Generic Coverage Verification without Location Information Using Dimension Reduction

Gaurav S. Kasbekar, Yigal Bejerano and Saswati Sarkar

***Abstract*—Wireless Sensor Networks (WSNs) have recently emerged as a key sensing technology with diverse civilian and military applications. In these networks, a large number of small sensors or nodes perform distributed sensing of a target field. Each node is capable of sensing events of interest within its sensing range and communicating with neighboring nodes. The target field is said to be k-covered if every point in it is within the sensing range of at least k sensors, where k is any positive integer. We present a comprehensive framework for verifying k-coverage of a d-dimensional target field for arbitrary positive integers k, d. Our framework uses a divide and conquer approach based on the technique of dimension reduction, in which the k-coverage verification problem in d-dimensions is reduced to a number of coverage verification problems in (d-1) dimensions, which are then recursively solved. Our framework leads to a distributed polynomial-time coverage verification algorithm that does not require knowledge of the locations of nodes or directional information, which is difficult to obtain in WSNs. Each node can execute the algorithm using only the distances between adjacent nodes within its transmission range and their sensing radii. We analytically prove that the scheme detects a coverage hole if and only if the target field has a coverage hole.**

I. Introduction

Recent advances in wireless communications and electronics have enabled the development of *low-cost sensor nodes* [1]. Each sensor node is capable of sensing specific events in its vicinity and of communicating with adjacent nodes. Thus, for event sensing applications, a large number of sensor nodes are deployed in a *target field* and they collaborate to form an ad-hoc network, referred to as a *wireless sensor network* (WSN). WSNs have the potential to become the dominant sensing technology in many civilian and military applications, such as intrusion detection, environmental monitoring, object tracking, traffic control, and inventory management. In many of these applications, WSNs need to monitor the target field for detecting events of interest, e.g., entrance of an intruder in an intrusion detection application.

Coverage of the target field is essential for reliable detection of events of interest, and the quality of the coverage is considered a measure of the *quality of Service* (QoS) delivered by a WSN [2]. However, sensor nodes are prone to failures that may cause coverage holes in the target field, which in turn adversely affects event detection capabilities of the WSN. Thus, sensor nodes must execute simple and efficient coverage holes detection mechanisms for ensuring network reliability and providing the required QoS.

Coverage verification in sensor networks has received considerable attention in the last few years. Nevertheless, researchers have mostly focused on WSNs where the sensors are deployed in a straight line or a 2-dimensional plane (refer to Section II for a review of related work). Several important applications however require that the sensors be deployed in a 3-dimensional space. For example, in 3D underwater acoustic sensor networks [24], sensors are suspended at different depths in the water, which allows observation of phenomena that cannot be adequately observed using a 2D sensor network deployed on the ocean bottom. Also, sensors need to be deployed in a 3-dimensional space in the atmosphere for weather forecasting and climate monitoring [23]. Most of the coverage verification schemes developed for 3-dimensional sensor networks assume that the precise locations (i.e., coordinates) of the sensors are known, which cannot be guaranteed for WSNs (Section II).

In this paper, we consider a WSN where the sensors are deployed in a d-dimensional space, and focus on detecting *coverage holes*, which are regions in the target field that are covered by $k-1$ or fewer sensors. Here, d, k can be any positive integer. We describe the system model in Section III and the detection problem in Section IV. We provide a generic coverage verification algorithm that detects a coverage hole if and only if one such is present. Our algorithm only requires that each sensor knows its distances from its neighbors and the distances between its neighbors that are also neighbors of each other, and does not need any information on their locations otherwise. The algorithm is distributed and requires only simple computations.

Our coverage verification algorithm uses a *divide and conquer* approach based on a *dimension reduction* mechanism. We first show that the coverage verification problem in d-dimension can be solved by reducing it to a number of coverage verification problems in $d-1$ dimensions (when $d > 1$), and can therefore be recursively reduced to a number of coverage verification problems in 1 dimension (Section V). This *dimension reduction* is based on a projection process; we provide the details for this projection process for $d = 3$ and $d = 2$ in Section VI. Finally, we show that it is straightforward to verify coverage when all sensors are in 1-dimension, that is on a straight line (Section VII).

II. Related Work

Comprehensive surveys for coverage verification in WSNs can be found in [3], [4]. In this section we compare our results with existing results that are closely related. A genre of papers focus on sensor deployment, topology control, motion control and routing for satisfying several quality of service requirements such as maximizing coverage, maximizing network lifetime, minimizing the number of sensors deployed,

etc. [5], [17], [6], [23], [22]. Specifically, Choudhury *et al.* [17] present an interesting topology control heuristic that activates a subset of sensors so as to maintain coverage and connectivity. This algorithm however does not guarantee coverage in every circumstance [17]. We focus on verifying coverage with guarantees on detection performance when sensors are deployed in a d-dimensional space.

Coverage verification has primarily focused on WSNs where sensors have been deployed on a 2-dimensional plane [2], [5], [6], [8], [9], [10], [20]. Most of these papers assume knowledge of either the locations of the sensors [2], [6], [8], [9], [10] or the angles between neighboring sensors [5]. The coverage verification scheme presented by Bejerano [20] however attains proven detection guarantees without using location information. But, this scheme cannot be readily generalized for WSNs with sensor deployment in an arbitrary d-dimensional space.

The coverage verification problem for sensors deployed in a 3-dimensional space has been addressed in [21], [22], [7], [13], [14]. The algorithms proposed by Huang *et. al.* [21] and So *et. al.* [7] require precise knowledge of sensor locations, which cannot be easily obtained for WSNs [11], [12]. We now consider the existing coverage verification schemes for 3-dimensional deployment that are oblivious to the nodes' locations, referred to as *coordinate-free* solutions. Ghrist *et. al.* [13] describe an innovative holes detection scheme based on homology, which is however a centralized solution that cannot be easily implemented in WSNs. Li *et al.* [14] introduced distributed schemes for detecting large holes. To the best of our knowledge, our scheme is the first distributed, coordinate-free solution that is guaranteed to detect holes of any size for sensors deployed in an arbitrary d-dimensional space. Our scheme does not use directional information and relies only on communication with neighbors.

III. The Network Model

We consider a *wireless sensor network* (WSN) consisting of a set V of sensors that are also called *nodes*. The sensors are distributed over a large d-dimensional *target field*, where $d \geq 1$. In practical WSNs, sensors are either deployed on a plane ($d = 2$) or in three-dimensional space ($d = 3$). However, our framework is applicable to an arbitrary positive integer d; hence we do not assume any particular value for d.

Each node u can sense events of interest in its *sensing range* and communicate with nodes in its *transmission range*. We assume that the sensing and transmission ranges of a node u are *open* d-dimensional balls[1] centered at u, with radiuses r_u and R_u, respectively, where $R_u > r_u$. Let $\hat{r} = \max_{u \in V} r_u$, and $\hat{R} = \min_{u \in V} R_u$.

We refer to the boundary of a d-dimensional ball as a *d-boundary-sphere*. For example, a 3-boundary-sphere is a sphere in the usual sense (boundary of a 3-dimensional ball), a 2-boundary-sphere is a circle (boundary of a 2-dimensional ball *i.e.,* a disc) and a 1-boundary-sphere is a pair of points (boundary of a 1-dimensional ball *i.e.,* a line segment). The boundary of the sensing range of any node u is a d-boundary-sphere, which we refer to as the *sensing border* of node u.

Let $d_{u,v}$ denote the Euclidean distance between nodes u and v. Nodes u and v are termed *adjacent* or *neighbor* if they are included in the transmission range of each other. Let N_u be the set of neighbors of u. For simplicity, we assume that there are no two sensors at the same location and each sensor has a unique identification number.

We refer to the set of points of the target field which are at a distance of at least $\hat{r}$ from any boundary point of the target field as the *internal space* of the target field. We distinguish between *internal nodes*, which lie within the internal space versus the other nodes, referred to as *periphery nodes*. We assume that only internal nodes need to verify their coverage. The sensors are not aware of their locations. Yet, every sensor knows if it is a periphery or an internal node, possibly using the mechanisms in [16], [15]. We assume that nodes only have *localized distance information*. Specifically, each node u knows (a) r_u, (b) $d_{u,v}$ and r_v for each $v \in N_u$ and (c) $d_{v,w}$ for each pair $w, v \in N_u$. Thus, we assume that each node can estimate its sensing range, and its distances from its neighbors without learning their orientations, and communicates this information to its neighbors. This is a realistic assumption as recent studies [18], [19] have introduced accurate distance estimation techniques that are applicable for wireless sensors.

We say that a point in the target field is *k-covered* if it is in the interior of the sensing ranges of at least k nodes. Similarly, any set of points in the target field is considered *k-covered* if every point in the set is k-covered. In particular, we say that a node u's sensing border is *k-covered* if every point on it is k-covered (*by nodes other than node u*). Note that since the sensing range of a node is an *open* ball, no point on u's sensing border is covered by u itself. We define a *k-coverage hole*, or simply coverage-hole, as a continuous area of the target field comprised of points that are not k-covered. For instance, if $k = 1$ then every point of the coverage hole is not monitored by any sensor.

Finally, no coordinate-free coverage verification scheme can guarantee the detection of every k-coverage hole, if $\hat{R} < 2 \cdot \hat{r}$ [20]. Thus, we assume that $\hat{R} \geq 2 \cdot \hat{r}$.

A summary of the paper main notation is given in Table I.

IV. The Coverage Holes Detection Problem

Our objective is to verify that a given d-dimensional target field does not contain any k-coverage hole for arbitrary $k \geq 1, d \geq 1$, using only localized distance information. We now present a proposition which has been proved by Huang *et al.* in [8], [21] and which we use in our solution.

Proposition 1: Assume that no two nodes are located in the same location. Then, the internal space of the target field is k-covered if and only if the sensing border of every internal node $u \in V$ is k-covered.

[1]The sensing and the transmission of a sensor may have various geometric shapes, however, we assume that each one of these ranges subsume a ball with radius r_u or R_u, which are the commented sensing and transmission ranges, respectively.

Symbol	Semantics
V	The set of sensors in the given WSN.
u	The node that executes our scheme.
d	The dimension of the target field.
k	The coverage requirement.
$d_{u,v}$	The Euclidian distance between nodes u and v.
N_u	The neighbors of node u.
R_u	The *transmission radius* of node u.
r_u	The *sensing radius* of node u.
$\hat{R}$	A lower bound on a node trans. radius, ($R_u \geq R$).
$\hat{r}$	An upper bound on a node sensing radius, ($r_u \leq r$).
AB	The distance between two points A and B
$[A,B)$	The ray originating at A and passing through B
$\angle ABC$	Angle between the rays $[B,A)$ and $[B,C)$ and its size
N_u^s	The set of nodes whose sensing range subsumes u's sensing border
k_u	$k - \lvert N_u^s \rvert$
$C_{u,v}$	The set of intersection points of the sensing borders of nodes u and v
$(uv)'$	The virtual sensor at the center of $C_{u,v}$
t'	The virtual sensor that is the projection of sensor t

TABLE I
GENERAL NOTATION.

Recall that Proposition 1 does not apply to points of the target field near its edge since the sensing borders of periphery nodes may not be k-covered. Hence, our solution guarantees accurate k-coverage-verification only for the internal space of the target field. Thus, our objective can be formulated as follows.

Problem Definition: *(The d-dimensional k-Coverage Verification Problem)*: The d-dimensional k-coverage verification problem is a decision problem whose goal is to determine whether the sensing border of every internal node u is k-covered, by using only localized distance information.

We distinguish between *detecting the presence* of coverage holes and *finding their exact locations*. Since the sensors are oblivious to their locations, they cannot report about the exact location of a coverage hole when they detect one. We assume that once a coverage hole has been detected, other means are applied for inferring the hole location. For instance, backtracking the paths of the coverage-hole report messages or by using a coarse positioning system that provides a rough location estimations of the nodes. We do not investigate the mechanisms for inferring the hole locations.

V. THE COVERAGE VERIFICATION SCHEME

In this section we present a distributed scheme for solving the d-dimensional k-coverage problem where k and d are arbitrary. Each sensor $u \in V$ independently verifies that its vicinity is k-covered by using the sensing border concept presented above. Whenever a node determines that its sensing border is not k-covered, it reports the presence of a hole. If the sensing borders of all nodes are k-covered, and therefore the presence of hole is not reported, clearly there is no hole (Proposition 1). In our divide-and-conquer approach, each sensor verifies that its sensing border is k-covered by dividing the problem into simpler instances, in which the coverage requirement k is reduced or the problem dimension is reduced to $d-1$. In section VII, we propose a strategy for verifying coverage in 1-dimension which concludes the algorithm. In the following we elaborate on the algorithm executed by each individual sensor u.

A. Subsumption and Intersection

We now present two properties that can be easily used to confirm or rule out k-coverage of the sensing border of a sensor in some special cases. It is easy to check the correctness of these properties.

*Property 1 (***Subsumption):** The sensing border of sensor u is entirely subsumed in the sensing range of sensor w if and only if $d_{u,w} + r_u \leq r_w$.

By using Property 1, a sensor u can easily verify if its sensing border is entirely subsumed in the sensing range of another sensor w. Suppose w's sensing border subsumes u's sensing border. Then, since every point on u's sensing border is covered by sensor w, we can check $(k-1)$-*coverage* of u's sensing border *by sensors other than sensor* w. Let $N_u^s \subseteq N_u$ be the set of sensors such that u's sensing border is entirely subsumed in the sensing range of each sensor in set N_u^s. N_u^s is found by using property 1. If $|N_u^s| \geq k$, then u's sensing border is k-covered. Now, let $|N_u^s| < k$, and $k_u = k - |N_u^s|$. Clearly, in this case, u's sensing border is k-covered if and only if it is k_u-covered by sensors in the set $N_u \backslash N_u^s$. To check whether the above condition holds, we need to detect intersecting spheres.

*Property 2 (***Intersection):** The sensing border of sensor $v \in N_u \backslash N_u^s$ intersects u's sensing border (but is not tangent to it) if and only if $d_{u,v} < r_u + r_v$ and $d_{u,v} + r_v > r_u$.

The first condition in Property 2 states that there is overlap between balls u and v and the second condition states that sphere v is not subsumed in the interior of sphere u. Now, if the sensing border of no sensor in $N_u \backslash N_u^s$ intersects sphere u, then u's sensing border is not k_u-covered by sensors in the set $N_u \backslash N_u^s$. This condition can be verified using property 2.

Hence, in the next subsection, we assume that $|N_u^s| < k$ and that the sensing border of at least one sensor in $N_u \backslash N_u^s$ intersects u's sensing border, and focus on checking k_u-coverage of u's sensing border by sensors in the set $N_u \backslash N_u^s$.

B. Coverage verification through dimension reduction

A key step in our divide and conquer scheme is mapping a given k-coverage verification instance in d-dimensions into a number of k_u-coverage problems in $d-1$ dimensions.

We first show that when the sensing borders of u and v intersect, the intersection constitutes a $(d-1)$-boundary-sphere, which we call $C_{u,v}$. To show this, suppose node u lies at the origin and node v lies at the point with x_1 coordinate equal to $d_{u,v}$ and all other coordinates equal to 0. Then the equations of the sensing borders of u and v are given by:

$$x_1^2 + x_2^2 + \ldots + x_d^2 = r_u^2 \tag{1}$$

$$(x_1 - d_{u,v})^2 + x_2^2 + \ldots + x_d^2 = r_v^2 \tag{2}$$

Subtracting (1) from (2) and rearranging, we get:

$$x_1 = \frac{d_{u,v}^2 + r_u^2 - r_v^2}{2d_{u,v}} \tag{3}$$

Substituting this value of x_1 from (3) into (1), we get:

$$x_2^2 + \ldots + x_d^2 = r_u^2 - \left(\frac{d_{u,v}^2 + r_u^2 - r_v^2}{2d_{u,v}}\right)^2 \tag{4}$$

$C_{u,v}$ is the set of points that satisfy (3) and (4). This shows that $C_{u,v}$ is a $(d-1)$-boundary-sphere, with radius equal to the square root of the expression on the right-hand side in (4).

We develop a divide and conquer approach using the following proposition.

Proposition 2: Suppose $|N_u^s| < k$ and that the sensing border of at least one sensor in $N_u \backslash N_u^s$ intersects u's sensing border. u's sensing border is k_u-covered by sensors in the set $N_u \backslash N_u^s$ if and only if for every sensor v such that the sensing borders of u and v intersect, $C_{u,v}$ is k_u-covered by nodes in $N_u \backslash (N_u^s \cup v)$.

Proof: The necessity follows from the fact that $C_{u,v}$ lies on u's sensing border. Let us prove sufficiency. Suppose every $(d-1)$-boundary-sphere $C_{u,v}$ is k_u-covered by nodes in $N_u \backslash (N_u^s \cup v)$. Assume, to reach a contradiction, that u's sensing border is not k_u-covered. Thus, there is an area on u's sensing border that is not k_u-covered. Consider a point p on the boundary of a coverage hole. Such a point must exist since all the $(d-1)$-boundary-spheres $C_{u,v}$ are k_u-covered and hence some parts of u's sensing border are k_u-covered. Recall that we define the sensing range of a sensor as an *open* ball. From this, it follows that the boundary points of a coverage hole are not k_u-covered. So p is not k_u-covered.

Now, for any ϵ, however small, there is a point on u's sensing border within distance ϵ from p that is k_u-covered. This implies that p must be on the boundary of the sensing range of some sensor v that intersects with u's sensing border.

Thus, p is located on $(d-1)$-boundary-sphere $C_{u,v}$, which contradicts the assumption that every $(d-1)$-boundary-sphere $C_{u,.}$ is k_u-covered. ■

Note that results similar to Proposition 2 have been proved in [21] and [22]. However, these papers use location information for checking coverage. Our innovation is to show how this proposition can be used to develop a coverage verification algorithm that does not *use location information*.

In the light of Proposition 2, we need to check whether $C_{u,v}$ is k_u-covered. To do this, we first project all sensors in the set $N_u \backslash (N_u^s \cup v)$ onto the $(d-1)$-dimensional space in which $C_{u,v}$ lies. Let $w \in N_u \backslash (N_u^s \cup v)$ be a sensor. We call the projection of w onto the $(d-1)$-dimensional space of $C_{u,v}$ as *virtual sensor* w'. The intersection of the sensing border of (real) sensor w with the $(d-1)$ dimensional space in which $C_{u,v}$ lies is regarded as the *sensing border of virtual sensor* w'. Similarly, we say that virtual sensor $(uv)'$ lies at the center of $C_{u,v}$ and we regard $C_{u,v}$ as the sensing border of virtual sensor $(uv)'$. The *sensing range of a virtual sensor* is the interior of its sensing border in the $(d-1)$ dimensional space in which $C_{u,v}$ lies.

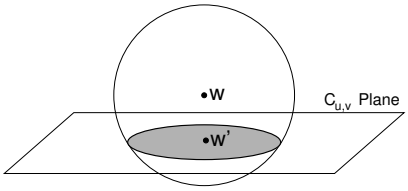


Fig. 1. Illustration of projection concepts for $d = 3$

Fig. 1 illustrates these concepts for the case $d = 3$. The figure shows the sensing border of sensor w, which is a sphere and the plane in which circle $C_{u,v}$ lies. The projection of w on the $C_{u,v}$ plane is virtual sensor w'. The darkened circle on the $C_{u,v}$ plane is the intersection of w's sensing border with the $C_{u,v}$ plane, and is the sensing border of virtual sensor w'. The shaded region within the darkened circle is the interior of the sensing range of virtual sensor w'. Note that this shaded region is within the interior of the sensing range of real sensor w.

In general, a point in the $(d-1)$-dimensional space in which $C_{u,v}$ lies is in the interior of the sensing range of a virtual sensor if and only if it is in the interior of the sensing range of the corresponding real sensor. From this fact and from the definition of virtual sensors and their sensing ranges, it follows that $C_{u,v}$ is k_u-covered by real sensors in the set $N_u \backslash (N_u^s \cup v)$ if and only if the sensing border of virtual sensor $(uv)'$ is k_u-covered by virtual sensors that are the projections of real sensors in the set $N_u \backslash (N_u^s \cup v)$.

Using the distances between pairs of real sensors and the sensing radii of the real sensors, u calculates the distances between pairs of virtual sensors and the sensing radii of the virtual sensors (see Section VI). Subsequently, it can check k_u-coverage of the sensing border of virtual sensor $(uv)'$ by calculations in the $(d-1)$ dimensional space in which $C_{u,v}$ lies. In fact, doing this is exactly identical to the problem of checking coverage of the sensing border of a real sensor when real sensors are deployed in $(d-1)$-dimensions. Thus, we have reduced a coverage verification problem in d dimensions to a number of coverage verification problems in $(d-1)$ dimensions. This problem can be recursively solved using the above steps. Specifically, each coverage verification problem in $(d-1)$ dimension can again be reduced to a number of coverage verification problems in $(d-2)$ dimensions and so on until we get problems in 1 dimension. We describe how such problems can be solved in 1-dimension in Section VII.

C. Examples

Fig. 2 illustrates the procedure. Parts (a) and (b) show the case $d = 3$. Part (a) shows some sensors and their sensing borders which are spheres. The sensing borders of sensors u and v intersect in circle $C_{u,v}$. Part (b) shows the virtual sensors and their sensing borders obtained by projecting the sensors onto the plane of circle $C_{u,v}$. In this figure, the sensing border of virtual sensor $(uv)'$ is subsumed in the sensing range

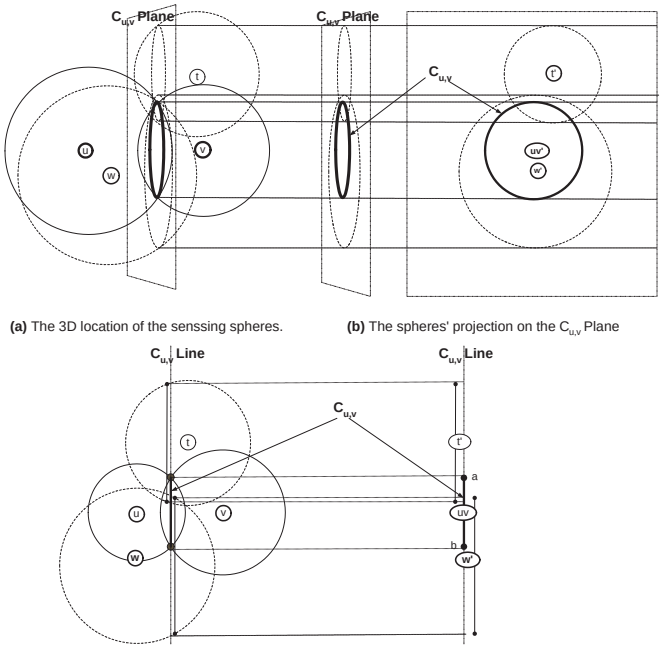


Fig. 2. Parts (a) and (b) show projection of sensors placed in three-dimensions on the $C_{u,v}$ plane. Parts (c) and (d) show projection of sensors placed in two-dimensions on the $C_{u,v}$ line.

of virtual sensor w' and intersects with the sensing border of virtual sensor t'.

Parts (c) and (d) show the case $d = 2$. Part (c) shows some sensors and their sensing borders which are circles. The sensing borders of sensors u and v intersect in the pair of points a and b, which together form 1-boundary-sphere $C_{u,v}$. Part (d) [2] shows the virtual sensors and their sensing borders obtained by projecting the sensors onto the line of 1-boundary-sphere $C_{u,v}$. In part (d), point a is in the interior of the sensing range of virtual sensor t' and point b is in the interior of the sensing range of virtual sensor w'.

In Fig. 3, we summarize the computations executed by each node u for determining k-coverage.

VI. Projection from a Higher Dimension to a Lower Dimension

We now describe how sensors in d-dimensions can be projected onto a $(d-1)$-dimensional space without any knowledge of their coordinates. For concreteness, we describe in detail the projection process for $d = 3$ (Sections VI-A to VI-D). The projection process for arbitrary d is analogous. For $d = 3$, we consider a sensor u which needs to determine k_u-coverage of $C_{u,v}$, the circle formed by the intersection of its sensing border with that of another sensor v in its transmission range. Towards that end, in the projection process, it determines information about virtual sensor $(uv)'$, which is the center of circle $C_{u,v}$, and *projected virtual sensors* which are the projections of sensors in $N_u \backslash (N_u^s \cup v)$ onto the $C_{u,v}$ plane. Specifically, it

[2]In part(d), note that the sensing ranges of virtual sensors $(uv)'$, t' and w' lie on the same straight line, shown dotted. They are shown on different lines so as not to clutter the figure.

Verify_Coverage(u, N_u, k, d)

begin

/* This procedure checks whether the sensing border of node u is k-covered by sensors in the set N_u when u and sensors in the set N_u lie in a d-dimensional target field. */

if d=1 **then**
  Use the algorithm in Section VII to check k-coverage of u's sensing border
else
  Determine N_u^s using Property 1
  if $|N_u^s| \geq k$ **then**
    Return that u's sensing border is k-covered
  else
    Set $k_u = k - |N_u^s|$
    for Every node $v \in N_u \backslash N_u^s$ **do**
      Check, using Property 2, whether the sensing borders of u and v intersect
    end for
    if the sensing border of no sensor $v \in N_u \backslash N_u^s$ intersects u's sensing border **then**
      Return that u's sensing border is not k-covered
    else
      for Every node v such that u's and v's sensing borders intersect **do**
        Project all sensors in the set $N_u \backslash (N_u^s \cup v)$ onto the $(d-1)$-dimensional space containing $C_{u,v}$
        /*Virtual sensor $(uv)'$ is at the center of $C_{u,v}$ and virtual sensors in the set $N_{(uv)'}$ are projections of sensors in the set $N_u \backslash (N_u^s \cup v)$. Recursively check k_u-coverage of $C_{u,v}$ by sensors in the set $N_u \backslash (N_u^s \cup v)$.*/
        Verify_Coverage($(uv)'$, $N_{(uv)'}$, k_u, $d-1$)
      end for
      if for all nodes v such that u's and v's sensing border intersect, $C_{u,v}$ is k_u-covered by sensors in the set $N_u \backslash (N_u^s \cup v)$ **then**
        Return that u's sensing border is k-covered
      else
        Return that u's sensing border is not k-covered
      end if
    end if
  end if
end if

end

Fig. 3. The algorithm run by node u to check k-coverage of its sensing border. The statements delimited by /* and */ are comments

calculates the distances between pairs of these virtual sensors that are in transmission range of each other (sections VI-B and VI-D) and the sensing radius of each of these virtual sensors (in sections VI-A and VI-C). In the projection process, u uses the sensing radii of the sensors in its transmission range, and the distances between pairs of sensors that are in its transmission range and also in transmission range of each other as inputs. Note that u does not use the coordinates of the real sensors or calculate the coordinates of the virtual sensors.

After completing the above projection process from 3-dimensions to 2-dimensions, u needs to check k_u-coverage of the sensing border of $(uv)'$. To this end, after checking for subsumption of the sensing border of $(uv)'$ in the sensing ranges of other virtual sensors, u considers each virtual sensor t' in the $C_{u,v}$ plane, whose sensing border intersects with that of $(uv)'$. The intersection is a pair of points, say $a_{t'}$ and $b_{t'}$. u needs to check coverage of $a_{t'}$ and $b_{t'}$, which can be done by projecting virtual sensors in the set $N_{(uv)'} \backslash (N_{(uv)'}^s \cup t')$ onto the line $a_{t'}b_{t'}$. In this projection from 2-dimensions to 1-dimension, u determines information about the resulting 1-dimensional virtual sensors, which consist of (a) the mid-point of the pair of points $a_{t'}b_{t'}$ and (b) *projected virtual*

sensors which are the projections of the virtual sensors in $N_{(uv)'} \backslash (N^s_{(uv)'} \cup t')$ onto the $a_{t'}b_{t'}$ straight line. As in the 3-dimensions to 2-dimensional projection, u calculates the distances between pairs of these 1-dimensional virtual sensors that are in transmission range of each other and the sensing radius of each of these virtual sensors. As before, u uses as inputs, the sensing radii of the virtual sensors in the transmission range of $(uv)'$, and the distances between pairs of virtual sensors that are in the transmission range of $(uv)'$ and also in transmission range of each other. In Section VI-E, we mention some salient points of the projection process from 2-dimensions to 1-dimension.

We now return to the calculations for projection from 3-dimensions to 2-dimensions. We use the following notation throughout the section. Let O be the center of circle $C_{u,v}$. Thus, the virtual sensor $(uv)'$ is located at point O. If A and B are two points, then the length of the segment AB is denoted by simply AB.

A. Calculation of the Sensing Radius of Virtual Sensor $(uv)'$

We first find lengths Ou and Ov, which are needed throughout this section. For the purpose of this calculation, let u be the origin and let v be the point $(d_{u,v}, 0, 0)$. As derived in section V (see (3)), the x_1 coordinate of every point on circle $C_{u,v}$ is the same, say $x_{1,C_{u,v}}$ and is given by:

$$x_{1,C_{u,v}} = \frac{d^2_{u,v} + r^2_u - r^2_v}{2d_{u,v}}. \tag{5}$$

Since O is the center of the circle $C_{u,v}$, it lies on the straight line joining u and v. Hence, O lies on the x_1-axis and its x_1-coordinate is $x_{1,C_{u,v}}$. So, we get:

$$Ou = |x_{1,C_{u,v}}| = \left| \frac{d^2_{u,v} + r^2_u - r^2_v}{2d_{u,v}} \right| \tag{6}$$

$$Ov = |d_{u,v} - x_{1,C_{u,v}}| = \left| d_{u,v} - \frac{d^2_{u,v} + r^2_u - r^2_v}{2d_{u,v}} \right|. \tag{7}$$

Note that if either u or v is on the $C_{u,v}$ plane, then it is at point O. We can find whether this is the case from (6) and (7). Fig. 4 shows the case in which v lies on the $C_{u,v}$ plane. The case in which u lies on the $C_{u,v}$ plane is symmetrical. Fig. 5 and Fig. 6 respectively show the cases in which u and v lie on the same side and on the opposite side of the $C_{u,v}$ plane. In all these cases, at least one of u and v is not on the $C_{u,v}$ plane. Henceforth in this section, we assume without loss of generality, that u does not lie on the $C_{u,v}$ plane.

Now, u calculates the sensing radius of virtual sensor $(uv)'$, which is equal to $r_{C_{u,v}}$, the radius of circle $C_{u,v}$. As shown in Fig. 6, let G be any point on circle $C_{u,v}$. Note that G lies on sphere u. We have,

$$r_{C_{u,v}} = OG = \sqrt{r^2_u - Ou^2} \tag{8}$$

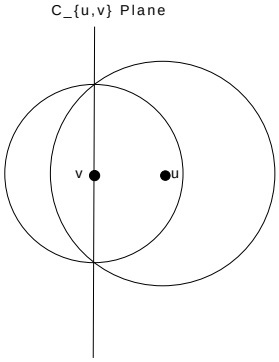


Fig. 4. v lies on the $C_{u,v}$ Plane

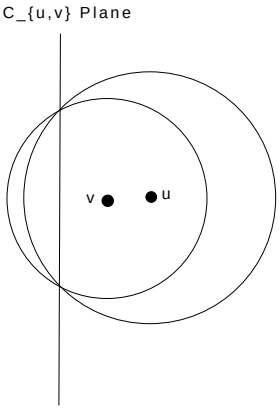


Fig. 5. u and v lie on the same side of the $C_{u,v}$ Plane

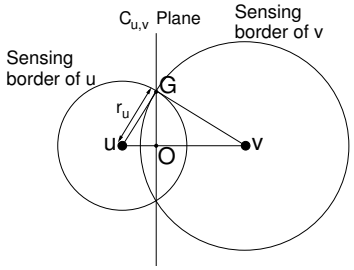


Fig. 6. Calculation of the radius of circle $C_{u,v}$

B. Calculation of the Distance of a Projected Virtual Sensor from $(uv)'$

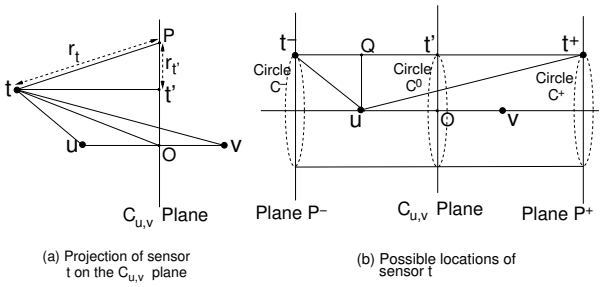
Let $t \in N_u \backslash (N^s_u \cup v)$ be any sensor. We assume henceforth in this section that sphere t intersects with both sphere u and sphere v. (Otherwise, since $C_{u,v}$ is the intersection of sphere u and sphere v, it follows that no point on $C_{u,v}$ can be in the interior of t's sensing range. In this case, sensor t can be ignored for the purposes of checking coverage of circle $C_{u,v}$). We show that t is in the transmission range of both u and v. Since sphere u and sphere t intersect, a point, say p, on sphere u is in the interior of the sensing range of t. Then by the triangle inequality and the assumption that $\hat{R} \geq 2\hat{r}$, we get:

$$d_{t,u} \leq tp + pu < \hat{r} + \hat{r} \leq \hat{R}$$

So t and u are in the transmission range of each other. Similarly, t and v are in the transmission range of each other.

So distances $d_{t,u}$ and $d_{t,v}$ are known to sensor u.

Let t' be a projected virtual sensor that is the projection of real sensor t on the $C_{u,v}$ plane. We now show how u can calculate Ot', the distance between virtual sensors $(uv)'$ and t'. See Part(a) of Fig. 7. The lengths of the sides of $\triangle tuv$ are



(a) Projection of sensor t on the $C_{u,v}$ plane

(b) Possible locations of sensor t

Fig. 7. Part (a) shows projection of the sensor t on the $C_{u,v}$ plane. Part(b) shows possible locations of t for fixed distances Ot' and tt'

$d_{t,u}$, $d_{t,v}$ and $d_{u,v}$. $\angle tuv$ is given by the *cosine rule* as:

$$\angle tuv = \cos^{-1} \frac{d_{t,u}^2 + d_{u,v}^2 - d_{t,v}^2}{2d_{t,u}d_{u,v}} \tag{9}$$

By the cosine rule in $\triangle Otu$:

$$Ot = \sqrt{Ou^2 + d_{t,u}^2 - 2(Ou)(d_{t,u})cos\angle tuv} \tag{10}$$

Again, by the cosine rule in $\triangle Otu$, $\angle tOu$ is given by:

$$\angle tOu = \cos^{-1} \frac{Ot^2 + Ou^2 - d_{t,u}^2}{2(Ot)(Ou)} \tag{11}$$

Now, if t and u lie on the same side of the $C_{u,v}$ plane, as in Part(a) of Fig. 7, then $\angle tOt' = 90° - \angle tOu$. If t and u lie on opposite sides of the $C_{u,v}$ plane, then $\angle tOt' = \angle tOu - 90°$. In either case:

$$\angle tOt' = |90° - \angle tOu| \tag{12}$$

Since t' is t's projection on the $C_{u,v}$ plane, $\angle tt'O$ is a right angle. From $\angle tOt'$, we get:

$$Ot' = Ot \cos \angle tOt' \tag{13}$$

Thus, Ot' can be determined from equations (6) and (9) to (13).

C. Calculation of the Sensing Radius of a Projected Virtual Sensor

We show how u can calculate $r_{t'}$, the sensing radius of virtual sensor t'. See Part(a) of Fig. 7. From Ot and $\angle tOt'$, which were calculated in (10) and (12) respectively, we get:

$$tt' = Ot \sin \angle tOt' \tag{14}$$

Now, if $r_t \leq tt'$, then sphere t does not intersect with the $C_{u,v}$ plane and virtual sensor t' can be ignored for the purpose of checking coverage of circle $C_{u,v}$. If $r_t > tt'$, then let P be a point where sphere t intersects the $C_{u,v}$ plane. Since $t'P$ equals $r_{t'}$, the radius of the circle formed by the intersection of sphere t with the $C_{u,v}$ plane, we get:

$$r_{t'} = t'P = \sqrt{(tP)^2 - (tt')^2} = \sqrt{r_t^2 - (tt')^2} \tag{15}$$

Thus, $r_{t'}$ can be calculated from (14) and (15).

D. Calculating the Distance Between Two Projected Virtual Sensors

We show how u can calculate the distance between two projected virtual sensors s' and t' that are in transmission range of each other. We define two virtual sensors s' and t' to be in the transmission range of each other if and only if their sensing ranges intersect. The motivation behind this definition is that, the projection from 2-dimension to 1-dimension and verification of coverage in 1-dimension will use the distances between virtual sensors that are in transmission range of each other as per the above notion.

Consider two sensors $s, t \in N_u \setminus (N_u^s \cup v)$. We now show that the corresponding projected virtual sensors s' and t' are in transmission range of each other only if the corresponding real sensors s and t are in the transmission range of each other. Suppose s and t are not in the transmission range of each other. In this case, it can be easily shown, from the assumption that $\hat{R} \geq 2\hat{r}$, that the sensing ranges of s and t do not intersect. Since the sensing range of a virtual sensor is a subset of the sensing range of the corresponding real sensor, it follows that the sensing ranges of virtual sensors s' and t' do not intersect as well. Thus, henceforth, we assume that s and t are in transmission ranges of each other, and subsequently compute $s't'$.

First, assume that u can determine whether t and s are on the $C_{u,v}$ plane, and whether t lies on the same side of the $C_{u,v}$ plane as s. If both t and s are on the $C_{u,v}$ plane, then $s't' = d_{t,s}$. Now, suppose at least one of t and s is not on the $C_{u,v}$ plane. Assume, without loss of generality, that t is not on the $C_{u,v}$ plane. If s is also not on the $C_{u,v}$ plane, node u checks whether s and t lie on the same side or opposite sides of the $C_{u,v}$ plane. Parts (a) and (b) of Fig. 8 show these cases. In both these figures, a perpendicular is drawn from s to the line joining t and t'. Suppose it meets the line at point H. We now consider these cases separately:

1) *s is either on the $C_{u,v}$ plane or on the same side of the $C_{u,v}$ plane as t*: Assume, without loss of generality, that $ss' \leq tt'$. We have:

$$tH = tt' - Ht' = tt' - ss' \tag{16}$$

   tt' and ss' can be calculated as in (14).

2) *s and t are on opposite sides of the $C_{u,v}$ plane*: We have:

$$tH = tt' + Ht' = tt' + ss' \tag{17}$$

In both the above cases, $s't'$ can be calculated as follows:

$$s't' = sH = \sqrt{d_{s,t}^2 - (tH)^2} \tag{18}$$

Thus, u can compute $s't'$ using (16), (17) and (18).

Recall that we have assumed that sensor u does not lie on the $C_{u,v}$ plane. We now show how u can find whether a sensor t is on the $C_{u,v}$ plane, and if not whether t is on the same side of the $C_{u,v}$ plane as u. Using this procedure, u can determine whether two sensors s and t are on the same side of the $C_{u,v}$ plane. If t is any sensor, $\angle tOt'$ can be calculated as in (12). t lies on the $C_{u,v}$ plane if and only if $\angle tOt' = 0$.

Now suppose sensor t does not lie on the $C_{u,v}$ plane. Suppose distances Ot' and tt' have been calculated as in (13) and (14). See Part(b) of Fig. 7. Let P^- and P^+ be the planes parallel to the $C_{u,v}$ plane at a distance equal to distance tt' from the $C_{u,v}$ plane, on the same side as u and on the opposite side respectively. Let C^0 be the circle on the $C_{u,v}$ plane with center O and radius equal to distance Ot'. For a fixed distance Ot', every point on circle C^0 is a possible location for t'. Let circles C^- and C^+ be the circles that are the projections of circle C^0 on planes P^- and P^+ respectively. Since t' is the projection of t on the $C_{u,v}$ plane, t is the projection of t' on either plane P^- or P^+. So for fixed distances Ot' and tt', every point on circle C^+ and circle C^- is a possible location for t. For some candidate location of t' on circle C^0, let t^- and t^+ be the corresponding candidate locations for t on circle C^- and C^+ respectively. Note that t^- and t^+ lie on the normal to the $C_{u,v}$ plane passing through that particular candidate location of t' as shown in Part(b) of Fig. 7. Suppose the normal from u to line t^-t^+ meets the line at point Q. Then, $uQ = Ot'$ and $Qt' = Ou$. Thus, $t^-Q = t^-t' - Qt' = tt' - Ou$ and similarly, $t^+Q = tt' + Ou$. So distances ut^- and ut^+ can be calculated as follows:

$$ut^- = \sqrt{(tt' - Ou)^2 + (Ot')^2} \tag{19}$$

$$ut^+ = \sqrt{(tt' + Ou)^2 + (Ot')^2} \tag{20}$$

In the above expressions, tt', Ot' and Ou can be calculated as in (14), (13) and (6) respectively. Note that every point on circle C^- (respectively, C^+) is at distance ut^- (respectively, ut^+) from u. If $d_{t,u} = ut^-$, then t lies on circle C^-, hence, on the same side of the $C_{u,v}$ plane as u. Otherwise $d_{t,u} = ut^+$ and then t lies on circle C^+, hence, on the opposite side of the $C_{u,v}$ plane as u.

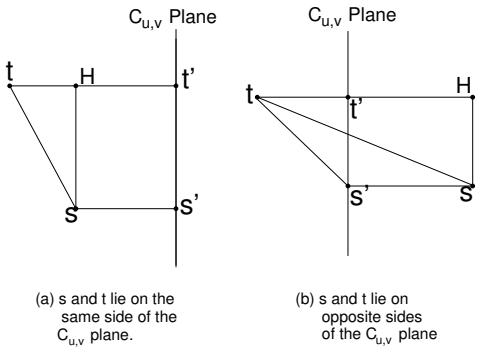


Fig. 8. Parts (a) and (b) shows the cases in which s and t are on the same side and opposite sides of the $C_{u,v}$ Plane respectively

Remark 1: Note that when sensors are projected on a plane, two or more virtual sensors may end up at the same location even when no two real sensors are located at the same position. So for the two-dimensional problem, the requirement of Proposition 1 is not met. However, this is not a difficulty because in the two-dimensional problem, we need only to check whether the sensing border of virtual sensor $(uv)'$ is k_u-covered. We do not need to check k_u-coverage of the target field. So we do not apply Proposition 1.

E. Projection from Two Dimensions to One Dimension

We now comment on some salient aspects of the projection from a plane to a line (2-dimensions to 1-dimension).

First, as for the virtual sensors obtained after projection onto a plane, two 1-dimensional virtual sensors (i.e., those obtained after projection onto a line) are said to be in the transmission range of each other if and only if their sensing ranges intersect.

The same 2-dimension to 1-dimension projection scheme applies both when the sensors on a plane are virtual sensors obtained after projection from a 3-dimensional space, and also when the sensors on a plane are real sensors.

VII. k-Coverage Verification Algorithm in One Dimension

In this section, we describe the k-coverage verification algorithm for the case in which sensors are placed on a straight line. The one-dimensional case may arise if in practice, real sensors are placed on a straight line or if sensors are placed in a higher dimension and we reduce the problem to several one-dimensional coverage verification problems. Each sensor knows the distances between adjacent sensors in its transmission range and their sensing radii either from distance measurements and exchanges (if sensors are real) or from calculations in the projection process (if sensors are virtual).

Consider a sensor u, which may either be real or virtual, that checks k-coverage of its sensing border. u sets up a coordinate system in which u is the origin. The sensing border of u consists of two points, say a_u and b_u that are located at r_u and $-r_u$ respectively. An arbitrary sensor v can be at any one of the points $d_{u,v}$ and $-d_{u,v}$. Denote these points by v^+ and v^- respectively. We now outline the algorithm that u uses to check k-coverage of a_u and b_u. u finds out whether each sensor v in its transmission range contains at least one of the points a_u and b_u. This is the case if and only if $d_{v^+,a_u} < r_v$ (since $d_{v^+,a_u} \leq d_{v^+,b_u}$). u discards sensors which do not cover any of the points a_u and b_u. It then divides the remaining sensors into three sets S^+, S^- and S^0 such that sensors in the set S^+ are on the right of the origin (*i.e.,* on the same side as a_u), sensors in the set S^- are on the left of the origin (on the opposite side from a_u) and sensors in the set S^0 are at the origin. For each sensor s in the transmission range of u, u knows the distance $d_{s,u}$. Since, in addition, it determines on which side of the origin it lies, it knows its coordinate. Thus, the locations of all sensors in sets S^+, S^- and S^0 and the locations of a_u and b_u are known. Using these, for each sensor $s \in S^+ \cup S^- \cup S^0$, u can find distance d_{s,a_u} and d_{s,b_u}.

Now, a_u (b_u, respectively) is in the interior of the sensing range of s if and only if $d_{s,a_u} < r_s$ ($d_{s,b_u} < r_s$, respectively). Thus, u can determine whether a_u and b_u are k-covered.

We now describe how u divides the sensors into sets S^+, S^- and S^0. The sensors v for which $d_{u,v} = 0$ are put into set S^0. Next, let $t \notin S^0$ be a sensor. For every other sensor $s \notin S^0$, u finds out whether t and s lie on the same side or opposite sides of the origin. This can be done as follows. If t and s are in the transmission range of each other, u knows the distance $d_{t,s}$ between them from the projection processes. Note that $d_{t^+,s^+} = |d_{u,t} - d_{u,s}|$ and $d_{t^+,s^-} = |d_{u,t} + d_{u,s}|$. If $d_{t,s} = d_{t^+,s^+}$, then t and s lie on the same side of the origin, else $d_{t,s} = d_{t^+,s^-}$ and then t and s lie on opposite sides of the origin. Suppose t and s are not in the transmission range of each other, which implies that their sensing ranges do not intersect. We show by contradiction that they are on opposite sides of the origin. Suppose they are on the same side, say on the right of the origin. Then, since they both contain at least one of a_u and b_u, it follows that they both contain the point a_u. So their sensing ranges intersect, which is a contradiction.

Now, let S^+ (S^-, respectively) consist of the sensors that are on the same (opposite) side as t. If it is the other way round, then by symmetry of the points a_u and b_u around the origin, the conclusion about k-coverage of a_u and b_u will not change. This is because we are interested in whether or not *both* a_u *and* b_u are k-covered. Thus, u can divide the sensors into sets S^+, S^- and S^0.

VIII. Correctness and Complexity Analysis

The following theorem establishes that the algorithm presented in Fig. 3 solves the coverage verification problem.

Theorem 1 (Correctness): When $\hat{R} \geq 2\hat{r}$, the coverage verification algorithm in Fig. 3 reports the presence of a k-coverage hole if and only if the internal space of the target field has a k-coverage hole, and does not use location information.

Proof: From Proposition 1, it follows that in order to verify k-coverage of the internal space of the target field, it is sufficient to check k-coverage of the sensing border of each node u. When $|N_u^s| < k$, it follows from Proposition 2 that in order to check k-coverage of u'sensing border, it is sufficient to check k_u-coverage of $C_{u,v}$ for each $v \in N_u \backslash N_u^s$ by sensors in the set N_u $(N_u^s \cup v)$. Note that $C_{u,v}$ is k_u-covered by real sensors in the set $N_u \backslash (N_u^s \cup v)$ if and only if the sensing border of virtual sensor $(uv)'$ is k_u-covered by virtual sensors that are the projections of real sensors in the set $N_u \backslash (N_u^s \cup v)$. The coverage verification algorithm in Fig. 3 checks k_u-coverage of $C_{u,v}$ for each $v \in N_u \backslash N_u^s$ by (a) projecting sensors in the set N_u $(N_u^s \cup v)$ onto the $(d-1)$-dimensional space containing $C_{u,v}$ when $d > 1$ and recursing; and subsequently (b) checking k_u coverage in 1-dimension of the sensing borders of virtual sensors projected in 1-dimension. In Section VI (Section VII, respectively), we have presented the algorithms that accomplish steps (a) (and (b), respectively), when $\hat{R} \geq 2\hat{r}$. These algorithms do not use location information. The result follows since we have shown that the algorithm in step (b) correctly determines whether or not the sensing borders of virtual sensors are j-covered for any j. ■

We now compute the running time of the algorithm run by sensor u in terms of $|N_u|$, the number of sensors in its transmission range.

A. One Dimension

Suppose real or virtual sensors are located in one dimension. We now give the complexity of the algorithm described in Section VII. The following are constant time operations:

1) Checking whether at least one of a_u and b_u is in the sensing range of a sensor v,
2) If t is a sensor not at the origin, checking for a sensor s whether s and t are on the same side or opposite sides of the origin,
3) Once sensors have been divided into sets S^+, S^- and S^0, checking whether a_u and b_u are in the interior of the sensing range of each sensor.

Since there are $|N_u|$ sensors in N_u, the total time required for performing the above operations for all sensors in N_u is $O(|N_u|)$. Hence, the complexity of the algorithm run by sensor u is $O(|N_u|)$.

B. Projection from d to $d-1$ dimensions

Let $d \geq 2$. We now find the complexity of projecting sensors in $N_u \backslash (N_u^s \cup v)$ onto the $(d-1)$ dimensional space in which $C_{u,v}$ lies, using the algorithm in Section VI. Calculation of the sensing radius of sensor $(uv)'$ takes $O(1)$ time. Calculation of the distance of a projected virtual sensor t' from virtual sensor $(uv)'$ and the sensing radius of t' are constant time operations. So the total time required over all projected virtual sensors is $O(|N_u|)$. Again, calculation of the distance between two projected virtual sensors takes $O(1)$ time. There are $O(|N_u|^2)$ such pairs. So the total time taken is $O(|N_u|^2)$.

Adding the times for all the projection calculations, the complexity of the projection calculations is $O(|N_u|^2)$.

C. Two Dimensions

We now compute the complexity of the calculations performed by a real or virtual sensor u to check k-coverage of its sensing border when u and other sensors are located in two-dimensions. Checking whether the sensing border of u is subsumed in the sensing range of another sensor v is a constant time operation. So the total time for all $v \in N_u$ is $O(|N_u|)$. Now, suppose the sensing borders of u and v intersect. Projection of sensors in $N_u \backslash (N_u^s \cup v)$ onto the $C_{u,v}$ line takes $O(|N_u|^2)$ time and checking coverage in the resulting one-dimensional problem takes $O(|N_u|)$ time as computed above. Thus, the time for checking coverage of the intersection points of the sensing borders of u and v is $O(|N_u|^2)$. Adding over all $v \in N_u \backslash N_u^s$, we get the total time as $O(|N_u|^3)$. Adding the computation times for subsumption and intersection, the total running time taken by sensor u for checking k-coverage of its sensing border is $O(|N_u|^3)$.

D. Three Dimensions

Now suppose sensors are located in three-dimensions. Analogous to the two-dimensional case, the total time for checking subsumption is $O(|N_u|)$. If sphere u and sphere v intersect, projection of sensors in $N_u \backslash (N_u^s \cup v)$ onto the $C_{u,v}$ plane takes $O(|N_u|^2)$ time and checking coverage in the resulting two-dimensional problem takes $O(|N_u|^3)$ time as computed above. So the total time for checking coverage of circle $C_{u,v}$ is $O(|N_u|^3)$. Adding over all sensors v, the total running time is $O(|N_u|^4)$. Adding the running times for subsumption and intersection calculations, the total running time taken by sensor u for checking coverage is $O(|N_u|^4)$.

E. d dimensions, $d \geq 4$

It is easy to show by induction that when sensors are located in d dimensions where $d \geq 4$, the running time taken by sensor u for checking coverage is $O(|N_u|^{d+1})$.

F. Running Time for Coverage Verification

In the above calculations, we found the complexity of the algorithm run by node u for different dimensions in terms of $|N_u|$. Now, let $\Delta = \max_{u \in V} |N_u|$ be the maximum number of neighbors of any node in the network. When sensors are located in d dimensions, the time taken by any sensor to check k-coverage of its sensing border is [3]:

1) $O(\Delta)$ if $d = 1$ and
2) $O(\Delta^{d+1})$ if $d > 1$.

IX. Simulation Results

We analytically proved that the coverage verification scheme accurately verifies k-coverage of the target field when the ratio $\hat{R}/\hat{r} \geq 2$. In our simulations, we demonstrate how to make our scheme more robust to errors in the measurements of distances between nodes. In all our simulations, we consider a three-dimensional target field and the case $k = 1$. Each internal node in the target field uses the coverage verification algorithm described in this paper to verify coverage of its sensing border and reports the presence of a hole if it detects one.

First, we simulated some simple test cases. We developed a simulator that allows us to create a coverage hole of controllable size at the center of the target field and to place the nodes randomly around it, while ensuring that the space around the coverage hole is fully covered. We ran a large number of simulations with different sizes of the central coverage hole. We observed that when the hole size was 0, no node reported a hole and when the hole size was greater than 0, some nodes reported a hole. This experimentally confirms the correctness of the scheme.

Next, we considered a $50 \times 50 \times 50$ units^3 target field. Each sensor had a sensing radius of 10 units and a transmission radius of 22 units. Each sensor was placed uniformly at random in the target field. We varied the number of nodes and ran 100 simulations in each case. We searched for a number of nodes to place, for which the probability of a coverage hole is roughly 0.5. We found that when the number of nodes was equal to 370, in 51 instances out of 100, a coverage hole was reported. We fixed the number of nodes at 370 for the rest of the simulations. We considered the case in which there may be errors in the distances measured by the nodes. In our simulator, the evaluated distance between two adjacent nodes u and v is given by,

$$Eval\_Dist_{u,v} = d_{u,v} \cdot (1 + X \cdot Error\_Index)$$

where, $d_{u,v}$ is the actual distance between the nodes, $X \sim N(0,1)$ is a normal random variable and $Error\_Index$ is a simulation parameter that controls the variance of the measurement errors. We generated 100 random seeds, each of which results in a particular placement of the nodes. We increased $Error\_Index$ from 0% to 5% in increments of 0.5% and in each case, ran a simulation with each one of the 100 random seeds. We define a *false alarm* to be a simulated instance in which there is actually no hole, but a hole is reported. A *misdetection* is defined to a simulated instance in which there is actually a hole, but it is not reported. With a positive value of $Error\_Index$, there are false alarms and misdetections, which can be found by comparing with the simulation for $Error\_Index = 0$ for the same random seed. The probility of false alarm (respectively misdetection) is the fraction of the total simulated instances that are false alarms (respectively misdetections). We define the *risk* to be the sum of the probabilities of false alarm and misdetection.

In order to improve the robustness of our scheme to distance measurement errors, we propose threshold-type of policies in which a hole is reported by the network if and only if the number of nodes who report that they border a hole is greater than the threshold. The top plot of Fig. 9 shows the probabilities of false alarm and misdetection and the risk as a function of threshold for $Error\_Index = 2\%$. The plot shows that the false alarm probability decreases as a function of threshold. This is because, if the threshold is T, a hole is reported by the network if the number of nodes who report a hole is greater than T. So for a higher value of T, a hole is reported in fewer simulated instances. For the same reason, the misdetection probability increases with the threshold. Roughly, the risk first decreases and then increases. The risk is minimized at a threshold value of 6. This is the *optimal threshold*.

The bottom plot of Fig. 9 shows the risks with the optimal threshold, with a threshold of 0 and with a threshold equal to twice the optimal threshold as a function of $Error\_Index$. It can be seen that the risk with the optimal threshold is always lower than the risks with the other two thresholds. This shows that a threshold-type of policy can reduce the risk if the threshold is chosen judiciously.

Fig. 10 shows the optimal threshold as a function of $Error\_Index$. The plot shows that the optimal threshold increases with $Error\_Index$. The reason for this is that for any fixed placement of nodes (*i.e.,* for a fixed random seed), the number of nodes who report a hole increases with $Error\_Index$. Fig. 11 illustrates this for a particular value of

[3]Note that in practice, $d \leq 3$ and hence the algorithm is polynomial-time.

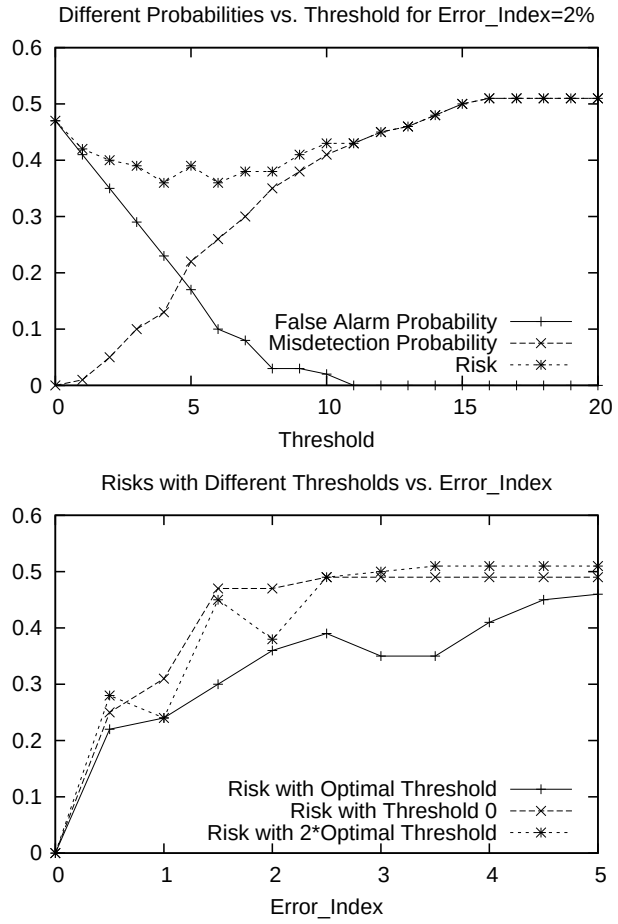


Fig. 9. The top plot shows the false alarm probability, misdetection probability and the risk as a function of threshold for $Error\_Index = 2\%$. The bottom plot shows the risks with the optimal threshold, with a threshold of 0 and with a threshold equal to twice the optimal threshold as a function of $Error\_Index$

the random seed. We now intuitively explain this phenomenon. Suppose there are random errors in the distance measurements, as is the case for positive $Error\_Index$. It is a likely event that if a node's sensing border is actually covered, with the erroneous distances at least a small hole is detected on its sensing border. However, it is unlikely that if there was actually a hole on its sensing border, the random errors are such that the hole is found to be covered. Thus, the number of nodes who erroneously report that they border a hole increases with $Error\_Index$. Now, suppose for $Error\_Index = e_1$, the optimal threshold is T_1. If $Error\_Index$ is increased to $e_2 > e_1$, then for every random seed, the number of nodes who report a hole increases. Hence, for any random seed, for $Error\_Index = e_2$ it is possible to choose a threshold $T_2 \geq T_1$ such that if there is no misdetection for this random seed for threshold equal to T_1, then there is no misdetection for threshold equal to T_2. Also, since $T_2 \geq T_1$, it is possible that there is no false alarm for threshold T_2 even if there is a false alarm for threshold equal to T_1. Since this is the case for all random seeds, it is intuitive that the optimal threshold increases with $Error\_Index$.

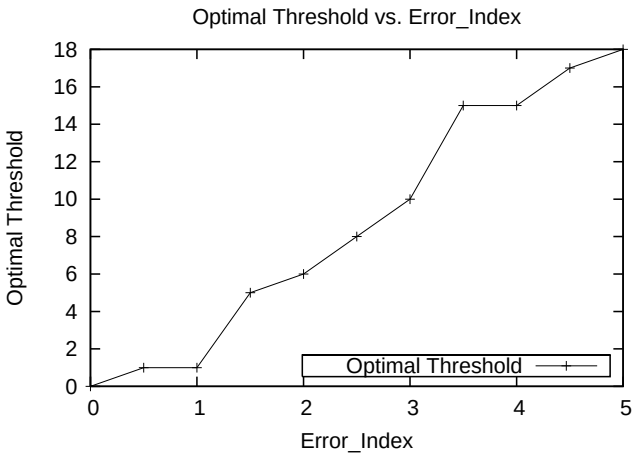


Fig. 10. The optimal threshold as a function of $Error\_Index$

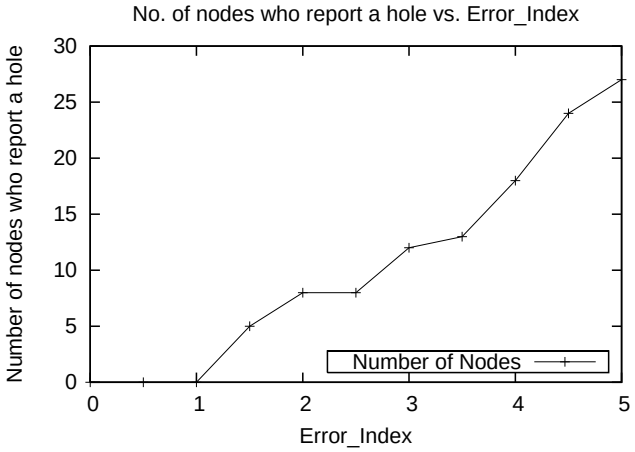


Fig. 11. The number of nodes who report a hole as a function of $Error\_Index$ for a fixed placement of nodes

X. Conclusion

We presented an efficient, distributed, coordinate-free algorithm for verifying k-coverage of a d-dimensional target field for arbitrary integers k and d. We analytically proved that the scheme detects a coverage hole if and only if there is a coverage hole in the target field. Our simulation results show how the robustness of the scheme to distance measurement errors can be improved by using a threshold type policy. We believe that the methods developed in this study are fundamental for wireless sensor network management and they will affect the design of new network protocols.

References

[1] F. Zhao and L Guibas, *"Wireless Sensor Networks: An Information Processing Approach"*. Morgan Kaufmann, 2004.

[2] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks". In Proc. of *Infocom'01*, Anchorage, Alaska, U.S.A., April 2001.

[3] N. Ahmed, S.S. Kanhere, S. Jha, "The holes problem in wireless sensor networks: a survey". *Mobile Computing and Communications Review*, Vol. 9, No. 2, pp. 4-18, 2005.

[4] M. Cardei and J. Wu. *"Coverage in Wireless Sensor Networks"*, Handbook of Sensor Networks. CRC Press 2004.

[5] Q. Fang, J. Gao, and L. Guibas. "Locating and bypassing routing holes in sensor networks.". In Proc. of *Infocom'04*, March 2004.

[6] G. Wang, G. Cao, and T. La-Porta. "Movement-assisted sensor deployment". In Proc. of *Infocom'04*, Hong Kong, China, March 2004.

[7] A. Man-Cho So and Y. Ye. "On Solving Coverage Problems in a Wireless Sensor Network Using Voronoi Diagrams". In Proc. of *WINE 2005*, LNCS 3828, pp. 584-593, 2005.

[8] C-F. Huang and Y.-C Tseng, "The Coverage Problem in a Wireless Sensor Network". In Proc. of *ACM WSNA'03*, Sep. 2003.

[9] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks". *In International Journal of Wireless Ad Hoc and Sensor Networks*, vol. 1, num. 1-2, pp. 89-123, January 2005.

[10] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. "Integrated coverage and connectivity configuration in wireless sensor networks". In Proc. of *ACM SenSys'03*, Los Angeles, CA, Nov. 2003.

[11] D. Niculescu, "Positioning in ad hoc sensor networks", In *IEEE Network*, Volume 18, Issue 4, July-Aug. 2004 Pages: 24 – 29

[12] , Q. Shi, S. Kyperountas, N. S. Correal and F. Niu. "Performance Analysis of Relative Location Estimation for Multihop Wireless Sensor Networks". *IEEE Journal On Selected Areas In Communications (JSAC)*, Vol 23, No. 4, April 2005.

[13] R. Ghrist, A. Muhammad, "Coverage and hole-detection in sensor networks via homology". In Proc. of *IPSN 2005*, April 2005.

[14] X. Li, D. K. Hunter, and K. Yang, Distributed Coordinate-free Hole Detection and Recovery, In Proc. of *Globecom '06*, November 2006.

[15] Y. Wang, J. Gao, J. S. B. Mitchell, Boundary Recognition in Sensor Networks by Topological Methods, In Proc. of *Mobicom '06*, Sep. 2006.

[16] C. Zhang, Y. Zhang and Y. Fang "Detecting Coverage Boundary Nodes in Wireless Sensor Networks", In Proc. of *ICNSC '06*, April 2006.

[17] R. R. Choudhury and R. Kravets, "Location-Independent Coverage in Wireless Sensor Networks" Technical Report, UIUC, 2004

[18] B. Alavi and K. Pahlavan "Modeling of the TOA-based Distance Measurement Error Using UWB Indoor Radio Measurements". In *IEEE Communications Letters*, Vol. 10, No. 4, April 2006, pages 275-277.

[19] C. Y. Wen, R. D. Morris, and W. A. Sethares, "Distance Estimation Using Bidirectional Communications Without Synchronous Clocking", Accepted for publication in IEEE Trans. Signal Processing.

[20] Y. Bejerano, "Simple and Efficient k-Coverage Verification without Location Information". In Proc. of *Infocom'08*, Phoenix, Arizona, U.S.A., April 2008.

[21] C.-F. Huang, Y.-C. Tseng and L.-C. Lo, "The coverage problem in three-dimensional wireless sensor networks". In Proc. of *GLOBECOM '04*, Vol. 5, pp 3182-3186, Nov-Dec 2004.

[22] M. K. Watfa and S. Commuri, "The 3-Dimensional Wireless Sensor Network Coverage Problem". In Proc. of *ICNSC '06*, pp 856- 861, April 2006.

[23] S.M.N. Alam and Z.J. Haas, "Coverage and Connectivity in Three-Dimensional Networks", In Proc. of *Mobicom '06*, Sep. 2006.

[24] , I.F. Akyildiz, D. Pompili and T. Melodia, "Underwater Acoustic Sensor Networks: Research Challenges", *Ad Hocs Networks Journal, (Elsevier)*, March 2005.