusters using clustering protocols [56]. Every sensor node reports its readings to its cluster head. The cluster heads aggregate the readings by calculating fire weather indexes. The indexes along with cluster head location information are communicated to the data processing center. The indexes are then used to determine the potential fire danger and to decide appropriate actions for fighting the fire. If the indexes show a high risk of fire, the local residents, industries, and fire departments are alarmed.

3.4.2 Clustering, Aggregation, and Routing

In forest fire detection systems, most of the data collected by sensor nodes are likely to be geographically correlated. Therefore, these readings can be efficiently aggregated into a few packets to be communicated to the data processing center. We propose an application specific aggregation which includes calculating fire weather indexes and reporting these indexes instead of individual sensor readings. We use clustering algorithms to organize sensor nodes into small clusters [27,56]. Each cluster head aggregates all the sensor readings inside the cluster into two fire weather indexes namely FFMC and FWI. We recommend using HEED [56] as the clustering algorithm. HEED achieves a fairly uniform distribution of cluster heads across the network without any assumption about the homogeneity of sensor deployment. It also uses cluster head rotation to balance the load on nodes due to cluster head operations.

The large size of the forest field removes the possibility of single-hop communication between cluster heads and the data processing center. Therefore cluster heads use multi-hop routing to communicate to the data processing center. Routing based on location information [30, 58] is more suitable for forest fire detection systems for several reasons. Geographic routing protocols are helpful in exploiting the correlation between collected data and supporting geographic queries. In a forest fire detection system, users may prefer to query a small geographical region rather than the entire network. In addition, such routing schemes operate on local information without requiring global transfer of routing tables. Recent research in localization protocols have made it easy to obtain precise location information in GPS-less sensor networks [10].

3.4.3 K-Coverage for Forest Fire Detection

Many sensor network applications including forest fire detection require a high degree of coverage to increase the detection accuracy and measurement reliability. Having multiple sensors report an event helps removing false readings. In addition, some areas in the forest with high fire potential and human neighborhoods are more important and need to be monitored with a higher coverage degree.

We establish a relationship between the coverage degree k, and the desired error margin in weather data readings. In addition, since calculation of fire weather indexes only depend on these factors, we are able to relate the coverage degree to the desired error margin in

calculating fire weather indexes. Therefore, given a desired accuracy for fire weather indexes, the required coverage degree can be obtained.

Aside from sensor characteristics, the effective sensing range of sensors is ruled by field conditions such as vegetation type. Thus, on site experiments with conservative assumptions are required for sensing range estimation. We assume that the minimum sensing range of all sensors is r_s. Consider a disk of radius r_s centered at arbitrary point p. Since this disk is small, we can assume that the weather readings are stable in the disk and represented by the values at point p. We use the temperature for the discussion, however, the same argument is applicable to other factors. Define random variable T, as the reading of an individual sensor inside the disk. The statistical population is the set of all readings from sensors inside the disk. Each sample reading is thus drawn from a normal distribution with mean μ_T and standard deviation σ_T, where μ_T is the mean of all sensor readings, and σ_T depends on the error in readings.

Calculating the exact value of μ_T, which also represents the temperature at point p, requires averaging over all readings from infinite number of sensors inside the disk. However, an estimate can be easily obtained by sampling a number of readings. The estimate known as sample mean, $\hat{\mu_T}$, is calculated as the average of all samples.

$$\hat{\mu_T} = \frac{1}{k}\sum_{i=1}^{k} t_i$$

where t_is are the individual sensor readings and k is the number of samples. As the number of samples is increased, the sample mean becomes closer to the exact value of the population mean. The error between the sample mean and the population mean, $\delta_T = |\mu_T - \hat{\mu_T}|$, is calculated as follows [50].

$$\delta_T = z_{\frac{\alpha}{2}} \frac{\sigma_T}{\sqrt{k}}$$

where z is the standard normal distribution, α is the length of the confidence interval, σ_T is the population standard deviation, and k is the sample size. $z_{\frac{\alpha}{2}}$ can be derived from the table of standard normal distribution. Rearranging the formula, we can solve for the sample size necessary to produce results accurate to a specified confidence and error margin.

$$k = (\frac{z_{\frac{\alpha}{2}}\sigma_T}{\delta_T})^2$$

Now, given a confidence value of $100(1-\alpha)\%$, and standard deviation of σ_T, we can determine the sample size required to estimate the population mean μ_T within $\pm\delta_T$ error

margin. μ_T depends on the current temperature at point p, and σ_T depends on the error in readings and is ruled by sensor specifications. For example, the Sensirion SHT11 temperature sensor used in fireboard has an error margin of $\pm 0.5°C$. The error is usually interpreted as $2\sigma_T$, thus in this case we have $\sigma_T = 0.25$.

The number of required samples can be interpreted as the number of required sensors inside the disk of radius r_s. Since each sensor is not more than distance r_s from point p, the number of required sensors is equal to the coverage degree at point p. The following example verifies and illustrates the above analysis.

Assume fireboards are used in the forest fire detection system. From sensor specifications, it is known that the error in temperature readings is 0.5°C, thus $\sigma_T = 0.25$. Suppose that we need to measure the temperature with maximum error of 0.25°C with a confidence value of 95%.

$$k = (\frac{z_{\frac{\alpha}{2}}\sigma_T}{\delta_T})^2 = (\frac{1.96 \times 0.25}{0.25})^2 = 4$$

Assume the temperature is 30°C. We generate 4 random samples from the normal distribution with mean $\mu_T = 30$, and standard deviation $\sigma_T = 0.25$ as follows.

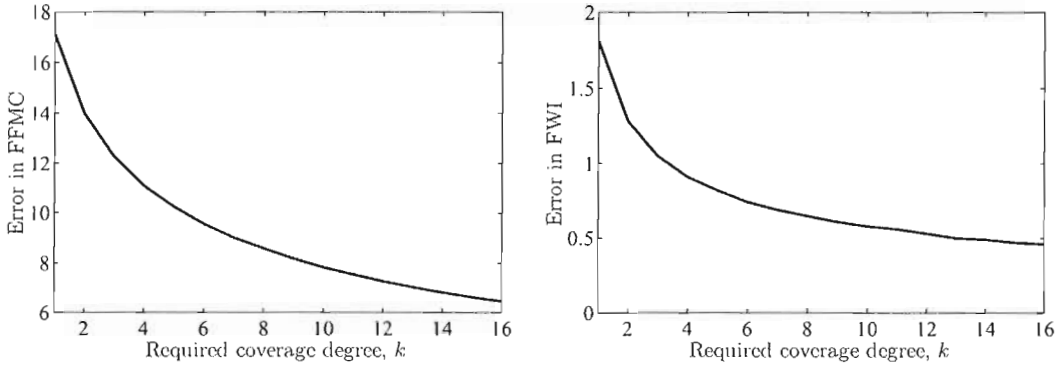
$$t_1 = 28.36, t_2 = 31.60, t_3 = 29.60, t_4 = 30.72$$

The sample mean is $\hat{\mu_T} = 30.07$ and thus the error is $|\mu_T - \hat{\mu_T}| = 0.07$ which is within the required error bound. To verify the confidence interval, the experiment is repeated 1000 times, and it is observed that in more than 95% of the times, the error is within the required range verifying the confidence value.

From the above argument it is easy to see that the required coverage degree impacts the accurate measurement of weather data and consequently fire weather indexes. Since the formulas for calculating fire weather indexes are quite complicated, a simple closed form relationship cannot be derived [52]. However, we can calculate the error bound numerically.

Accuracy in Fire Weather Indexes

The coverage degree k, influences the accuracy of weather data measurement and thus calculation of fire weather indexes. In this section we study the relationship between coverage degree and errors in calculation of fire weather indexes. Therefore, given a desired accuracy for fire weather indexes, the required coverage degree can be obtained. As indicated by Fig. 3.7 and 3.8, a small change in temperature causes different amount of displacement in



(a) (b)

Figure 3.10: Error in calculating fire weather indexes (a) FFMC, (b) FWI.

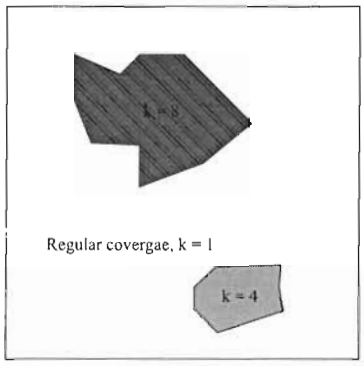
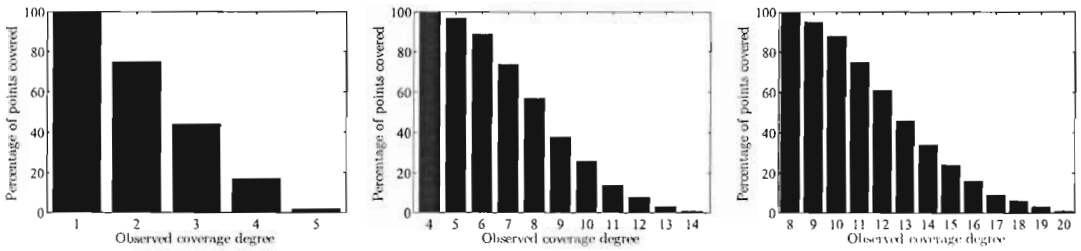


Figure 3.11: A field with two hot spots.

FFMC and FWI calculations based on the current value of relative humidity. This can be related to the slope of the tangent line on the corresponding curve. We choose the range at which the slope is maximum and calculate the errors in measurement of temperature and relative humidity. We use these values to compute the maximum errors in fire weather indexes. Fig. 3.10 shows the observed error in FFMC and FWI for various coverage degrees. As indicated by the figure, a higher coverage degree results in smaller errors in calculation of fire weather indexes. Therefore, given a desired error margin, the required coverage degree can be obtained.



(a) $k = 1$ (b) $k = 4$ (c) $k = 8$

Figure 3.12: Coverage achieved in the field and host spots.

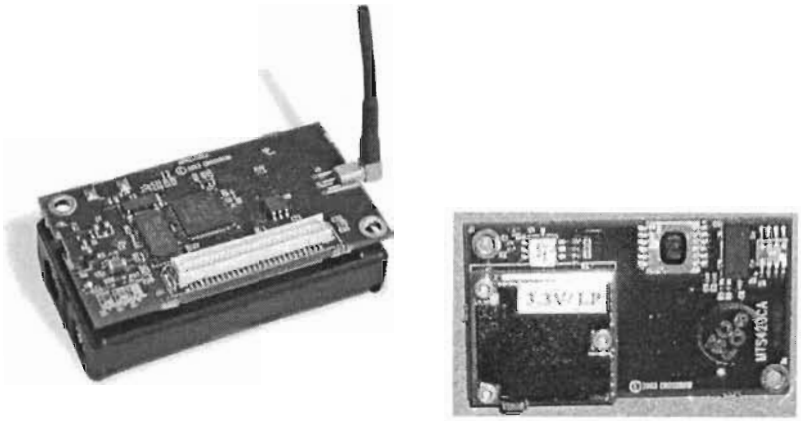
Hot spot Coverage

In forest fire detection systems, some areas with higher fire potential and human neighborhoods are considered more important and thus should be monitored with higher coverage degrees. These areas are commonly referred to as hot spots. We extend our distributed k-coverage algorithm, DRKC, to provide different degrees of coverage in different areas as required by the application. This requires sensor nodes to know their locations through GPS or localization algorithms [10]. The location of the hot spots, and the required coverage degrees are specified by the application and communicated to all nodes. Knowing the location of the hot spots, each node can determine the required coverage degree for itself. For example, in Fig. 3.11, nodes inside the small polygon choose a coverage degree of 4. In DRKC, the initial weights of sensor nodes are assigned based on their required coverage degree. Each node decides to activate some of its neighbors unless it finds itself sufficiently covered. Thus, the required coverage degrees can be achieved in a straightforward manner.

A sample result of the algorithm is shown in Fig. 3.12, where it is required to 1-cover the field while two regions in the middle (hot spots) need higher coverage degrees, 4 and 8. The figure shows the percentage of points inside each region which achieve the required coverage degree. The results indicate that in each spot, our DRKC algorithm indeed achieves the required coverage degree while it provides 1-coverage in the rest of the area.

3.4.4 Hardware and Software Requirements

For our design, we recommend using MICAz motes, MPR2400, manufactured by Crossbow Technology Inc. [45]. The mote platform as shown in Fig. 3.13(a), hosts the processor, memory modules, and radio transceiver. Sensor boards are connected to the mote platform



(a) (b)

Figure 3.13: (a) The Crossbow MICAz mote platform, and (b) The Crossbow fireboard (MTS420)

via a 51-pin connector. Fig. 3.13 (b) shows the sensor board, MTS420, commonly known as fireboard. Fireboard is compatible with MICAz and has temperature and humidity, barometric pressure, GPS, ambient light sensors, and dual-axis accelerometer. The GPS module on fireboard is optional and is not present in MTS400 model. While Mica motes are mechanically robust by design, to protect them from damage, we use MIH2400 housings manufactured by Crossbow. While protecting the mote, the housing does not obstruct sensing and communication functionality.

The Crossbow mote platform runs TinyOS [51] which is an open source, component-based, event-driven operating system for sensor node with very limited resources. TinyOS is largely written in nesC [43] programming language. TinyOS provides a set of libraries for controlling radio communication, and operating various sensor boards connected to the mote. The biggest advantage of TinyOS is providing application code portability meaning that application level code is independent of the underlying mote platform. Thus changing the platform requires simply recompiling the source code for the new platform. All the protocols need to be implemented in nesC and compiled for the specified mote platform. The protocols can use higher level services provided by TinyOS in order to implement their functionality.

The data processing center runs a relational database to record the data reported by cluster heads. The database is available to remote clients and web based applications. The

applications provide remote access to the forest field for experimental, data analysis, and management purposes.

Chapter 4

Conclusions and Future Work

This chapter summarizes the conclusions of this thesis and outlines some future research directions for this research.

4.1 Conclusions

We presented a novel approach to solve the k-coverage problem in wireless sensor networks. We modelled the k-coverage problem as a set system for which an optimal hitting set corresponds to an optimal solution for k-coverage. We proposed an approximation algorithm for computing near-optimal hitting sets efficiently. We proved that our algorithm produces a solution that is at most a logarithmic factor from the optimal. We compared our algorithm against the currently-known k-coverage algorithms and showed that it runs up to four orders of magnitude faster, while producing same or better solution sizes than the other algorithms.

The target of our k-coverage algorithm is large-scale sensor network applications such as forest fire detection systems deployed over wide areas. However, it is difficult to control such applications in a centralized manner. Thus, we designed our algorithm in a way that minimizes the reliance on global information, and therefore, can be implemented in a distributed manner. We designed and implemented a fully distributed version of our algorithm that uses only local information. Our distributed algorithm has low message complexity and it does not require sensors to know their locations. Location unawareness is a valuable feature especially for large-scale networks where many sensors are deployed. This is because sensors do not need to be equipped with GPS systems, a significant cost saving. Moreover, while localization protocols exist for sensors without GPS, these protocols impose

communication and computation overheads on sensors. Our k-coverage algorithm saves these overheads by not requiring localization protocols. We implemented our distributed k-algorithm and compared it against two other distributed algorithms in the literature. Our comparison showed that our algorithm: (i) converges much faster than the others, (ii) activates near-optimal number of sensors, and (iii) significantly prolongs the network lifetime because it consumes much less energy than the other algorithms. We also extend DRKC to provide hot spot coverage which is required for some applications where more important areas such as human neighborhood need a higher coverage degree. Simulation results verify that our algorithm indeed achieves the required coverage degree for different regions inside the field.

Finally, we presented the design of a wireless sensor network for early detection of forest fires. Our design is based on the Fire Weather Index (FWI) System designed by the Canadian Forest Service [12]. The FWI System uses weather observations to produce a set of indexes which indicate the likelihood of the current weather conditions to cause a fire. We presented a new aggregation paradigm to efficiently calculate these indexes in a wireless sensor network. We established a relationship between the desired accuracy of the system and the required coverage degree.

4.2 Future Work

The research presented in this thesis can be extended in several directions. We summarize some of these directions below.

- In this work, we used a disk sensing model which is widely used in the literature for its simplicity. In this model, the sensing capability of a sensor node is represented by a disk and assumed to be uniform in all directions. An event that occurs within the disk is always detected with probability 1 while any event outside the disk goes undetected. This model, does not effectively reflect the probabilistic nature of sensing. Thus, several probabilistic sensing models have been developed by the research community. These models may help design a more realistic framework for coverage algorithms. Experimental studies need to be carried out to study the sensing behavior of various sensor in order to verify and customize these models.

- We used the necessary and sufficient node density for k-coverage presented in [32].

These bounds, however, are not tight to provide a good measure for evaluation and comparison of k-coverage algorithms. Providing better bounds could be another direction for our future research.

- While simulation provides a good starting point for evaluation and comparison of algorithms, many factors are not revealed until a practical experiment is carried out. Implementation of our algorithm and building a prototype of our design will provide valuable insights by studying the performance of the system in action.

- In forest fire detection, most of the data reported by sensors are likely to be correlated. Thus aggregation models are extremely helpful in reducing the communication overhead and energy consumption. Studying and modelling the data correlation helps designing a better correlation scheme. In addition, aggregation may be integrated with routing to further exploit such correlations.

- Finally, addressing environmental concerns are important directions for future research. Although there has been a vast amount of research on sensor network applications, little has been done to address the environmental pollution caused by sensor networks themselves. Ironically, one of the applications of sensor networks is environmental study and pollution monitoring. The effect of sensor network deployments on wild life and natural habitat is largely ignored. Unless sensor nodes are equipped with a continuous energy source such as solar power, they need to be undeployed to clean the field for a new deployment. The question of how to undeploy sensor networks has not been considered by the research community yet, while there has been a lot of work on deployment algorithms and strategies.

Appendix A

Equations for the Fire Weather Index System

This Appendix lists all equations of the FWI System used in this thesis. These equations are from [52] and are reproduced here for the sake of completeness of the thesis.

A.1 Symbols in the Equations

All quantities in the numbered equations are represented in the following list by single letters, sometimes with subscripts. The symbols are arranged in groups according to their place in the whole.

Weather

T – noon temperature, °C

H – noon relative humidity, %

W – noon wind speed, Km/h

r_o – rainfall in open, measured once daily at noon, mm

r_f – effective rainfall, FFMC

r_e – effective rainfall, DMC

r_d – effective rainfall, DC

Fine Fuel Moisture Code (FFMC)

m_o – fine fuel moisture content from previous day

m_r – fine fuel moisture content after rain

m – fine fuel moisture content after drying

E_d – fine fuel EMC for drying

E_w – fine fuel EMC for wetting

k_o – intermediate step in calculation of k_d

k_d – log drying rate in FFMC, $\log_{10}$ m/day

k_l – intermediate step in calculation of k_w

k_w – log wetting rate, $\log_{10}$ m/day

F_o – previous day's FFMC

F – FFMC

Duff Moisture Code (DMC)

M_o – duff moisture content from previous day

M_r – duff moisture content after rain

M – duff moisture content after drying

K – log drying rate in DMC, $\log_{10}$ M/day

L_e – effective day length in DMC, hours

b – slope variable in DMC rain effect

P_o – previous day's DMC

P_r – DMC after rain

P – DMC

Drought Code (DC)

Q – moisture equivalent of DC, units of 0.254 mm

Q_o – moisture equivalent of previous day's DC

Q_r – moisture equivalent after rain

V – potential evapotranspiration, units of 0.254 mm water /day

L_f – day length adjustment in DC

D_o – previous day's DC

D_r – DC after rain

D – DC

Fire Behavior Indexes (ISI, BUI, FWI)

$f(W)$ – wind function

$f(F)$ – fine fuel moisture function

$f(D)$ – duff moisture function

R – Initial Spread Index (ISI)

U – Buildup Index (BUI)

B – FWI (intermediate form)

S – FWI (final form)

A.2 Equations and Procedures

Fine Fuel Moisture Code (FFMC)

$$
\begin{aligned}
m_o &= 147.2(101 - F_o)/(59.5 + F_o) \\
r_f &= r_o - 0.5 \\
m_r &= \begin{cases} m_o + 42.5 r_f (e^{-100/(251-m_o)})(1 - e^{-6.93/r_f}) & \text{if } m_o \leq 150 \\ m_o + 42.5 r_f (e^{-100/(251-m_o)})(1 - e^{-6.93/r_f}) + 0.0015(m_o - 150)^2 {r_f}^{0.5} & \text{if } m_o > 150 \end{cases} \\
E_d &= 0.942 H^{0.679} + 11 e^{(H-100)/10} + 0.18(21.1 - T)(1 - e^{-0.115H}) \\
E_w &= 0.618 H^{0.753} + 10 e^{(H-100)/10} + 0.18(21.1 - T)(1 - e^{-0.115H}) \\
k_o &= 0.424[1 - (H/100)^{1.7}] + 0.0694 W^{0.5}[1 - (H/100)^8] \\
k_d &= k_o \times 0.581 e^{0.0365T} \\
k_l &= 0.424[1 - (\frac{100 - H}{100})^{1.7}] + 0.0694 W^{0.5}[1 - (\frac{100 - H}{100})^8] \\
k_w &= k_1 \times 0.581 e^{0.0365T} \\
m &= E_d + (m_o - E_d) \times 10^{-k_d} \\
m &= E_w - (E_w - m_o) \times 10^{-k_w} \\
F &= 59.5(250 - m)/(147.2 + m)
\end{aligned}
$$

Table A.1: Effective day lengths (L_e) for DMC.

Month	Jan	Feb	Mar	Apr	May	June	July	Aug	Sep	Oct	Nov	Dec
L_e	6.5	7.5	9.0	12.8	13.9	13.9	12.4	10.9	9.4	8.0	7.0	6.0

Table A.2: Day length factors (L_f) for DC.

Month	Jan	Feb	Mar	Apr	May	June	July	Aug	Sep	Oct	Nov	Dec
L_f	-1.6	-1.6	-1.6	0.9	3.8	5.8	6.4	5.0	2.4	0.4	-1.6	-1.6

Duff Moisture Code (DMC)

$$
\begin{aligned}
r_e &= 0.92 r_o - 1.27 \\
M_o &= 20 + e^{(5.6348 - P_o/43.43)} \\
b &= \begin{cases} 100/(0.5 + 0.3P_o) & \text{if } P_o \leq 33 \\ 14 - 1.3 \ln P_o & \text{if } 33 < P_o \leq 65 \\ 6.2 \ln P_o - 17.2 & \text{if } P_o > 65 \end{cases} \\
M_r &= M_o + 1000 r_e / (48.77 + b r_e) \\
P_r &= 244.72 - 43.43 \ln(M_r - 20) \\
K &= 1.894(T + 1.1)(100 - H) L_e \times 10^{-6} \\
P &= P_r + 100K
\end{aligned}
$$

Drought Code (DC)

$$
\begin{aligned}
r_d &= 0.83 r_o - 1.27 \\
Q_o &= 800 e^{-D_o/400} \\
Q_r &= Q_o + 3.937 r_d \\
D_r &= 400 \ln(800/Q_r) \\
V &= 0.36(T + 2.8) + L_f \\
D &= D_r + 0.5V
\end{aligned}
$$

Initial Spread Index (ISI)

$$\begin{aligned} f(W) &= e^{0.05039W} \\ f(F) &= 91.9e^{-0.1386m}[1 + m^{5.31}/(4.39 \times 10^7)] \\ R &= 0.208f(W)f(F) \end{aligned}$$

Buildup Index (BUI)

$$U = \begin{cases} 0.8PD/(P + 0.4D) & \text{if } P \leq 0.4D \\ P - [1 - 0.8D/(P + 0.4D)][0.92 + (0.0114P)^{1.7}] & \text{if } P > 0.4D \end{cases}$$

Fire Weather Index (FWI)

$$\begin{aligned} f(D) &= \begin{cases} 0.626U^{0.809} + 2 & \text{if } U \leq 80 \\ 1000/(25 + 108.64e^{-0.023U}) & \text{if } U > 80 \end{cases} \\ B &= 0.1Rf(D) \\ S &= \begin{cases} e^{2.72(0.434 \ln B)^{0.647}} & \text{if } B > 1 \\ B & \text{if } B \leq 1 \end{cases} \end{aligned}$$

Bibliography

[1] Aerovision Web Page. http://www.aerovision-uav.com.

[2] M.E. Alexander and W.J. De Groot. Fire behavior in Jack Pine stands as related to the Canadian Forest Fire Weather Index System. Technical report, Canadian Forest Service, Northern Forestry Centre, Edmonton, Alberta, 1988.

[3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, and M. Gouda. A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5), December 2004.

[4] AVHRR Web Page. http://noaasis.noaa.gov/noaasis/ml/avhrr.html.

[5] B.C. Fire Lookout Towers. http://www.firelookout.com/bc.html.

[6] B.C. Ministry of Forests and Range. Fire review summary for okanagan mountain fire (K50628).

[7] B.C. Ministry of Forests and Range Web Page. http://www.for.gov.bc.ca.

[8] E. Breejen, M. Breuers, F. Cremer, R.A.W Kemp, M. Roos, K. Schutte, and J.S. Vries. Autonomous forest fire detection. In *Proc. of Third International Conference on Forest Fire Research and Fourteenth Conference on Fire and Forest Meteorology*, Luso, Portugal, November 1998.

[9] H. Bronnimann and M. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, 14(4), April 1995.

[10] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications*, 7(5):28–34, 2000.

[11] Canadian Forest Fire Danger Rating System (CFFDRS), Web Page. http://www.nofc.forestry.ca/fire.

[12] Canadian Forest Service (CFS), Web Page. www.nrcan.gc.ca/cfs.

[13] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers*, 51(12), 2002.

[14] W. J. de Groot. Interpreting the Canadian Forest Fire Weather Index (FWI) System. In *Proc. of the Fourth Central Region Fire Weather Committee Scientific and Technical Seminar*, Edmonton, Canada, 1998.

[15] D. M. Doolin and N. Sitar. Wireless sensors for wildfire monitoring. In *Proc. of SPIE Symposium on Smart Structures and Materials*, San Diego, CA, March 2005.

[16] Fire Watch Web Page. http://www.fire-watch.de/.

[17] J. Fleming and R. G. Robertson. Fire Management Tech Tips: The Osborne Fire Finder. Technical Report 0351 1311-SDTDC, USDA Forest Service, October 2003.

[18] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[19] P. Gupta and P.R. Kumar. Internets in the sky: The capacity of three dimensional wireless networks. *Communications in Information and Systems*, 1(1):33–49, 2001.

[20] D.L. Hall and J. Llinas. *Handbook of Multisensor Data Fusion*. CRC Press, 2001.

[21] C. Hartung, R. Han, C. Seielstad, and S. Holbrook. FireWxNet: A multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments. In *Proceedings of the 4th International Conference on Mobile systems, Applications and Services (MobiSys'06)*, Uppsala, Sweden, June 2006.

[22] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete and Computational Geometry*, 2(1), December 1987.

[23] M. Hefeeda and M. Bagheri. Efficient k-coverage algorithms for wireless sensor networks. Technical Report TR 2006-22, School of Computing Science, Simon Fraser University, September 2006.

[24] M. Hefeeda and M. Bagheri. Efficient k-coverage algorithms for wireless sensor networks. *IEEE Transactions on Networking, submitted*, 2007.

[25] M. Hefeeda and M. Bagheri. Randomized k-coverage algorithms for dense sensor networks. In *Proceedings of INFOCOM 2007 Minisymposium*, Anchorage, AK, May 2007.

[26] M. Hefeeda and M. Bagheri. Wireless sensor networks for early detection of forest fires. Technical Report TR 2007-08, School of Computing Science, Simon Fraser University, April 2007.

[27] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS'00)*, Maui, Hawaii, January 2000.

[28] Y. Huang and Y. Tseng. The coverage problem in a wireless sensor network. In *Proc. ACM International Conference on Wireless Sensor Networks and Applications*, San Diego, CA, September 2003.

[29] R. Iyengar, K. Kar, and S. Banerjee. Low-coordination topologies for redundancy in sensor networks. In *Proc. of ACM Mobihoc'05*, Urbana-Champaign, IL, May 2005.

[30] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of 6th Annual International Conference on Mobile Computing and Networking (MobiCom'00)*, Rome, Italy, 2000.

[31] E. Khrt, J. Knollenberg, and V. Mertens. An automatic early warning system for forest fires. *Annals of Burns and Fire Disasters*, 14(3), 2001.

[32] S. Kumar, T. H. Lai, and J. Balogh. On k-coverage in a mostly sleeping sensor network. In *Proc. of ACM International Conference on Mobile Computing and Networking (MOBICOM'04)*, pages 144–158, Philadelphia, PA, September 2004.

[33] Z. L, S. Nadon, and J. Cihlar. Satellite-based detection of Canadian boreal forest fires : development and application of the algorithm. *International journal of remote sensing*, 21:3057–3069, 2000.

[34] D.T. Lee. On k-nearest neighbor voronoi diagrams in the plane. *IEEE Transactions on Computers*, 1(C31), 1982.

[35] A. Lohi, T. Ikola, Y. Rauste, V., and Kelha. Forest fire detection with satellites for fire control. In *Proc. of IUFRO Conference on Remote Sensing and Forest Monitoring*, Rogow, Poland, June 1999.

[36] LPSOLVE Web Page. http://lpsolve.sourceforge.net/5.5.

[37] A.M. Mainwaring, D.E. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proc. of the First International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, pages 88–97, Atlanta, Georgia, September 2002.

[38] J. Matousek, R. Seidel, and E. Welzl. How to net a lot with little: Small ϵ-nets for disks and halfspaces. In *Proc. of the 6th Annual ACM Symposium on Computational Geometry (SoCG'90)*, Berkeley, CA, June 1990.

[39] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and Srivastava M. Coverage problems in wireless ad-hoc sensor networks. In *Proc. of The IEEE Conference on Computer Communications (INFOCOM'01)*, April 2001.

[40] D. Mehta, M. Lopez, and L. Lin. Optimal coverage paths in ad-hoc sensor networks. In *Proc. of IEEE International Conference on Communications (ICC'03)*, May 2003.

[41] MODIS Web Page. http://modis.gsfc.nasa.gov.

[42] National Fire Danger Rating System, Web Page. http://www.wrh.noaa.gov/sew/fire/olm/nfdrs.htm.

[43] nesC Web Page. http://nescc.sourceforge.net.

[44] T. Okabe, B. Boots, K. Sugihara, and S. Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley, 2nd edition, 2000.

[45] Crossbow Inc. Web Page. http://www.xbow.com/support/appnotes.htm.

[46] G. Pearce. The science of fire behaviour and fire danger rating. Technical report, New Zealand Forest Research Institute Ltd., 2000.

[47] J. San-Miguel-Ayanz, J.D. Carlson, M. Alexander, K. Tolhurst, G. Morgan, and R. Sneeuwjagt. Chapter 2: Current methods to assess fire danger potential. In *Wildland Fire Danger Estimation and Mapping - The Role of Remote Sensing Data.* World Scientific Publishing Co. Pte. Ltd., 2003.

[48] A. So and Y. Ye. On solving coverage problems in a wireless sensor network using voronoi diagrams. In *Proc. of Workshop on Internet and Network Economics (WINE'05)*, Hong Kong, December 2005.

[49] B. Son, Y. Her, and J. Kim. A design and implementation of forest-fires surveillance system based on wireless sensor networks for South Korea mountains. *International Journal of Computer Science and Network Security (IJCSNS)*, 6(9):124–130, 2006.

[50] J. R. Taylor. *Introduction to Error Analysis.* University Science Books, second edition, 1997.

[51] TinyOS Web Page. http://www.tinyos.net.

[52] C.E. Van Wagner and T.L. Pickett. Equations and FORTRAN program for the Canadian Forest Fire Weather Index System. Technical Report 33, Canadian Forest Service, Ottawa, Ontario, 1985.

[53] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Transactions on Sensor Networks*, 1(1), August 2005.

[54] S. Yang, F. Dai, M. Cardei, and J. Wu. On connected multiple point coverage in wireless sensor networks. *Journal of Wireless Information Networks*, May 2006.

[55] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proc. of the 23rd International Conference on Distributed Computing Systems (ICDCS'03)*, May 2003.

[56] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *Proceedings of IEEE INFOCOM'04*, Hong Kong, March 2004.

[57] L. Yu, N. Wang, and X. Meng. Real-time forest fire detection with wireless sensor networks. In *Proc. of International Conference On Wireless Communications, Networking and Mobile Computing (WiMob'05)*, Montreal, Canada, September 2005.

[58] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA/CSD-TR-01-0023, UCLA Computer Science Department, May 2001.

[59] H. Zhang and J.C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc and Sensor Wireless Networks: An International Journal*, 1(1-2), January 2005.

[60] Z. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proc. of The Third ACM Conference on Embedded Networked Sensor Systems (Sensys'03)*, Los Angeles, CA, November 2003.

[61] Z. Zhou, S. Das, and H. Gupta. Connected k-coverage problem in sensor networks. In *Proc. of International Conference on Computer Communications and Networks (ICCCN'04)*, Chicago, IL, October 2004.