Ad Hoc & Sensor Wireless Networks Vol. 00, pp. 1–31
Reprints available directly from the publisher
Photocopying permitted by license only

Operational Range Assignment in Sensor and Actor Networks

OSSAMA YOUNIS[1], SRINIVASAN RAMASUBRAMANIAN[2], AND MARWAN KRUNZ[2]

[1]*Applied Research, Telcordia Technologies One Telcordia Dr., Piscataway, NJ 08854*
Email: oyounis@research.telcordia.com
[2]*Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721*
Email: {srini,krunz}@ece.arizona.com

Received: September 25. Accepted: October 4, 2007.

Networks of sensors and actors can be used in a variety of applications, including the detection of chemical activity or controlling pests in farms. In these applications, nodes (sensors and actors) are typically deployed in large numbers. Thus, not all of them need to operate at their maximum *operational* (sensing/acting) ranges, especially when the underlying operation is expensive. In this work, we study how to minimize the operational range in a distributed fashion. Unlike previous work, we consider environments in which location estimation is not possible (e.g., dense forests). We develop a distributed protocol for assigning the operational ranges of nodes (ORAP). ORAP assigns longer ranges to nodes that are less resource constrained (e.g., in terms of energy) than their peers in order to prolong the operational lifetime of every individual node. If the nodes are not aware of their locations, ORAP employs a novel localized algorithm (VICON) for determining the virtual coordinates of the neighbors of each node. VICON relies on conservative estimates of neighbor distances and advertised 1-hop neighborhood information. We show that ORAP introduces negligible overhead and meets its intended goals. Extensive simulations indicate that ORAP balances resource consumption among nodes, resulting in a significant network lifetime improvement even under inaccurate distance estimation.

Keywords: Sensor and actor networks, location-unawareness, range assignment, localized protocols.

I INTRODUCTION

Maximizing the network lifetime is a key design objective for applications that employ networks of sensors and actors. In sensor network applications, the network lifetime is defined here as the time until the external observer (i.e., the sink) no longer receives any sensing reports. This is why it can also be referred to as "coverage time." Under such a definition, one or more nodes may still be alive even when the observer no longer receives any sensed data. In wireless sensor networks (WSNs), coverage time depends on the sensor battery, whose lifetime is limited and must be efficiently used. On the other hand, in wireless actor networks (WANs), coverage time depends on the availability of "action" material such as pesticides for pest control or foam for fire extinguishers. In this paper, we refer to the battery in a WSN (or the pesticide in a WAN) as the *critical resource*.

In a WSN, we define the sensing range of a sensor as the distance at which an event or a phenomenon can be detected. We focus on active sensing applications, such as RADAR systems that require sensors to send continuous long-range ultrasound pulses for object detection. In a WAN, we define the acting range as the farthest distance from an actor at which the resource is effective. For example, the acting range can be the farthest distance that a pesticide or foam can reach. We refer to the sensing (or acting) range as the *operational range*. In this paper, we assume that each node (sensor/actor) has k different operational levels, each defining an operational range. In such model, every node does not need to operate at its maximum operational level. This is demonstrated in Figure 1, where node A only needs to set its operational level to cover the farthest point that is not covered by any of its neighbors.

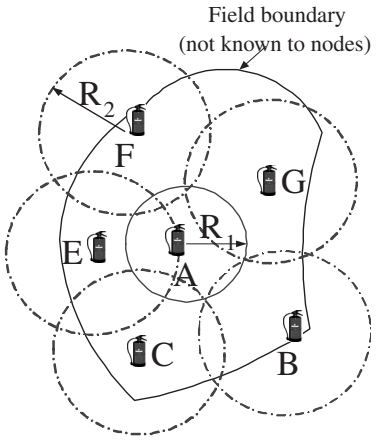


FIGURE 1
A network of fire extinguishers in which every node can operate at one of two levels that correspond to ranges R_1 and R_2.

In this work, we focus on applications in which the operational range is the dominant factor for resource consumption. The benefits of reducing the operational range is apparent in WANs, especially when the material needed for action is not renewable (e.g., in disaster areas). It is also imperative to reduce redundant actions when using potentially dangerous materials such as pesticides or tear gas to avoid lethal effects on plants or people. In sensing applications like RADAR systems, the sensing process is continuously active, while processing and communication only take place when an object of interest is detected. Reducing the operational range in such applications results in significant energy savings. In other WSN applications, the sensed information might require analysis before reporting, which typically results in significant time/energy consumption. Reducing the sensing range in such applications significantly reduces the dataset to be analyzed, thus reducing the processing overhead and the consumed energy. Currently, some commercially available sensors (e.g., Sandia National Lab's chemiresistors [8] and the Osiris photoelectric sensor [25]) are capable of adjusting their sensing levels to control the cost associated with the sensing process.

Selecting the optimal operational ranges for all the nodes in the network so as to minimize the consumption of the critical resource is an NP-hard problem [6] (in fact, the simplified version of this problem in which each node is either ON or OFF is also NP-hard [11]). This is why heuristic techniques were proposed in previous research for range assignment. For example, the authors in [6] considered range assignment for nodes that monitor a fixed set of targets in the field, while [37] focused on constructing connected covers. However, the problem is more challenging in location-unaware networks in which a node is not capable of determining its location or the relative directions of its neighbors. This occurs, for example, when the node cannot perform network-wide localization based on location-aware anchor nodes (e.g., in forests or caves), and multilateration or multitriangulation algorithms cannot be applied (as explained below). Selecting the operational ranges for nodes should also consider the capabilities of each node to balance the load in the network and prolong the lifetime of every node.

Contributions: In this work, we develop a distributed operational-range assignment protocol (ORAP) for resource-constrained networks. In our model, we assume that the nodes may not be aware of their locations or of the field boundary. Therefore, the objective of every node[1] is to adjust its operational range such that its maximum sensing/acting region is covered. ORAP balances the load among nodes by assigning operational ranges based on the availability of the critical resource at each node. Our analysis indicates that ORAP introduces negligible overhead in the network. We compare ORAP to extended algorithms from the literature that are proposed for different network

[1] A "node" is used to refer to either a sensor in a WSN or an actor in a WAN.

models and applications. Extensive simulations show that ORAP significantly prolongs coverage time of the field.

Since knowledge of neighbors' coordinates by a node v is needed to compute v's operational range, ORAP employs a novel algorithm for determining the *virtual* (relative) coordinates of neighbors (VICON) prior to range selection. VICON is "customized" to conservatively set the neighbor coordinates so as not to leave uncovered regions. At a node v, VICON exploits the available neighborhood information and the estimated 1-hop distances to v's neighbors to determine the virtual coordinates of these neighbors. Inaccurate distance estimation, inconsistent neighborhood information, and disconnected neighbor graphs may result in a failure to determine the coordinates of some neighbors. Thus, VICON employs heuristics to find virtual coordinates for all the neighbors under these challenges.

The rest of the paper is organized as follows. Section II briefly surveys related work. Section III provides our system model and assumptions, and formulates the problem under study. Section IV introduces our VICON (VIrtual COordinates of Neighbors) algorithm and illustrates how it handles inaccurate distance estimation. Section V provides details of ORAP and studies its properties in terms of correctness, complexity, and overhead. Section VI evaluates the performance of ORAP in the context of a WSN application in which measurements from sensors are collected by an observer placed in the network field. Finally, Section VII gives concluding remarks and directions for future research.

II RELATED WORK

A large body of research has studied coverage problems in WSNs. Much of this research can also be applied to WANs. We categorize the related work in this area into two parts; one that assumes fixed operational (sensing) ranges, and the other that assumes variable operational ranges. Our work belongs to the latter category.

A Selecting Covers Under Fixed Operational Ranges

Most of the proposed work under this category has studied the problem of selecting active covers[2] from sensors with fixed sensing and transmission ranges. The proposed protocols often assume the ability to estimate node locations or at least the directionality of neighbor's received messages. In [34], we employed geometric arguments and intelligent exploitation of node density to develop distributed protocols for computing covers without the need for location or directionality information. Cardei *et al.* [5] computed a number of set covers for a fixed number of targets to maximize the lifetime of

[2]A "cover" is a subset of nodes whose sensing regions cover the field.

the WSN. They proposed two centralized heuristic techniques for target coverage; one uses linear programming and the other is a greedy approach. Huang and Tseng [17] provided the necessary conditions for achieving k-coverage. Kumar *et al.* [18, 19] provided theoretical bounds on the number of nodes required to achieve k-coverage under different node deployment scenarios. The CCP protocol [32] probabilistically provides different degrees of coverage according to the application requirements. The OGDC protocol [36] computes the minimum required set of active nodes by reducing the overlap among deployed sensors. The treatment also gives the necessary conditions on the ratio between the sensing (R_s) and transmission (R_t) ranges that guarantees connectivity ($R_t \geq 2R_s$). Tian and Georganas [28] proposed a simple approach for selecting sensor covers that is based on checking the *sponsored area*, defined as the area covered by other working neighbors. Carle and Simplot-Ryl [7] proposed a technique for determining redundancy that is based on dominating sets. They extended a dominating set algorithm to elect nodes that cover the field instead of just covering their neighboring nodes. Nodes with higher residual energy than their peers are favored in joining the active set. Cao *et al.* [3] proposed a sleeping schedule for sensors that guarantees a bounded-delay sensor coverage. Yan *et al.* [33] proposed a protocol for collaborative sleep-and-wakeup among neighboring nodes. Gallais *et al.* [12] proposed several distributed protocols that account for different ratios between the sensing and communication ranges. Finally, Nazrul Alam and Haas [24] studied how to compute the sensing range to guarantee coverage and connectivity under three-dimensional deployments.

In the context of WANs, Melodia *et al.* [21] proposed selecting actors based on an auction-like mechanism. According to this mechanism, actors that bid to be part of the cover should be able to satisfy an application-specific delay bound. Vedantham *et al.* [30] studied how to select a cover that achieves the best *mutual exclusion* in terms of reducing the intersection among actors' regions of operation.

B Selecting Covers Under Variable Operational Ranges

The problem of sensing-range assignment has been recently addressed in a few studies. Cardei *et al.* [6] considered variable sensing ranges and proposed centralized and distributed heuristics for maximizing the number of set covers. Their approach assumes synchronized nodes, base station intervention, and knowledge of node positions. We do not assume any of these capabilities in our work. Although we propose a more general model than [6], we extend the approach in [6] and use it as a baseline for comparison (in addition to a centralized approach). Gupta *et al.* [15] proposed a greedy algorithm for selecting a connected cover to optimize query execution. In [37], the authors extended [15] to support variable sensing and transmission ranges and focused on maintaining connectivity and coverage. Our approach can be integrated with the one in [37] to maintain connected covers in location-unaware networks.

Our work belongs to this category of protocols. However, in contrast to previous works, we assume that every node may need to compute its local map prior to range assignment. We propose a novel protocol that assigns the operational ranges to the nodes according to their *weights*, where the weight of a node v is defined as the amount of resource(s) remaining in v (e.g., remaining battery or action material).

III PROBLEM STATEMENT

A System Model

We consider a WSN/WAN in which nodes are randomly deployed in the field according to some arbitrary distribution. Let R_t be the longest transmission range for a node. We refer to a node within distance $\leq R_t$ as a "neighbor." We assume the following:

- Nodes are quasi-stationary.
- Each node (sensor/actor) has k different operational levels $l_1, \ldots, l_k$, each defining an operational range $R_1, R_2, \ldots, R_k$, where $R_1 < R_2 < \cdots < R_k$.
- The function of a node relies on a non-renewable "resource" (e.g., battery energy) and its operation is resource-intensive. Resource depletion is proportional to R_i^m, where $1 \leq i \leq k$ and m is a constant ≥ 1.
- The radio component in a node employs a low duty cycle.
- A node has k operational levels, as described earlier. The operational ranges are not necessarily circular and may even be characterized probabilistically (how a node determines if a point within its operational range is covered will be discussed in Section V–A). Turning off a node corresponds to $R_0 = 0$.
- A node can estimate the distance between itself and a neighbor based on well-known ranging techniques, such as the time of flight or received signal strength [35]. For example, the Cricket sensor [10] uses ultrasound and RF beacons to estimate ranges based on time of flight. Experimental results show that range estimation using Cricket is reasonably accurate even in indoor environments [10]. Note that one-hop distance estimation is possible, even in harsh environments, though with less accuracy. We account for distance inaccuracy in designing our localization mechanism (Section IV) and in our simulation experiments (Section VI).

To fully exploit the redundancy of nodes in the network, we assume that $R_t \geq R_k$. Having $R_t < R_k$ may result in selecting operational ranges that are

longer than necessary, since a node may not be aware of the presence of some nodes that lie within its operational range.

B Design Objectives

Given a set of N nodes in the field, our goal is to allow every node i, $1 \leq i \leq N$, to assign to itself the minimum operational level l_j, where $0 \leq j \leq k$, such that the maximum operational region of i (determined by l_k) is covered by i and/or its neighbors. The assigned ranges should be periodically reassigned based on the resource availability to balance the load among nodes and to maximize the operational lifetime of every node.

The field boundary may be skewed and thus not known to individual nodes if they are not aware of their absolute coordinates. So, the objective of every node is to ensure that its maximum operational region is covered as long as possible. To determine how much of a node's operational region is covered, each node needs to be aware of the relative locations of its neighbors. If typical localization techniques cannot be used, a node uses the VICON algorithm to compute the virtual coordinates for its neighbors, as described in Section IV. The ORAP protocol, provided in Section V, executes VICON at every node and employs a distributed node collaboration technique to assign operational ranges to nodes (refer to Figure 8 for a flowchart that describes the range assignment process). Our ORAP protocol has the implicit objective of maximizing the coverage time of the network. Next, we discuss the details of VICON and ORAP.

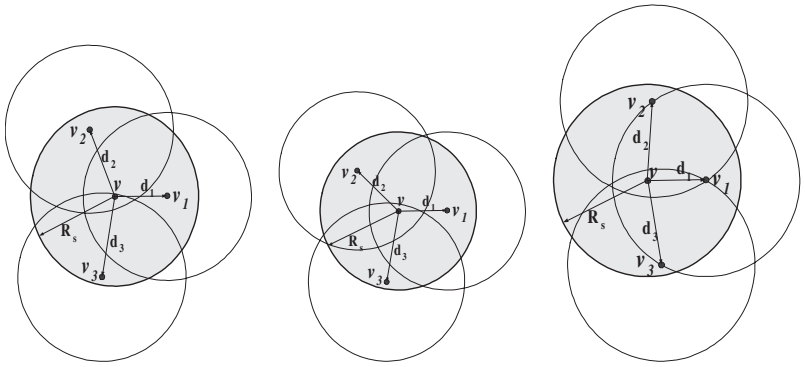
IV THE VICON ALGORITHM

Knowledge of neighbors' locations is essential for the appropriate adjustment of the operational ranges. In VICON, a node v computes "virtual" coordinates of its neighbors and uses these coordinates to determine its operational range. A virtual coordinate space (VCS) of node v's neighbors is one that *tries* to maintain the *connectivity profile* of the real coordinate space (RCS). That is, the VCS tries to maintain the relative distances and angles between every two neighbors of v, say u_1 and u_2. The VCS also tries to maintain the "non-neighborhood" property of u_1 and u_2 if they are not neighbors in the RCS. The VCS generated by VICON maintains the above properties under accurate distance estimates. VICON's basic triangulation approach resembles the one in [4]. However, VICON is novel in that it addresses some practical challenges not previously dealt with, as will be described shortly. During network operation, VICON can be triggered to recompute the VCS whenever better estimates of neighbor distances are provided using calibration techniques, such as the ones in [31] and [9].

The problem of assigning node coordinates is a special instance of the general "graph embedding" problem in the 2D space, which has been extensively

studied in the literatures of graph theory and computational geometry [13]. Computing virtual coordinates was also studied in the networking literature (e.g., [2, 14, 16, 20, 23, 26, 27]). In these studies, the objective was to assign coordinates to all the nodes in the network. This is typically achieved by exploiting connectivity information, hop counts, estimated distances between nodes, and/or known anchor node positions. Such approaches are not applicable to our setup for several reasons. First, we are not concerned with accurate node positioning as in previous research works, since our primary objective is determining coverage. We are rather interested in identifying what combinations of neighboring nodes can cover the region covered by a given node v. Second, we consider several practical challenges that were not studied in previous positioning techniques, namely, localizing partitioned neighbor graphs and using inconsistent connectivity information. In addition, our model does not assume the presence of anchor nodes in the network. Finally, we only require each node to compute the relative coordinates of its neighbors, and do not require network-wide node positioning.

Although some multitriangulation and multilateration localization approaches (e.g., [16, 22]) have previously studied localization in the presence of inaccurate distance estimates, they may not work if the connectivity information is not sufficient. For example, consider the topology in Figure 2(a), where node v is computing coordinates for its neighbors v_1, v_2, and v_3. For simplicity, assume that that the communication range is equal to the sensing range R_s. The figure shows that the three neighbors do not cover the sensing range of node v. Multitriangulation (e.g., [16]) and multilateration techniques (e.g., [22]) cannot compute coordinates for nodes in that topology due to lack of connectivity. Multidimensional scaling (MDS) techniques (e.g., [14,20,27]) can be applied. However, there is no unique solution to localization in this topology. A slight rotation in v_2 will not affect the connectivity in the topology



(a) Original topology (b) Possible neighbor placement (c) Neighbor placement by VICON

FIGURE 2
Illustration of the uniqueness and conservativeness of VICON.

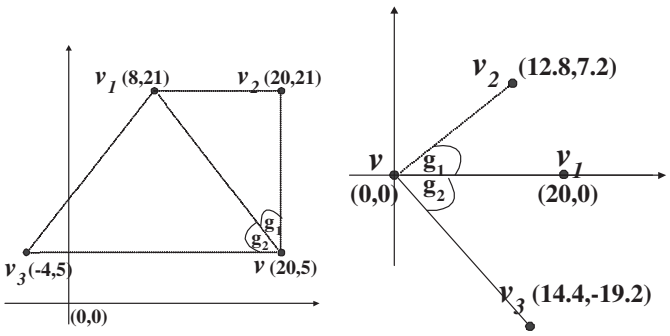
and will also preserve the distances between v and its neighbors, as shown in Figure 2(b). An MDS-based localization technique may erroneously accept the node placement shown in the figure in which the neighbors cover the sensing range of v. On the other hand, coordinates need to be assigned conservatively (as shown in Figure 2(c)) to preserve the state that node v's range is not covered.

We propose a localized, lightweight localization algorithm that relies on ranging (distance estimation) to be able to determine how a node is covered[3]. The algorithm can be easily employed in dynamic scenarios where nodes are deployed at different times. It can also be used in mobile networks if the speed of the nodes allow the computed virtual coordinates to remain usable for a reasonable duration that is specified by the application. VICON does not preserve the absolute directionality of the neighbors, which is not a problem since our objective is to determine "how much area is uncovered," and not "which area." We first describe the basic operation of the VICON algorithm assuming accurate estimates of distances. Then, we extend it to handle the challenges mentioned above.

A Basic Triangulation Algorithm

Before we delve into the details of VICON, we first summarize its assumptions and basic approach. Prior to executing VICON, each node v is aware of its 2-hop connectivity information (reachability and distances). This is done by having every node locally broadcast the distances between itself and its 1-hop neighbors. This information is later used for triangulation of neighbor coordinates. Assume that v has a list of N_{nbr} neighbors. Node v executes VICON as follows. Without loss of generality, assume that v has three neighbors v_1, v_2, and v_3, as depicted in Figure 3(a) (finding virtual coordinates for more than three nodes is done similarly). Node v assumes that it is positioned at the origin. It places its first neighbor (v_1) at (d_1,0), where d_1 is the distance between v and v_1 (see Figure 3(b)). From the distance information that is broadcast by neighbor v_1, v is aware of the distance between v_1 and v_2 ($\|v_1, v_2\|$). Using $\|v_1, v_2\|$, $\|v, v_1\|$, and $\|v, v_2\|$, v can compute the angle g_1 shown in Figure 3(a). To determine the virtual coordinates of v_2, v_2 is rotated by an angle g_1 from the origin in the counter-clockwise direction. Similarly, angle g_2 is computed using the distances $\|v_1, v_3\|$, $\|v, v_1\|$, and $\|v, v_3\|$. To place v_3 in the VCS, v_3 is rotated in the counter-clockwise direction with an angle g_2 and is assigned a tentative coordinate. The validity of this coordinate is then tested against all the already placed nodes to determine whether the original connectivity is preserved. In this example, rotating v_3 in the counter-clockwise direction causes it to be a neighbor of v_2, which is not the case according to the RCS. Therefore, v_3 is rotated by an angle g_2 in the clockwise direction.

[3] Several range-free positioning techniques, e.g., [2, 16], were proposed to compute coordinates based on anchor node positions.

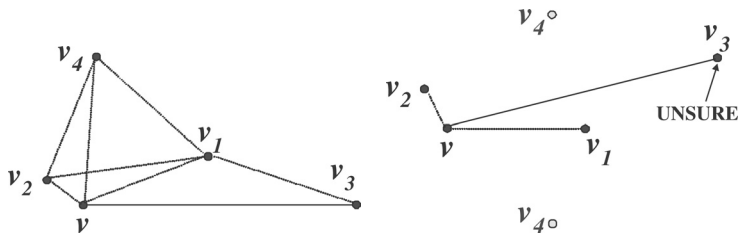


(a) RCS (b) VCS

FIGURE 3
Executing VICON at a node v to determine the virtual coordinates of v's neighbors.

Figure 3(b) illustrates that v is still covered by three nodes that are within $g_1 + g_2$ total angle, and are all on one side of v. Although v_1 was used as the "reference" node for v_2 and v_3, a new node v_4 to be placed in the virtual coordinate space may not necessarily be a neighbor of v_1. In this case, v searches for a neighbor of v_4 among the already-placed nodes. Node v_4 is then placed according to its reference while preserving the connectivity of the original configuration.

One complication that may occur during neighbor placement is that a node may satisfy the connectivity requirement with respect to the already placed neighbors, regardless of the placement of this node in the clockwise or counter-clockwise direction of its reference. This may result in subsequent failures to place neighbors in the VCS because no direction of rotation (clockwise or counter-clockwise) can maintain the connectivity as in the RCS. The problem is demonstrated in Figure 4(a), where v_3 is a neighbor of v_1 but not of v_2 or v_4. In the VCS at v, v_1 and v_2 are placed first. Node v_3 can be placed in either direction from v_1. Suppose that v_3 is placed the counter-clockwise



(a) RCS (b) VCS

FIGURE 4
Failure to compute the virtual coordinates of neighbor v_4 due to incorrect placement of v_3.

direction from v_1 (as shown in Figure 4(b)). When v_4 is to be placed, it has to go in the counter-clockwise direction to satisfy its connectivity with v_1 and v_2. However, this will make v_4 a neighbor of v_3. Therefore, v fails to determine a virtual coordinate for v_4. This problem arises because of the order by which neighbors are added to the VCS, and also due to the non-planar nature of the graph.

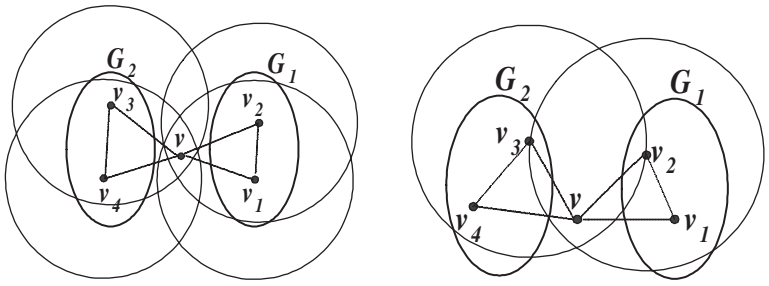
The aforementioned problem can be addressed using the following approach. Node v processes its neighbors in sequence and pushes the IDs of successfully placed neighbors in a stack named *FinishedNbr*. A neighbor that can be successfully placed in two positions is marked "UNSURE" in *FinishedNbr*, whereas a neighbor that can only be placed in one position is marked "SURE." If v fails to compute coordinates for a neighbor i, it pulls out neighbor IDs from *FinishedNbr* until it finds one that is marked UNSURE. This neighbor is then placed in the alternative direction, is marked SURE, and is pushed back in the stack. VICON then attempts to re-process the pushed-out neighbors. This approach ensures that incorrectly selected coordinates are corrected as more neighbors are placed. In our example in Fig. 4(b), v_3 is marked UNSURE when placed. When v fails to place v_4, it pops v_3 from *FinishedNbr*, places it in clockwise direction relative to v_1, and successfully places v_4.

B Placement Challenges

Application of the basic triangulation approach requires ideal conditions, in terms of connected neighborhood graphs, accurate distance estimation, and consistent distance/connectivity information. The lack of such conditions poses challenges on the process of computing virtual neighbor coordinates. Below, we discuss these challenges and propose extensions to the basic triangulation mechanism to cope with them.

1 Disconnected Neighbor Graphs

This problem is depicted in Figure 5(a), where v's neighbors form more than one connected component $G_1 = \{v_1, v_2\}$ and $G_2 = \{v_3, v_4\}$. This results in having a subset of the neighbors unable to find reference nodes that are already placed in the VCS. VICON handles this problem as follows. First, v's neighbors are divided into groups, where each group represents a connected component. This is done by exploiting the advertised connectivity information of the neighbors. In Figure 5(a), two neighbor groups are available: $\{v_1, v_2\}$ and $\{v_3, v_4\}$. Second, the coordinates of the neighbors in each group are computed independently from the other groups. Finally, each group, other than the first one, is rotated to preserve connectivity as in the original configuration. This is depicted in Figure 5(b) where the two groups are placed closest to each other while preserving their disjointedness. Setting the neighbor groups closer to each other in the VCS may result in reducing the portion of v's region that



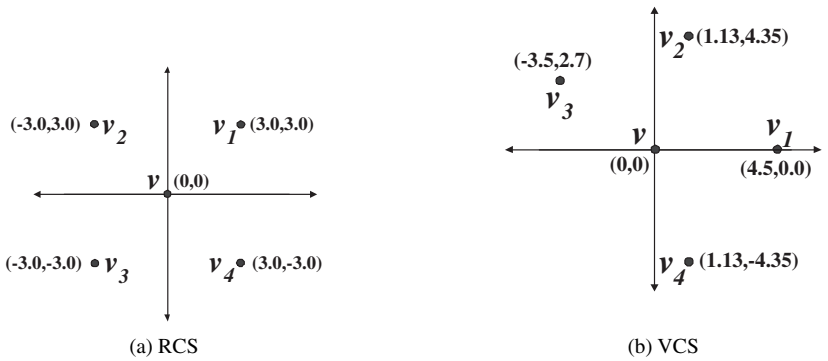
(a) RCS (b) VCS

FIGURE 5
Assigning coordinates to disjoint neighbor groups.

is covered by its neighbors, and consequently setting a more conservative operational range for v.

2 Inaccurate Distance Estimation

Because distance estimation is based on measurements, inaccuracies might be introduced. In the basic triangulation algorithm, inaccuracies in distance estimation may result in failing to place some neighbors. We demonstrate this problem in Figure 6(a). Let $R_t = 10$ and let the maximum error in the estimated distances be $\Delta_{\max} = 10\%$. With estimated distances $\|v, v_1\| = \|v, v_2\| = 5$ and $\|v_1, v_2\| = 6$, the angle $\angle v_1 v v_2 \simeq 74°$, and v will fail to place the subsequent neighbors v_3 and v_4. Figure 6(b) shows the corresponding VCS. All nodes are successfully placed in this configuration without losing the coverage properties. However, under high densities, the basic triangulation algorithm does not ensure that all the neighbors will be placed due to the accumulation of distance errors. In addition, the effect of inaccurate distance estimation on the



(a) RCS (b) VCS

FIGURE 6
Placement of neighbors under inaccurate distances ($R_t = 10$, $\Delta_{\max} = 10\%$).

coverage of node v is most apparent when the distances vv_i $(1 \leq i \leq N_{nbr})$ are underestimated.

We propose the following remedy to the problem of inaccurate distance estimation. Assume that the maximum distance estimation error is $\Delta_{\max}$. To be conservative, v adds $\Delta_{\max}/2$ to all of its estimated neighbor distances before executing VICON. On average, this results in having v overestimate the distance 75% of the time. We conducted some numerical experiments to study the effect of this approach on coverage uncertainty (erroneously deciding that a range is covered)[4]. In these experiments, we assumed circular operational ranges. Our experiments revealed the following:

- When v's neighbors do not cover the operational range of v, v will decide to use its maximum operational range. Therefore, this case has no effect on the coverage of v's range.
- When the neighbors cover v's operational range and the error in distance estimation Δ $(\Delta \leq \Delta_{\max})$ is random, the observed coverage uncertainty is negligible. This typically holds when the number of neighbors, N_{nbr}, is *reasonably* large (e.g., >6).

To use conservative distance estimates for neighbor placement, we first compute tentative positions for a neighbor u in the clockwise and counterclockwise directions. Each tentative position of u results in a new computed distance to each of the already placed neighbors in the VCS. For each of the tentative positions, we compute the *position uncertainty* for neighbor u $(PU(u))$, defined as the sum of distance errors between u and the already placed neighbors. Specifically, $PU(u) = \sum_{i=1}^{k} |\hat{d}_i - d_i|$, where k is the number of neighbors placed so far in the VCS, $\hat{d}_i$ is the new distance between u and a placed neighbor i in the VCS, and d_i is the original distance estimate between u and i (computed using time of flight or signal strength, as mentioned in Section III). We then select the position that results in a smaller $PU(u)$. Note that the selected position may result in a VCS that is does not preserve all the relative directions of neighbors as in the RCS. Moreover, some non-neighbor nodes may become neighbors in the VCS. This, however, does not cause any coverage violations due to the overconservative distance estimation.

3 Inconsistent Connectivity Information

Another source of inaccuracy is inconsistent connectivity information. This may be caused, for example, by link symmetry. For instance, node v may have two neighbors v_1 and v_2 that are themselves neighbors in the network graph. However, due to link asymmetry, v_1 may broadcast its table without including v_2.

VICON handles inconsistency in connectivity information as follows. Without loss of generality, consider the same example above and assume that

[4]We study coverage uncertainty due to inaccuracies in the operational range in Section V–A.

v_1 has already declared v_2 as a neighbor while v_2 has not. In this case, v relies on v_1's unilateral declaration and assumes that v_1 and v_2 are neighbors. This will not cause any problem for the coverage computations, which only rely on estimated distances. In fact, by assuming that v_1 and v_2 are neighbors, the uncertainty in the placement of these neighbors is reduced. Another scenario that might occur is if both v_1 and v_2 do not declare each other as neighbors. This case does not require any action from v, and the placement of v_1 and v_2 can proceed normally as in the original VICON algorithm.

C Pseudo-code of VICON

Figure 7 provides a pseudo-code for the VICON algorithm. The function *ComputeNeighborGroups(v)* is used to determine the disjoint neighbor groups, whereas *RotateNbrGroups(v)* is used to place these groups closest to each other (as described earlier). The array NodesInGroup[] carries the number of nodes in every neighbor group. The function *CanTriangulate(u_i)* determines if neighbor u_i can be added to the VCS based on the already placed neighbors

```
VICON(v)
1. nGroups = ComputeNeighborGroups(v)
2. FOR k=1 to nGroups
2.1. u = Reference neighbor of group k
2.2. u.VirtualCoordinate = (||u, v||,0)
2.3. index = (k-1) · NodesInGroup[k]
2.3. Push u in FinishedNbr stack
2.4. Push 0 in FinishedAngle stack
2.5. update = TRUE, nFinished = 1
2.6. LastUnsureNbr = -1
2.7. WHILE (update = TRUE)
2.7.1. WHILE (update=TRUE and nFinished<NodesInGroup[k])
2.7.1.1. update = FALSE
2.7.1.2. FOR every u_i ∈ neighbors(v)
2.7.1.2.1. IF CanTriangulate(u_i)
2.7.1.2.1.1. Push u_i in FinishedNbr
2.7.1.2.1.2. Push angle of u_i in FinishedNbrAngle
2.7.1.2.1.3. IF (DualPosition(u_i)) Mark u_i as UNSURE
2.7.1.2.1.4. nFinished++, update = TRUE
2.7.2. update = FALSE
2.7.3. IF (nFinished < NodesInGroup[k])
2.7.3.1. Pop nodes until the last node u_i marked UNSURE
2.7.3.2. Place u_i in its alternative position
2.7.3.3. Mark u_i as SURE
2.7.3.4. Reduce nFinished accordingly
2.8. WHILE (nFinished<NodesInGroup[k])
2.8.1. Place neighbor u by minimizing PU(u)
2.8.2. increment nFinished
3. RotateNeighborGroups(v)
```

FIGURE 7
Pseudo-code for the VICON algorithm when executed at node v.

(as described above). Finally, *DualPosition(u_i)* determines if neighbor u_i can be successfully placed in more than one position.

D Properties of VICON

1 Correctness

Proposition 1. *Assuming accurate distance estimation, when VICON terminates, all the neighbors of a node v are assigned coordinates that preserve their original connectivity profile.*

Proof. Without loss of generality, assume that the neighbors of v form one connected component and only one node v_i ($i \leq nFinished$) is marked UNSURE in *FinishedNbr*. We prove the lemma by contradiction. Assume that VICON terminates and a neighbor v_j is assigned a wrong coordinate or that v fails to place v_j. If v_j is assigned a wrong coordinate, this assignment will conflict with at least one of v_j's neighbors in the VCS, triggering VICON to subsequently place v_j in an alternative position. Suppose that v fails to place v_j. This may occur because of two reasons: (1) no already placed neighbor is in the neighborhood list of v_j, or (2) v_i was placed in the wrong direction of its reference neighbor. In the former case, computing the coordinates for v_j will be postponed until one of its neighbors is placed in the VCS. Failing to find a coordinate for v_j until VICON terminates means that it will not have any neighbors in *FinishedNbr*. This is impossible since v's neighbors form one connected component. In the latter case, VICON uses a methodology to fix the position of v_i, as described before. Therefore, in both cases v will eventually be able to assign v_j a virtual coordinate that fits with its reference neighbors. This argument applies to all neighbors of v. □

Now, assume that the neighbors of v form multiple disjoint groups. The above proof applies to each individual component. The placement of these components in the VCS ensures that they remain non-neighbors (as described before). Therefore, the connectivity profile is also preserved in the presence of multiple neighbor groups. □

2 Complexity

VICON requires that every node broadcasts its neighborhood table. The number of messages needed for such a broadcast is constant. Each message requires $O(N_{nbr})$ bits. For every neighbor, its unique identifier (ID) and distance from v should be stored in addition to its neighborhood table. Assume that the ID requires 2 bytes and the maximum transmission range is ≤ 200 meters, so that the value of the neighbor distance can be encoded using one byte. One more byte will be needed for the neighbor's state, described in Section V. Therefore, v needs to store 4 bytes for every neighbor u in addition to u's neighborhood table. On average, v will need to store a total of $4N_{nbr}(4N_{nbr}+1)$ bytes, which is insignificant for typical values of N_{nbr} (e.g., ≤ 20).

VICON's worst-case complexity is $O(N_{nbr}^2)$ ($\leq \alpha N_{nbr}^2$, where α is a constant). Because there can be at most five non-neighboring neighbors of a node v [34] and each of these non-neighboring neighbors can be successfully assigned to at most two positions, α is at most 2^5. Therefore, VICON does not introduce significant computational complexity.

V ORAP PROTOCOL

In this section, we introduce the Operational Range Assignment Protocol (ORAP) for location-unaware sensor/actor networks. ORAP relies on VICON or a similar approach for estimating distances. It can also be applied if approximate location information is available. Before introducing ORAP, we show how a node determines the coverage of a point within its operational region.

A Coverage Determination

Every node that executes ORAP needs to determine whether or not the points in its maximum coverage range are covered (assuming that the field area is discretized into a number of points). As mentioned in Section III, the operational ranges of a node v are not necessarily circular. Assume that a node v is using an operational range R, where $R_1 \leq R \leq R_k$. Also assume that a point pt that lies at a distance D_0 from v is covered with probability $P_{v,R}(pt)$, which depends on R and D_0. If $D_0 > R$ then $P_{v,R}(pt) = 0$. Otherwise, $P_{v,R}(pt)$ depends on v's probability distribution function for point coverage, $f_v(r)$, that is specified by v's manufacturer. $f_v(r)$ is a continuous distribution function that decreases as the distance from v increases. To compute whether or not pt is covered by node v, we consider a small distance δ around pt and slightly modify the above definition of $P_{v,R}(pt)$ to be:

$$P_{v,R}(pt) = \begin{cases} \int_{D_0-\delta/2}^{D_0+\delta/2} f_v(r)dr, & D_0 \leq R \\ 0, & \text{Otherwise,} \end{cases}$$

where δ is the distance between two adjacent points in the field. Node v decides that pt is covered if the probability that pt is covered by v or any neighbor j of v exceeds a threshold γ, i.e.,

$$\left(\left[1 - (1 - P_{v,R}(pt)) \prod_{j \in Nbrs(v)} (1 - P_{j,R_j}(pt)) \right] \stackrel{\text{def}}{=} P_{cov}(pt) \right) \geq \gamma, \quad (1)$$

where $Nbrs(v)$ is the set of v's neighbors and R_j is the current operational range for neighbor j.

B Design Rationale and Overview

The goal of ORAP is to assign operational ranges to nodes such that: (1) the field area is covered with probability $\geq \gamma$, and (2) the operational lifetime of every individual node is prolonged as much as possible. To achieve the first objective, every node v ensures that every point pt in the field that lies in v's maximum operational range is covered with probability $P_{cov}(pt) \geq \gamma$. To achieve the second objective, two measures are taken. First, ORAP assigns longer ranges to nodes with higher "weights," where a dynamic parameter is used to represent the weight of a node (e.g., remaining critical resource). Furthermore, ORAP is re-triggered at fixed intervals of time, referred to as the *cover update interval* t_{cu}. This ensures that new ranges are assigned every t_{cu} interval, resulting in efficient load balancing among nodes. It was shown in [1] that favoring high-weight nodes in cover selection and refreshing the active covers at small-time granularity can asymptotically lead to optimal network lifetime. The optimal cover refreshment rate is still an open issue that is beyond the scope of this paper.

The ORAP protocol is executed at every node in the network without the intervention of any centralized entities. We assume that a node can be in one of two states: DECIDED or UNDECIDED. All the nodes start in the UNDECIDED state. We consider two possibilities for assigning operational ranges. One possibility is to let neighboring nodes negotiate reducing their ranges in several steps until they decide on their final operational ranges. Despite its obvious efficiency in terms of minimizing the operational ranges, this approach requires a significant number of message exchanges in the network and is highly sensitive to packet losses. Another possibility is to let nodes decide in the order of their weights such that every node uses information about already assigned ranges to select its range. We adopt this design option due to its simplicity and robustness, although it may achieve less efficient range assignment than the first design option (the first design option will be explored further in a centralized approach that we use in our simulations in Section VI).

Because node clocks are typically unsynchronized and ORAP is typically timer-triggered, the node with the fastest clock in its 1-hop neighborhood sends a message to its neighbors to trigger the execution of ORAP. Consequently, every node that receives this message sends a similar triggering message prior to executing ORAP. The message initiation process synchronizes the start of ORAP execution among nodes within a given locality, which is sufficient for our protocol. ORAP can be easily incorporated in real systems such as TinyOS [29], and can be used on nodes such as Cricket [10].

It is worth noting that ORAP uses every neighbor's p.d.f. (Section V–A) to compute if the neighbor covers a particular point. However, if the sensing range is completely arbitrary and cannot be modeled, then every neighbor needs to approximate its sensing range to the largest possible circle and advertise its radius. If node coordinates are known and the sensing range is arbitrary, then every node may choose to advertise the set of points that it covers. This is not

useful, however, in WSNs, in which the nodes have limited memory that will typically not permit them to store all the points that their neighbors cover.

C Protocol Details

We describe the operation of ORAP in the context of an arbitrary node v. ORAP has three phases: Phase I is for initialization, Phase II is the core operation of ORAP in which v decides on an operational range R, and Phase III is for the optimization of R. A summary of the three phases of ORAP is shown in Figure 8.

1 Phase I

In this phase, v computes a real-valued weight $wgt(v)$, as follows:

$$wgt(v) = \frac{E(v)}{E_{\max}} \tag{2}$$

where $E(v)$ is the remaining energy in v's battery and $E_{\max}$ is the maximum battery capacity. A neighbor discovery process is then initiated in which v broadcasts $wgt(v)$. Based on the replies that v receives, it broadcasts its neighborhood table (which includes the estimated distances). Neighbor discovery is not necessary if nodes periodically exchange heartbeat messages for discovering routes and maintaining connectivity to the observer. In the

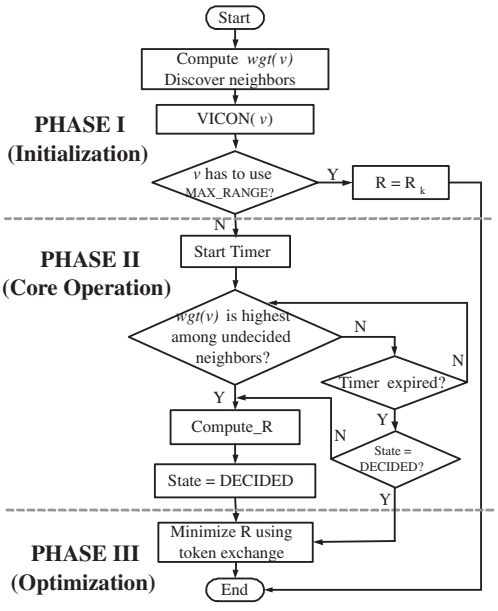


FIGURE 8
The ORAP protocol executed at node v.

second step of Phase I, v executes VICON to compute the virtual coordinates of its neighbors. Note that this step is independent of ORAP, since ORAP can also use physical neighbor coordinates, if available. The final step is to check whether v *has to use* its maximum operational range R_k or not. This is determined by having v assume that all its neighbors are using R_k and checking if any part of v's operational region is not covered. Nodes residing on the periphery of the network (without knowing so) will always find themselves using R_k since each one will have at least one point in its operational range that is not covered by any neighbor. Thus, these nodes will deplete their critical resource faster than those inside the field. If v passes this test, it quits ORAP and sets R to R_k. Otherwise, v executes Phase II.

2 Phase II

In this phase, v computes its operational range based on its weight and the weights of its neighbors. Node v does not make a decision regarding its working range R unless it has the highest weight among all of its undecided neighbors. This gives the chance for nodes with higher weights to decide first and choose longer ranges. At the same time, v sets a timer (say 5 seconds) for this phase. The "seconds" granularity of the timer is chosen to enable v receive messages from all its neighbors. If the timer expires before a decision is made, v computes its range assuming that all undecided neighbors use R_0. The purpose of the time is to avoid indefinite waiting for neighbors with higher weights that may have failed or have their update messages lost. For example, if v has five neighbors $u_1, \ldots, u_5$, where u_4 and u_5 are not decided, then v ignores u_4 and u_5 and considers the remaining neighbors only in its range computation (explained below).

As mentioned in Section III, we assume that the energy consumption of the sensing or acting stage is proportional to R^m, where m is a constant ≥ 1. The function *Compute_R* proceeds as follows. Node v first sets its range R to R_{k-1} and sets the range of every undecided neighbor u to the largest R_j that is smaller than $[(wgt(u)/wgt(v))^{1/m} \times R]$, where $j \leq k-1$. Note that $wgt(u)/wgt(v) < 1$, which means that v's undecided neighbors are assumed to use operational ranges $\leq R_{k-1}$. Decided neighbors are set to the ranges that they have decided on. If this assignment covers the operational region of v, v sets its range R to R_{k-2} and the same process is repeated. If range R_i, $0 \leq i < k$, fails to ensure complete coverage of v's region, then v decides to use R_{i+1} and changes its state to DECIDED. The selected R is advertised to v's neighbors. Note that a good feature in ORAP is the flexibility of the function *Compute_R* that can be implemented in different ways to support different coverage objectives. Pseudo-code for range computation is given in Fig. 9.

3 Phase III

After Phase II, v can terminate ORAP and use its selected R. However, redundancies may have been introduced due to the order of the decision-making

```
Compute _R(v)
1. R = R_{k-1}, i = 0, found = FALSE
2. While (R > R_0 and not found)
2.    For i = 1 to N_nbr
3.       If (nbr(i) is not decided)
4.          R' = max_{R_j<R} {R_j : R_j = [(wgt(u)/wgt(v))^{1/m} × R]}
5.          Set range of nbr(i) to R'
6.    If range of v is not covered
7.       found = TRUE
8.    Else i = i + 1, R = R_{k-i-1}
9. If (found)
10.   R = R_{k-i}
9. return R
```

FIGURE 9
Pseudo-code for the operational range computation algorithm executed at node v that has N_{nbr} neighbors.

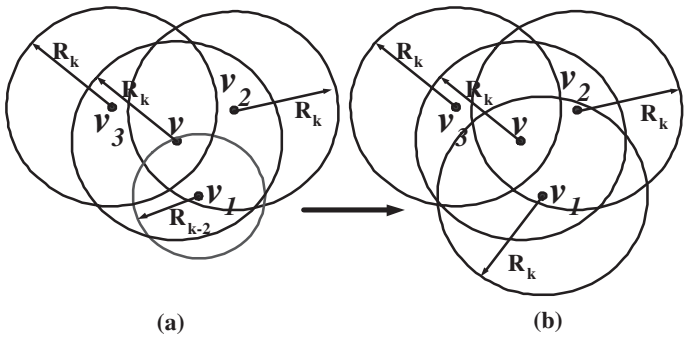


FIGURE 10
Example of redundancy resulting from the sequence of execution in Phase II. (a) Node v selects R_k assuming that v_1 will select R_{k-2}. (b) Node v_1 selects R_k, which makes v redundant.

process in Phase II. This problem is present in all previous approaches that use a bottom-up approach in selecting covers, i.e., start from an empty cover and gradually add nodes. An example is illustrated in Figure 10, where v_2 and v_3 are decided, v_1 is undecided, and $wgt(v) > wgt(v_1)$. Initially, v assumes that v_1 will use R_{k-2} and thus v uses R_k to cover its entire region. When v_1 gets its turn, it selects the range R_k, which is larger than what was assumed by v. This makes v completely covered. Although v may easily detect this redundancy, it can not reduce its range without coordinating with its neighbors. Therefore, we propose the following iterative approach for removing redundancies.

When the node that has the least weight in its neighborhood selects its operational range, it sends a token to its neighbors, allowing them to proceed to Phase III. A node v that receives such a token starts another timer (around one second). Node v has to wait to receive tokens from all the neighbors with

less weights than its own. Once these tokens are available, v computes its final operational range and releases a token that advertises the new range of v. If the timer expires before v has received enough tokens, v keeps its range as computed in Phase II and releases its token.

D Analysis of ORAP

We analyze the ORAP protocol in terms of its correctness and overhead. For the purpose of this analysis, we assume knowledge of node locations and circular sensing ranges.

Proposition 2. *Once ORAP terminates, the operational region of every node with non-depleted resource is covered with probability* $\geq \gamma$.

Proof. When ORAP is executed at a node v, two operations affect the final coverage of v's operational region:

1. Selection of v's operational range in Phases I and II. In Phase I, if v determines that its operational region can not be completely covered by its neighbors, it sets its operational range to R_k and terminates ORAP. This results in having v's region completely covered. If v goes through Phase II, it selects its operational range based on both *real* ranges of its decided neighbors and *hypothetical* ranges of its undecided neighbors. An undecided neighbor will not be able to select an operational range that is less than the largest hypothetical range made by any of its neighbors, unless its region is completely covered. This also ensures that v's operational region is covered with the target probability.

2. Reduction of operational ranges in Phase III. A hole in field coverage may occur when two neighboring nodes (e.g., v_1 and v_2) are allowed to reduce their operational ranges simultaneously. Such scenario implies that both v_1 and v_2 had all the tokens they needed from the neighbors with less weights to start Phase III simultaneously. This is not possible since we assume that the weight is a real number and the likelihood that v_1 and v_2 will have the same weight is infinitesimally small. □

Proposition 3. *Every node v selects the minimum operational range that covers its operational region, provided that accurate distances are used and the neighbors of v do not form multiple disjoint components.*

Proof. Assume that v has selected R_i as its range, although R_j $(j < i)$ was sufficient to have v's region covered. This may occur in Phase II depending on the order of ORAP execution among v's neighboring nodes. However, when Phase III is executed, v will be able to compute the minimum range R_j based on its neighbors' selections. Because nodes that receive tokens after v are only allowed to *reduce* their operational ranges, R_j is final and minimal. Consequently the entire cover selected by ORAP is minimal. □

Note that if inaccurate distances are used to obtain the virtual coordinates, minimality may not be achieved. The redundancy introduced in this case depends on the density and distribution of nodes in the field. In Section VI, we study the minimality of the selected covers when using both accurate and inaccurate distances.

Message overhead. In typical multi-hop wireless networks, periodic updates of routing information are used to maintain connectivity to the observer. By piggybacking the control information of ORAP on these route-update messages, ORAP's control overhead can be carried at a free cost. Even in the absence of routing updates, ORAP has low message overhead, which is computed as follows. Four types of message exchange are required: (1) neighbor discovery, which requires two messages per node, (2) advertisement of a node's weight, which requires only one message whenever ORAP is re-triggered, (3) advertisement of the selected range, which requires one message, and (4) token exchange, which requires one message. Therefore, the total number of messages per node is five.

Note that if more parameters are added to the weight computation (e.g., mobility, remaining uncovered area, etc.), then a node v's weight has to be advertised whenever one of v's neighbors decides to recompute its range. This raises the message overhead to $O(N_{nbr})$. Phase III will have to be modified in this case, as it relies on a parameter that is assumed to be fixed during the range assignment process.

Time complexity. The time complexity of ORAP has two components: (1) convergence speed, and (2) processing complexity. Ignoring the protocol timers, Phase I has $O(1)$ convergence speed. The average-case convergence speed is proportional to the average number of neighbors of any node. The worst-case convergence speed for Phases II and III is proportional to the number of nodes[5]. This justifies the use of timers to limit the convergence time and avoid indefinite waiting in case of failures. Note that node density is not the primary factor affecting the convergence speed. The more significant factor is the distribution of weights across the network. The ORAP protocol distributes the critical resource consumption among all the nodes in the network, resulting in much faster convergence speed than the worst-case described above.

The main processing complexity in ORAP is in testing whether the node's operational region is covered. This test is performed once in Phases I and III, and then every time a neighbor selects its range in Phase II. If we discretize the operational region into a fixed number of P points, then the complexity of the test is $O(PN_{nbr})$.

[5] In our experiments, we observed that the convergence speed of ORAP is significantly less than the worst case.

VI PERFORMANCE EVALUATION

We evaluate our proposed protocol via simulation in the context of a WSN application (an equivalent scenario can be defined for WANs). We developed a discrete event-driven simulator for this purpose, which is scalable and efficient for large-scale networks. We study the operation of ORAP for sensing range assignment under accurate distance measurements (which we refer to as "SRAP-A") and under inaccurate distances (which we refer to as "SRAP-D"). In the studied scenario, a number of nodes send their measurements to an observer via multi-hop routes. We assume that nodes are uniformly distributed and that an observer is in the middle of the field. All nodes start with full batteries. The network is considered dead when the observer cannot receive any more reports from the sensors. The simulation parameters used in our experiments are shown in Table 1. In this table, t_{cu} is the duration after which SRAP-A or SRAP-D is re-triggered to reassign operational ranges. The parameter m specifies the rate of energy consumption, as defined in Section III. For radio communications, we adopt a simple energy consumption model in which a fixed amount of power is consumed from every active node during its operation. This is inspired by real systems, such as Berkeley Motes [10]. Since we assume that node operation is quite expensive, we set the energy consumed in communication to correspond to the energy consumed at the least operational range (R_1). We use a simple PDF (probability density function) for coverage: $f_v(r) = 1$ if $r \leq R_i$, and 0 otherwise, where R_i is the sensing range that correspond to operational level i. We also set γ to 0.99.

We compare our protocol to a centralized greedy algorithm (which we refer to as "CentralizedApp") [1]. In this algorithm, a centralized entity that is aware of the locations of all the nodes in the network is responsible for range assignment. The network operation is divided into phases of equal duration.

Number of nodes (N)	900
Field location	From (0,0) to (50,50)
Location of observer	(25,25)
Communication range (R_t)	5 meters
n_d (discretized distances)	5
Number of operational levels (k)	4
Maximum operational Range (R_k)	5 meters
Maximum battery capacity	1.0 Joule
Communication energy (E_{comm})	10^{-6} Watt
Energy consumption parameter (m)	2
Cover update interval (t_{cu})	2000 seconds

TABLE 1
Simulation parameters

Given the energy spent by each node at the end of phase i, the minimal cover for phase $i + 1$ is chosen such that the maximum energy spent by a node at the end of phase $i + 1$ is minimized. The algorithm selects a minimal cover as follows. All the nodes are assumed to employ the maximum operational range for phase $i + 1$. The nodes are arranged in the descending order of the expected energy spent at the end of phase $i + 1$. The algorithm selects the node with the highest value (say v). If reducing v's range by one step (i.e., from R_j to R_{j-1}, $0 < j \leq k$) violates coverage, then v's operational range is kept at R_j. Otherwise, v's operational range is reduced to R_{j-1}. The expected energy spent at the end of phase $i + 1$ is updated, as well as the ordered set of nodes. The procedure is repeated until a minimal cover is obtained. As the nodes reduce their range one step at a time, the worst-case running time of the algorithm is $O(Nk)$. The difference between this algorithm and ours is that the former is centralized and assumes precise knowledge of the location of all the nodes while the latter is distributed and location-unaware. The centralized execution of CentralizedApp enables it to detect coverage of a point in the range of a node v by *any* node in the network, and not only by the neighbors of v as in any localized protocol.

We also compare ORAP to the AR-SC approach proposed in [6], which is the earliest range assignment protocol in sensor networks. AR-SC is a distributed protocol proposed for target coverage. However, we extend it to area coverage by discretizing the field into a large number of points. AR-SC gives priority in decision-making (range assignment) to nodes seeing more uncovered targets. We assume ideal conditions for the operation of AR-SC, which include full node synchronization, optimal sequence of decision-making according to node priorities, and knowledge of the exact node coordinates through GPS. This algorithm is a good baseline for comparison because it is also distributed and uses a typical greedy range-assignment mechanism.

For a fair comparison, we assume that the network boundary is not known by the application employing any of the compared techniques. Our simulator ignores packet losses at the MAC layer since it may only add some redundancies to field coverage but have no impact on the operation of the compared protocols. We first study the properties of VICON and ORAP by considering one selected cover. Then, we study the operation of the network and compute the coverage time. The results provided below are the average of several experiments with different random topologies.

A General Properties

In this section, we focus on the selection of one cover by any coverage algorithm. All the nodes are assumed to be alive. We compute the number of nodes selected at the k different operational levels (in addition to $R_0 = 0$), the energy cost of the selected cover, and the ratio of successfully placed neighbors per node for SRAP-D. In Figure 11, we demonstrate the number of nodes at each

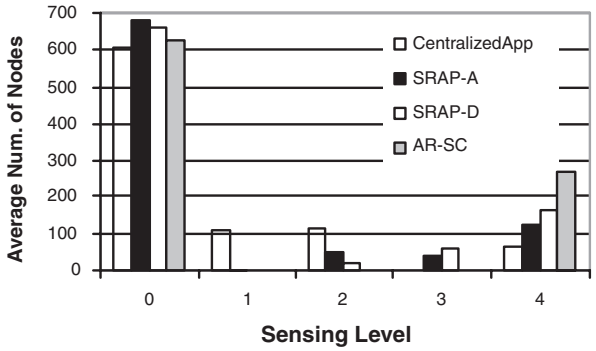


FIGURE 11
Average number of nodes at each operational level.

operational level for different node densities. One observation that can be made from the figure is that the AR-SC approach assigns ranges at only R_0 and R_4. This is because AR-SC is oriented toward target coverage and not area coverage. It gives preference to nodes having more uncovered targets (points) and thus all the selected nodes typically use the maximum operational range. SRAP-A and SRAP-D show a more node collaborative behavior, while CentralizedApp shows the best collaborative behavior because of reducing the ranges iteratively and not at one step like the other protocols.

Figure 12 shows the energy consumed in the network during the operation of one cover (cover cost), given the selected operational ranges of each node. With maximum operational ranges, the energy consumed in the network is about 0.0234 Joule for $N = 900$. As expected, CentralizedApp gives the

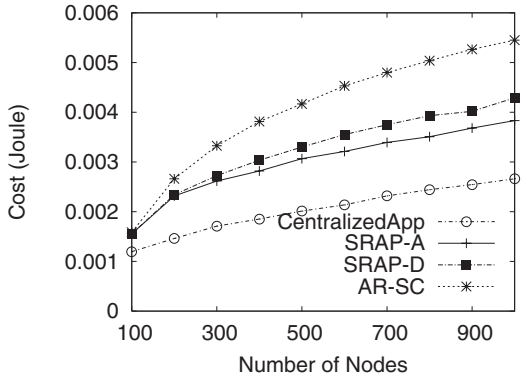


FIGURE 12
Energy cost of a selected cover after range assignment.

smallest cover cost. The figure shows that our protocols significantly reduce the cover cost over AR-SC, especially when distances are accurately estimated. SRAP-D and AR-SC show, respectively, about 10–20% and 30–70% increase in cover cost over SRAP-A. The redundancy introduced by SRAP-D is due to the conservative estimation of distances to guard against false positives (erroneously deciding that a point is covered).

B Network Operation

In this section, we evaluate the network operation when operational range assignment is employed. We focus on three metrics. The first metric is the coverage quality over time as the network operates. Coverage quality is the fraction of the field that is covered by sensors or actors at specific instances of time. The second metric is the duration while the coverage quality of the field is within a specified range. The third metric is coverage redundancy, which is the number of nodes covering the least covered point within a node's region. A coverage redundancy of 1 means that there is at least one point within any node's region that is covered by only one node. This metric indicates how *minimal* the selected cover and ranges are. We have noticed that the number of operational levels does not have significant impact on performance. The results below are averaged over 10 experiments of different random topologies.

Figure 13 shows the coverage time during which the percentage of field coverage is within a specific range of coverage quality. CentralizedApp shows about 50% coverage time improvement over our protocols. It is clear that SRAP-A and SRAP-D significantly improve coverage time over AR-SC, especially at the higher coverage quality ranges (60–80% and 80–100%). This is a desirable effect for applications requiring maximized field coverage time. For the quality range 80–100%, SRAP-A shows about 40–50% improvement over AR-SC, while SRAP-D shows about 30–40% improvement. By closely

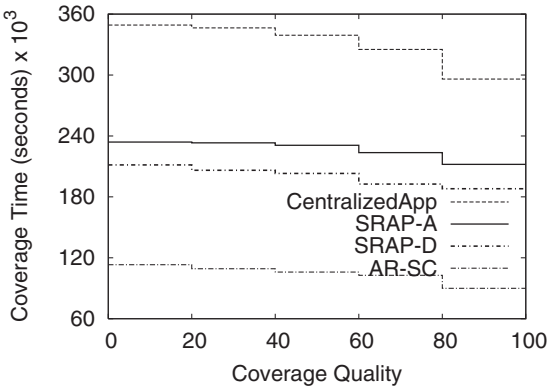


FIGURE 13
Coverage time vs. quality while the coverage quality is within five ranges: 0–20%, 20–40%, 40–60%, 60–80%, and 80–100%.

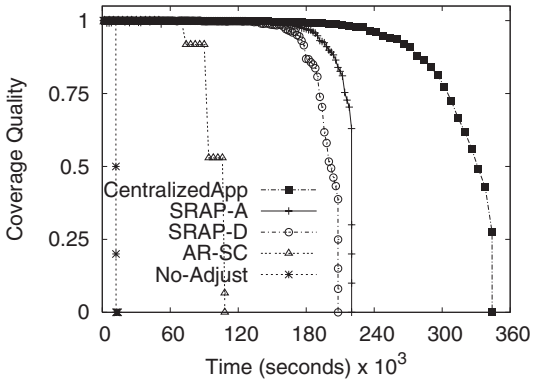


FIGURE 14
Coverage quality over time in a complete simulation run.

studying the behavior of ORAP, we can see that it delays the death of most of the nodes in the network as long as possible. The reason behind this behavior will shortly be apparent when other metrics are studied.

In Figure 14, we demonstrate the coverage quality of the field over time as the network is operating. We include results of the application when operated without operational range adjustment (referred to as "No-Adjust"), i.e., all the nodes use their maximum operational ranges (R_k). The figure shows that under CentralizedApp, SRAP-A, and SRAP-D nodes die smoothly over time because of periodically refreshing the selected operational ranges based on a dynamic parameter (battery level). This makes it possible to assign completely different ranges at consecutive network operation intervals (t_{cu}). For AR-SC, the change in coverage quality over time can be very abrupt since it changes the range assignment at a much slower rate. Note that if $t_{cu} \to 0$, most of the time will be spent in communication than covering. On the other hand, if $t_{cu} \to \infty$, the network lifetime will be equal to that of a node.

We also study the redundancy of the selected covers under all protocols. A cover is *minimal* if for the operational region A of every node, there exists at least one point in A that is covered by no more than one node. We compute the average minimum number of nodes covering any point within a node's region for all the compared protocols. A value of 1 means no redundancy. CentralizedApp guarantees no redundancy in the selected cover and thus is not included in this experiment. The redundancy of each protocol is reported in Figure 15. For SRAP-A, the redundancy does not exceed 1 by more than 2–3%. We closely examined these redundancies and found that they occur at nodes which are assigned range R_0. This means that the minimality requirement proven in Section V–D is not violated, even when VICON is used. The redundancy increases by about 5% if Phase III of ORAP is not applied. For SRAP-D, redundancy may reach 9–10% due to inaccurate neighbor placement. The difference in redundancies among SRAP-A,

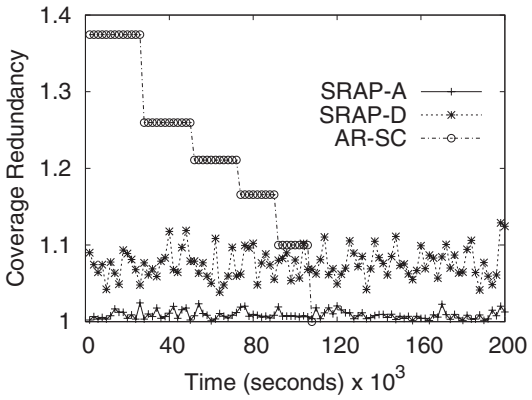


FIGURE 15
Coverage redundancy vs. time for $N = 900$.

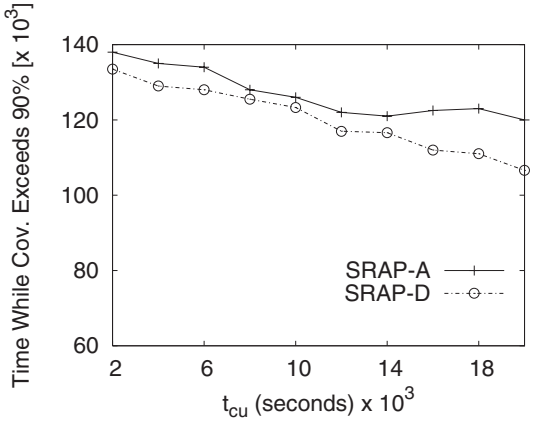


FIGURE 16
Effect of the cover update interval t_{cu} for $N = 500$.

SRAP-D, and AR-SC accounts in part for the difference in coverage time reported above.

Finally, we study the effect of the cover update interval (t_{cu}) on the performance of SRAP-A and SRAP-D. Figure 16 shows the time until coverage drops below 90% with different values oft_{cu} for $N = 500$ and $R_t = 5$ meters. It is clear that a smaller t_{cu} results in better coverage time. The result is not surprising since our model assumes that the operational process has dominant overhead and thus the cover construction overhead is negligible. In other models where the radio imposes significant overhead (as in our previous work [34]), coverage time deteriorates as t_{cu} approaches 0 or ∞.

VII CONCLUSION

In this work, we studied the problem of operational range assignment in sensor/actor networks. To handle the possibility of location-unawareness, we proposed a novel algorithm (VICON) for computing virtual coordinates for the neighbors of each node in the network based on localized information. We then proposed a distributed protocol (ORAP) that periodically assigns operational ranges to nodes based on their remaining resources. We have shown that ORAP has negligible message overhead and computational complexity. Our simulation results also indicate that ORAP significantly improves coverage time, even under inaccurate distance estimation. To extend the functionality of ORAP for different applications, we plan to study how to incorporate other parameters in the node weights, such as mobility, node degree, or potential coverage. We also plan to implement ORAP in real systems, such as [29].

REFERENCES

[1] R. Balasubramanian, S. Ramasubramanian, and A. Efrat, "Coverage time characteristics in sensor networks," University of Arizona, Tech. Rep., November 2006, http://www.ece.arizona.edu/∼sriniftinyos/papers/Sensor-Coverage-TechReport.pdf.

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

[3] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards optimal sleep scheduling in sensor networks for rare-event detection," in *Proc. of the International Symposium on Information Processing in Sensor Networks (IPSN)*, April 2005.

[4] S. Capkun, M. Hamdi, and J.-P. Hubaux, "GPS-free positioning in mobile ad hoc networks," in *Proc. of the International Conf. on System Sciences (HICSS)*, 2001.

[5] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *Proc. of the IEEE INFOCOM Conf.*, March 2005.

[6] M. Cardei, J. Wu, M. Lu, and M. Pervaiz, "Maximum network lifetime in wireless sensor networks with adjustable sensing ranges," in *Proc. of the IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob)*, August 2005.

[7] J. Carle and D. Simplot-Ryl, "Energy-efficient area monitoring for sensor networks," *IEEE Computer*, (2), 40–46, February 2004.

[8] Chemiresistor, http://www.sandia.gov/sensor/, 2007.

[9] P.-C. Chen, "A non-line-of-sight error mitigation algorithm in location estimation," in *Proc. of the IEEE Wireless Communications and Networking Conference*, 1999.

[10] Crossbow Technology Inc., http://www.xbow.com/, 2007.

[11] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM*, **45**(4), 634–652, July 1998.

[12] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic, "Localized sensor area coverage with low communication overhead," in *The Fourth Annual IEEE International Conference on Pervasive Computer and Communications (PerCom)*, March 2006.

[13] J. L. Gross and T. Tucker, *Topological Graph Theory*. John Wiley and Sons, 2001.

[14] S. Guolin, W. Yi, and G. Wei, "On localization in sensor network with inaccurate and incomplete distance measurements," *Journal of Electronics*, **24**(1), 60–63, January 2007.

[15] H. Gupta, S. Das, and Q. Gu, "Connected sensor cover: Self organization of sensor networks for efficient query execution," in *Proc. of the ACM International Symposium on Mobile and Ad-Hoc Networking and Computing (MobiHoc)*, June 2003.

[16] T. He, C. Huang, B. M. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proc. of the ACM MobiCom Conf.*, San Diego, September 2003.

[17] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," in *Proc. of the ACM Workshop on Sensor Networks and Applications (ACM WSNA)*, September 2003.

[18] S. Kumar, T. H. Lai, and J. Balogh, "On k-coverage in a mostly sleeping sensor network," in *Proc. of the ACM MobiCom Conf.*, September 2004.

[19] L. Lazos and R. Poovendran, "Stochastic coverage in heterogeneous sensor networks," *ACM Transactions on Sensor Networks*, **2**(3), 325–358, August 2006.

[20] Z. Li, W. Trappe, Y. Zhang, and B. Nath, "Robust statistical methods for securing wireless localization in sensor networks," in *Proc. of the International Symposium on Information Processing in Sensor Networks (IPSN)*, April 2005.

[21] T. Melodia, D. Pompili, V. C. Gungor, and I. Akyildiz, "A distributed coordination framework for wireless sensor and actor networks," in *Proc. of the ACM International Symposium on Mobile and Ad-Hoc Networking and Computing (MobiHoc)*, May 2005.

[22] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range assignment," in *Proc. of the ACM Conf. on Embedded Networked Sensor Systems (ACM SenSys)*, November 2004.

[23] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer, "Virtual coordinates for ad hoc and sensor networks," in *Proc. of DIALM-POMC*, October 2004.

[24] S. M. Nazrul-Alam and Z. Haas, "Coverage and connectivity in three-dimensional networks," in *Proc. of the ACM MobiCom Conf.*, September 2006.

[25] Osiris Photoelectric Sensors, http://schneider-electric.ca/www/en/products/sensors2000/html/osiris.htm, 2007.

[26] A. Rao, C. Papadimitriou, S. Ratnasamy, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proc. of the ACM MobiCom Conf.*, September 2003.

[27] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, **15**(11), 961–974, November 2004.

[28] D. Tian and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proc. of the First ACM Workshop on Wireless Sensor Networks and Applications*, September 2002.

[29] TinyOS, http://www.tinyos.net, 2006.

[30] R. Vedantham, Z. zhuang, and R. Sivakumar, "Mutual exclusion in wireless sensor and actor networks," in *Proc. of the IEEE Communications Society Conf. on Sensors and Ad Hoc Comm. and Networks (SECON)*, September 2006.

[31] K. Whitehouse and D. Culler, "Calibration as parameter estimation in sensor networks," in *Proc. of the ACM Workshop on Sensor Networks and Applications (ACM WSNA)*, September 2002.

[32] 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), 36–72, August 2005.

[33] T. Yan, T. He, and J. Stankovic, "Differentiated surveillance for sensor networks," in *Proc. of the ACM Conf. on Embedded Networked Sensor Systems (ACM SenSys)*, November 2003.

[34] O. Younis, M. Krunz, and S. Ramasubramanian, "Coverage Without Location Information," *Proc. of the IEEE International Conference on Network Protocols (ICNP)*, Beijing, China, October 2007.

[35] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *Proc. of the ACM International Conf. on Mobile Systems, Applications, and Services (ACM MobiSys)*, June 2005.

[36] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc & Sensor Wireless Networks*, **1**, 89–124, January 2005.

[37] Z. Zhou, S. Das, and H. Gupta, "Variable radii connected sensor cover in sensor networks," in *Proc. of the IEEE Communications Society Conf. on Sensors and Ad Hoc Comm. and Networks (SECON)*, September 2004.