Ad Hoc & Sensor Wireless Networks Vol. 7, pp. 169–224
Reprints available directly from the publisher
Photocopying permitted by license only

Forest Fire Modeling and Early Detection using Wireless Sensor Networks

MOHAMED HEFEEDA AND MAJID BAGHERI

School of Computing Science, Simon Fraser University, Canada
E-mail: {mheefeda,mbagheri}@cs.sfu.ca

Received: May 2, 2008. Accepted: June 26, 2008.

Early detection of forest fires is the primary way of minimizing their damages. We present the design of a wireless sensor network for early detection of forest fires. We first present the key aspects in modeling forest fires according to the Fire Weather Index (FWI) System which is one of the most comprehensive forest fire danger rating systems in North America. Then, we model the forest fire detection problem as a node k-coverage problem ($k \geq 1$) in wireless sensor networks. We propose approximation algorithms for the node k-coverage problem which is shown to be NP-hard. We present a constant-factor centralized algorithm, and a fully distributed version which does not require sensors know their locations. Our simulation study demonstrates that our algorithms: activate near-optimal number of sensors, converge much faster than other algorithms, significantly prolong (almost double) the network lifetime, and can achieve unequal monitoring of different zones in the forest.

Keywords: Wireless Sensor Networks, Forest Fire Modeling, Forest Fire Detection Systems, Coverage Protocols, k-Coverage Protocols, Fire Weather Index.

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. Forest fires eradicate forests, burn the infrastructure, and may result in high human death toll near urban areas. Common causes of forest fires 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 forest ecosystem and they are important to the life cycle of indigenous habitats. However, in most cases, the damage caused by fires to public safety and natural resources is intolerable and early detection and suppression of fires deem crucial. For example, in August 2003,

Year	Number of fires	Hectares burned	Total cost (millions)
2006	2,590	131,086	$156.0
2005	976	34,588	$47.2
2004	2394	220,516	$164.6
2003	2473	265,050	$371.9
2002	1783	8,539	$37.5
2001	1266	9,677	$53.8
2000	1539	17,673	$52.7
1999	1208	11,581	$21.1
1998	2665	76,574	$153.9
1997	1175	2,960	$19.0
1996	1358	20,669	$37.1
1995	1474	48,080	$38.5

TABLE 1
Forest fires in the Province of British Columbia, Canada since 1995

a forest 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 turned into a firestorm. The fire forced the evacuation of 45,000 residents and 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 [7]. In the province of British Columbia alone, there have been 2,590 forest fires during 2006 [8]. These burned 131,086 hectares and costed about $156 million. Table 1 summarizes the extent and cost of wild fires in BC in previous years. The situation of forest fires is even worse if we look at the national level. Over the past ten years, on average, there have been 4,387 and 52,943 forest fires in Canada and the United States, respectively, per year [12]. Preventing a small fraction of these fires would account to significant savings in natural and human resources.

Apart from preventive measures, early detection and suppression of fires is the only way to minimize the damage and casualties. Systems for early detection of forest fires have evolved over the past decades based on advances in related technologies. We summarize this evolution in the following, motivating the need and potential of wireless sensor networks for this critical application.

1.1 Evolution of forest fire detection systems

Traditionally, forest fires have been detected using fire lookout towers located at high points, where a person looks for fires using special devices such as

Osborne fire finder [23]. 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 [9,22,33]. Most systems use Charge-Coupled Device (CCD) cameras and Infrared (IR) detectors installed on top of towers. 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 (UAV) are used to monitor forests [2]. More advanced forest fire detection systems are based on satellite imagery [6,40]. 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, the 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 (WSNs) can potentially provide such solution. Recent advances in WSNs support our belief that they make a promising framework for building near real-time forest fire detection systems. Current sensing modules can sense a variety of phenomena including temperature, relative humidity, and smoke [18] which are all helpful for fire detection systems. In the future, several sensing modules will be explicitly designed and optimized for forest fire detection systems. In addition, sensor nodes can operate for weeks on a pair of AA batteries to provide constant monitoring during the fire season. Moreover, recent protocols make sensor nodes capable of organizing themselves into a self-configuring network, thus removing the overhead of manual setup. Large-scale wireless sensor networks can be easily deployed using aeroplanes at a low cost compared to the damages and loss of properties caused by forest fires.

1.2 Contributions and paper organization

In this paper, we present the design and evaluation of a wireless sensor network for early detection of forest fires. Our design is based on solid forestry research conducted by the Canadian Forest Service [12] over several decades. In particular our contributions can be summarized as follows:

- We present the key aspects in modeling forest fires. We describe the Fire Weather Index (FWI) System [12,20], and show how its different

components can be used in designing efficient fire detection systems. We go well beyond mere description of the FWI System. We collect real data and analyze the behavior of the fire indexes for wide ranges of weather conditions such as temperature and humidity. This analysis could be of interest to computer science and engineering researchers working on the sensor networks area and their applications. It can also be beneficial for sensor manufacturers who can optimize the communication and sensing modules of their products to better fit forest fire detection systems.

- We model the forest fire detection problem as a k-coverage problem ($k \geq 1$) in wireless sensor networks.
- We propose efficient approximation algorithms for the node k-coverage problem which is shown to be NP-hard. We first present a centralized constant-factor approximation algorithm, then, we present a fully distributed version of the algorithm. The distributed algorithm employs only local information, has low message complexity, and it does *not* require sensors to know their locations. Location unawareness is a significant advantage in large-scale sensor networks, because location information is acquired either by equipping sensors with GPS modules which increases the cost, or by using localization protocols which consume energy and may shorten the network lifetime.
- We show how our distributed k-coverage algorithm can be extended to address several issues relevant to forest fire detection systems, such as providing different coverage degrees at different subareas of the forest. This is important because parts of the forest need to be monitored with higher accuracy than others. The need for unequal coverage is confirmed by real data collected from forests.
- We conduct an extensive simulation study to evaluate and compare our algorithms against others in the literature. Our simulation study demonstrates that our algorithms: (i) activate near-optimal number of sensors, (ii) converge much faster than other algorithms, (iii) significantly prolong (almost double) the network lifetime because they consume much less energy than other algorithms, and (iv) they can indeed achieve unequal monitoring of different zones in the forest.

Preliminary parts of the work presented here appear in [29,30]. Substantial extensions and enhancements have been made including improved modeling of the k-coverage problem, detailed analysis of the Fire Weather Index System, and rigorous evaluation. The rest of the paper is organized as follows. In Section 2, we summarize the related work. Section 3 describes and analyzes the FWI System which is the basis of our design. An overview of the wireless sensor network design for forest fire detection is presented in Section 4, where we model the forest fire detection problem as a k-coverage problem. The solution

of the k-coverage problem is addressed in Sections 5 and 6. Section 5 presents the theoretical foundation of our solution and it also presents the centralized k-coverage algorithm. Section 6 presents the distributed k-coverage algorithm and its extension to achieve unequal monitoring of forest zones. In Section 7, we rigorously evaluate the proposed k-coverage algorithms as well as various aspects of the proposed wireless sensor network for forest fire detection. Finally, we conclude the paper in Section 8.

2 RELATED WORK

We summarize the related works in the following two subsections.

2.1 Applications of wireless sensor networks

Sensor networks have several appealing characteristics for environmental monitoring applications such as habitat monitoring [4, 38], and forest fire detection systems [16,21,46,55]. For example, in [38], 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. Different issues such as deployment, data collection, and communication protocols are discussed and design guidelines are provided. The system is comprised of patches of sensor nodes reporting their readings to a base station through gateway nodes. 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.

The authors of [21] 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 [18] 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. In contrast to this system which reports raw weather data, our design processes weather conditions based on the Fire Weather Index System [11] and reports more useful, summarized, fire indexes.

In [26], the authors address the problem of fire behavior rather than fire detection. They present FireWxNet, a portable fire sensor network to measure the weather conditions surrounding active fires. The system is comprised of sensor nodes, webcams, and base stations which are capable of long distance communication. FireWxNet is deployed at the fire site to study the fire behavior using the collected weather data and visual images. Temperature, relative

humidity, wind speed and direction are collected 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 proposed system is designed for a different application which is early detection of forest fires.

A Forest fire surveillance system for South Korea mountains is designed in [46]. 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. Sensor nodes use a minimum cost path forwarding to send their readings to a sink which is connected to the Internet. The sink calculates the risk level of a forest fire. The calculation depends on daily measurement of relative humidity, precipitation, and solar radiation. In contrast, our proposed system calculates fire indexes according to the FWI System at cell heads where the data is more likely to be correlated. This removes the need for communicating all sensor data to the sink, only a few aggregated indexes are reported to reduce energy consumption.

2.2 Coverage in wireless sensor networks

Due to its importance, the coverage problem in wireless sensor networks has received significant research attention, see [15] for a survey. Several distributed coverage protocols have been proposed to maintain 1-coverage, i.e., to ensure that each point in the area is within the sensing range of at least one active sensor. For example, OGDC [56] tries to minimize the overlap between the sensing circles of activated sensors. While sensors in PEAS [53] probe their neighbors to decide whether to be in active or sleep mode. Centralized and distributed algorithms with guaranteed bounds are presented in [1]. The algorithms solve a variation of the set k-cover problem, where sensors are partitioned into k covers and individual covers are iteratively activated to achieve 1-coverage (not k-coverage) of the monitored area. 1-coverage protocols that do not require node location information have also been considered in [48, 50]. The authors of [48] propose three node scheduling schemes that estimate the distance to nearest neighbor, number of neighbors, or a probability of a node being off duty and use one of these metrics to put some sensors in sleep mode. These schemes may not guarantee full 1-coverage of the area. The algorithm in [50] tries to find uncovered spots and activate sensors in these areas using information from nearby active sensors. Different from these works, our proposed algorithms are more general and can provide k-coverage ($k \geq 1$). k-coverage is needed in many applications for accuracy and reliability of the data collected from the sensor network.

The authors of [51] 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. In [31], the authors first propose a k-coverage determination algorithm, and then present a distributed sleep control protocol to achieve k-coverage by exchanging several types of

messages. This protocol requires 2-hop location and coverage information. Unlike our distributed algorithm, these works assume that nodes know their locations, which either increases the per-unit cost if sensors are equipped with GPS, or requires running localization protocols on the sensor network. Localization protocols impose communication and processing overheads and therefore consume sensors' energy and shorten the network lifetime. In addition, the estimation errors of the localization protocols, which may not be small [35], could negatively impact the operation of the k-coverage protocols that rely on location information.

In [17], 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 [24]. 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 authors of [52] address the problem of selecting the minimum number of sensors to activate from a set of already deployed sensors to achieve k-coverage. They prove that the problem is NP-hard since it is an extension of the dominating set problem [24]. 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 algorithm, PKA, which uses pruning to reduce the number of active sensors. The idea of the pruning method is similar to the algorithms for constructing connected dominating sets, e.g., [19]: nodes are assigned unique priorities and they broadcast their neighbor set information. Then each node can go to a sleep mode by checking whether the coverage and connectivity can be maintained by other higher priority nodes in its neighborhood.

The work in [57] presents a centralized k-coverage 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 two distributed algorithms. The first one is a distributed greedy algorithm, which requires carrying around a central state. The second algorithm, called distributed priority algorithm (DPA), is localized and more robust. The DPA algorithm, which is also used in [25] with some modifications to activate a minimal subset of sensors to answer a query, employs multi-hop neighborhood information to turn off nodes that are not needed to k-cover the area.

3 UNDERSTANDING AND MODELING 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 [43].

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 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 [42]. The last factor, precipitation, plays an important role in wetting fuels.

As shown in Figure 1, 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, we present how these codes and indexes can be interpreted and utilized in designing a wireless sensor network for early detection of forest fires.

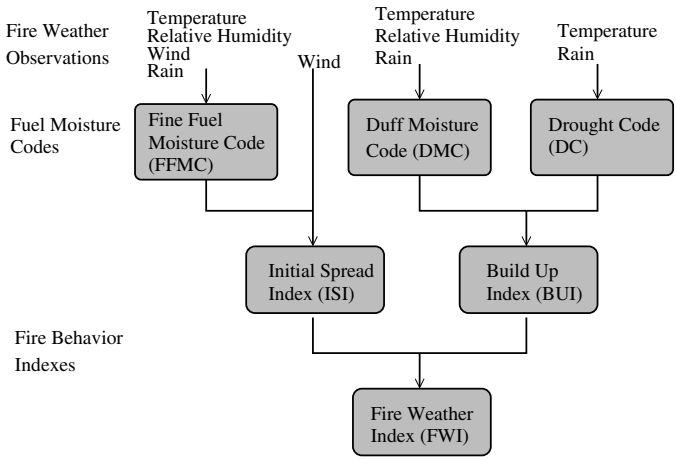


FIGURE 1
Structure of the Fire Weather Index (FWI) System.

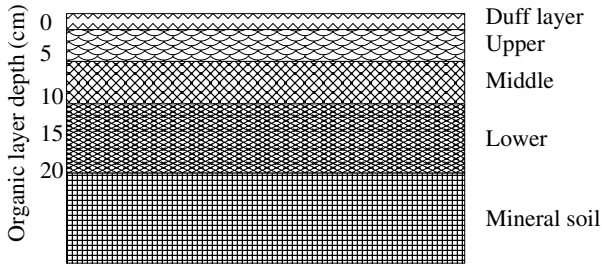


FIGURE 2
Forest soil layers.

3.1 Fuel codes of the FWI system

The forest soil can be divided into five different layers [12, 20] as shown in Figure 2. Each layer has specific characteristics and provides different types of *fuels* for forest fires. These characteristics are reflected in fuel codes of the FWI System. Related to each fuel type, there 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 moisture content with a noon temperature reading of 21°C, relative humidity of 45%, and a wind speed of 13 km/h [20]. Also, each fuel type has a fuel loading metric, which describes the average amount (in tonnes) of that fuel which 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 [20], 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 [20]. The DMC determines the probability of fire ignition due to lightning and also shows the rate of fuel consumption in moderate depth 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 deep layers. The DC scale is also open-ended, although the maximum probable value is about 800 [20].

3.2 Fire indexes of the FWI system

Fire indexes of the FWI System describe 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 Figure 1, 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 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.

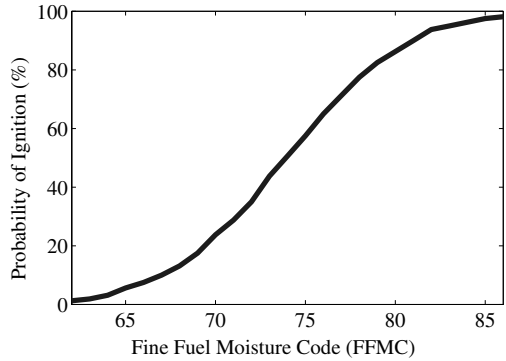
3.3 Interpreting, analyzing, and using the FWI system

The FWI System has been proposed in the forestry research community, and we claim no credit for it. In this paper, however, we distill and analyze the key features of the FWI System that could be used in designing a wireless sensor network for early detection of forest fires. This analysis is based on surveying many works from forestry research, collecting real data and images from various sources, and numerically analyzing complex formulas relating the FWI System to basic weather conditions. Since forest fire detection is frequently cited as one of the important applications of wireless sensor networks, we believe that our analysis of the FWI System is useful for the sensor networks community, and it could stimulate more research to solve the important forest fire detection problem.

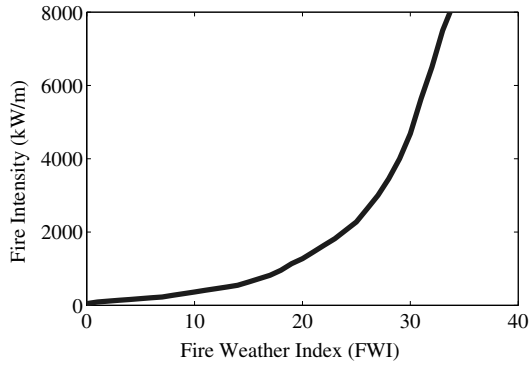
There are two goals of the proposed wireless sensor network for forest fires: (i) provide early warning of a potential forest fire, and (ii) estimate the scale and intensity of the fire if it materializes. Both goals are needed to decide on required measures to combat a forest fire. To achieve these 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. In the following, we justify the choice of these two components by collecting and analyzing data from several forestry research publications.

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 [20] to plot the probability of ignition as a function of FFMC. The results are shown in Figure 3(a). The FFMC scale ranges from 0–101 and is the only component of the FWI System without an open-ended scale. Generally, fires begin to ignite at FFMC values around 70, and the maximum probable value



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



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

FIGURE 3
Using two main components of the Fire Weather Index System in designing a wireless sensor network to detect and combat forest fires.

Ignition potential	FFMC value range
Low	0–76
Moderate	77–84
High	85–88
Very High	89–91
Extreme	92+

TABLE 2
Ignition Potential versus the FFMC value

that will ever be achieved is 96 [20]. Based on data available from the web site of The Sustainable Resource Development Ministry of the Province of Alberta, Canada, we classify in Table 2 the potential of fire ignition versus the FFMC 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 estimates 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 studies. For example, in 1974, the Alberta Forest Service performed a short term study of experimental burning in the Jack pine forests in north eastern Alberta. Snapshots of the resulting fires and the computed FWI indexes are shown in Figure 4 for three fires with different FWI values [5], we obtained a permission to reproduce these images. Another study [20] relates the fire intensity with the FWI index. We plot this relationship in Figure 3(b) by interpolating data from [20]. In Table 3, we provide a classification of fire danger as a function of the FWI index based on the data available from [12].

Both 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 sensing fidelity and deployment distribution of sensors impact the accuracy of the corresponding FFMC and FWI. Therefore, we need to quantify the impact of these weather conditions on FFMC and FWI. Using this quantification, we can design a wireless sensor network to produce the desired accuracy in FFMC and FWI. In addition, this quantification could help other researchers and sensor manufacturers to customize or develop new products that are more suitable for the forest fire detection application. To do this quantification, we contacted the Canadian Forest Service to obtain the closed-form equations that describe the dependence of FFMC and FWI on the weather conditions. We were given access to these complex equations as well as a program that computes them [49], we post an electronic copy of this report at [41] for interested researchers in this area.

(a) Moderate surface fire (FWI = 14)

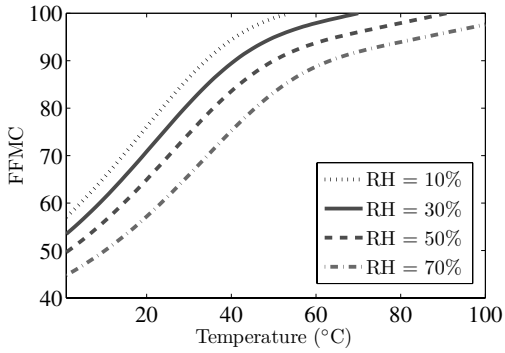
(b) Very intense surface fire (FWI = 24)

(c) Developing active fire (FWI = 34)

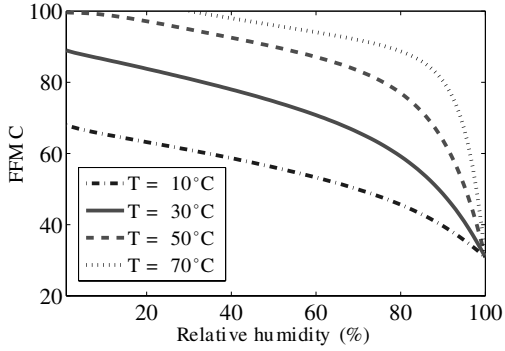
FIGURE 4
Experimental validation of the FWI index. Pictures shown from experiments conducted by the Alberta Forest Service, and are reproduced with permission.

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	Moderate 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

TABLE 3
Potential fire danger versus the FWI value



(a) FFMC versus Temperature

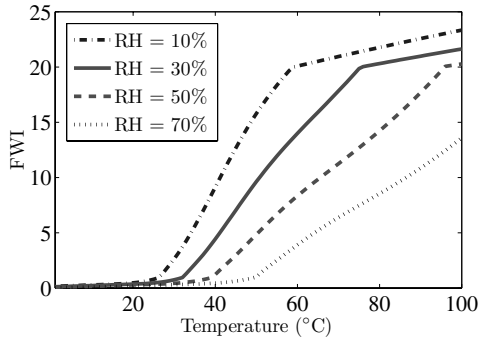


(b) FFMC versus Humidity

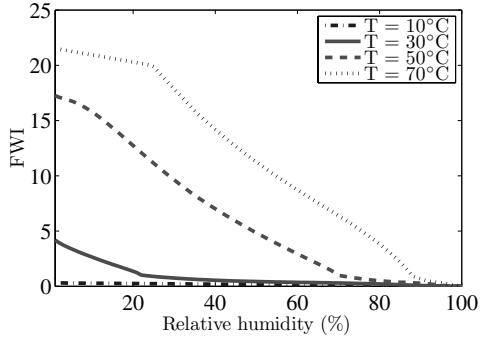
FIGURE 5
Sensitivity of the FFMC code to basic weather conditions.

We numerically analyze the sensitivity of FFMC and FWI to air temperature and relative humidity under different conditions and wide ranges of these parameters. A representative sample of our results are shown in Figure 5 and Figure 6. The sensitivity of FFMC to temperature and relative humidity is shown in Figure 5 for fixed wind speed at 5 km/h and precipitation level of 5 mm. Figure 6 shows the sensitivity of FWI to temperature and relative humidity under similar conditions. An interesting observation for sensor manufacturers is that the accuracy of the sensor readings is critical in high temperature ranges and when humidity is low, while fine accuracy is not that important outside these ranges. 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



(a) FWI versus Temperature



(b) FWI versus Humidity

FIGURE 6
Sensitivity of the FWI index to basic weather conditions.

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.

4 EARLY DETECTION OF FOREST FIRES USING WIRELESS SENSOR NETWORKS

In this section, we present the design of a wireless sensor network for forest fire detection. Indeed there are many research problems in such large-scale sensor network. We focus on a subset of them, and we leverage solutions for other problems in the literature, as outlined below.

The system considered in this paper is depicted in Figure 7. A sensor network deployed in a forest reports its data to a processing center for possible actions, such as alerting local residents and dispatching fire fighting crews. Sensors are deployed uniformly at random in the forest by, for example, throwing them from an aircraft. A single forest fire season is approximately six months (between April and October), and it is desired that the sensor network lasts for several seasons. Since the lifetime of sensors in active mode is much shorter than even a fraction of one season, sensor deployment is assumed to be relatively dense such that each sensor is active only during a short period of time and the monitoring task is rotated among all sensors to achieve the target network lifetime. Therefore, during the network operation, a small fraction of

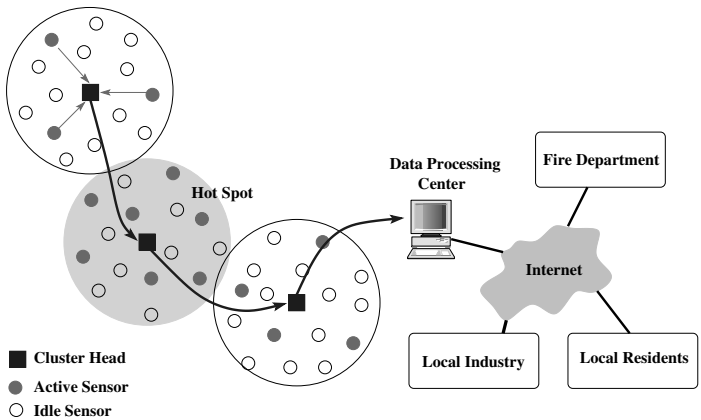


FIGURE 7
The architecture of the proposed forest fire detection system. Nodes self-organize into clusters, where cluster heads aggregate collected data using the FWI system. The shaded area represents a forest zone with higher fire potential and thus needs higher monitoring accuracy.

the deployed sensors are kept in *active* mode, while the rest are put in *sleep* mode to conserve energy. It is important to mention that the forest fire detection application considered in this paper works on a large time scale. Thus, active sensors are not continuously monitoring the area. Rather, they periodically (e.g., every 30 minutes) perform the sensing task. Therefore, sensors in the active mode are further divided into *active-sense* and *active-listen* modes. In the former, all modules (transmission, receiving, and sensing) of the sensor are turned on, while in the latter only the receiving module is on.

Sensors are assumed to self-organize into clusters using a distributed protocol. After the termination of the clustering protocol, sensors know their cluster heads and the whole network is connected. Any of the protocols described in the recent survey in [54] can be employed. Our proposed system does not restrict the cluster size, and it allows single- and multi-hop intra-cluster communications. The sensor clustering and data routing problems are outside the scope of this paper. We consider three problems in this paper. First, modeling the forest fire detection application as a coverage problem in wireless sensor networks, which we describe in the following subsection. Second, designing a new distributed coverage protocol. This is presented in two sections: Section 5 presents the theoretical foundation of our new algorithm and a centralized k-coverage algorithm; and Section 6 presents the distributed version of the new k-coverage algorithm. The final problem is achieving unequal fire protection in different zones in the forest, e.g., forest zones near industrial plants and residential areas, or forest zones with drier conditions and higher temperatures (denoted by hot spots). This is illustrated in Figure 7 by activating more sensors in the shaded hot spot area. We make the case for this unequal protection using real data and present a method to achieve it in Section 6.4.

4.1 Modeling forest fire detection as a coverage problem

We discussed in the previous section the relevance and importance of the FWI System, especially its FFMC and FWI components. We design our wireless sensor network for forest fire detection based on the FWI System. As shown in Figure 7, the deployed sensors are grouped into clusters, and each cluster elects a cluster head. Each cluster head periodically computes the FFMC and FWI for its cluster by sampling weather conditions from active sensors inside the cluster. This information is then forwarded—through multi hop routing—to a processing center for possible actions. Recall that FFMC and FWI are computed from basic weather conditions such as temperature and humidity (see Figure 1).

To be useful in detecting fires and assessing their intensity, FFMC and FWI need to be estimated within specific error bounds. For example, if the error in the estimated FWI is high (e.g. 5 units), the fire would be misclassified as indicated by Table 3. To achieve the desired accuracy in FFMC and FWI, basic weather conditions should, in turn, be measured accurately. The accuracy level of measuring basic weather conditions is determined from the curves

relating FWI and FFMC to weather conditions, such as Figures 5 and 6. For instance, the worst-case slope of the FWI-Temperature curve in Figure 6(a) at RH $= 10\%$ is about 0.62. Thus, an error up to 1 unit in FWI requires measuring the temperature with 1.6 degree accuracy. Knowing the needed accuracy in measuring weather conditions, the sensor network should be designed to collect data with that accuracy. We illustrate this design using temperature as an example, the same can be done for other metrics.

Consider measuring the temperature in an arbitrary cluster. Sensors in the cluster should be activated in a way that the samples reported by them represent the temperature in the whole cluster. This means that the cluster area should be covered by the sensing ranges of active sensors. This is called 1-coverage, or coverage with degree 1, because each point in the area is supposed to be within the sensing range of at least one sensor. In dense sensor networks and when sensors are deployed uniformly at random in the area—which is the case for forest fire detection systems as described above—area coverage can be approximated by sensor location coverage [52]. That is, we need to activate a subset of sensors to ensure that the locations of all sensors are 1-covered.

In real forest environments, sensor readings may not be accurate due to several factors, including: (i) different environment conditions (e.g., some sensors happen to be in the shade of trees, while others are not), (ii) inaccurate calibration of sensors, (iii) aging of sensors, and (iv) unequal battery levels in sensors. In addition, to cover large forests, sensing ranges of deployed sensors will have to be large (in order of hundreds of meters), which may introduce more errors in the sensor readings. Therefore, multiple (k) samples may be needed to estimate the temperature at a location with the target accuracy. That is, each location needs to be sensed by k different sensors. This is called k-coverage, where $k \geq 1$. The actual value of k depends on the expected error in the sensor readings and the tolerable error in the FFMC and FWI indexes. One way to estimate k is described in the following.

We define a random variable T as the reading of a sensor inside the cluster. It is reasonable to assume that T follows a normal distribution because of the many factors contributing to it, which all are naturally stochastic. We denote the mean and standard deviation of T as μ_T and σ_T, respectively. The estimated mean $\widehat{\mu}_T$, also known as the sample mean, is given by: $\widehat{\mu}_T = \frac{1}{k}\sum_{i=1}^{k} t_i$, where t_is are the individual sensor readings, and k is the number of samples. As the number of samples increases, the sample mean becomes closer to the actual mean. The error between the sample mean and the population mean, $\delta_T = |\mu_T - \widehat{\mu}_T|$, is calculated as follows [47]: $\delta_T = z_{\frac{\alpha}{2}} \frac{\sigma_T}{\sqrt{k}}$, 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 obtained from tables of the standard normal distribution. Rearranging the formula, we get:

$$k = \left\lceil \left(z_{\frac{\alpha}{2}} \frac{\sigma_T}{\delta_T} \right)^2 \right\rceil. \tag{1}$$

Thus, 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 δ_T error margin. σ_T can be calculated from the specifications of the sensing board. The error in sensor measurements is usually interpreted as $2\sigma_T$. To illustrate, suppose we want to measure the temperature with a maximum error of 1°C and with a confidence value of 95%. Assume that sensors have temperature sensing boards with an error up to 2°C, i.e., $\sigma_T = 1$. Therefore, we need a coverage degree $k = (1.96 \times 1/1)^2 = 4$. In the evaluation section, we study and validate the relationship between the coverage degree k and the error in FFMC and FWI. We also study the tradeoff between the error in the sensor readings σ_T and the required coverage degree k to meet given target errors in FFMC and FWI.

To summarize, in this section we have established a mapping between the forest fire detection system and the k-coverage problem ($k \geq 1$) in sensor networks. We showed how k can be estimated based on the error in sensor readings and the maximum tolerable errors in estimating the FFMC code and FWI index. The tolerable errors in FFMC and FWI can be estimated from Figures 5, 6 and Tables 2, 3, based on the application requirements. After computing k, we need to activate a subset of sensors to ensure k-coverage, and keep other sensors in sleep mode to conserve energy. In Section 6, we present a distributed protocol to achieve this.

5 THE k-COVERAGE PROBLEM IN WIRELESS SENSOR NETWORKS

To achieve k-coverage ($k \geq 1$) in different clusters of the monitored forest, we need a distributed, energy-efficient, algorithm. In this section, we first present the theoretical foundation of our new k-coverage algorithm. Then, we present a centralized k-coverage algorithm, and we show that it activates a number of sensors that is at most a constant factor from the optimal one. Although the theoretical analysis seems somewhat complex, the algorithm itself is quite simple. More importantly, the algorithm can be implemented in a distributed manner with low message complexity. In Section 6, we present and analyze the distributed algorithm. In the evaluation section, we rigorously validate the correctness and efficiency of both the centralized and decentralized k-coverage algorithms.

5.1 Problem statement and our solution approach

Dense sensor networks are typical for many real-life applications that are expected to last for long periods, such as forest fire detection and habitat monitoring. In dense sensor networks, area coverage can be approximated by node coverage as indicated by [14, 15, 52]. In addition, the authors of [34] show that if a given set of grid points are k-covered by a set of sensors each of

sensing radius r, then the entire area is k-covered by the same set of sensors but each with a slightly larger radius. Therefore, k-coverage of the entire area can easily be guaranteed by node coverage algorithms if we configure them to use a slightly smaller radius. In this paper we assume that the set of points to be covered are the same as the set of sensor locations and therefore present node coverage algorithms to solve the k-coverage problem.

The node k-coverage problem is formally stated as follows.

Problem 1 (Node *k*-Coverage Problem). *Given n already-deployed sensors, and a desired protection degree $k \geq 1$, select a minimal subset of sensors to monitor all sensor locations such that every location is within the sensing range of at least k different sensors.*

The above k-coverage problem is proved to be NP-hard by reduction to the minimum dominating set problem in [52]. 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. Finding the minimum number of nodes to provide 1-coverage for 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 efficient approximation algorithms for solving this problem.

The proposed algorithms may not be suitable for low-density sensor networks. In low-density sensor networks, however, most or all sensors will have to be activated to maintain coverage, and therefore, there is little room for optimization using any algorithm. Furthermore, we assume that the sensing range of sensors is a disk of radius r. This assumption has been made by numerous previous works, including [14, 15, 17, 32, 34, 44, 45, 51, 52]. This is a reasonable assumption for several types of sensors in which a threshold distance can be defined such that the sensing of an event happening within this threshold is detected with high probability. Nonetheless, sensing ranges of some sensors may be better modeled by probabilistic distributions [3, 13, 36], and there have been a few coverage protocols that account for this probabilistic nature [28, 58]. All of these protocols can only provide up to 1-coverage. Since the k-coverage problem is much more challenging than the 1-coverage problem, we focus on k-coverage with the disk sensing model and we leave the extension to probabilistic sensing models for future work.

Next, we describe our solution approach for Problem 1. We start with the following definition of set systems and hitting sets [10].

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 $R \cap H \neq \emptyset$.

We map the k-coverage problem in sensor networks to a set system. We define X to be the set of discrete points representing all locations of the n

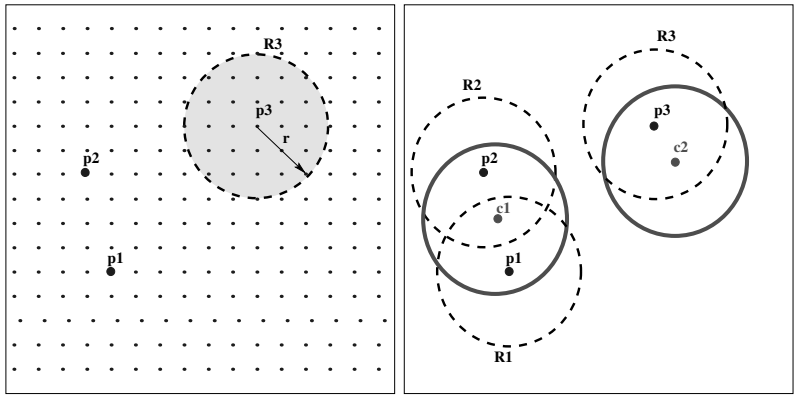


FIGURE 8
Modeling the coverage problem as a set system $(X, \mathcal{R})$.

deployed sensors in the monitored area. 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 (the sensing range) centered at p. All points in X that fall within that circle constitute one element in $\mathcal{R}$. Figure 8(a) shows sensors deployed in the area and highlights one element of $\mathcal{R}$ (the shaded circle R_3). We say that a point $c \in X$ *hits* an element $R \in \mathcal{R}$ if c intersects with R. For example, in Figure 8(b), c_2 hits R_3. Notice that if the sensor at c_2 is activated, it will cover the center of R_3 (point p_3). A single point may hit multiple elements of $\mathcal{R}$. Figure 8(b) shows two elements of $\mathcal{R}$ being hit by c_1. The goal of this modeling is to find the minimum number of points that hit all elements of $\mathcal{R}$. These points constitute a minimum hitting set. Since there is an element in $\mathcal{R}$ for each point in X, the minimum hitting set will provide protection for all points in X. This is 1-coverage. For example, if we have only the five points $\{p_1, p_2, p_3, c_1, c_2\}$ in X, then, as shown in Figure 8(b), $\{c_1, c_2\}$ will form a minimum hitting set. Therefore, activating sensors at c_1 and c_2 will achieve 1-coverage for all five points.

For k-coverage, we repeat the procedure for 1-coverage up to k times, while making sure that nodes are not over-covered. The problem now reduces to finding a minimum hitting set for the proposed set system. Since finding the minimum hitting set is NP-hard, we design an approximation algorithm to find a near optimal hitting set. The proposed algorithm uses the concept of ε-nets [27], which is defined as follows.

Definition 2 (ε-net). Let ε be a constant $(0 < \varepsilon \leq 1)$. 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 $\varepsilon|X|$, that is, $\forall R \in \mathcal{R}$ where $|R| \geq \varepsilon|X|$ 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 $\mathcal{R}$ (the ones that are greater than or equal to $\varepsilon|X|$), while the hitting set must hit every element of $\mathcal{R}$. Under certain conditions on the set system, ε-nets can be computed efficiently (in constant time). It is why we use them. The idea of our algorithm is to find ε-nets of increasing sizes till one of them hits all elements of $\mathcal{R}$, i.e., an ε-net becomes a hitting set. And we develop bounds on how large this ε-net is compared to the minimum hitting set. We also show hat this ε-net can indeed be found in a finite number of steps.

ε-nets can be computed efficiently if the considered set system has a finite VC-dimension (defined below) [27]. Therefore, we need to prove that the set system for the coverage problem has this property, which we will do shortly. We first present the definition of the VC (Vapnik and Červonenkis) dimension of a set system and the associated concept of *set shattering* [10].

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 $Y \cap B = A$. The cardinality of the largest set that can be shattered by $\mathcal{R}$ is called the VC-dimension of the set system.

Informally, Y is shattered by $\mathcal{R}$ if all subsets of Y can be constructed by intersecting Y with some elements of $\mathcal{R}$. To illustrate, consider a set of points X and a collection $\mathcal{R}$, where the elements of $\mathcal{R}$ are disks of radius r. Define $Y \subseteq X$. Let Y contain only two points y_1, y_2. Thus, there are only four possible subsets of Y: $\{\emptyset\}$, $\{y_1\}$, $\{y_2\}$, $\{y_1, y_2\}$. As shown in Figure 9(a), all subsets of Y can be constructed by interesting different elements (disks) of $\mathcal{R}$ with Y. For example, intersecting R_1 with Y produces $\{y_1\}$, while interesting R_4 with Y produces $\{\emptyset\}$. Therefore, the VC-dimension of the set system $(X, \mathcal{R})$ is *at least* 2 because $|Y| = 2$. Following a similar argument for $Y = \{y_1, y_2, y_3\}$, Figure 9(b) shows that the VC-dimension of $(X, \mathcal{R})$ is greater than or equal

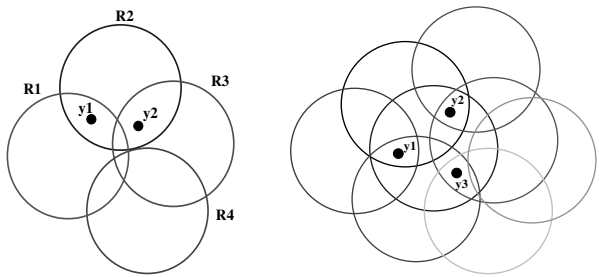


FIGURE 9
The concept of set shattering: (a) two points shattered by four disks, and (b) three points shattered by eight disks.

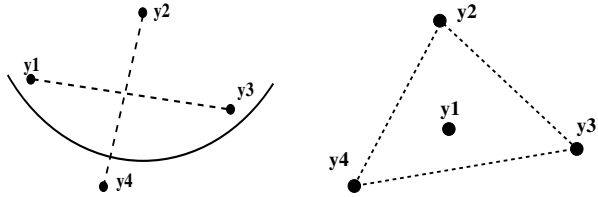


FIGURE 10
No four points can be shattered by disks of same radius.

to 3, because all eight subsets of Y can be constructed. The following lemma shows that shattering cannot be done if Y has four points, and therefore the VC-dimension of $(X, \mathcal{R})$ is exactly 3.

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.*

Proof. First we need to show that there exists a set of three points that can be shattered by $\mathcal{R}$. Figure 9(b) shows this by construction, as discussed above. Second, we show that no four points can be shattered by $\mathcal{R}$. As illustrated in Figure 10, any four points are either on their convex hull (Figure 10(a)), or one point is contained by the triangle formed by the other three points (Figure 10(b)). In the first case, two opposite points cannot be separated from the other two. For instance no disk in $\mathcal{R}$ can contain only points $\{y_1, y_3\}$ without including either point y_2 or y_4, or both. Thus, not all subsets of four points can be constructed by interesting the elements of $\mathcal{R}$ with the set of four points. Notice that the four points cannot be too far apart, otherwise no disk in $\mathcal{R}$ can contain all of them. Similarly, in the second case it is not possible to separate the points on the triangle from the point inside it. Hence, no four points can be shattered and the VC-dimension of the set system $(X, \mathcal{R})$ is 3.

To summarize, we modeled the coverage problem as a set system $(X, \mathcal{R})$, where X is the set of sensor locations and $\mathcal{R}$ is a collection of subsets of X. We showed that this set system has a VC-dimension of 3. And because it has a finite VC-dimension, ε-nets for this set system can be constructed efficiently using simple methods (explained in Section 5.3). The idea of our solution is to find ε-nets of increasing sizes till one of them becomes a hitting set. Point in the hitting set represent sensors which can provide sufficient coverage upon activation. □

5.2 Overview of the centralized k-coverage algorithm

In the previous section, we presented the theoretical foundations of our solution approach. In this section, we present an overview of the proposed algorithm.

This is followed by more details and analysis of the algorithm in later subsections. An 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 Section 6.

The proposed centralized k-coverage algorithm is denoted by RKC (Randomized k-Coverage algorithm). The pseudo code of RKC is given in Figure 11. The algorithm takes as input the set of points representing the 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,

Centralized k-Coverage Algorithm: RKC(X, r, k)

```
1.  H = ∅;
2.  for i = 1 to k
3.      set weights of all active nodes to 0;
5.      ε = 1; // sets the initial size of ε-net
6.      while (S(ε) ≤ n) do
7.          set weights of all inactive nodes to 1;
8.          for j = 1 to T(n, ε)
9.              N = net-finder(X, ε);
10.             u = verifier (X, N, i, r);
11.             if (u == null)
12.                 add N to H;
13.             else
14.                 double weights of neighbors of u;
15.         ε = ε/2;
16.   if S(ε) > n
17.     return ∅;
18.  return H;
```

Function net-finder(X, ε)

```
1.  net = ∅;
2.  for i = 1 to S(ε)
3.      q = getRandomPoint(X); //biased
4.      add q to net;
5.  return net;
```

FIGURE 11
A centralized approximation algorithm for the k-coverage problem.

it has to be in the sensing range of at least k sensors. Thus, for each node p, there should be at least k nodes inside a disk of radius r centered at p.

The algorithm runs in rounds, where each round tries to increase the coverage of the nodes by at least 1. Thus, at round i ($1 \leq i \leq k$), all sensor nodes are at least i-covered. All nodes are assigned a weight which is initialized to 0 for active nodes and to 1 for all other nodes at the beginning of a new round. This prevents the algorithm from activating a node more than once thus producing only feasible solutions. As detailed later, weights accelerate the process of finding a near-optimal hitting set and help in establishing an upper bound on its size.

In each round, the algorithm tries ε-nets of increasing sizes and proceeds to the next round whenever one of them increases coverage of all nodes by at least 1. This is checked using a coverage verifier function. The algorithm terminates if the size of the required ε-net ever becomes larger than the total number of sensors. This happens when node density is not enough to achieve coverage. In every round, the algorithm starts with the smallest possible ε-net ($\varepsilon = 1$) and in every iteration of the while loop, the algorithm doubles the size of the ε-net (i.e., ε is divided by 2). ε-nets are constructed using the function net-finder (described in Section 5.3).

In every single iteration of the while loop, the algorithm tries up to $T(n, \varepsilon)$ ε-nets one at a time (the for loop in lines 8–14), all have the same size. For each ε-net, the verifier checks whether it is a hitting set, i.e., it completely 1-covers all points. If so the algorithm adds all nodes in the ε-net to the solution for k-coverage and proceeds to the next round. Otherwise, the algorithm doubles the weight of one set that is not hit by the current ε-net. This means that the weights of all the neighbors of an under-covered point are doubled. Then, the algorithm tries another ε-net of the same size. Points with increased weights will have higher probability of being included in the new ε-net and thus providing coverage for the under-covered points. The number of ε-nets $T(n, \varepsilon)$ is chosen such that if we try all of them—while doubling the weights after each trial—and none of them is shown to be a hitting set, it is guaranteed that no ε-net of this size can be a hitting set. Thus, the size of the ε-net has to be increased to find a hitting set. The exact value of $T(n, \varepsilon)$ is given in the next subsection.

5.3 Details of the centralized algorithm

The RKC algorithm uses two functions: coverage verifier and net-finder. The verifier algorithm returns an under-covered point, if there is any, or null if all points are i-covered ($1 \leq i \leq k$). In case that the required coverage is not achieved, the weight of neighbors of the returned point will be doubled before constructing the next ε-net. We employ a simple coverage verifier that checks all points one by one. More complex verifiers can also be used with our centralized algorithm. For example, the authors of [45] design a k-coverage verifier using order-k Voronoi diagrams. However, such verifiers are difficult to decentralize and they may require nodes to know their locations.

The net-finder function constructs an ε-net based on the ε value passed to it and the weights of points in X. ε-nets are known *abstract* concepts that have been studied before in theoretical Computer Science literature., e.g., [27, 39]. We employ the method in [27] for constructing ε-nets. This method states that randomly selecting at least

$$\max\left(\frac{4}{\varepsilon}\log\frac{2}{\delta}, \frac{8d}{\varepsilon}\log\frac{8d}{\varepsilon}\right) \tag{2}$$

points of the set X yields an ε-net with a probability at least $1-\delta$, where δ is a constant ($0 < \delta < 1$), and d is the VC-dimension of the set system (X, R). We showed in Lemma 1 that $d = 3$ for the set system modeling the coverage problem. We choose this method for constructing ε-nets because implementing it in a distributed manner is not difficult, as it chooses points randomly. In contrast, the method in [39] involves triangulation, which requires sensors to be aware of their locations, and more importantly, it is not clear how it can be decentralized.

Next, we determine $S(\varepsilon)$, which is the size of the ε-nets returned by the net-finder function and used by the RKC algorithm. $S(\varepsilon)$ will also be used in Section 5.4 to bound the approximation factor of the output size produced by our algorithm compared to the optimal size. For mathematical convenience, let us set $\delta = 2 \times 10^{-C}$, where C is a constant. Thus, in Equation (2) for the first term to dominate the second in the max() function, we must have $\frac{4}{\varepsilon}\log 10^C \geq \frac{24}{\varepsilon}\log\frac{24}{\varepsilon}$. This leads to $\varepsilon \geq 24 \times 10^{-C/6}$. The minimum value for ε occurs when its corresponding (largest) ε-net is required to hit each individual element of X (of cardinality 1), that is, $\varepsilon|X| = 1$. Therefore, the minimum value of ε is $1/|X| = 1/n$. Using this minimum value of ε, we get $n \leq \frac{1}{24}10^{C/6}$ as a sufficient condition for the first term in Equation (2) to always be larger than or equal to the second. In this case, the size of the ε-net is given by $S(\varepsilon) = 4C/\varepsilon$. The condition $n \leq \frac{1}{24}10^{C/6}$ does not limit our algorithm in any way, because the constant C can always be chosen to satisfy it. For example, if we set $C = 60$, a sensor network using our algorithm can have up to approximately a billion sensors, and the probability of finding an ε-net by randomly selecting $4C/\varepsilon$ points will be at least $1 - 2 \times 10^{-60}$.

Now, we describe the pseudo code of the net-finder function. The net-finder iterates $S(\varepsilon)$ times, and in every iteration it selects a random point q biased on the weights and adds it to the *net* variable. 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.

Finally, we determine $T(n, \varepsilon)$, which is the number of ε-nets (of the same size) that the RKC algorithm tries for each ε value. Note that if an ε-net becomes a hitting set (checked by the coverage verifier), the RKC algorithm adds it to the final solution and proceeds to the next round. If the checked ε-net is not a hitting set, the algorithm doubles the weight of neighboring nodes of

an under-covered node, effectively doubling the weight of one set that is not hit by the current ε-net. The weight-doubling process is explicitly included in the RKC algorithm to leverage a previous result on ε-nets [10], which states that, for a given ε if the weight-doubling process is repeated at least $4h \log \frac{n}{h}$ times and none of the ε-nets is found to be a hitting set, then it is guaranteed that there is no hitting set of size h. We set $T(n, \varepsilon) = 4S(\varepsilon) \log \frac{n}{S(\varepsilon)}$, and will use this result in computing the approximation factor of the RKC algorithm in the following subsection.

Remarks. We should clarify our contributions from what we leverage from previous works. The abstract concept and the method of constructing ε-nets have been proposed before, and we claim no credit for them. However, the mapping of the coverage problem in sensor networks as a set system and proving the conditions under which we can use ε-nets are our own contributions. In addition, our contributions include putting various pieces into a complete k-coverage algorithm, as well as proving the constant-factor approximation on the output size and analyzing the complexity of the algorithm. Furthermore, the distributed k-coverage algorithm presented in Section 6 is totally new. Finally, all of our results and algorithms are validated through extensive simulations on networks with thousands of sensors.

5.4 Analysis of the centralized algorithm

In this subsection, we prove the correctness of the RKC algorithm, show that it has a constant-factor approximation, and analyze its time complexity. The following theorem proves that the RKC algorithm is correct, and provides the upper bound on the solution returned by it.

Theorem 1. *The k-coverage algorithm (RKC) in Figure 11 ensures that every sensor location is k-covered, and returns a solution of size at most a factor of $2k$ larger than the minimum number of sensors required for k-coverage.*

Proof. Suppose that the algorithm terminates by providing a set Y of sensors. By construction, this set of points is guaranteed to hit every disk of radius r at least k times. Since for our set system $(X, \mathcal{R})$, we put a disk in $\mathcal{R}$ for each node $p \in X$, there should be at least k elements in Y that hits the disk centered at p. In addition, each of these k sensors is within a distance r from p. Therefore, p is k-covered by this set of sensors. Hence, all nodes are k-covered by sensors in Y.

Next, we consider the bound on the solution size. Consider any arbitrary round and assume that the algorithm finds a hitting set when the value of ε is ε', that is ε'-net is a hitting set. Denote the size of this solution as $S(\varepsilon')$. Since the algorithm proceeds to the next round as soon as it finds a first hitting set, it must have not found a hitting set for $2\varepsilon'$. As explained in Section 5.3, therefore, the hitting set must be larger than $S(2\varepsilon')$. That is, $S(2\varepsilon') < N^*$, where N^* is the minimum number of sensors required for 1-coverage. It is shown in Section 5.3 that $S(\varepsilon') = 4C/\varepsilon'$, where C is a constant. Thus, $S(2\varepsilon') = 4C/(2\varepsilon') = S(\varepsilon')/2$. Therefore, the size of the hitting set is $S(\varepsilon') < 2N^*$.

The algorithm repeats for k rounds and adds a hitting set of size at most $2N^*$ to the final solution H at each round. Thus, the final solution for k-coverage includes at most $2kN^*$ sensors where N^* is the minimum number of sensors required for 1-coverage. Note that the actual approximation factor of our algorithm is indeed much smaller than $2k$ since the optimal solution for k-coverage contains fewer number of sensor than kN^*.

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

Theorem 2. *The k-coverage algorithm terminates in time* $O(n^2 \log^2 n)$*, where n is the number of sensors.*

Proof. The algorithm runs for k similar rounds, so we perform the analysis for an arbitrary round. Since the size of the ε-net is doubled (because ε is halved) in every iteration of the While Loop (lines 7–15 in Figure 11), the While Loop can iterate at most $\log n$ times, and the algorithm indeed terminates. In each round, the algorithm iterates for $T(n, \varepsilon) = \frac{4}{\varepsilon} \log(n\varepsilon)$ steps for each ε in the For Loop (lines 9–14 in Figure 11). In the worst case, there are up to $O(n)$ iterations in the For Loop. Therefore, the running time of the algorithm is $\log n[O(n) + O(n)(T_F + T_V)]$, where T_V and T_F are the time complexities of the verifier and the net-finder functions, respectively. The verifier runs in $O(n)$. The net-finder runs in $O(n \log n)$, because the maximum $S(\varepsilon)$ is $O(n)$, and the function getRandomPoint() takes $O(\log n)$ steps. The running time of getRandomPoint() is justified as follows. To select a random point based on weights, we maintain weights in a cumulative list, where 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. Substituting T_F, T_V in the running time of the algorithm completes the proof. □

We note that the centralized algorithm is designed as an intermediate step for the distributed algorithm presented in the next section, where the computation will be distributed across all nodes. Therefore, the time complexity of the centralized algorithm is not a big concern, because no single node will perform the whole computation. Also, recall that the problem is NP-hard.

6 DISTRIBUTED 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 distributed 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. Finally, we show how our algorithm can be extended to achieve unequal monitoring of different subareas in the forest, and we present a simple data aggregation scheme based on the FWI System.

A an important feature of the proposed distributed k-coverage algorithm is that it does not require sensors to know their locations. This is a significant advantage in large-scale sensor networks, because location information is acquired by either equipping sensors with GPS modules, or using localization protocols. The *relative* cost of adding a GPS module is indeed non-negligible. For example, the price of the Crossbow MTS400 Weather Sensor Board is \$210, while for MTS420 it is \$299; the only difference is that MTS420 is equipped with a GPS module. That is, there is more than 40% increase in the cost of sensor boards because of the GPS module. These prices were obtained from Crossbow's web site [18] in September of 2007. Localization protocols, on the other hand, consume a non-trivial fraction of the energy of sensors, because they require exchanging messages and running location estimation algorithms. Therefore, they could shorten the network lifetime. In addition, the estimation errors of the localization protocols, which may not be small [35], could negatively impact the operation of the k-coverage protocols that rely on location information.

6.1 Overview of the distributed algorithm

The centralized k-coverage algorithm (shown in Figure 11) maintains two global variables: the size of the current ε-net, and weights of all points. At every iteration of the While loop, the size of the ε-net is doubled, and at every iteration of the For loop, the weights of neighbors of one under-covered node is doubled. The size of the ε-net is used to determine the number of nodes that need to be activated to achieve coverage. The weight doubling, on the other hand, ensures an upper bound on the solution size. The basic idea of of our distributed algorithm, which we call DRKC (Distributed Randomized k-Coverage algorithm), is to emulate the centralized algorithm by trying different ε-nets of growing size and *locally* verifying the coverage. Similar to the centralized algorithm, DRKC uses the weights to control the solution size. To simplify the presentation, we first assume that all nodes are time-synchronized. In Section 7, we show through simulations that only coarse-grained time synchronization is sufficient for our algorithm.

The ε-net size is computed 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 the same time, they can know the size of the ε-net for the current iteration. 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.

In the centralized algorithm, node weights are used in the net-finder algorithm to add nodes to the current ε-net biased on their weights. Similarly, in DRKC, a node becomes part of the ε-net with a probability proportional to its weight relative to total weight of all nodes. Knowing 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 becomes active and notifies its neighbors to increase their coverage, *curCoverage*.

Finally, coverage verification in the centralized algorithm is done by checking all nodes one by one. In the distributed algorithm, each node independently checks the coverage of its neighbors which is already known through UPDATE messages.

If a node finds its neighbors sufficiently covered it does not contribute to the ε-net. This is achieved by setting a 0 weight for itself. Moreover, a node that is already active takes a 0 weight too. Therefore, the probability of becoming active in the next iteration $p = (weight/totalWeight) \times netSize$ is 0 for such nodes. These nodes continue running the iterations which helps keep all nodes synchronized.

In each iteration all nodes double the *netSize*, and therefore run for at most $\log n$ steps until $netSize > n$. Hence, if a node is not redundant (i.e. all its neighbors are not already covered) it becomes active eventually. Thus, the algorithm terminates by finding a solution to the coverage problem if there exists one.

6.2 Details of the distributed algorithm

DRKC works in rounds of equal length. The round length 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 till 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 till the beginning of the next round.

The pseudo code of DRCK is given in Figure 12. A node can be in one of three states: ACTIVE, SLEEP, and TEMP. A node starts a round in the TEMP state, where it initializes parameter *curCoverage*. 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. In ACTIVE state, all modules (transmission, receiving, and sensing) are turned on, while all modules are turned off in SLEEP state.

Distributed k-Coverage Algorithm (DRKC)

DRKC Sender

```
1. while (true) {
2.       curCoverage = 0;
3.       state = TEMP;
4.    for i = 1 to k {
5.       /* initialize parameters */
6.       weight = 1, totalWeight = n, netSize = 1;
7.       while (netSize ≤ n) {
8.          if (netSize × (weight/totalWeight) > rand()) {
9.             state = ACTIVE;
10.            broadcast a NOTIFY message to neighbors;
11.         }
12.         wait for UPDATE messages;
13.         /* verify covergae of neighbors*/
14.         if (all neighbors are i–covered or state == ACTIVE )
15.            weight = 0;
16.         /* update variables for next iteration */
17.         netSize = netSize × 2;
18.      }
19.   }
20.    if (state ≠ ACTIVE ) state = SLEEP;
21.    wait till end of round;
22.}
```

DRKC Receiver

```
/* upon receiving a message msg */
1.    update (neighbors, curCoverage);
2. if (msg.type == NOTIFY ) {
3.    broadcast an UPDATE message to neighbors;
4. }
```

FIGURE 12
A distributed algorithm for the k-coverage problem.

After initialization, the algorithm iterates up to $\log n$ times in the while loop. 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 becomes active and broadcasts a NOTIFY message to its neighbors to increase their coverage. When a neighbor receives a NOTIFY message, it increases its coverage and broadcasts an UPDATE

message informing all its neighbors of its current coverage. To reduce collisions between UPDATE messages, a node waits a random period between 0 and $neighborSize \times T_m$, before sending the message, where T_m is the average transmission time of a message and *neighborSize* is the number of neighbors.

After waiting enough time to receive UPDATE messages, a node verifies its neighbors' coverage. If the neighbors are sufficiently covered, it sets its own weight to 0 that prevents it from being selected as a member of the ε-net. After repeating the steps for $\log n$ times, the node starts a new stage for the next coverage degree. This is repeated k times until all nodes are fully k-covered.

6.3 Analysis of the distributed algorithm

The distributed DRKC coverage algorithm emulates the centralized RKC algorithm. Therefore, if nodes are time-synchronized and execute the iterations of the algorithm at about the same time, the output result of DRKC will be the same as RKC. The simulation experiments in Section 7 confirm that DRKC produces very close results to RKC even when nodes are not perfectly synchronized.

Next, we analyze the communication and computation complexities of the proposed distributed k-coverage algorithm. 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 (UPDATE and NOTIFY) have fixed sizes. Therefore, we analyze the communication complexity in terms of number of messages exchanged.

Theorem 3. *The number of messages sent by a node in any round of the DRKC protocol is* $O(1)$ *in the worst case.*

Proof. A node broadcasts a NOTIFY message if it becomes active which could happen only once. Moreover, each node may broadcast an UPDATE message when it receives a new NOTIFY messages from its immediate neighbors. Therefore, the total number of messages sent by a node is constant, $O(1)$. □

The time complexity is presented in the following theorem.

Theorem 4. *The worst-case time complexity for a node in any round of the DRKC protocol is* $O(\log n)$.

Proof. A node runs two components: DRKC Sender and DRKC Receiver. In any round, a node running the DRKC Sender code may iterate in the while loop (lines 7–17) up to $O(\log n)$ times, because the *netSize* parameter is doubled in every iteration. Each iteration of the loop takes $O(1)$ steps. The DRKC Receiver processes up to a constant number of messages from neighboring nodes

The low communication and computation complexity of the DRKC algorithm, coupled with the fact that it does not require node location information, makes it appealing for practical sensor network applications such as forest fire detection. □

6.4 Unequal monitoring of forest zones and data aggregation

In this subsection, we demonstrate how our DRKC algorithm can be extended to address an application-specific requirement, which is unequal monitoring of different zones in the forest. We also present a simple data aggregation scheme suitable for the forest fire detection problem.

We are interested in the realistic behavior of forest fires. To that end, we have collected and analyzed real data on the fire danger rating produced by the Protection Program of the Ministry of Forests and Range in the Province of British Columbia, Canada. Sample of the data is shown in Figure 13 for 23 July 2007. The figure shows several hot spots with 'High' danger rating within larger areas with 'Moderate' rating. The number, size, and locations of the hot spots are dynamic, because they depend on weather conditions. Maps such as the one shown in Figure 13 are produced daily, and they exhibit similar patterns. This is intuitive because some areas may have higher fire potential than others. For example, dry areas at higher elevations are more susceptible to fires than lower and more humid areas. Moreover, it is usually important to monitor parts of the forest near residential and industrial zones with higher reliability and accuracy. Therefore, based on the analysis of real data, we can conclude that unequal monitoring of different forest zones is an important issue in forest fire detection systems.

To support unequal monitoring of forest zones, we propose to cover the forest with different degrees of coverage at different zones. Intuitively, in

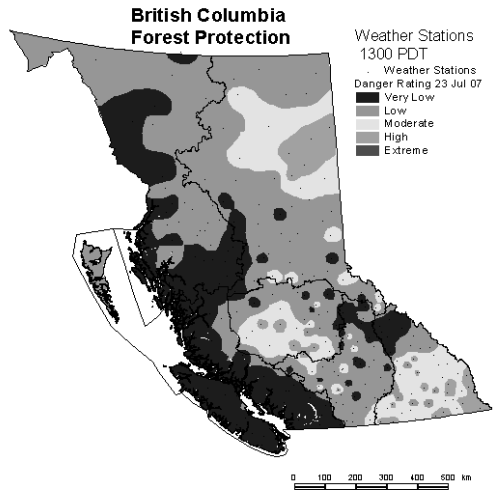


FIGURE 13
The need for coverage with different degrees in forest fires. The picture shows different fire danger levels at different zones. Reproduced with permission from the Ministry of Forests and Range, Protection Program, BC, Canada.

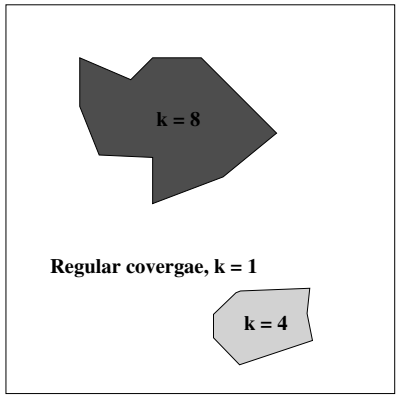


FIGURE 14
Modeling forest zones that require different degrees of coverage as polygons.

hot spots, the FFMC and FWI are expected to be in the high ranges of their scales, and small errors in these ranges could lead to mis-classifying a fire and/or taking wrong re-actions. For example, the 'Very High' range of FFMC in Table 2 is 89–91 (only two units), while the 'Low' range is 0–76. As discussed in Section 4.1, higher accuracy in computing FFMC and FWI require collecting weather conditions more accurately, which can be achieved by controlling the coverage degree k.

We extend our distributed k-coverage algorithm (DRKC), described in Section 6, to support coverage with various degrees at different zones in the forest at the same time. We are not aware of any other coverage protocol in the literature that supports this feature. We first model areas requiring different coverage degrees as polygons, an example is shown in Figure 14. Then, the vertices of each polygon are communicated to all cluster heads in the network. Each cluster head in turn can determine whether they are within the area with the different coverage. If this is the case, it notifies the sensors in its own cluster to adjust their operation to achieve the new requested coverage degree. This is easily done by our DRKC algorithm, because coverage verification in DRKC is done by individual nodes. In the evaluation section, we verify that coverage with various degrees can indeed be achieved by DRKC.

As discussed in Section 6, our DRKC does not use any location information. Thus, it saves the overhead of localization protocols, or the cost of equipping sensors with GPS, which is a significant saving considering the scale of the forest fire detection system. However, cluster heads need to determine whether or not they are inside some hot spots. This can be achieved by associating sensor IDs to their approximate locations during the deployment process. For example, during deployment, sensors with specific ID ranges can be thrown by the aircraft in target geographical locations. This mapping is maintained by the data processing center to dynamically configure the sensor network. It

is important to emphasize that the approximate locations do not impact the operation of our DRKC protocol, they are only used to delineate hot spots. Hot spots are usually measured in kilometers, and thus approximate locations are suitable for specifying them.

Finally, we present a simple data aggregation scheme explicitly designed for forest fire detection applications. Based on our analysis of the FWI System in Section 3.3, the application can interpret and uses only the FFMC code and the FWI index. Thus, individual sensor readings of various weather conditions may not be of interest to the application. Therefore, there is no need to deliver all these detailed data to the processing center. We propose that cluster heads aggregate individual sensor readings by computing the FFMC and FWI using their respective closed-form equations [49]. Each cluster head periodically collects weather conditions from sensors in its cluster and computes FFMC and FWI. Cluster heads carry out significant load, because they compute FFMC and FWI from complicated equations and participate in data forwarding across clusters. Hence, unless the role of the cluster head is rotated, heads run out of energy and die earlier than other nodes. This may cause coverage holes in some areas, or it could partition the network and disrupt data forwarding. To balance the load across all nodes, we propose to scale the probability of a node activating itself by its level of remaining energy. Thus, a node that has been a cluster head before will have a smaller probability of becoming cluster head again. Our simulation results (Section 7) show that this simple extension balances the load across all nodes and significantly prolongs the network lifetime.

7 EVALUATION

In this section, we rigorously evaluate the proposed k-coverage algorithms as well as various aspects of the proposed wireless sensor network for forest fire detection. We start by describing our experimental setup in the following subsection. Then, in Section 7.2, we show that the centralized RKC algorithm ensures that the requested coverage degrees are satisfied, produces near-optimal results, and runs much faster than other centralized algorithms in the literature. In Sections 7.3 and 7.4, we analyze the performance of the decentralized DRKC 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 other distributed k-coverage algorithms in the literature. Then, in Section 7.5, we demonstrate that the decentralized DRKC algorithm can achieve unequal monitoring of different zones in the forest, and in Section 7.6, we show that DRKC balances the load across all sensors, is robust to node failures, and prolongs the network lifetime. Finally, in Section 7.7, we validate the relationship between the coverage degree and the accuracy in estimating FFMC and FWI.

7.1 Algorithms implemented and experimental setup

We compare our work to the works in [57] and [52]. Thus, we implemented six algorithms in total: three centralized and three distributed. We had to implement the algorithms in our own packet-level simulator, because simulators like NS-2 did not support the scale of our experiments (thousands of nodes), which is important to evaluate the k-coverage algorithms that are designed for large-scale sensor networks.

We implemented our RKC and DRKC algorithms in Figures 11 and 12. The centralized algorithm in [57] 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 [57], 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 [57] and obtained the same results. The centralized algorithm in [52], 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 [37] to solve the linear program. LPSOLVE is an open source tool for solving mixed integer linear programming problems. The distributed algorithm in [52], called, PKA, uses a similar pruning idea as DPA. We implemented PKA and validated our implementation by comparing the results with those in [52].

To conduct fair comparisons, we use the same experimental setup as in [57]: We deploy 5,000 sensors in an area of 40 m $\times$ 40 m. We vary the 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 realistic energy model in [53] and [56], 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. 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.

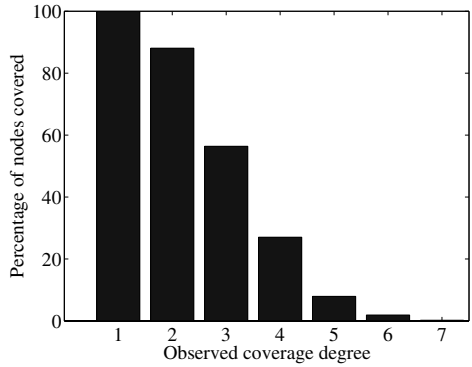
Unless otherwise specified, the above parameters are used, and each experiment is repeated 10 times with different seeds and the average over all of them is reported.

7.2 Evaluation of the centralized k-coverage algorithm (RKC)

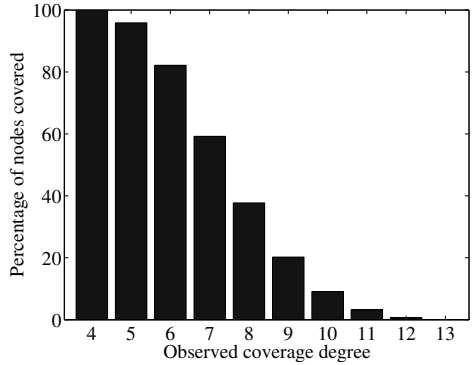
The first set of experiments studies the correctness of our centralized algorithm. That is, whether it indeed achieves the desired degree of coverage. We vary the requested coverage degree k between 1 and 8 and observe the achieved coverage at every single point in the area. Some of the results are shown in Figure 15, where the x-axis shows the observed coverage degree and the y-axis shows the percentage of points which achieve that degree. The figure shows that the RKC algorithm achieves the goal: 100% of the points are sufficiently covered. Moreover, 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: CKC [57] and LPA [52]. As shown in Figure 16(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 Figure 16(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.

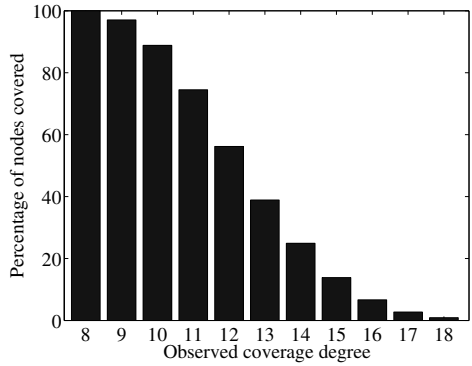
In the above experiment we could not use large network sizes in the comparison, because CKC took very long time (days) in some cases and it did not even terminate in others. Our algorithm is designed for large-scale sensor networks, thus we need to study its efficiency in these networks. By efficiency 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 [34] for uniformly deployed sensors. These conditions are obtained for 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 [34] in their simulations. We use a large area of size 1,000 m $\times$ 1,000 m with 30,000 nodes and vary the sensing range r. The results for $k = 4$ are shown in Figure 17, where Nec_cond and Suff_cond



(a) $k = 1$

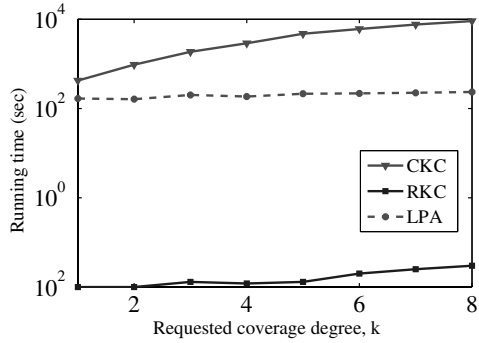


(b) $k = 4$

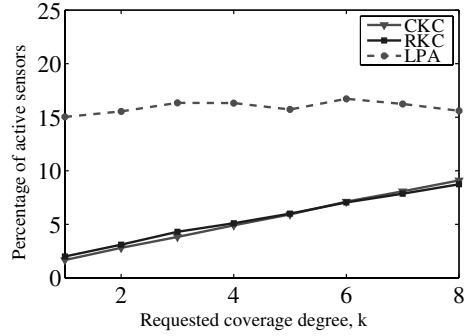


(c) $k = 8$

FIGURE 15
Correctness of the RKC algorithm. The figure shows the achieved coverage distribution for various requested coverage degrees.



(a)



(b)

FIGURE 16
Comparing the performance of our centralized k-coverage algorithm (RKC) versus two others: (a) Running time, and (b) Percentage of active sensors.

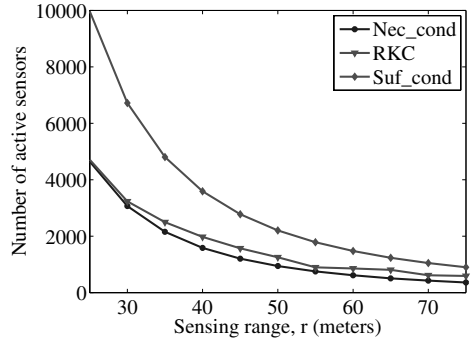


FIGURE 17
Efficiency of our centralized k-coverage algorithm (RKC). The figure compares the number of active sensors produced by our RKC algorithm versus the necessary (Nec_cond) and sufficient (Suf_cond) conditions proved in [34].

denote the necessary and sufficient conditions, respectively. The figure shows that the output of our algorithm is very close to the necessary condition. The results of this experiment confirm our analysis in Theorem 1, and that our centralized algorithm produces near-optimal number of active sensors.

7.3 Evaluation of the distributed k-coverage algorithm (DRKC)

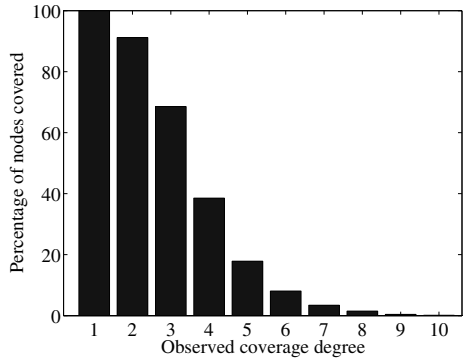
We start by verifying that the DRKC algorithm achieves the requested coverage degree. As in the case of the centralized algorithm, we vary the requested coverage degree k between 1 and 8 and observe the achieved coverage at every point in the area. The results for $k = 1, 4$ and 8 are shown in Figure 18. 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 Figure 19(a), we vary the requested coverage degree k and fix all other parameters for both the centralized and distributed algorithms. And in Figure 19(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.

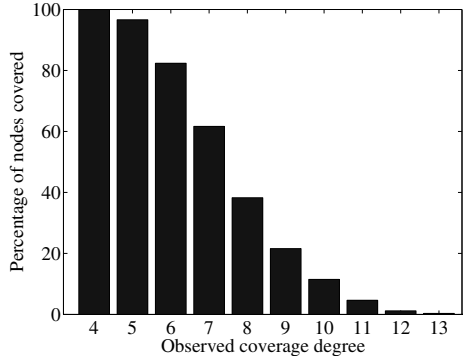
In Section 6, we assumed that sensors start a round of the DRKC algorithm at the same time, i.e., they are time-synchronized. In this experiment, we examine the effect of clock drift on the performance of DRKC. By clock drift we mean that sensors may start a round at different points in time. We add 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. Figure 20 summarizes the impact of clock drift on the performance on the DRKC algorithm. Figure 20(a) indicates that only a minor variation in the average number of messages exchanged between sensors may result from large clock drifts. In addition, as shown in 20(b), the convergence time of the DRKC algorithm did not change much. The 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. 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.

7.4 Comparing DRKC versus other distributed algorithms

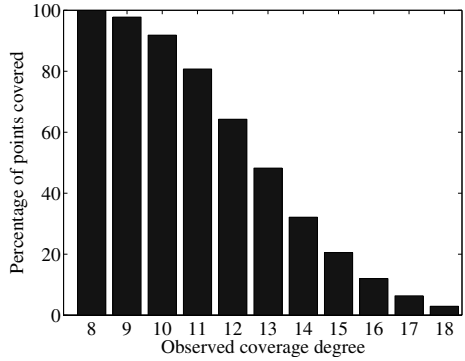
We compare our distributed DRKC algorithm against two other distributed k-coverage algorithms: DPA [57] and PKA [52] along various performance metrics. We first vary the coverage degree k and compute the number of sensors activated by each algorithm to achieve the requested coverage degree. We



(a)

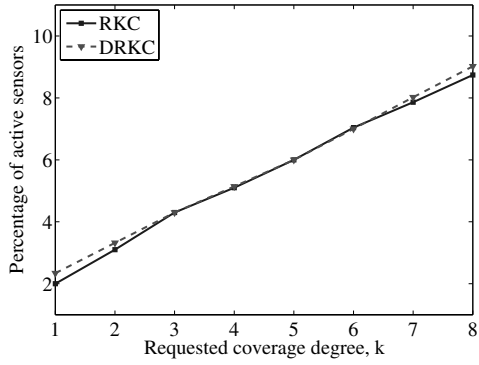


(b)

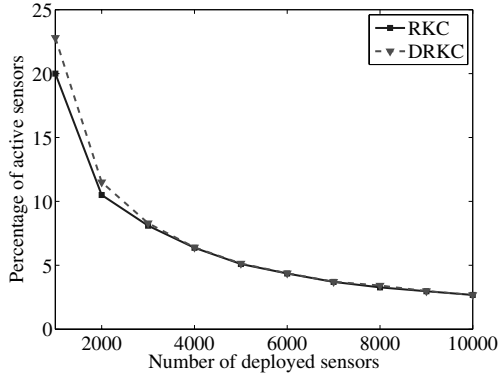


(c)

FIGURE 18
Correctness of our distributed k-coverage algorithm (DRKC). The figure shows the achieved coverage distribution for various requested coverage degrees: (a) $k = 1$, (b) $k = 4$, and (c) $k = 8$.



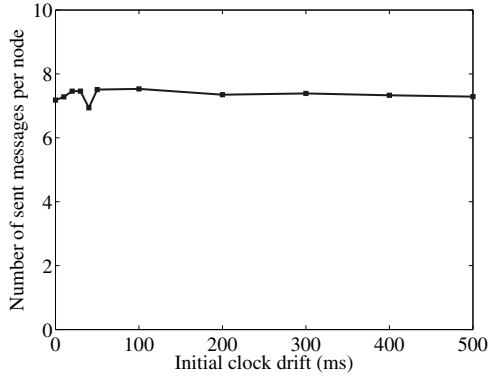
(a)



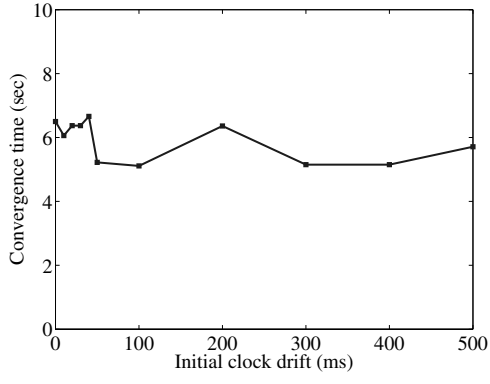
(b)

FIGURE 19
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.

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 Figure 21. Figure 21(a) indicates that the DRKC algorithm converges much faster than the other two algorithms: It converges in less than 8 second compared to 25 seconds for the others. Short convergence times are desirable because the network reaches steady state faster. This reduces the energy consumed by sensors as we will show later in this section. Figure 21(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%).



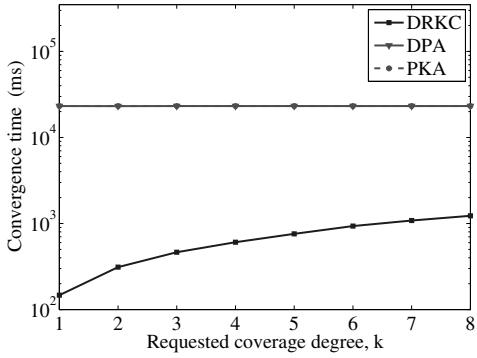
(a)



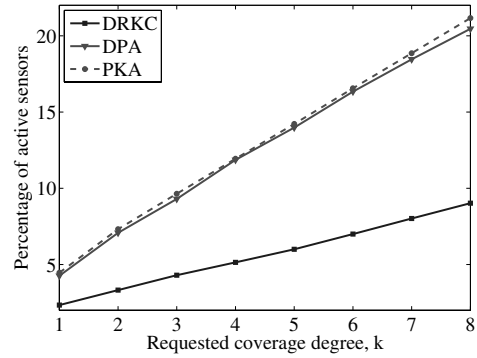
(b)

FIGURE 20
The impact of clock drift on the performance of our distributed k-coverage algorithm on: (a) average number of messages sent per node, and (b) convergence time.

In the next set of experiments, we compare the energy consumption of the DRKC, DPA, and PKA distributed k-coverage algorithms. In Figure 22(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 Figure 21, 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 Figure 22(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 Figure 22(b), the amount of energy consumed per node is much lower (about one-fifth) in DRKC than the



(a)

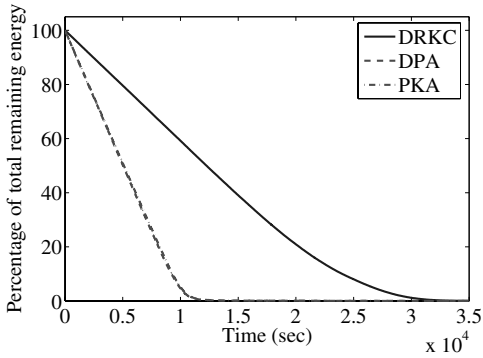


(b)

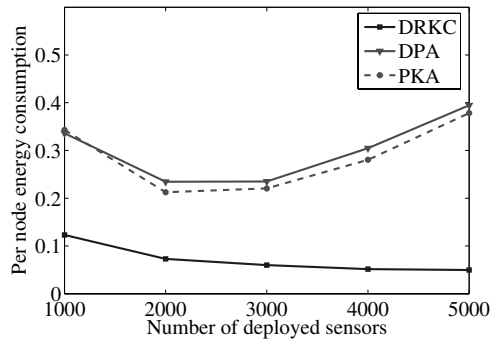
FIGURE 21
Comparing the performance of our DRKC algorithm versus two other distributed k-coverage algorithms for various coverage degrees: (a) convergence time, and (b) percentage of active of sensors.

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, increase chances of collision, and ultimately consume more energy.

In the last experiment, we look at the lifetime of the sensor network under different distributed algorithms. Figure 23 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



(a)



(b)

FIGURE 22
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.

network, because it consumes much smaller amount of energy than the other two algorithms, as shown in the previous experiment.

7.5 Unequal monitoring using different coverage degrees

In this experiment, we validate that our distributed k-coverage algorithm can maintain coverage with various degrees to achieve unequal monitoring of different zones in the forest. We assume there are two hot spots inside the forest that need higher coverage degrees than other areas, as shown in Figure 14. The two spots are modeled as two polygons. The requested coverage degree in one spot is 8 and in the other is 4. Nodes outside the hot spots are requested to have a coverage degree of 1. We run our algorithm and notify nodes inside the hot spots of the different coverage degrees. We let the algorithm converge, and we check the coverage degree of every single point in the area. We plot the

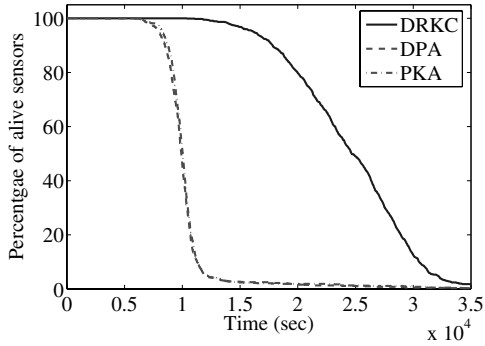


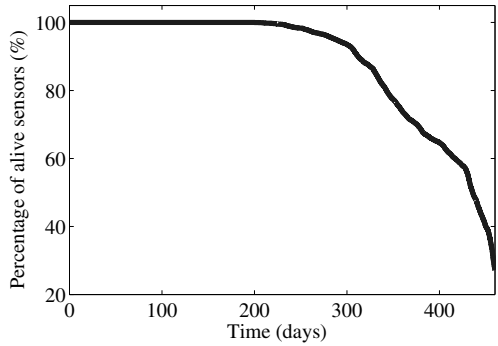
FIGURE 23
Comparing the network lifetime under different distributed k-coverage algorithms.

achieved coverage distribution in each area in Figure 28. The results indicate that in each of the hot spots, our algorithm indeed achieves the requested coverage degree while it provides 1-coverage in the rest of the area. Figure 28 also shows that our algorithm does not over cover areas, because the fraction of nodes having higher-than-requested coverage degrees decreases fast. This is important to save energy and prolong network lifetime.

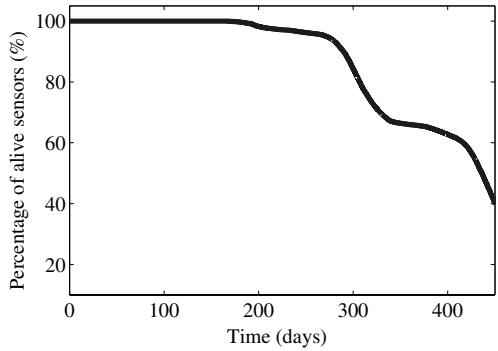
7.6 Load balancing, node failures, and network lifetime

We study the average load on individual nodes and on the network lifetime under our k-coverage algorithm. We measure the load on a node by the energy consumed by that node. Once a node runs out of energy, it is assumed to be failed or dead. We run our algorithm till all nodes are dead. After each round of the algorithm, we count the number of alive nodes. We plot the percentage of alive nodes versus time. We repeat the whole experiment for various coverage degrees, from $k = 1$ to 8. Sample of the results are shown in Figure 24. As the figure shows, most of the nodes stay alive for a long period (more than 200 days). Then, they gradually die. This means that the algorithm did not over utilize some nodes in early rounds, otherwise, they would have died earlier. Notice that the energy of a node is enough for it to be active in a few days, and if a node were chosen as a cluster head for several times, it will probably survive for only a few hours. These results confirm that our algorithm distributes the load uniformly across all deployed nodes. This is critical in order to keep nodes alive for the longest possible period and achieve more reliable coverage. This also extends the network lifetime as shown by our next experiment.

Next, we analyze the sensor network lifetime under our k-coverage algorithm as sensors dynamically fail when they run out of energy. We define the network lifetime as the time till the coverage drops below 100%, i.e., there are some points in the area that have coverage less than k. Analyzing the network lifetime is critical in a forest fire detection system, because the sensor network



(a) $k = 1$

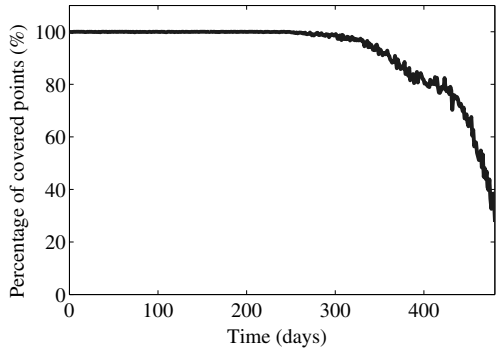


(b) $k = 8$

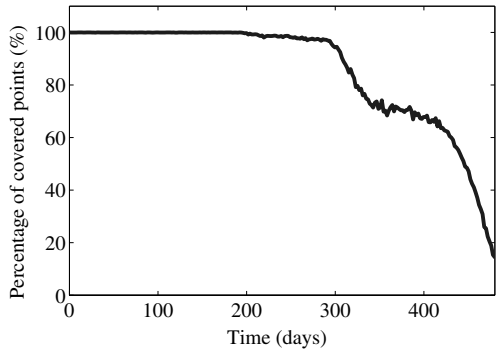
FIGURE 24
Our k-coverage algorithm balances load across all nodes, since most of them stay alive for long periods.

should last for at least one fire season. We use the same setup as in the previous experiment, except we measure coverage not the number of alive nodes. We run the simulation for a long time and periodically check the coverage degree for every single point in the area. A point is considered covered if its coverage degree is at least k. We vary k between 1 and 8 and plot some of the results in Figure 25. The figure shows that 100% coverage of the area is maintained through a long period of time, more than 200 days. This is because our algorithm uniformly distributes load on nodes.

Figure 25 also shows that coverage decreases at a slower rate than the number of alive nodes in Figure 24. For example, in Figure 24(a), the number of alive nodes starts to drop below 100% around day 200, while 100% coverage is maintained till almost day 300 as shown in Figure 25(a). This demonstrates



(a) $k = 1$



(b) $k = 8$

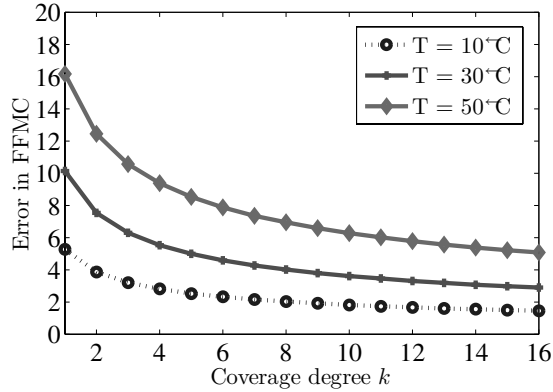
FIGURE 25
Our k-coverage algorithm prolongs network life, because 100% of the area is k-covered is over long periods.

the robustness of our algorithm against node failures. In addition, the results in Figure 25 imply that alive nodes are not grouped in certain subareas, rather, they are uniformly distributed in the whole area. Therefore, our k-coverage algorithm prolongs the network lifetime because it uniformly balances the load across all nodes and it keeps alive nodes distributed throughout the whole area.

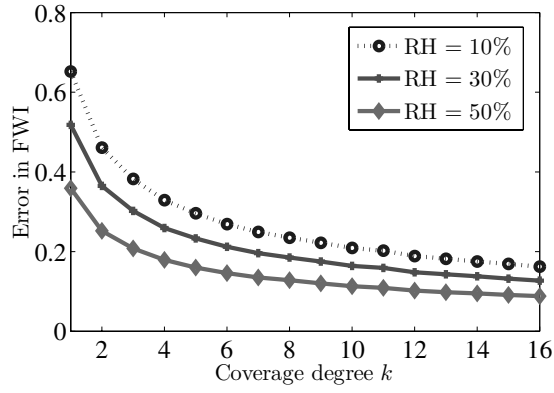
7.7 Accuracy of FFMC and FWI

In Section 4.1, we established a relationship between the coverage degree and the accuracy in estimating FFMC and FWI. We numerically analyze this relationship. We vary the coverage degree k between 1 and 16. We assume that the accuracy of the temperature sensing board is 4°C, i.e., $\sigma_T = 2$. All calculations are done for a confidence level of 95%. For each value of k, we

compute the error in estimating the temperature δ_T. Then, we use the software program that computes the FFMC and FWI indexes [49] to determine the maximum error in these indexes, given a $\pm\delta_T$ error in the temperature T. We repeat the experiment for several values of the temperature and humidity. Some of the results are given in Figure 26. First, as predicted by the analysis in Section 4.1, the figure shows that higher coverage degrees result in smaller errors in FFMC and FWI. Second, the figure exposes an important issue: the error in FFMC and FWI is amplified in extreme conditions (high temperatures and low humidity), which is due to the non-linearity of the complex equations that determine FFMC and FWI. For example, Figure 26(b) indicates that an error up to 2 units in FFMC could result when $k = 2$ and the temperature



(a)



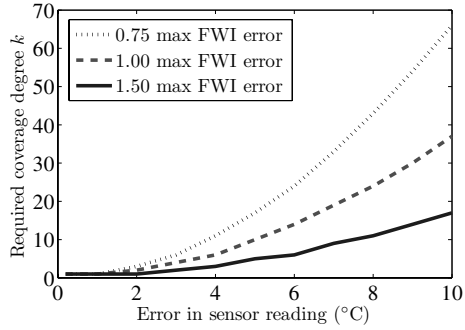
(b)

FIGURE 26
Error in calculating: (a) FFMC Code, (b) FWI index for various coverage degrees and in different weather conditions.

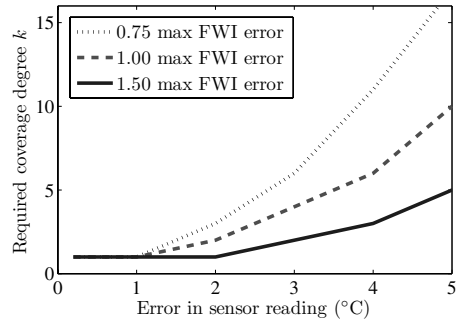
is 10°C, while this error could be as high as 12 units if the temperature is 50°C with the same k value. This means that in extreme conditions, which are the most important for the forest fire detection system, even small errors in sensing the temperatures could lead to significant errors in FFMC and FWI, which may lead the sensor network operators to take incorrect actions. This also highlights the importance of unequal monitoring of forest zones: host spots of the forest need to be covered with higher degrees to provide accurate assessment of the potential and intensity of fires. Furthermore, the results in Figure 26 can be used to *dynamically* configure the sensor network such that higher coverage degrees are enforced as the weather conditions get more severe. Dynamic configuration of the sensor network (or parts of it) is easily achieved by our k-coverage algorithm because of its distributed nature, this is demonstrated in the next subsection.

In the previous experiment, the error in sensor reading is fixed. In our next experiment, we analyze the tradeoff between the accuracy of the sensing boards and the required coverage degree such that a given maximum error in FFMC and FWI is not exceeded. Since forest fire detection is an important application for sensor networks, sensor manufacturers may customize or even create new products explicitly for this application. In this case, understanding the needed accuracy of the sensing module could result in significant savings especially for mass production of sensors.

We consider a wide range of accuracy for sensing boards; the results presented here are for temperature sensors, but the analysis can be carried out for other weather conditions as well. As mentioned in Section 4.1, the error in sensor reading is specified as $2\sigma_T$. We vary the error in sensor reading from 0.25°C to 10°C, which captures the the range of accuracy achieved by very accurate and expensive sensors to rough and cheap sensors. For each value of the error reading, we compute the required coverage degree k to meet the given error in FWI and FFMC using their equations. We repeat for a few target errors in FWI and FFMC. We plot the results for the FWI index in Figure 27(a) shows the results for the full error range, while Figure 27(b) zooms in the small error range between 0–5 for illustration. The figure clearly exposes the tradeoff: for larger errors in sensor readings (i.e., cheaper sensors), higher coverage degrees are required to meet the target error in FWI and FFMC. For example, for a maximum error in FWI of 1.0 unit, a coverage degree of 1 is needed when sensors that have temperature error readings up to 1°C are deployed, whereas a coverage degree of 8 would tolerate temperature error readings up to 4.5°C while achieving the same accuracy in FWI. Higher coverage degrees require keeping more sensors active, which means that they will be depleted from energy faster. Therefore, to achieve a target network lifetime, more sensors will need to be deployed for higher coverage degrees. However, with mass production of less-accurate sensors, increasing the degree of coverage could result in more cost-effective sensor networks that achieve the same function.



(a)

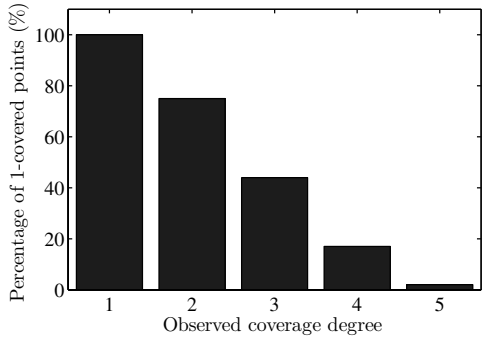


(b)

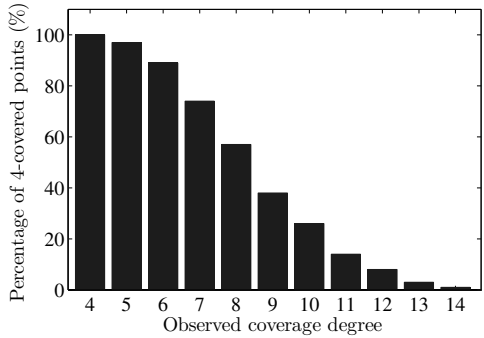
FIGURE 27
The tradeoff between the accuracy in sensor reading and the required coverage degree, given a maximum tolerable error in the FWI index. (a) considers a wide range for sensor accuracy, while (b) zooms in the small range between 0–5.

8 CONCLUSION

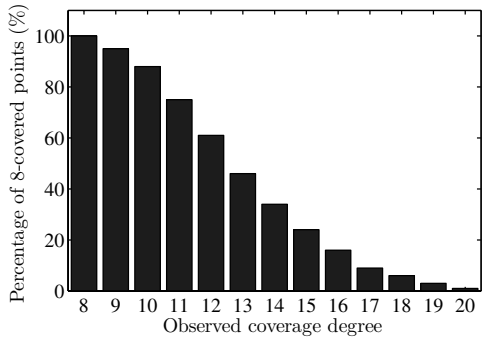
We presented the design of a wireless sensor network for early detection of forest fires. Our design is based on the Fire Weather Index (FWI) System, which is backed by decades of forestry research. The FWI System is comprised of six components: three fuel codes and three fire indexes. The three fuel codes represent the moisture content of the organic soil layers of forest floor, whereas the three fire indexes describe the behavior of fire. By analyzing data collected from forestry research, we showed how the FWI System can be used to meet the two goals of a wireless sensor network designed 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. To achieve these goals, we designed our sensor network based on two main components of the FWI System: the Fine Fuel



(a) Requested $k = 1$



(b) Requested $k = 4$



(c) Requested $k = 8$

FIGURE 28
Coverage with different degrees achieved by our algorithm.

Moisture Code (FFMC), and 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 modeled the forest fire detection problem as a k-coverage problem, with $k \geq 1$. We computed the required coverage degrees to achieve a given accuracy level in estimating different components of the FWI System. We then presented approximation centralized and distributed algorithms to efficiently solve the node k-coverage problem, which is NP-hard. Our distributed algorithm is simple to implement and does not require any specific node deployment schemes. Therefore, nodes can be deployed by, for example, throwing them from an aircraft. This significantly facilitates node deployment in real life. In addition, 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, which is 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 conducted extensive simulation study to validate our theoretical analysis and to compare our coverage algorithms against others in the literature. The comparisons showed that our algorithms outperform other algorithms along several performance metrics, including convergence time, number of sensors activated, and total energy consumption. Furthermore, our simulations show that our distributed algorithm: (i) balances load across all deployed nodes, and therefore maintains reliable coverage and significantly prolongs the network lifetime; and (ii) can provide various coverage degrees at different areas of the forest, and thus can achieve higher detection accuracy in important areas such as near residential or industrial neighborhoods.

ACKNOWLEDGEMENT

This work is partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. We would like to thank the Ministry of Public Works and Government Services of Canada, and the Protection Program of the Ministry of Forests and Range in the Province of British Columbia, Canada for allowing us to reproduce Figures 4 and 13, respectively.

REFERENCES

[1] Z. Abrams, A. Goel and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *Proc. of International Symposium on Information Processing in Sensor Networks (IPSN'04)*, Berkeley, CA, April 2004, pp. 424–432.

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

[3] N. Ahmed, S. Kanhere and S. Jha. Probabilistic coverage in wireless sensor networks. In *Proc. of IEEE Conference on Local Computer Networks (LCN'05)*, Sydney, Australia, November 2005, pp. 672–681.

[4] I.F. Akyildiz, S. Weilian, Y. Sankarasubramaniam and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine* **40**(8) (2002), 102–114.

[5] 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.

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

[7] B.C. Ministry of Forests and Range. Fire Review Summary for Okanagan Mountain Fire (K50628).

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

[9] 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, pp. 2003–2012.

[10] H. Bronnimann and M. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry* **14**(4) (1995), 463–479.

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

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

[13] Q. Cao, T. Yan, T. Abdelzaher and J. Stankovic. Analysis of target detection performance for wireless sensor networks. In *Proc. of International Conference on Distributed Computing in Sensor Networks*, Marina Del Rey, CA, June 2005, pp. 276–292.

[14] M. Cardei and J. Wu. Coverage in wireless sensor networks. In M. Ilyas and I. Mahgoub, editors, *Handbook of Sensor Networks*. CRC Press, 2004.

[15] M. Cardei and J. Wu. Energy-efficient coverage problems in wireless ad hoc sensor networks. *Elsevier Computer Communications* **29**(4) (2006), 413–420.

[16] Z. Chaczko and F. Ahmad. Wireless sensor network based system for fire endangered areas. In *Proc. of Third International Conference on Information Technology and Applications (ICITA'05)*, Sydney, Australia, July 2005.

[17] 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), 1448–1453.

[18] Crossbow Inc. Web Page. http://www.xbow.com/.

[19] F. Dai and J. Wu. An extended localized algorithm for connected dominating sets formation in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, **15**(10) (2004), 908–920.

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

[21] 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, pp. 477–484.

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

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

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

[25] H. Gupta, Z. Zhou, S. Das and Q. Gu. Connected sensor cover: self-organization of sensor networks for efficient query execution. *IEEE/ACM Transactions on Networking* **14**(1) (2006), 55–67.

[26] 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 *Proc. of International Conference on Mobile systems, Applications and Services (MobiSys'06)*, Uppsala, Sweden, June 2006, pp. 28–41.

[27] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete and Computational Geometry* **2**(1) (1987), 127–151.

[28] M. Hefeeda and H. Ahmadi. A probabilistic coverage protocol for wireless sensor networks. In *Proc. of IEEE International Conference on Network Protocols (ICNP'07)*, Beijing, China, October 2007.

[29] M. Hefeeda and M. Bagheri. Randomized k-coverage algorithms for dense sensor networks. In *Proc. of IEEE INFOCOM 2007 Minisymposium*, Anchorage, AK, May 2007, pp. 2376–2380.

[30] M. Hefeeda and M. Bagheri. Wireless sensor networks for early detection of forest fires. In *Proc. of International Workshop on Mobile Ad hoc and Sensor Systems for Global and Homeland Security (MASS-GHS'07), in conjunction with IEEE MASS'07*, Pisa, Italy, October 2007, pp. 2376–2380.

[31] C. Huang, Y. Tseng and H. Wu. Distributed protocols for ensuring both coverage and connectivity of a wireless sensor network. *ACM Transactions on Sensor Networks* **3**(1), 2007.

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

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

[34] 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)*, Philadelphia, PA, September 2004, pp. 144–158.

[35] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: A quantitative comparison. *Elsevier Computer Networks* **43** (2003), 499–518.

[36] B. Liu and D. Towsley. A study on the coverage of large-scale sensor networks. In *Proc. IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, Fort Lauderdale, Florida, October 2004.

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

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

[39] J. Matousek, R. Seidel and E. Welzl. How to net a lot with little: Small ϵ-nets for disks and halfspaces. In *Proc. of 6th Annual ACM Symposium on Computational Geometry (SoCG'90)*, Berkeley, CA, June 1990, pp. 16–22.

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

[41] Network Systems Lab Web Page. http://nsl.cs.sfu.ca/wiki/.

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

[43] 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.

[44] S. Shakkottai, R. Srikant and N. Shroff. Unreliable sensor grids: Coverage, connectivity, and diameter. *Ad Hoc Networks* **3**(6) (2005), 702–716.

[45] 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, pp. 584–593.

[46] 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) (2006), 124–130.

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

[48] D. Tian and N. Georganas. Location and calculation-free node-scheduling schemes in large wireless sensor networks. *Elsevier Ad Hoc Networks* **2** (2004), 65–85.

[49] 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.

[50] T. Wu and K. Ssu. Determining active sensor nodes for complete coverage without location information. *International Journal of Ad Hoc and Ubiquitous Computing* **1**(1–2) (2005), 38–46.

[51] 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) (2005), 36–72.

[52] S. Yang, F. Dai, M. Cardei and J. Wu. On connected multiple point coverage in wireless sensor networks. *International Journal of Wireless Information Networks* **13**(4) (2006), 289–301.

[53] F. Ye, G. Zhong, J. Cheng, S. Lu and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS'03)*, Providence, RI, May 2003, pp. 28–37.

[54] O. Younis, M. Krunz and S. Ramasubramanian. Node clustering in wireless sensor networks: Recent developments and deployment challenges. *IEEE Network* **20**(3) (2006), 20–25.

[55] 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, pp. 1214–1217.

[56] H. Zhang and J.C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc and Sensor Wireless Networks* **1**(1–2) (2005), 89–123.

[57] 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, pp. 373–378.

[58] Y. Zou and K. Chakrabarty. A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks. *IEEE Transactions on Computers* **54**(8) (2005), 978–991.