# Energy Efficient Resource Distribution for Mobile Wireless Sensor Networks

Mohamed M.Ali Mohamed
Department of Electrical
and Computer Engineering
University of Illinois at Chicago
Email: mali25@uic.edu

Ashfaq Khokhar
Department of Electrical
and Computer Engineering
University of Illinois at Chicago
Email: ashfaq@uic.edu

Goce Trajcevski
Department of Electrical Engineering
and Computer Science
Northwestern University
Email: goce@eecs.nothwestern.edu

***Abstract*—This work addresses the problem of energy efficient management of mobile resource distribution in Wireless Sensor Networks (WSN), subject to particular Quality of Service (QoS) constraints. In practice, when phenomena over large geographical regions are being monitored, certain events (e.g., fast growing temperature exceeding a given threshold) may require an increased coverage within a particular area. In this paper we present novel methodologies for optimizing the "bargaining stage" when deciding how to select the mobile resources within the sensing field which will be re-located in response to a QoS request for an increased coverage. Our proposed techniques are capable of incorporating constraints – i.e., certain desirable properties, such as not overly-depleting any regions (i.e., drop the coverage QoS below certain threshold) when relocating mobile sensors. In addition to the energy-efficiency, we also tackle the aspect of finishing the "bargaining" and the transition in a manner that attempts to reduce the overall latency between signaling and completion of catering to a request. Our experimental results demonstrate that, when compared to the naive method of centralized decision-making about relocations of mobile units, our methodology offers significant energy savings, both in terms of communication overheads and maintenance of the hierarchical routing structures, as well as the quality assurances in terms of the turnaround time for servicing a given request.**

## I. Introduction

The advancement in development of wireless sensor networks (WSN) have revolutionized many application domains and generated a large body of research works over the past decade [1], [2]. Typically, a WSN – a collection of individual nodes equipped with sensing, computation and communication capabilities – monitors one or more physical phenomena in a given field, and provides a "map" of the values' variations in that field, which can be used for answering various queries, as well as detecting occurrences of events of interest. The two extreme-points along the spectrum of WSN-based data management are: (1) transmitting the raw data to a dedicated sink, possibly via multi-hop connections, and perform all the processing centrally; and (2) in-network distributed storing of the data and extracting the information relevant for processing the queries/events [3], [4]. The latter is often a preferred choice from the perspective of energy-efficiency (conversely, network's lifetime) [5] .

A common method used in managing the data gathering and aggregation in WSN is to construct a kind of a spatial indexing structure [6], which is maintained/updated in a distributed manner, subject to particular Quality of Service (QoS) constraints [7]. Contrary to the centralized settings, in-network coupling of data/information management and indexing structure maintenance involves not only the spatial regions "covered" by a particular node, but also roles/responsibilities of the nodes along the hierarchy of the particular index. Within a given region, the residing group of sensor nodes is assigned the task of monitoring and reporting the sensed values of the phenomena of interest. A particular problem that has been identified and addressed in the literature is how to ensure certain coverage criteria with a given deployment of a set of sensor nodes so that the spatial distribution of the monitored phenomena is matched with a satisfactory accuracy [8], [9]. A simple to define instance of this problem is how to ensure connectivity and coverage without having any "holes" in the network [10].

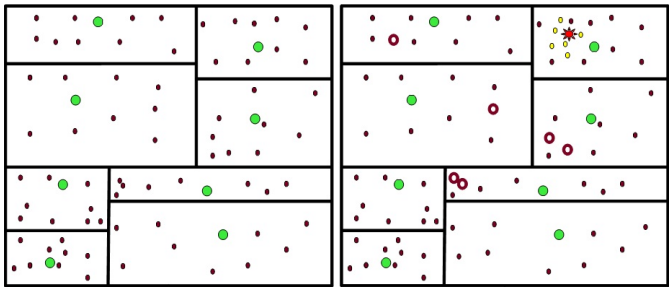


Figure 1. Relocation of sensors in response to an event.

Availability of mobile sensor nodes offers an immense flexibility to WSN, in the sense that the location of the sensors can change in order to adapt to certain changes of the values of the sensed phenomenon [11], [12], [13]. If a pre-defined event of interest is detected during the monitoring of the sensed field – e.g., a sudden increase in temperature and CO-concentration, indicating a possibility of a forest fire – then increasing the coverage in that locale may be needed, in order to provide denser coverage and more accurate measurements. However, that process cannot be executed "in isolation" – meaning, completely ignoring the quality of coverage in the rest of the region(s) monitored by the (static and mobile)

nodes [14]. The left part of Figure 1 depicts a stable scenario of sensor nodes distribution in a given field. The right portion illustrates how some of the sensor nodes (indicated as blank disks with red circular boundary) are selected to-be-moved in new locations inside the region in which an event of interest has been detected, requiring increased coverage.

In this paper, we propose efficient distributed algorithms for managing the relocation of mobile sensors upon detection of event of interest in a particular geographic location. The methodology also caters to the constraint of satisfying the minimum number of nodes required by each of the regions not co-located with events of interest, in order to meet connectivity, coverage, or any other application-dependent demands. Our methodology is able to handle multiple simultaneous requests, and provide the resources for them from the nearest region capable of supplying. The proposed methodologies aim at minimizing the communication cost for the "bargaining" process, and seek to supply the resources in a manner that minimizes the total motion, as well as the response time. The algorithm proceeds in a "cascading manner" – meaning, if the regions neighboring the one in which request-generating event is located are not able to satisfy fully the demand, then they provide a partial supply and recursively propagate the request to their (other) neighbors. We assume that the maximum number of nodes required by simultaneous of events in the field is no more than the total number of existing sensor nodes in the field, after satisfying the other regional constraints, i.e, coverage, connectivity, ..etc. Specifically, in this work we also consider the management of such request when a hierarchical structure is present and maintained in a distributed manner (cf. partitions in Figure 1).

In the rest of this paper, Section II introduces some preliminary background and notations/assumptions used for presenting our algorithms. In section III we present the techniques for efficient dissemination of the requests for increased coverage, and the methods of supplying resources are proposed in section IV. The experimental results are discussed in section V, followed by a related work discussion in section VI. The paper is concluded and avenues for future work are presented in section VII.

## II. Preliminaries

We assume a sensor network consisting of $N$ nodes $SN = \{sn_1, sn_2, \ldots, sn_N\}$, grouped into geographically collocated $K$ clusters $(C_1, C_2, \ldots, C_K)$. We also assume that under normal initial conditions, each cluster $C_i$ contains $\approx N/K$ motes, which may be of two basic kinds: *static* – $S_{C_i}$, and *mobile* – $M_{C_i}$ where $S_{C_i} \subseteq SN$ and $M_{C_i} \subseteq SN$, and $SN = \cup_{(i)}(S_{C_i} \cup M_{C_i})$. We assume that the location of the individual nodes are known, either via GPS or via some collaborative trillateration technique [1].

In order to ensure some "desirable" properties – both from the pure networking aspect (e.g., connectivity, coverage), as well as application Quality of Service (QoS) requirements (e.g., density of coverage) – we assume that each cluster has a predefined lower-bound threshold $\Theta_{C_i} \leq (N/K)$ so that $|M_{C_i}| + |S_{C_i}| \geq \Theta_{C_i}$. With this in mind, we note that part of the mobile nodes in each cluster may be "free" to move outside that cluster, without violating the $\Theta_{C_i}$ constraint. Hence, $M_{C_i} = M^b_{C_i} \cup M^f_{C_i}$, where $M^b_{C_i}$ denotes the mobile nodes *bound* to $C_i$ and $M^f_{C_i}$ denotes the *free* nodes which can cross the boundaries between neighboring clusters. While the membership of a particular mobile node $sn_j$ may vary – i.e., its state may transfer from *bound* to *free* and vice-versa, we assume that at any time-instant $M^f_{C_i} \cap M^b_{C_i} = \emptyset$.

Each cluster is assumed to have one designated sensor node that will act like a *local cluster-head*, and we use $H(C_i)$ to denote the local cluster-head of the cluster $C_i$. $H(C_i)$ is in charge of tasks such as gathering and maintaining the information about the status (e.g., locations, expected-lifetime) of cluster's population of nodes; coordinate the operation of the nodes (e.g., increase the sampling frequency); perform analysis/aggregation and information extraction of the measurements from the nodes in the cluster; ..etc. Based on the spatial partitions used, we assume a hierarchy which is constructed from the local cluster-heads, and rooted at a designated *sink*.

For this work, the important responsibilities of a local cluster-head are:

1) *Event detection* – event denotes an occurrence of something of interest [15] and, based on the reported values from the sensors in the cluster, the local cluster-head is in charge of detecting them[1]. We assume an application-dependent specification of "interesting" events, for which the location of the sensors detecting them is known, along with two values:
   - The bounding rectangle – which is, some safety or quality based boundary around the location of an event, approximated by the perimeter of a rectangle.
   - The number of sensors needed to be placed around the perimeter of the bounding rectangle – again, an application-dependent parameter. We assume that the sensors will be located at a uniform distance around the boundary.

   Hence, we use $E_{C_i,j}(L, B, n_e)$ to denote that the $j$-th event $E$ has been detected at location $L$ in the cluster $C_i$, for which $n_e$ nodes are needed around the perimeter of the rectangle $B$.
2) *Mobility coordination* – in order to ensure a desired QoS, the local cluster-head may need to direct the mobile sensors toward the location of a given event. We assume the existence of efficient techniques to orchestrate the trajectories of the mobile sensor for the purpose of positioning them at the respective locations around the perimeter of the bounding rectangle which can be viewed as a simplified instance of the techniques in [16] (cf. Sec. IV).

There are various spatial partitioning methods [17] and although throughout this work we use rectangular regions, the

[1]In this work, we do not consider any composite events.

results can be directly extended to the cases when the field of interest is convex polygon (subsequently split into a set of non-overlapping regions). A hierarchical spanning tree data structure (indexing tree) is constructed to manage the WSN, which is rooted at the sink node, and has the local-cluster-heads as the leaf nodes. The intermediate nodes are called global-cluster-heads, and we use the term cluster-heads to refer to both local and global-cluster-heads. Various widely used data structures conform to the aforementioned description, and have been used in the existing state-of-the-art indexing systems – e.g., K-D Trees, Octrees [18], [6], and Voronoi Treemaps [19]. We note that optimizing the energy consumption due to altering the indexing structure is not considered in this work (i.e., no local cluster-head will ever drop the number of actual sensors in its region below $\Theta_{C_i}$).

## III. Resource Requesting

We now proceed with the details of handling the requests for additional mobile nodes, to be made available in a cluster in which an event of interest has been detected.

When a local cluster-head $H(C_i)$ detects an event $E_{C_i,j}(L, B, n_e)$ within its region, it firstly checks whether the mobile nodes from $M_{C_i}$ are sufficient to cover the requirements – i.e., whether $|M_{C_i}| \geq n_e$. If so, the QoS requirements can be satisfied locally. Otherwise, $H(C_i)$ may need to request additional resources.

In the rest of this section, we focus on two basic techniques for handling the request from the local cluster-head that needs more resources to cover an event within its region to the other cluster-heads in the field. First, we present the centralized protocol, followed by two variants of a distributed protocol.

### A. Centralized Requesting

Under the centralized requesting scheme, once a local cluster-head recognizes the need of resources because of the detection of an event in its region, following is the protocol that is executed:

1) $H(C_i)$ sends a request to its parent node in the indexing tree by generating the message *Request*$(H(C_i), r, j)$, the semantics of which is: *The local cluster-head of $C_i$ is requesting $r, j$ nodes to satisfy the request of servicing the detection of its $j$-th event*. See Figure 2(a).
2) Once the sink node has received a particular request, it broadcasts the message *RequestSink*$(m_{id}, H(C_i), r, j)$ to its children, which recursively propagate it down the hierarchy, until it has reached the leaves (recall that leaves are actually the local cluster-heads). The parameter $m_{id}$ is a unique message-ID, in case the sink needs to process multiple requests from the same local cluster-head. See Figure 2(b).
3) Upon receiving the *RequestSink*$(m_{id}, H(C_i), r, j)$ message, each local cluster-head responds with a message containing an information about its free mobile nodes, which could be used to cater the given request. Thus, the local cluster-head of the $l$-th cluster, $H(C_l)$ will send the message *RequestCater*$(m_{id}, (H(C_l), A_l))$, indicating that it has $A_l$ available nodes. We note that $A_l \leq |M^f_{C_l})|$ (the number of the "free" mobile nodes) since $H(C_l)$ may be processing multiple request (including some due to events in its own geographical region). The *RequestCater*$(m_{id}, H(C_l), A_l)$ message is sent to the parent node in the hierarchy, and each parent aggregates the $(H(C_l), A_l)$ pairs for a corresponding $m_{id}$ before propagating it further up the hierarchy. See Figure 2(c).
4) Once the sink has received the availabilities of individual clusters, it calculates the best manner to move resources in response to *Request*$(H(C_i), r, j)$, and individual messages are sent down the hierarchy, notifying individual local cluster-heads how many of their available nodes should be forwarded towards $C_i$. Each local cluster-head $H(C_l))$ whose resources will need to be moved, will receive the message *Allocate*$(m_{id}, H(C_i), L(H(C_i)), H(C_l), a_l)$, where $a_l \leq A_l$ is the number of nodes to be moved towards $H(C_i)$, located at $L(C_i)$. See Figure 2(d).

Figure 2 depicts the steps of the centralized requesting protocol. The hierarchical index in this figure is based on a K-D Tree structure (orthogonal bisections). It shows an example scenario prior to the event. We note that the criterion for selecting the resources to be forwarded is discussed in section IV.

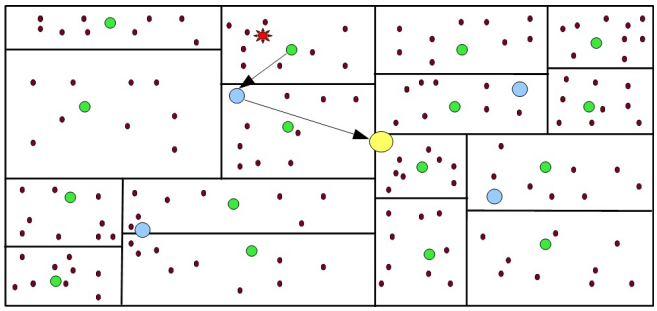
Assuming an $n$ nodes K-D Tree as an example indexing structure, the complexity of executing the centralized protocol can be characterized as follows:

- $\log n$ messages are needed to propagate the request from the $H(C_i)$ to the sink node.
- $n/2 = (O(n))$ messages to send the information request from the sink back to all the local cluster-heads. However, since some of them may be transmitted in parallel in different sub-trees. , the time, in terms of hops, is bounded by $O(\log n)$.
- $O(n)$ messages which are from the local cluster-heads towards the sink – again, bounded by $O(\log n)$ in terms of time (although two children will need sequential transmission towards their parent, the bound is still $O(\log n)$).
- Let $k$ denote the number of local cluster-heads selected to participate in sharing their resources with $H(C_i)$ ($k \leq n/2$).

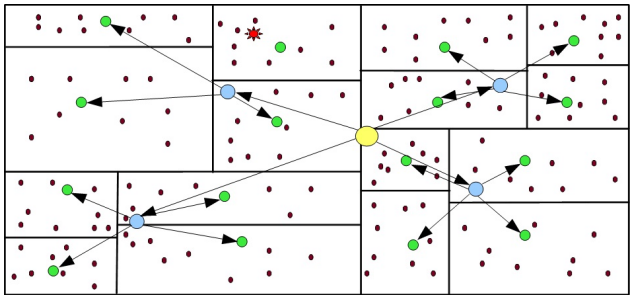
Thus, the overall communication complexity (message cost) is bounded by $O(n)$, in terms of number of messages, and $O(\log n)$ in terms of time.

### B. Distributed Request Management

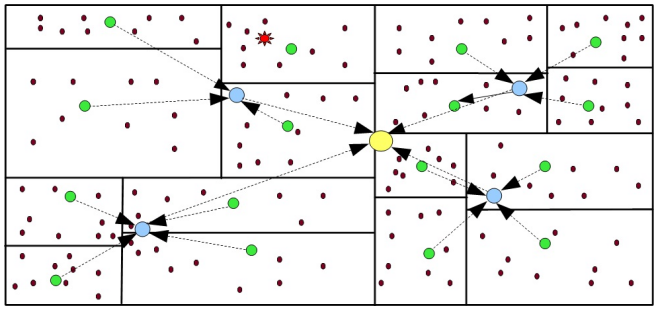
The distributed structural requesting protocol aims at minimizing the overall communication cost via exploiting spatial locality. We now present two approaches for handling a request for additional resources. The first one, called *structure-based* is relying solely on an existing hierarchical index structure, whereas the second one proposes a coupling between the indexing structure and geographical proximity.



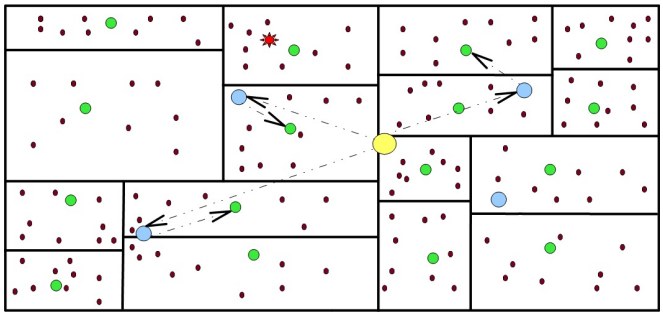
(a) Step 1: The local-cluster-head (Green) detecting the event sends a request to its parent node (Blue), which is forwarded to the sink node (Yellow).



(b) Step 2: The sink node (Yellow) requests information about available resources from all the local-cluster-heads (Green) through the data structure hierarchy.



(c) Step 3: Available resources information is sent to the sink node (Yellow) through the data structure hierarchy.



(d) Step 4: Sink node (Yellow) sends out the decision about resource forwarding to the involved local-cluster-head (Green) nodes throughout the data structure hierarchy.

Figure 2. The centralized requesting process in a sensed field spatially split with orthogonal bisection, with a K-D Tree indexing structure. The middle large yellow node represent the sink node, the four blue medium nodes are the global-cluster-heads, the sixteen green small nodes are the local-cluster-heads, and the tiny red nodes represent the sensor nodes distributed in the field

*1) Structure-Based Distributed Request (SBDR) Management:* Assuming a Binary Space Partitioning (BSP) structure which has recursively divided a given space into contiguous non-overlapping regions, each border/hyperplane corresponding to node in the indexing BSP tree, has a unique *level* (i.e., distance from the root).

When $E_{C_i,j}(L, B, n_e)$ is detected and $H(C_i)$ determines that additional sensors are needed to satisfy the QoS criteria, the SBDR protocol proceeds as follows:

1) $H(C_i)$ sends the message *Request*$(H(C_i), r, j)$ requesting additional mobile nodes to *its sibling node(s) in the indexing tree* which is sharing a common border and parent. See Figure 3(a).
2) In the case that $H(C_{s,i})$, the sibling of $H(C_i)$, can cater to the request, it responds with *Granted*$(H(C_i), j, r_s)$. Clearly, for this we need that $r_s > r$, and the nodes are selected from $M^f_{C_{si}}$ which is properly updated.
3) In the case that $H(C_{s,i})$ cannot cater to the request, it will send the message *Deny*$(H(C_i), j, r_s)$. The meaning is that, although it cannot fully grant the request, the sibling is still able to provide $r_s \geq 0$ nodes. In this case, $H(C_i)$ will forward *Request*$(H(C_i), r - r_s, j)$ to its parent.
4) The parent-node of $H(C_i)$, in turn, instead of propagating the request towards the sink, will actually forward the *Request*$(H(C_i), r - r_s, j)$ to its own sibling at the same level and sharing a common border whether it can cater to $H(C_i)$'s request. See Figure 3(b).
5) The procedure is repeated recursively until, in the worst case, the request has reached the root.

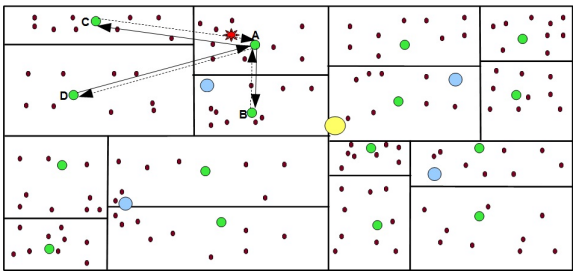
The SBDR protocol is illustrated in Figure 3. It depicts the chaining of the messages at two levels from the root, since the request cannot be satisfied at the first level – i.e, by sibling local cluster-heads.

Clearly, both the communication cost and the time for detecting the fulfilment of a particular request will vary for the SBDR protocol. We note that, in the worst-case scenario, the request needs to be propagated all the way to the sink node. Worse yet, the attempts to resolve it locally constitute additional overhead in terms of the time needed to determine the servicing of the request. However, as our experiments demonstrate, SBDR protocol does provide improvements over the centralized protocol.

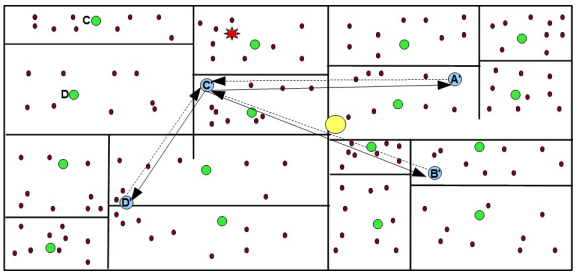
*2) Structure and Proximity based Distributed Request (SPDR) Management:* The objective of the SPDR variant is to decrease the overhead induced by the sibling-to-parent communication in the SBDR protocol. We observe that some local cluster-heads which are not siblings may still share a common border. To capitalize on this, in addition to the sensor nodes physically belonging to its cluster, each local cluster-head will maintain a list of its "cousins" – which is, the sibling node and the nodes sharing common border.

Upon detecting an event $E_{C_i,j}(L, B, n_e)$, the local cluster-head $H(C_i)$ executing CPDR protocol initiates the following:

1) $H(C_i)$ sends the message *Request*$(H(C_i), r, j)$ requesting additional mobile nodes to *its geographically neigh-*

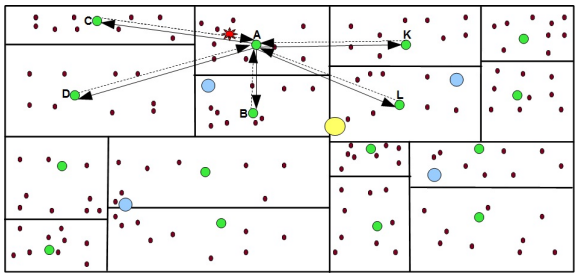



(a) Local-cluster-head A detects an event, and sends a resource request to its sibling local-cluster-heads B, C and D, which send back the response.

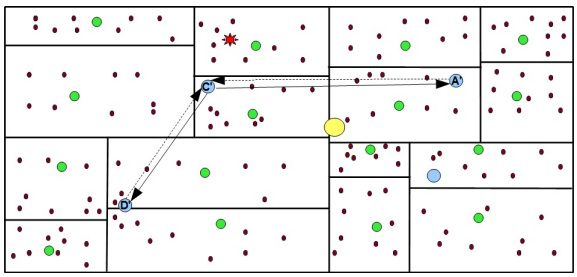



(b) When the request is not satisfied at the local-cluster-heads level, global-cluster-head C' checks the resource availability with its sibling nodes (A', B' & D')

Figure 3. SBDP with a K-D Tree indexing structure. The middle large yellow node represents the sink node, the four blue medium nodes are the global-cluster-heads, the sixteen green small nodes are the local-cluster-heads, and the tiny red nodes represent the sensor nodes distributed in the field




(a) Local-cluster-head A detects an event, and sends a resource request to its neighboring local-cluster-heads (B, C, D, K & L), which send back the responses.




(b) When the request is not satisfied at the local-cluster-heads level, global-cluster-head C' checks the resource availability with the neighboing global-cluster-heads (A' & D')

Figure 4. SPDR with a K-D Tree indexing structure. The middle large yellow node represent the sink node, the four blue medium nodes are the global-cluster-heads, the sixteen green small nodes are the local-cluster-heads, and the tiny red nodes represent the sensor nodes distributed in the field

*boring nodes with which it is sharing a border*. See Figure 4(a).

2) The sibling node $H(C_{s,i})$ and each of the Boarder-Neighbors ($BN(H(C_i))$ who can cater to the request, responds with *Granted*$(H(C_i), j, r_s)$. In this case, the request is no longer propagated. $H(C_i)$ notifies its sibling and its neighbors how many mobile nodes each of them should dispatch.
3) If the sibling node and some of the $BN(H(C_i))$ cannot cater to the request, they will each send *Deny*$(H(C_i), j, r_s)$. Note, however that, unlike the SBDR protocol, now the sum of the $r_s$ values from the sibling and the neighbors combined, may actually satisfy the request.
4) If not, the message *Request*$(H(C_i), r - \Sigma(r_s), j)$ is propagated to the parent of $H(C_i)$, and parent recursively repeats the procedure. See Figure 4(b).

Figure 4 shows the messaging at two different levels in the hierarchy, where the request cannot be satisfied at the first level, i.e, through sibling local-cluster-heads communication.

Again, we note that the worst-case scenario in terms of the upper-bound is the same as the centralized protocol – and, once again, in the worst case scenario we have the additional overheads of the attempts to resolve the request locally. However, in practice, one can obtain improvements – as demonstrated by our experiments.

## IV. Resource Supplying

In this section we present the methodology of fulfilling a resource request, starting with the process of acceptance of requests, selection of the nodes to be moved and moving them towards the cluster which has signaled a request.

### A. Strategy of Acceptance

Upon receiving a resource_request, a cluster-head node compares the number of requested sensor nodes $R_i$ to the number of available resources within its region $V_i$. If the available resources are sufficient to cater for the request, i.e, $V_i > R_i$, an acceptance message is sent to the requesting node, and the process of selecting the sensor nodes to be moved and moving them starts immediately.

Contrarily, if the available resources are less than the required resource, i.e, $V_i < R_i$, the request cannot be rejected. The reason for this, is that in a global view of the field, sometimes no single cluster might be able to suffice the needs for one request. However, a set of clusters can provide a number of the needed resources, which make them collectively able to suffice the new event needs. Therefore, in such scenario, the cluster-head receiving the request sends back a partial_acceptance message, indicating that it will be able to provide $V_i$ resources. Accordingly, when the requesting node receives this message, it forwards the request to another cluster-head node –according to the requesting strategy– with the required number of resources updated, i.e, $R_i = R_i - V_i$. Simultaneously, the partially accepting cluster-head node will

start forwarding the $V_i$ sensor nodes, which will create "some" sufficiency for the requesting cluster until further resources arrive. In other words, because of the partial acceptance feature, each request is –most likely– going to add some help to the requesting node, unless the whole region is starving.

### B. Nodes Selection (Which nodes to move?)

Once a local-cluster-head sends the requested resources –or some of them– to the requesting local-cluster-head, a criterion is needed to determine which specific sensor nodes are the ones to be forwarded. The selection criterion may involve the following metrics:

- Speed of Arrival/Travel Distance: The cluster-head node selects the sensor nodes to be moved such that they would arrive to the destination in the fastest way (or travel the shortest distance). This selection would vary according to the motion path (i.e, Manhattan, direct straight path, ...etc.).
- Local Configuration Balance: Maintaining the balance of the sensor nodes distribution inside the accepting cluster. Accordingly, the cluster-head selects the nodes to move in a way that minimizes –or better eliminates– the need of moving the remaining nodes inside the cluster to maintain its internal constraints (i.e, connectivity, coverage, ..etc).
- Global Configuration Balance: Maintaining balance of the sensor nodes distribution in the whole field. The goal of this balance is to keep the available of the $M_C$ mobile sensor nodes distributed across the field, which helps having resources available near to possible future events. This also balances the load on the indexing tree, which keeps –relatively– equal load of network information updates on the tree branches. We note that this option is possible in a straight forward fashion in the centralized solution. However, including this metric in distributed techniques would incur added overhead.
- Energy Consumption: The cluster-head node selects the sensor nodes to be moved according to an optimization function which minimizes the consumed energy. The optimization can focus on the energy consumed for communication, or the energy consumed in motion, or a weighted factor of each of them.

### C. Movement strategy (How to move the nodes?)

The sensor nodes selected to be moved towards the requesting cluster are informed by their local-cluster-head. The supply of sensor nodes to the requesting cluster can follow different methods. In this subsection, we present two methods to supply the requested resources from the –partially or fully– accepting local-cluster-head(s) towards the requesting cluster, then we follow with a discussion on handling the request inside local-cluster-heads. In the scope of this work, we assume an obstacle-free field, or that obstacle avoidance is implicitly taken care of.

*1) Direct Forwarding:* In direct forwarding, the sensor nodes move directly towards the requesting cluster. The motion can be in a straight path or Manhattan, depending on the application setup. Once the nodes are decided to move, they are informed by their local-cluster-head, and given the location of the cluster of destination. The nodes leave their cluster towards the destination cluster. On their way to the destination, the sensor nodes can turn off their sensing devices and radio transceivers until they arrive, where, when the sensor nodes pass through intermediate clusters, they do not need to report information. For some data intensive applications, the passing sensor nodes can turn on the sensing and reporting, depending on the speed of motion and the distance traversed inside the cluster. Figure 5 shows the direct forwarding of sensor nodes towards the cluster containing the event.

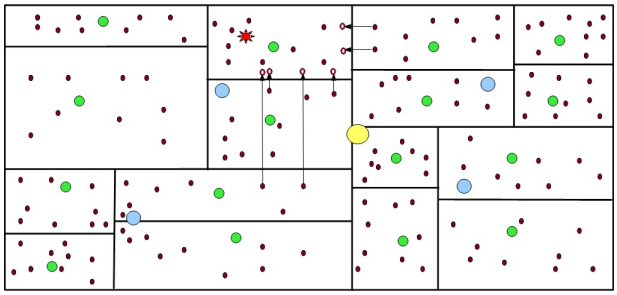


Figure 5. Six sensor nodes moved towards the cluster containing the event coming from three different supplying clusters.

*2) Relayed Motion:* The relayed motion depend on setting up the path of motion of the senor nodes through the intermediate clusters before starting the real motion. The goal of this type of motion is to minimize the traveled distance by each sensor node, and provide faster supply to the new events, especially when the available resources for supply are more than one cluster away, i.e, not direct neighboring. The method starts once resources are decided to be moved from cluster $C_{source}$ to cluster $C_{dest}$ passing, in sequence, through clusters $C_i$, where $i = 1, 2, .., k$. In the path setup, each local-cluster-head is informed with the local-cluster-head before it in the sequence, the one after it, and the number of resources to be supplied. After all the local-cluster-heads in the path are informed –and possibly requested to confirm, for some applications– the real motion starts. Each cluster-head sends the required amount of resources to the next cluster in the sequence, starting from $C_{source}$ through $C_i$ to $C_{dest}$. This method gives the advantage of faster delivery of the sensor nodes to the destination, regardless of the travel distance. However, it is at the cost of more unbalance during the transient period, where some intermediate clusters may have less number of nodes than its minimum requirements. Also, all the cluster head nodes along the path need to know that they are participating in this scenario. Figure 6 shows the relayed motion of sensor nodes towards the cluster containing the event, passing through intermediate clusters.

*3) Intra-Cluster Motion:* Once the sensor nodes from other clusters have reached the one who has requested resources, the local cluster-head will need to execute a re-allocation algorithm. As mentioned in Section II, in this work we assume that an event is associated with a bonding rectangle, and each

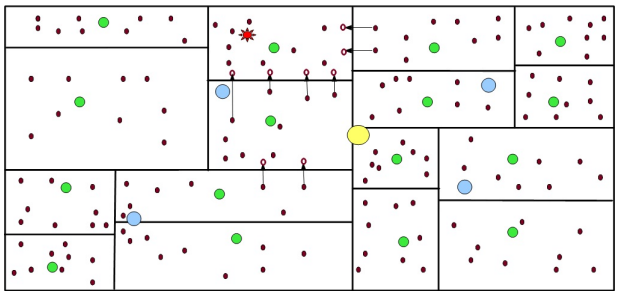


Figure 6. Six sensor nodes moved towards the cluster containing the event coming from two supplying clusters, where one of them (the bottom cluster) relays two more sensor nodes from its population for the cluster beneath it. It accordingly receives other two sensor nodes, which it can place at the appropriate positions inside the cluster.

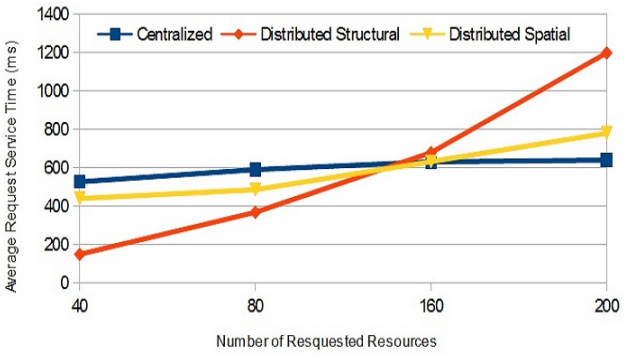



Figure 8. Average Request Service Time against Number of Requested Resources.

type of an event has distribution of locations for the sensors around the boundary.

With this in mind, the re-location inside a given cluster can be readily accomplished using the heuristics from [16]. We note that there are different variants of the re-location problem – e.g.: minimize the latest arrival time; determine locations that will maximize the reachability/coverage, and with a given time-budget, ..etc. [9]. In our work, we assume the simplest variant – minimizing the latest arrival time, with the known destinations' location. This is illustrated in Figure 7 – showing a zoomed-version of the cluster in which an event has been detected in Figure 6. As can be seen, some of the previously available sensors, along with the newly-arrived ones, are routed towards the predetermined locations along the perimeter of the rectangle bounding the event.

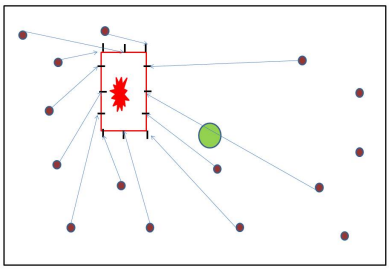


Figure 7. An example of intra-cluster movement of nodes.

## V. Experimental Results

We now discuss the simulation results illustrating the performance of the proposed methodologies. The proposed methods were implemented on top of SIDnet-SWANS [20] WSN simulator, based on Jist-SWANS discrete event simulation engine [21]. The data structure used for data indexing is an orthogonal bisection based K-D Tree implementation [17], [6]. The WSN has 500 nodes deployed in a square field of 300x300 meters square, using MAC802.15.4, and Shortest Geographic Path for routing. The power consumption characteristics are based on Mica2 Motes specifications, MPR500CA.

The number static sensor nodes in the field was set to $S_C = 80$ (also used as cluster-heads for the purpose of the K-D tree), which left $M_C = 420$ mobile nodes. The experiments were run for the proposed requesting techniques (Centralized, SBDR and SPDR). The number of requested nodes per event was varied from 40 to 200 nodes. We tested scenarios with 1, 4, 8 and 12 simulated events, simultaneously occurring in different clusters, and we report the averaged measurements over the parameters set. In our experiments, we compare the three proposed requesting methods according to several metrics, which we define as:

- **Request Service Time:** The time elapsed between the issuing of the initial request, till the request is accepted and the nodes are forwarded to the requesting cluster.
- **Average Travel Distance Per Sensor Node:** The average distance each resource (i.e, sensor node) needs to travel across the field to reach the requesting cluster.
- **Communication Cost:** The total number of messages transmitted during the requesting process, including the request messages, response messages and decision messages.
- **Resolution Level:** The leaf-based level in the K-D Tree hierarchy at which the request got accepted. We denote the local-cluster-heads as level 1, global-cluster-heads as level 2, and the sink node as level 3.

In Figure 8, the average request service time for the two distributed methods start lower than the centralized requesting method. However, with the increase in the number of requested resources, their service time exceeds that of the centralized method. This happens because the amount of resources availabel in the clusters neighboring (spatially or structurally) to the requesting cluster cannot suffice this large number of requested resources. Accordingly, further requesting iterations takes place to negotiate resources with more physically distant cluster heads up in the hierarchy. On the other side, the centralized requesting method provides a service time that is mostly stable with the increase of the amount of requested resources.

The average travel distance per sensor node is depicted in Figure 9. The distributed spatial requesting achieves the

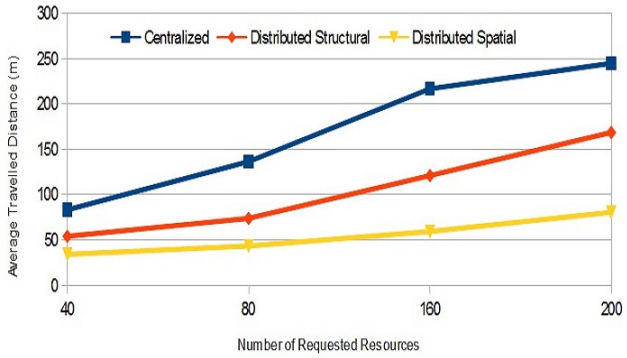



Figure 9. Average Travel Distance Per Sensor Node against Number of Requested Resources.

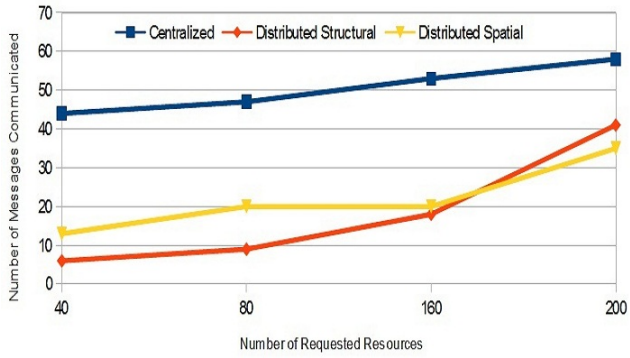



Figure 10. Number of Messages Communicated against Number of Requested Resources.

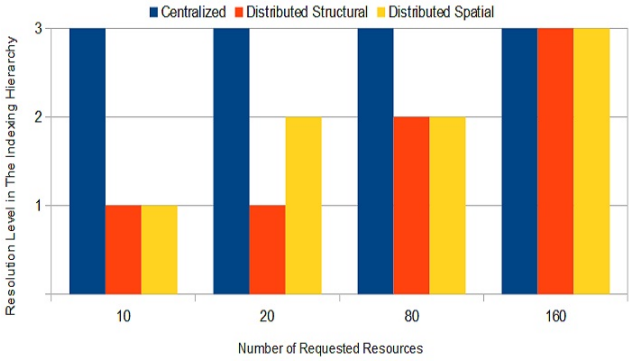



Figure 11. Resolution Level in the Indexing Hierarchy against Number of Requested Resources.

least average travel distance per sensor node, followed by the distributed structural method. However it seems counter-intuitive that the centralized method does not achieve the most optimal solution, this happens due to the less information it has. The decision in the centralized method is taken at the sink node, which has comprehensive information about the field until the local-cluster-heads level. Thus, it centrally calculates the most optimal distance based on the providing local-cluster-heads locations, which might have the available nodes within their clusters further from the border. On the contrary, the decentralized methods negotiate the resource supplying process first at the local-cluster-heads level, which makes them able to optimize based on the real location of the sensor nodes rather than the local-cluster-heads locations. If the centralized solution needed to be optimal, which is doable in terms of the logical capability, this would require the requesting information process to include the locations of the potential sensor nodes, which we consider as a significant communication overhead.

The communication cost of the centralized method is higher than the distributed methods, as shown in Figure 10, because of the information gathering rounds. In order to compare the communication cost of the two distributed methods, the resolution level depicted in Figure 11 gives us more clarity about the behavior inside the indexing hierarchy. The distributed methods were able to handle the requests below 160 sensor nodes without the need of propagating the request to the sink node. In this case, the communication cost of the distributed spatial method is hgiher than the distributed structural method, because of its need to communicate with the neighbors list, which is more than the sibling nodes in the K-D Tree. After this threshold, the communication cost of the distributed structural method slightly exceeds the distributed spatial method, because the number of decision messages is more likely to be higher for the structural method. This is because the partial acceptance across the hierarchy eliminates all the neighboring list clusters from being included in any further decision announcements, as they have already sent out their available resources to the requesting cluster.

Figures 12 and 13 compare the three requesting strategies when multiple simultaneous events are detected in different regions of the field, each requesting 30 sensor nodes. Thus, multiple requests are issued to the indexing structures from different leaf nodes (i.e, local-cluster-heads). The centralized method comes to be of the highest request service time and average travel distance, while the other two distributed methods achieve comparable results. The distributed methods outperform the centralized method because of their capability of handling the requests within their locality by providing resource supplies from the (spatially or structurally) neighboring clusters.

## VI. Related Work

In the recent years, mobility has contributed to the variety of application domains for WSNs and has brought a unique set of challenges and research results[11], [13]. From the basic setup-aspect, mobility facilitates deployment [22], [23], augments the monitoring [24], [18] and data gathering [25], [26] capabilities.

But one example of a formalism for relocation of sensor nodes in the deployment phase is using a virtual force based algorithms, proposed in [27], [22], [23]. Wang et. at. [23] propose an iterative algorithm in which coverage holes are detected by sensors using Voronoi diagrams [28]. The sensors are then moved from high density zones to low density zones increase coverage. Many sensor relocation algorithms have been

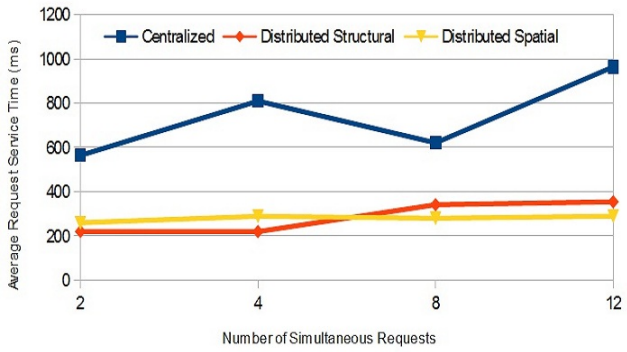



Figure 12. Average Request Service Time against Number of Simultaneous Requests.

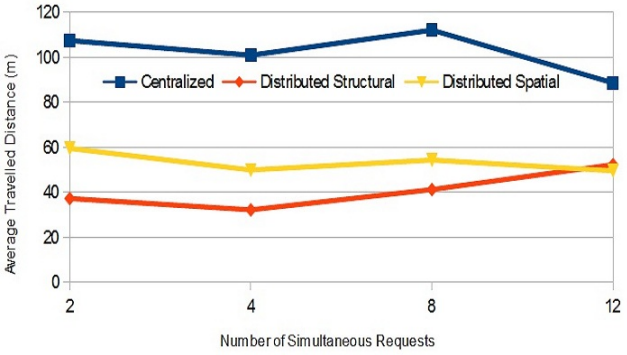



Figure 13. Average Travel Distance Per Sensor Node against Number of Simultaneous Requests.

proposed [29], [14], [30], [27], [24], [31], [32], presenting distributed algorithms in which the sensor nodes coordinate the relocation process themselves. While these approaches are useful for low density and small scale WSN, in this work we tried to capitalize on a hierarchical structure for settings in which WSNs have larger node population in relatively small spatial regions. The main benefit of our approaches is the separation of the bargaining process (requesting and supplying sensors) from the individual sensor nodes, and elevating is to clusters' level – thus savings in the communication and energy-expenditures.

In [29], [14], Cao et. al. proposed an algorithm, with physical implementation, of a Grid-Quorum solution for sensor relocation in WSN. The sensed field is split into cells arranged in a grid. Each cell has a cluster-head, which known the number of redundant nodes in its area. The information about redundant nodes are shared between cluster heads in the same row and column of the grid. When coverage is required in a specific region, the request is communicated in the row and column of its cell, where the supplier cells are identified using the intersection of the request with the previously advertised redundant nodes. The movement of nodes then follows a cascaded (relayed) path, which is negotiated between the sensor nodes along the path. The single level clustering decreases the scalable performance of the algorithm. If the network size is increased, the advertising and requesting processes will incur high communication cost and latency. The arrangement of cascaded movement in long paths will also be energy intensive, in terms of communication, as it will involve many sensor nodes. Our proposed approach, which operates via hierarchical scheme alleviates some of these drawbacks.

Several fully distributed algorithms were proposed for sensor nodes relocation, for which communication cost would highly increase for large scale and high density WSN. A vector algebra based algorithm to find the locations of potential redundant nodes for coverage compensation is proposed in[30]. The selection of the best redundant nodes is performed opportunistically by jointly considering the hole boundaries and the remaining energy of nodes. [27] proposed a distributed algorithm for node deployment and event-based relocation, where sensor nodes are moved by virtual forces. The algorithm requires knowledge of relative positions between neighboring nodes, by which they coordinate their movements. In [24], an iterative distributed relocation algorithm is presented, where each mobile sensor only requires local information in order to optimally relocate itself. The mobile sensors are assumed to be able to move only once over a short distance. Relocation of hopping sensors was investigated in rugged terrains in [31], [32]. The mobility model assumes that the sensor nodes move in fixed distance hops, and the algorithms are designed to fill sensing holes by optimizing the required number of hops using direct or relayed motion. Once again, the aspect of our work which complements the contexts addressed in these works is the scalability-benefits.

## VII. Conclusion and Future Work

In this paper, we proposed efficient methodologies for scalable management of relocation of mobile sensors in WSNs, in response to a detection of event of interest. The proposed work takes into consideration the minimum nodes count needed in each spatial region for guaranteeing certain QoS criteria. Capitalizing on a hierarchical structure, we distributed protocols which improve both the response time and the energy consumption due to communication, along with the choices of nodes to move seeking the optimization of the traveled distance. We presented three different requesting methods (centralized, SBDR and SPDR) and showed the difference in performance between them. The proposed approaches are capable of handling simultaneous detection of multiple events. The displacement of the mobile sensor nodes is performed using direct forwarding or relayed motion, which is handled between the cluster heads for large scale management.

Our future work is centered around two extensions. Firstly, we would like to investigate what are the costs involved in adjusting different hierarchical structures (e.g., Voronoi Treemaps [19]) when nodes move in response to an event, and develop efficient algorithms for optimizing those costs and identify the trade-offs involved. Secondly, we will investigate the problem of optimizing the motion plans of the nodes when the budget of available nodes across the network is not sufficient to cater to all the detected events.

## ACKNOWLEDGMENT

This research has been supported in part by the NSF grants CNS 0910988, 0910952 and III 1213038.

## REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004.

[3] C. Liu and G. Cao, "Distributed monitoring and aggregation in wireless sensor networks," in *INFOCOM*, 2010, pp. 2097–2105.

[4] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, 2005.

[5] I. Dietrich and F. Dressler, "On the lifetime of wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 5, no. 1, pp. 5:1–5:39, Feb. 2009. [Online]. Available: http://doi.acm.org/10.1145/1464420.1464425

[6] H. Samet, *The design and analysis of spatial data structures*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[7] W. Zhang, G. Cao, and T. L. Porta, "Data dissemination with ring-based index for wireless sensor networks," *IEEE Trans. Mob. Comput.*, vol. 6, no. 7, pp. 832–847, 2007.

[8] R. Mulligan and H. M. Ammari, "Coverage in wireless sensor networks: a survey," *Network Protocols and Algorithms*, vol. 2, no. 2, pp. 27–53, 2010.

[9] C. Caicedo-Nuez and M. Zefran, "A coverage algorithm for a class of non-convex regions," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 2008, pp. 4244–4249.

[10] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing holes in sensor networks," *MONET*, vol. 11, no. 2, pp. 187–200, 2006.

[11] S. F. Pileggi, C. Fernandez-Llatas, and T. Meneu, "Evaluating mobility impact on wireless sensor network," in *Proceedings of the 2011 UKSim 13th International Conference on Modelling and Simulation*, ser. UKSIM '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 461–466. [Online]. Available: http://dx.doi.org/10.1109/UKSIM.2011.94

[12] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. Sukhatme, "Robomote: enabling mobility in sensor networks," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, 2005, pp. 404–409.

[13] E. Ekici, Y. Gu, and D. Bozdag, "Mobility-based communication in wireless sensor networks," *Communications Magazine, IEEE*, vol. 44, no. 7, pp. 56–62, 2006.

[14] J. Teng, T. Bolbrock, G. Cao, and T. L. Porta, "Sensor relocation with mobile sensors: Design, implementation, and evaluation," in *MASS*, 2007, pp. 1–9.

[15] R. Adaikkalavan and S. Chakravarthy, "Formalization and detection of events using interval-based semantics," in *COMAD*, 2005, pp. 58–69.

[16] G. Trajcevski, P. Scheuermann, and H. Brönnimann, "Mission-critical management of mobile sensors: or, how to guide a flock of sensors," in *DMSN*, 2004, pp. 111–118.

[17] H. Samet, *Applications of spatial data structures: Computer graphics, image processing, and GIS*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[18] M. M. A. Mohamed, A. Khokhar, and G. Trajcevski, "Energy efficient in-network data indexing for mobile wireless sensor networks," in *Advances in Spatial and Temporal Databases*. Springer, 2013, pp. 165–182.

[19] M. Balzer, O. Deussen, and C. Lewerentz, "Voronoi treemaps for the visualization of software metrics," in *Proceedings of the 2005 ACM symposium on Software visualization*. ACM, 2005, pp. 165–172.

[20] O. C. Ghica, G. Trajcevski, P. Scheuermann, Z. Bischof, and N. Valtchanov, "Sidnet-swans: a simulator and integrated development platform for sensor networks applications," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, ser. SenSys '08. New York, NY, USA: ACM, 2008, pp. 385–386. [Online]. Available: http://doi.acm.org/10.1145/1460412.1460464

[21] http://jist.ece.cornell.edu/index.html. [Online]. Available: http://jist.ece.cornell.edu/index.html

[22] N. Heo and P. K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 35, no. 1, pp. 78–92, 2005.

[23] G. Wang, G. Cao, and T. La Porta, "Movement-assisted sensor deployment," *Mobile Computing, IEEE Transactions on*, vol. 5, no. 6, pp. 640–652, 2006.

[24] W. Wang, V. Srinivasan, and K.-C. Chua, "Coverage in hybrid mobile sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 11, pp. 1374–1387, 2008.

[25] S. Hanoun, D. Creighton, and S. Nahavandi, "Decentralized mobility models for data collection in wireless sensor networks," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1030–1035.

[26] G. Xing, M. Li, T. Wang, W. Jia, and J. Huang, "Efficient rendezvous algorithms for mobility-enabled wireless sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 1, pp. 47–60, 2012.

[27] M. Garetto, M. Gribaudo, C.-F. Chiasserini, and E. Leonardi, "A distributed sensor relocatlon scheme for environmental control," in *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*. IEEE, 2007, pp. 1–10.

[28] M. De Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational geometry*. Springer, 2008.

[29] G. Wang, G. Cao, T. La Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4. IEEE, 2005, pp. 2302–2312.

[30] N.-n. Qin, L.-x. Guo, Z.-g. Ding, and B.-g. Xu, "A vector algebraic algorithm for coverage compensation in hybrid wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.

[31] M. Kim, M. W. Mutka, and H. Choo, "On relocation of hopping sensors for rugged terrains," in *Computational Science and Its Applications (ICCSA), 2010 International Conference on*. IEEE, 2010, pp. 203–210.

[32] Y. Pei, F. J. Cintrón, M. W. Mutka, J. Zhao, and N. Xi, "Hopping sensor relocation in rugged terrains," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 3856–3861.