Distributed Real-Time Detection and Tracking of Homogeneous Regions in Sensor Networks*

Sharmila Subramaniam, Vana Kalogeraki
Department of Computer Science and Engineering
University of California, Riverside
{sharmi,vana}@cs.ucr.edu

Themis Palpanas
IBM T.J. Watson Research Center
themis@us.ibm.com

Abstract

In many applications we can deploy large number of sensors spanning wide geographical areas, to monitor environmental phenomena. The analysis of the data collected by such sensor network can help us to understand the field dynamics, and optimize the deployment of other solutions. We define a group of sensors having similar underlying distribution over a period of time as a homogeneous region. *In this paper we propose distributed algorithms to detect such regions, approximate their boundary with a piece-wise linear curve and track the boundary in real-time. Experimental results show the accuracy and efficiency of our detection and tracking algorithms.*

1. Introduction

Sensor networks are very useful tools for observing environmental phenomena. Typically, a large number of inexpensive sensors are deployed in some field to observe characteristics of interest [28]. Spatial division of the field, based on the similarity of the observed values, helps us to understand the physical properties of the observed phenomena. We call such regions in the sensor field, *homogeneous regions*. For example, an oil spill detected in the ocean is a homogeneous region (Figure 1). The sensors deployed around the origin of the spill can organize themselves into a network and communicate the measurements, to detect regions of varying oil concentrations.

Recent studies propose methods for delineating homogeneous regions by a boundary [7, 27]. However, these studies assume that the underlying phenomenon measured by the sensors can be quantified by a user specified predicate known a priori. For example, these predicates can be of the form "*all sensors observing temperature value* $>$ *40 degree*", or "*all sensors observing same mean temperature*". However, in several situations we need a more generalized grouping of the sensors, based on the sensor measurements over a time interval.

In this work, we address the problems of detecting and tracking such homogeneous regions in *real-time* when the definition of the phenomenon is *not* known in advance. Specifically, the problems we are considering are the following:

Problem 1. *(Homogeneous Region Discovery) Given N sensors monitoring an area of interest, find a group of sensors (corresponding to a region in space) such that the observations of the sensors belonging to the same group are* similar*, and are different from those of other sensors.*

Problem 2. *(Homogeneous Region Tracking) For a given region R defined by a set of* similar *sensors, track its movement over time.*

Since we may be dealing with fast moving events or with time-critical situations, we are required to meet real-time constraints to enable timely response to the events. In order to solve these problems, we need to address the following issues. First, to detect homogeneous regions, we need an efficient technique to identify sensors with similar readings. Such a technique has to be robust against spurious readings, and thus it has to estimate the distribution of several observations over a time interval. To be efficient it also has to operate in an online fashion. Second, we need to provide a formulation that allows us to define and discover homogeneous regions that differ from the surrounding area. Third, during tracking, we require that the user is informed about the *label* (i.e., the homogeneous region that the sensor belongs to) with minimum delays. Thus, we need an efficient tracking technique where only the updates of the labels are transmitted to the sink. Finally, we need to reduce resource consumption in the system during detection and tracking. This is essential in sensor networks since they are typically low-power systems.

*This research has been supported by NSF Awards 0330481 and 0627191.

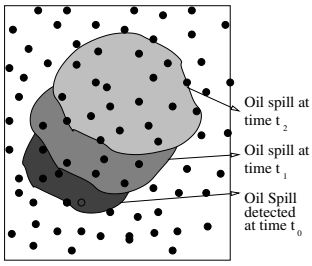


Figure 1. Spread of an oil spill detected in the ocean over time. Sensors are deployed in the ocean to detect and track the spread of the spill.

In a typical sensor network, the energy required to transmit data is higher than the energy required to process the data [29]. Hence it is often preferred to process the data *in-network* and communicate only the aggregates or results [15, 14]. Taking into account the above special characteristics of sensor networks, the framework we propose for detecting and tracking homogeneous regions operates in a distributed fashion. We assume a hierarchical, spatial decomposition of the field that enables the sensor network to perform localized processing, and then aggregate the results while moving up in the hierarchy. The spatial decomposition also enables real time tracking of the regions by enabling localized computation of the region *labels*. This is essential for real time tracking since with local label assignment the sensors can suppress transmission of values across the system to the *sink* sensor, thereby reducing delay and energy consumption. In addition, we observe that it suffices to monitor the movement of the boundary regions, and propose an algorithm based on a piecewise linear approximation of the boundary curve that can efficiently detect boundary lines and track their movement in a distributed manner.

The contributions of this work are as follows:

- We address the problem of detecting homogeneous regions by clustering together sensors with similar probability density functions, and we propose a distributed approach in order to minimize the energy dissipated through communication (Section 5).
- We achieve real-time tracking by approximating the boundary of the homogeneous regions by a piecewise linear curve, in a distributed manner (Section 6).
- We present experimental results to show the conditions under which we can detect homogeneous regions with high accuracy (Section 7).

2. Related Work

The problem of detecting boundaries between regions in a sensor network is addressed by Nowak and Mitra [27], and by Chintalapudi and Govindan [7]. Both studies assume that the underlying phenomenon is known a priori. In the first of these studies, *boundary* is defined as a delineation between regions of different mean values of sensor measurements. The method partitions the field to form a quad-tree structure, and prunes the tree back to get the tree which minimizes the *sum of squared errors*. In [7], the true boundary between the regions is approximated by an edge of finite thickness. Sensors gather information from their local neighborhood in order to decide if they lie on the boundary. Our approach is most similar in spirit with this second study. However, in our approach, we provide a robust computation of the region by using clustering techniques to dynamically find the regions with similar characteristics.

Ali et al. [1] propose an interesting approach to detect and track discrete phenomena (PDT) in sensor networks. A phenomena is said to take place in the sensor system when a set of sensors report similar discrete readings within a time window. Detecting phenomena with PDT is a centralized approach, demanding high energy for communication of all the measurements from the sensors to a sink. In our work, we consider a more general problem and we employ a distributed approach. Hellerstein et al. [16] propose algorithms to partition the sensors into *isobars*, that is, groups of neighboring sensors that have approximately equal values during an epoch. In our case we are partitioning the sensors according to the summary of their values over a time interval that spans several epochs. Moreover, we make no a priori decisions as to how to group sensors together based on their value ranges. Some recent studies propose an approach using dual space transformations, to track a non local phenomenon in sensor systems [21, 22]. This approach exploits the fact that when a region spreads, only the sensors at the frontier of the boundary change their region membership. Although this is an interesting approach, it is difficult to implement it in a distributed fashion to decide which sensors should be observing and which ones can be sleeping.

There is a sizable literature on the problem of tracking a point-target using a sensor network. An approach using linear regression and trigonometry methods is described by Brooks et al. [6]. A binary model for tracking a moving object in sensor networks is presented by Aslam et al. [2]. Hwang et al. [19] propose techniques for simultaneously tracking and maintaining identities of multiple targets. Nam et al. [24] propose a time-parameterized sensing task model that unlike previous models that assume that the sensing job's parameters are fixed at release time, allows the parameters to be described as time-varying functions. A distributed, collaborative approach is proposed with the *Dy-*

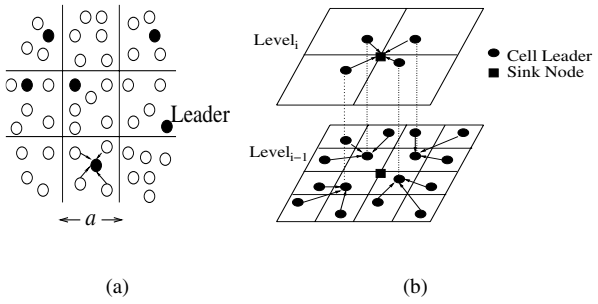


Figure 2. (a) Sensor field is divided into square cells of side a. (b) One of the sensors in the cell is selected as a *leader* sensor.

namic Convoy Tree-based Collaboration framework [36]. A cluster based approach for predictive tracking in sensor networks is proposed in [33]. In the same context, in [32] a *prediction-based* energy saving scheme is proposed that aims at reducing the energy consumption for object tracking, under the assumption that the target's movement remains constant for a certain period of time. In the context of real-time tracking, He et al. design a tracking framework that guarantees an end-to-end tracking deadlines [13]. The above studies describe efficient solutions for the problem of tracking a mobile object, represented by a single point in space. However our approach differs because we consider regions with changing spatial extends.

There is also work on multicast protocols that take into account spatio-temporal constraints [17, 18]. Finally, techniques have been proposed for clustering sensors that aim at minimizing the network's energy consumption [5, 35].

3. Sensor System and Communication Model

We assume a sensor network that consists of a set of sensors (each having a location on a 2-d plane) used to monitor and report observed measurements. When we deal with very large sensor networks, we have to consider the scalability of the query processing technique with respect to the size of the network. To that effect, we adopt a hierarchical organization for the sensor network, similar to the one used in [34]. The idea is to organize the network using overlapping virtual grids. We define several tiers for the grid with different levels of granularity, ranging from small local areas at the lowest tier, to the entire network area at the highest tier (see Figure 2).

At each cell at the lowest tier of the grid, there is one leader (or *parent*) node, that is responsible for processing the measurements of all the sensors in the cell. Moving up the hierarchy, the leader node of a cell collects values from the leader nodes of all its sub-cells in the lower level. The hierarchical decomposition of the sensor network, as well as the selection of the leaders for each level of the hierarchy, can be achieved using any of the techniques proposed in the literature [15, 26, 23]. These techniques ensure the leadership role rotates among the nodes of the network, and describe protocols that achieve this in an energy efficient way.

4. Estimating the Distribution of the Data

In this section we present a general framework for identifying sensors with similar readings. The fundamental idea is to compute an approximation of the underlying distribution of the sensor readings, and use this approximation to estimate how similar the readings of two different sensors are. Computing an approximation of the probability density function (PDF) of the readings allows us to efficiently compare the measurements from different sensors and combine the information from many sensors efficiently, thus minimizing the communication costs required to identify groups of similar sensors.

Since we are interested in detecting changes in the distribution of the values we consider the values in a sliding window W. The size of W can be kept small, but sufficient so that the process is robust in the presence of spurious readings. At each point in time, we want to approximate the distribution of the data values within the sliding window. This procedure is illustrated in Figure 3. The figure shows, for a particular time instance, the sliding window, and the distribution of the corresponding data.

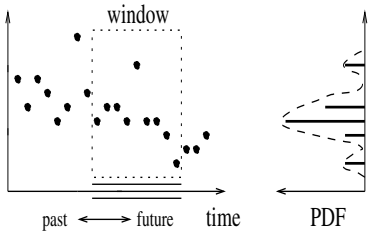


Figure 3. Estimation of the data distribution within a sliding window.

Estimating the Probability Density Function: There are several model estimation techniques that have been proposed in the literature, such as histograms [12], wavelets [11], kernel density estimators [30], and others. In our framework, we choose to estimate the distribution of the values generated by the sensors using *Kernel Density Estimators*, because of the following desirable properties: (i) they are efficient to compute and maintain in real-time

in a window streaming environment, (ii) they are non-parametric and can effectively approximate an unknown data distribution, (iii) they can easily be combined and (iv) they scale well in multiple dimensions.

Let M denote the set of measurements whose distribution we want to approximate, with values in the interval $[0, 1]$. (This requirement is not restrictive, since we can map the domain of the input values to the interval $[0, 1]$.) Let R be a random sample of M, and $k(x)$ a function, such that $\int_{[0,1]} k(x)dx = 1$, for all tuples in R. We call $k(x)$ the *kernel function*. We can now approximate the underlying distribution $f(x)$, according to which the values in M were generated, using the following function

$$f(x) = \frac{1}{|T|} \sum_{t_i \in R} k(x - t_i). \quad (1)$$

The choice of the kernel function is not significant for the results of the approximation [30]. Hence, we choose the Epanechnikov kernel that is easy to integrate:

$$k(x) = \begin{cases} \frac{3}{4B}\left(1 - (\frac{x}{B})^2\right) & \text{, if } |\frac{x}{B}| < 1 \\ 0 & \text{, otherwise} \end{cases} \quad (2)$$

where B is the bandwidth of the kernel function. In order to set B, we use Scott's rule [30]: $B = \sqrt{5}\sigma|R|^{-\frac{1}{5}}$, where σ is the standard deviation of the values in M.

We use kernel estimators for computing online an approximation of the data distribution at each sensor. The first step for creating a kernel estimator for a sliding window W is to maintain online a random sample of size $|R|$ of the set of the values in the most recent window W. The other quantity we need for the kernel estimator is the standard deviation σ of the values in the sliding window W. Both of these operations can be efficiently supported in a data streaming environment[31].

We use the "chain-sample" algorithm for producing a uniform random sample of size $|R|$, of a sliding window W (with $|R| \leq |W|$). The algorithm [3] starts with an initial random sample, and proceeds as follows. For each point in the sample, it picks at random the next element from the data stream that will replace it. The only restriction is that the new value must replace the old one, before the old one expires from the sliding window (that is, they should not be more than N values apart in the stream). The memory requirement is $O(|R|)$.

The estimate for the standard deviation of the sliding window is computed using a concise histogram along the time axis [4]. The estimate of the standard deviation is derived by combining the statistical information stored in all the buckets of the histogram. The memory required by this method is $O(\frac{1}{\epsilon^2}\log|W|)$, where ϵ is the maximum relative error we wish to tolerate in the estimation, and $|W|$ is the size of the sliding window.

Combining Multiple Estimator Models: This allows us to take the data distribution models of two different sensors in the network and construct a single model that describes the behavior of the data of both sensors. Our kernel estimators can be easily combined, and thus are well suited for our framework. There are two quantities that we have to combine, the sample set, R, and the bandwidth of the kernel function, B. We can combine the sample sets just by taking their union. We may then reduce the size of the resulting set by re-sampling, if necessary. In order to combine the bandwidths of two kernel functions, we only need to combine the two standard deviations upon which the bandwidths depend. This is accomplished using the same techniques as the ones for computing the standard deviation in a sliding window of streaming data [4].

Comparing Distributions: We now discuss how to compare the distributions of the values of two different sensors. To do that, we have to quantify the difference between two distributions. *Kullback-Liebler divergence* $D(p \parallel q)$ [8], is a well known and widely used technique, and is defined as $D(p \parallel q) = \int_y p(y)(logp(y) - logq(y))$ where $p(y)$ and $q(y)$ are probability distribution functions over y, and y is drawn from a finite set Y. However, the measure is undefined when $p(y) > 0$ but $q(y) = 0$ for some $y \in Y$. The $KL - divergence$ is therefore not applicable to the density distributions derived by kernel density estimation method, because this method may assign probability of zero for regions in the domain of the values. We use a variation of the KL-divergence, called the *Jensen-Shannon divergence* [20] which is defined as follows

$$JS(p, q) = \frac{1}{2}\left[D(p \parallel avg(p, q)) + D(q \parallel avg(p, q))\right] \quad (3)$$

where $avg(p, q)$ is the average distribution $(p(y) + q(y))/2$.

We estimate the JS-distance between two kernel estimator models $p(x)$ and $q(x)$ as follows. We approximate the estimated distribution with the values of the function with a finite set of grid points $b_1, b_2, \ldots, b_k$, where $b_{i+1} - b_i = bs$. Let $P_p(x, y) = \int_{x-y}^{x+y} p(x)dx$ (P_q is similar). We approximate the term $D(p \parallel avg(p, q))$ in Equation 3 as follows which can be done in $O(k|R|)$ time:

$$D(p \parallel avg(p, q)) = \sum_{i=1\ldots k} P_p(b_i, bs/2) \times \left[log(P_p(b_i, bs/2)) - log(\frac{P_p(b_i, bs/2) + P_q(b_i, bs/2)}{2})\right] \quad (4)$$

5. Distributed Detection of Homogeneous Regions (DDHR)

In this section we describe our technique for identifying homogeneous regions in a sensor field. Let us assume that each sensor s_i computes the values it observes,

v_i, and that there exists a user-specified threshhold δ such that two sensors are considered similar if $JS(v_i, v_j) \leq \delta$. To solve this problem, we have to cluster the sensors using the JS-divergence as a distance metric, thus identifying groups of similar sensors. Let us first consider the brute-force method where all the sensors transmit their pdf model to the sink and the sensors are clustered based on their v_i, $i = 1, \ldots, N$. This method demands high amounts of energy since it requires transmission of the kernel samples and the bandwidth from each sensor to the sink. Moreover, it incurs high latency in transmission due to the large number of packets sent across the network. We propose a distributed technique, where we detect homogeneous regions at each cell in the grid, and then communicate only the summary information of each cell to the leader in the next higher level in the network.

The sensors at the lowest level of the communication tree transmit their model parameters to their leader (as shown in steps of part (a) in Figure 4). This is followed by the two processes (Part (b) in Figure 4) described below:

1. Finding local regions within a cell: The leader of a cell collects the v_i from all the other sensors in the cell. These density functions are then clustered according to a clustering algorithm with *JS-divergence* as the distance measure.

The density estimator models belonging to a cluster are combined as described in Section 4, and the combined model is chosen to be the representative of that cluster. These representative estimator models are the *local representatives* of the homogeneous regions present inside the cell. These local representatives are then communicated to the leader at the next higher level. (We will use the term *representative* and *representative distribution model* synonymously.)

2. Finding homogeneous regions in the field: At any intermediate level of communication, the leader sensor collects local representatives from all the four leaders at its immediate lower level. These representative estimator models are then clustered. A new set of representative models for each cluster is transmitted to the leader at the next higher level (if any). Finally, at the sink, the combined estimator models of each cluster corresponds to one homogeneous region present in the field. We denote these representative models as *region representatives*.

We observe from the above two steps that only a few estimator models are clustered at a leader sensor, both at the cell level and at any intermediate level. We conducted experiments to compare two different clustering algorithms *hierarchical clustering* and *k-means* in our context and found that the hierarchical clustering approach gives better results. Therefore, we use hierarchical clustering of the estimator models to identify the homogeneous region within a cell, and in the field.

Distributed Detection of homogeneous regions(DDHR)
(a) At sensor j where j is not a cell leader
(v_j is the estimated *pdf* of the values observed at sensor j)
1. $CReps_j \leftarrow v_j$
2. Send $CReps_j$ to $parent(j)$

(b) At sensor i, where i is a cell leader (N_i denotes the number of clusters obtained at sensor i, by grouping the cluster representatives from its children. λ_{ik} denotes the representative *pdf* of the k^{th} cluster observed at sensor i)
1. Receive $[CReps_m]_{m \in Children(i)}$
2. $(\lambda_{i1}, \lambda_{i2}, ..., \lambda_{iN_i}) \leftarrow Cluster([CReps_m]_{m \in Children(i)})$
3. $CReps_i \leftarrow (\lambda_{i1}, \lambda_{i2}, ..., \lambda_{iN_i})$
4. if i is not the sink
5. Send $CReps_i$ to $parent(i)$
6. else
7. **Regions** $\leftarrow CReps_i$

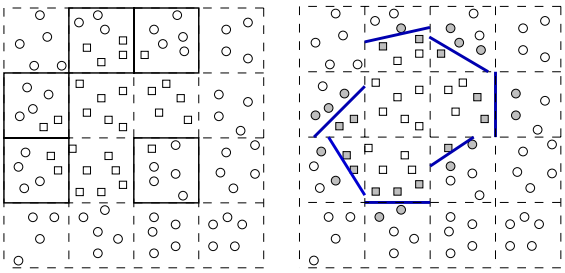
Figure 4. Outline of the detection algorithm.

The above steps for region identification are followed by a *label propagation process* as described below. Homogeneous regions in the sensor field are identified by their representative estimator models. Each of these estimator models i.e. the global representatives, are assigned a unique label. The labels are then communicated to the individual sensors following the same communication tree, as described below.

The sink and the leader nodes of each hierarchical level stores the mapping of the cluster representatives transmitted from the lower level to the cluster representatives (of the current level) that they are grouped under. The sink now transmits pairs of $(label, representativemodel)$ to the children nodes. At any intermediate level, a leader receives label pair(s) from its parent, assigns labels to its children representatives and communicates them to its children. This is repeated until the cell-leader level is reached. Thus, at the end of this step, the cell leaders have label(s) corresponding to the field region(s) present in their cell.

For example, let $(\lambda_{s1}, \lambda_{s2}, ..., \lambda_{sN_s})$ be the cluster representatives at the sink. For each k, the sink maintains a mapping of all the lower-level representatives $C' \subset [CReps_m]_{m \in Children(sink)}$ grouped under λ_{sk}. The sink communicates (k, λ_{sk}) to C' (assuming k is the label for the homogeneous region with representative representative model λ_{sk}.). This process is repeated until the leaf level sensors and the nodes are assigned labels accordingly.

We observe that there are two types of cells depending on the number of homogeneous regions their leaders identify - boundary cells, and interior (homogeneous) cells. This is illustrated with the highlighted cells in Figure 5(a). If



(a) (b)

Figure 5. (a) Region Detection: Sensors belonging to different regions are shown with different markers. The highlighted cells are the *boundary-cells* consisting of more than one region. (b) Tracking: The dark nodes are the active sensors i.e. the nodes within distance α from the estimated boundary.

we know in advance that the homogeneous regions that we want to identify are at least δ different from each other, then we can determine if a cell is a boundary cell in the initial steps of the algorithm in Figure 4. With hierarchical clustering algorithm, this is achieved by having a condition that two clusters should be combined at a stage when $JS(\lambda_j, \lambda_k) < \delta$, where λ_j and λ_k are the representative estimator models of the clusters.

If the cell leader has more that one representative model after this step, it labels itself to be a boundary cell and transmits the representatives models of the local regions to the leader at the next higher level. If on the other hand, the step results in only one representative, the cell is assumed to consist of only one homogeneous region, and the leader sensor *does not* communicate the representative model to its parent. We will refer to this variation of DDHR as DDHR-SC.

6. Tracking of Homogeneous Regions

In many practical applications involving sensor networks, it is necessary to not only detect the homogeneous regions, but also to track their spread over time. In our problem setup, the sink determines the homogeneous regions in the sensor field and identifies the interesting region, for example, the region under the influence of the spill, that needs to be tracked. It then transmits a query to the sensors to track the spread of this region.

The spread of a region is characterized by some sensors getting added to the *region* and some others getting removed. Therefore, it is required that we monitor the distribution of the observations at the sensors continuously to determine the regions they belong to. A simple approach to tracking the spread of a region is to iteratively detect the regions with the technique discussed in Section 5.

Our approach is to compute the new region labels for the sensors at the cell level leaders. This approach succeeds in reducing the latency and energy consumption, by evading transmission of model parameters to the sink sensor. However, the power analysis of sensors show that they dissipate high amount of energy being *alive* (in mW compared to μW for *sleep*), irrespective of whether they are measuring the environmental phenomena or listening or transmitting. Thus, we propose the following technique wherein only a fraction of the sensors continue observing over the tracking duration.

6.1. Tracking the boundary of the region

In this section, we propose a method to track the regions by approximating the region boundary with a boundary curve and alerting only the sensors that are within a distance α from the boundary curve to observe. The other sensors switch to *sleep* mode to conserve energy.

Distributed Detection of Boundary Lines: Once the sensors within the cells are assigned their global region labels as described in Section 5, the cells that contain multiple homogeneous regions approximate the boundary between the homogeneous regions by a line segment. In [7], the authors employ a *classifier based* approach ([10]) to examine if a sensor is located within certain threshold distance from the boundary line (we will refer to this as LED-ClassifierLine). In this section, we describe how we adapt the *classifier based* method to approximate the boundary of the region in a distributed way (DDBL). To simplify the discussion, we assume there are at most two regions in a cell, Region 1 and Region 2, and the classifier in a linear classifier. Let S_g be the set of sensors in cell g. Let the boundary line be of the form $l(a, b, c) = (ax + by + c = 0)$. For each sensor i in S_g, let F_i be defined as

$$F_i = \begin{cases} 1 \textit{ if } i \textit{ belongs to Region 1} \\ -1 \textit{ if } i \textit{ belongs to Region 2} \end{cases}$$

Classifier score $CS(a, b, c)$ for line l is defined as $CS(a, b, c) = |\sum_{i \in S_g} F_i S(ax_i + by_i + c)|$ where

$$S(x) = \begin{cases} -1 \textit{ if } x \leq 0 \\ 0 \textit{ if } x = 0 \\ 1 \textit{ if } x > 0 \end{cases}$$

and (x_i, y_i) is the location of sensor i. We consider l_{max}, the line with the highest classifier score, as the boundary line within the cell. When a cell consists of more than two homogeneous regions, boundary lines between all pairs of regions can be approximated in a similar way.

Cell Borders as the Boundaries: An additional step is required in cases where the boundary between regions coincides with the borders of the cells. Two adjacent cells which are completely inside two different homogeneous regions

will not detect a boundary between them with the above boundary line detection method. Such boundary lines i.e., boundaries lying on the cell borders, are easily identified by passing a message between the adjacent cells. In a field of two regions, if a non-boundary cell shares a cell border with another non-boundary cell, and if both of them have different region labels, the common cell border is identified as the boundary line between the two regions. In a field with more than two regions, this can be easily extended, where each region label in a cell is checked against labels of its adjacent cells.

6.2. Tracking Algorithm

Assuming we have an initial set of estimated region boundary lines L_0 at time t_0 in the field, we estimate the set of boundary lines L_1, L_2, ... at the successive times $t_\tau, t_{2\tau}$, ... where τ is the time interval between successive detections. The time interval τ is the time taken at a sensor to observe a window W of values i.e., $r\tau = |W|$ where r is the sample rate at the sensors.

Leader sensors of the boundary cells store information about the boundary lines that are within their cells (or on the cell borders). Given L_i, the number of sensors alive for estimating L_{i+1} can be minimized by keeping alive only those sensors which are within a specified distance α from any of the lines in L_i. For the time duration $t_{i\tau}$ to $t_{(i+1)\tau}$, we define the following *active* quantifier:

- *Active zone:* The area within a cell where for any point P in the area, *Euclidean Distance*$(P, l_{ke}) < \alpha$ for some $l_{ik} \in L_i$, where l_{ik} is the k^{th} line in the set L_i. (Note here that l_{ik} is the line segments within the cell, and not a line with infinite length.)
- *Active sensor:* if it lies within an active zone.
- *Active cell:* if any of the sensors within the cell is active.

Figure 6 illustrates the steps for updating the boundary information during the time interval $t_{i\tau}$ to $t_{(i+1)\tau}$. The active sensors continue observing during the time interval, and obtain a new estimator model of their observations. The leader sensor of an active cell receives the new estimator models and relabels the active sensors as shown in Steps 2-4 of the Algorithm (b). The function *ClosestRepresentative* is implemented by a simple look up table of distances from all the region representatives. The corresponding label of the closest representative is assigned to the sensor under consideration. When the new labels of a sensor are different from the labels during the earlier time interval, the leader sensor transmits this information to the sink.

Once all the sensors are assigned new labels, a new boundary line within the cell is estimated. If all the sensors within a cell have same label, the leader sensor communicates (as discussed earlier) with leaders of neighboring cell to check if any of its border is a boundary line. All member sensors within a distance α from the boundary line are set as active sensors and they are alerted to continue observing over the next time interval. With DDHR-SC, the methods described for detecting and tracking boundary lines are modified as follows. (a) The non-boundary cells are not labeled with the *label back propagation step*. Therefore, the boundaries which coincide with the cell borders are detected by comparing the representatives of the straddling cells, and not by comparing the labels. (b) During tracking, the region label pairs $[k, \lambda_k]$ are communicated along with the boundary line information to the neighboring cells. This enables the neighbor cell to label all the member sensors with the label information received, and alert the active sensors, if any, to observe.

Distributed Tracking of homogeneous regions during $(i+1)^{th}$ time step

(a) At an *active* sensor j where j is not a cell leader

1. $v_j \leftarrow$ Estimator model measurements during $t_{i\tau}$ to $t_{(i+1)\tau}$
2. send v_j to $parent(j)$

(b) At a sensor k where k is the leader of an *active* cell

1. Receive $[v_j]_{j \in Children(k)}$ from *active* children
2. for each $j \in Children(k)$ where j is an active sensor
3. $\lambda = ClosestRepresentative(v_j)$
4. $Label_{i+1}(j) \leftarrow RegionLabel(\lambda)$
5. if $(Label_{i+1}(j) \neq Label_i(j))$
6. transmit $Label_i(j)$ to $parent(k)$
7. Get new boundary line
8. Alert current $active$ sensors to continue observing and other sensors to *sleep*

Figure 6. Outline of the tracking algorithm.

While the above method conserves energy and reduces delay in tracking, any *new* regions (or holes) that evolve during the tracking process are not identified and are grouped with the existing region that is closest in terms of the estimator model. A typical example is tracking of oil spills where the concentration of the spill may decrease with time when the spill spreads over a wide area. In such a scenario, the new regions (i.e. the new concentration) are detected by having the system run the *detection* algorithm periodically. The frequency of this is computed based on the availability of the resources and the requirements of the tracking application.

7. Experimental Evaluation

Implementation. We built a simulator to evaluate our framework, implemented on top of the TAG [25] simulator. Specifically, we use the TAG simulator infrastructure in order to define the topology of the network and the type of messages exchanged, to disseminate queries, and to gather statistics. We also made the necessary modifications to enable the hierarchical organization of the nodes in the sensor network. We build the network hierarchy as described in Section 3.

Our implementation required $5,000$ lines of Java code. However, the code that implements our algorithm has very small footprint. For instance, the kernel density estimation, detection and tracking module (that is, the code that would have to run on a sensor to implement our algorithms) required a total of 200 lines of Java code.

For the experiments, the default value of the number of sensors is 1000 and they are assumed to be positioned at random locations in a square field of side $100m$. We assume that the field consists of two regions R_1 and R_2 with their representatives (distribution functions) as N_1 and N_2 respectively. The *distance* between the regions d_{R_1,R_2} is the JS-divergence between the representatives.

We generate measurements at sensors based on the representative distribution function of the homogeneous region they geographically belong to. In our experiments, each sensor keeps a window of 50 values of the observed feature.

For the experiments, we consider regions of elliptical (R_1 is inside the ellipse and R_2 is outside the ellipse) and linear (R_1 and R_2 on either side of the line) boundaries.

Accuracy of detecting homogeneous regions: In the first set of experiments, we evaluate the accuracy of the distributed detection algorithm (DDHR). We compare the accuracy with a **centralized approach** based on [9], where a multi-variate Gaussian model built from the data seen so far is used for detecting the homogeneous regions. We consider the Gaussian *pdf* over n sensors, $p(X_1, X_2, ..., X_n)$. The *pdf* is expressed in two parameters: length-n vector of means, μ and a $n \times n$ matrix of covariances, Σ. We detect the homogeneous groupings of the n sensors as follows.

1. For each pair of sensors (i, j), calculate the probability of both the sensors having the same measurement, within a confidence limit of ϵ, as $\int_{-\infty}^{\infty} \int_{x_i-\epsilon}^{x_i+\epsilon} p(X_i, X_j) dx_j dx_i$ where $P(X_i, X_j)$ is the density when $p(X_1, X_2, ..., X_n)$ is marginalized or projected over sensors X_i and X_j. This probability is a measure of similarity between the sensors i and j.

2. Cluster the sensors based on this similarity measure.

We refer to the above method as Gaussian-Fit and compare the performance of DDHR and Gaussian-Fit approaches based on the metric *percentage of mislabeled sensors*.

Figures 7(a) and (b) show the percentage of mislabeled sensors as a function of distance between the two regions d_{R_1,R_2}, for various sample sizes, where the representative distributions are Gaussians and Zipfian. The accuracy of DDHR and Gaussian-Fit depends on the number of observations used to build the *pdf* estimator. We consider the case where 50% and 70% of the total measurements observed are used. It can be seen that as the distance between the representatives of the regions, d_{R_1,R_2} decreases, i.e., as the regions become less distinguishable, more observations are required in order to accurately detect the homogeneous regions. For example, with DDHR, to obtain a 95% accuracy in labeling, we require only 50 observation if the distance between the regions is greater than 0.0035, whereas 70 observations are required if the distance between the regions is 0.003.

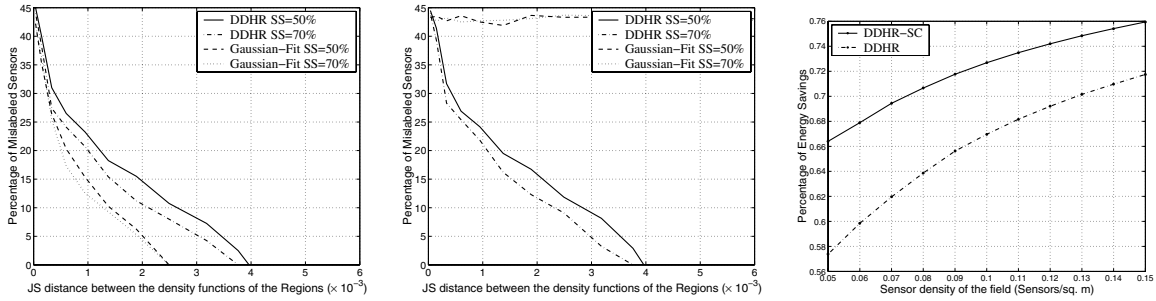
The figures also show that, as expected, Gaussian-Fit performs better than DDHR when the underlying distributions of the regions are Gaussians. However it fails to detect the homogeneous regions when the underlying distributions of the regions are Zipfian.

Communication Overhead: In the next set of experiments we compare the energy consumed for homogeneous region detection by DDHR and DDHR-SC, compared to the centralized approach (i.e., all the sensors communicate their estimated density parameters directly to the sink node). We use a simple energy model where the radio dissipates $E_{elec} = 50nJ/bit$ to run the transmitter or receiver circuitry and $\epsilon_{amp} = 100pJ/bit/m^2$ for transmission to achieve an acceptable signal to noise ratio. The sensors communicate through multi-hop communication.

We vary the sensor density of the field from 0.05 $sensors/sq.m.$ to 0.15 $sensors/sq.m$ to study the scalability of the algorithms with respect to energy consumption. In the experiments, 50% of the measurements (i.e, 400 bits) at each sensor are communicated as the kernel sample. The region boundary is circular with radius $20m$ and is centered at (62.5, 62.5).

Figure 7(c) shows the percentage of energy saved with DDHR and DDHR-SC compared to centralized detection of homogeneous regions with Gaussian-Fit. Due to the local clustering at each cell with DDHR, only the representatives of the clusters are communicated across the network saving 57% to 72% of the total energy dissipated when all the sensors communicate their estimator models to the sink. In the case where only the boundary cells communicate their estimated density (DDHR-SC), we observe savings between 66% to 76%. The figure also illustrates that our detection algorithm scales well with the increase in sensor density in the field.

Figure 8 shows the energy consumed for boundary detection with our distributed approach (DDBL) and with the classifier line approach described in [7] (LED-



(a) (b) (c)

Figure 7. Percentage of mislabeled sensors as a function of the JS distance between the density functions of the two regions, for various data sample sizes (SS), where (a) the representative distributions are Gaussians (b) the representative distributions are Zipfian. (c) Energy savings with DDHR and DDHR-SC compared to centralized detection of homogeneous regions.

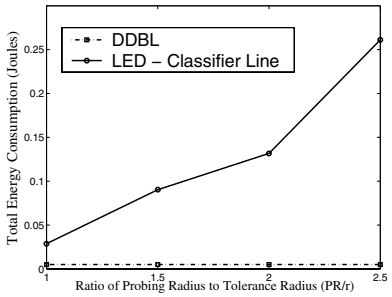


Figure 8. Energy consumption for boundary line detection using DDBL and LED-ClassifierLine.

ClassifierLine). The energy consumed in DDBL includes (a) the energy required for communicating the label information between cell leaders and sensors within the cells, and (b) the energy required for communication across straddling cell leaders to check if the common border is a boundary.

In LED-ClassifierLine, each sensor collects label information from all the sensors within a probing radius PR, to identify if it is within a tolerance radius r from the boundary line. The accuracy of the method increased with increase in $\frac{PR}{r}$. The authors show in [7] that a value of $\frac{PR}{r} = 2$ yields boundary detection rate of 90% or better. We assign the value of r to be equal to the communication range R, as in [7].

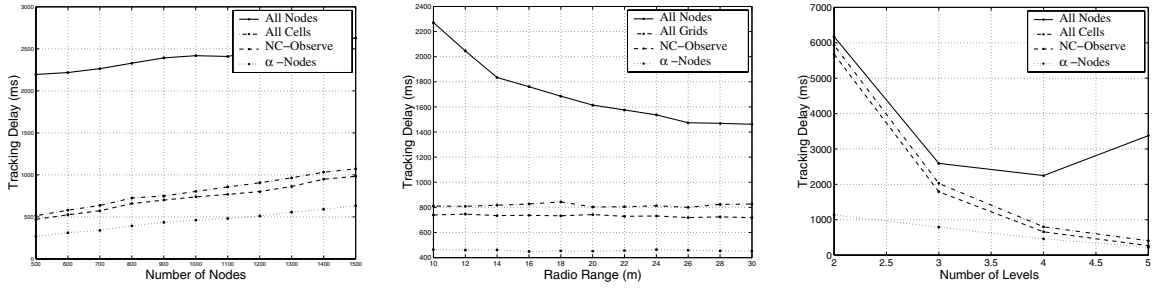
Figure 8 illustrates that DDBL is superior to LED-ClassifierLine in terms of energy consumption. This is due to the fact that, in DDBL, the cell leaders collect label information from the sensors within the corresponding cells and determine the boundary line whereas LED-ClassifierLine involves multiple communication across all the sensors to detect the boundary.

Delay in Tracking: In the first set of experiments for tracking, we evaluated the delays in tracking a region of interest. We compared the delay incurred while tracking a plume with the following approaches (i) *All-Nodes*, where a region is tracked by continuously querying the system to detect regions, (ii) *All-Cells*, where the labels are assigned at cell level, (iii) *NC-Observe*, where only the sensors in the boundary cells and their neighbor cells continue observing and report label updates and (iv) α*-Nodes*, where we track the boundary line of the region as described in Section 6.2 (with $\alpha = 5$).

In the experiments, the default value for the number of levels in the communication hierarchy is 4 and the communication radius is $30m$. Delays are caused due to the transmission of values from the nodes to their leaders and the time delay between tracking. Figure 9(a) illustrates the delays for various values of *node density* of the field. We observe that the delay for *All-Nodes* is 3 to 5 times higher than that obtained with the other approaches. This is expected since in the other approaches the new labels are computed locally, reducing the traffic in the system reduces significantly when compared to *All-Nodes* where the estimator models are transmitted to the sink for computation of the labels. Figures 9(b) and (c) present the delays for various values of communication radio range R and the number of levels in the communication hierarchy (which in turn determines the average number of nodes in a cell). We observe that localized computation of labels is a very efficient approach toward reducing the delay during tracking of regions.

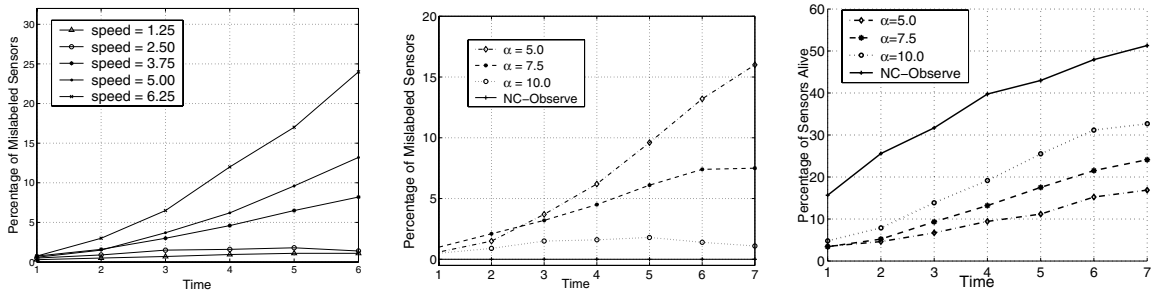
Tracking Accuracy: In our next set of experiments we evaluate the accuracy of our tracking technique in terms of the percentage of mislabeled sensors during successive time intervals of tracking. The accuracy in assigning correct labels to the sensors depends on (a) the ratio of the rate of spread of boundary to α, (b) accuracy of the linear approximation of the boundary between the regions inside cells, (c) distance between the representatives d_{R_1,R_2} and (d) the shape of the boundary.

We consider regions of $d_{R_1,R_2} > 0.005$ so that the simi-



(a) (b) (c)

Figure 9. Delay in tracking for various values of (a) Number of Nodes (b) Communication Radius and (c) Number of Levels in the hierarchy.



(a) (b) (c)

Figure 10. (a) Percentage of mislabeled sensors when varying the plume spreading speed. Tracking a plume ((b) and (c)).

larity between the homogeneous regions does not contribute to the error. (Refer to Figure 7(a).) Boundary lines within cells are approximated using the DDBL method described in Section 6.1.

Figure 10(a) shows the percentage of mislabeled sensors while tracking a plume centered at (62.5, 62.5), for various values of the speed (i.e. the rate of spread) of the plume (results for different plumes are similar, and are omitted for brevity.) α is kept constant at $5m$ and the rate of spread of the plume is varied between $1.25m$ and $10m$ per time interval of tracking. We run the experiment until the plume spreads out of the sensor field. The accuracy of tracking the plume decreases with increasing plume speed and is less than 15% when the speed of the plume is less than $5m$. There is a significant increase in the percentage of mislabeled sensors when the speed increases from 5 to 6.25. This is due to the fact that when the rate of spread of the plume is greater than α, the region boundary spreads beyond the set of live sensors, and thus becomes infeasible to track the boundary with high accuracy.

Figures 10(b), (c) show the percentage of mislabeled sensors while tracking the plume for various values of α. The speed of the plume is kept constant at $5m$ per time interval of tracking. As the boundary of the plume spreads out of the field after time instance 7, the experiment is conducted only till time instance 7. We observe in Figure 10(b) that the error remains below 17% for $\alpha = 5m, 7.5m$ or $10m$. Figure 10(c) shows the percentage of sensors alive while tracking the plume. The figure shows that only a maximum of 33% of the sensors are alive during tracking with $\alpha = 10m$, while 52% of the sensors are alive with $NC - Observe$.

We observe from the experiments that while all localized computation approaches are efficient in reducing the tracking delays, tracking in terms of the boundary of the region, with only α-*nodes* being *alive*, play a significant role in reducing the energy consumption of the system. Moreover the error due to tracking in terms of the boundary is under 5%, with favorable values for the parameter α. Thus our approach is efficient in tracking the regions in real time while reducing the energy consumption of the system.

8. Conclusion

This paper introduces the problem of identifying and tracking homogeneous regions in a sensor field. We present distributed techniques for clustering together sensors with similar observations over a time interval. We estimate the pdf of the data observed in each sensor and propose a distributed approach to clustering the pdfs from all the sensors to obtain homogeneous regions, and to efficiently track their boundaries in real-time. We experimentally evaluate and validate the performance of our approach.

References

[1] M. H. Ali, M. F. Mokbel, W. G. Aref, and I. Kamel. Detection and tracking of discrete phenomena in sensor-network databases. In *SSDBM*, pages 163–172, Santa Barbara, CA, 2005.

[2] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a Moving Object with a Binary Sensor Network. In *SENSYS*, Los Angeles, California, USA, Nov 2003.

[3] B. Babcock, M. Datar, and R. Motwani. Sampling From a Moving Window Over Streaming Data. In *SODA*, 2002.

[4] B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan. Maintaining Variance And k-medians Over Data Stream Windows. In *PODS*, pages 234–243, San Diego, CA, USA, 2003.

[5] S. Bandyopadhyay and E. J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *INFOCOM*, San Fransisco, CA, 2003.

[6] R. Brooks, P. Ramanathan, and A. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8), Aug. 2003.

[7] K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. *Ad-hoc Networks Journal*, 2003.

[8] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & sons, 1991.

[9] A. Deshpande, C. Guestrin, and S. Madden. Using probabilistic models for data management in acquisitional environments. In *CIDR*, Asilomar, CA, Jan 2005.

[10] R. O. Duda, P. E. Hart, and D. G. Stork. *Patterrn Classification*. John Wiley and Sons, Inc., 2001.

[11] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries. In *VLDB*, Rome, Italy, 2001.

[12] S. Guha and N. Koudas. Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation. In *ICDE*, pages 567–576, San Jose, CA, 2002.

[13] T. He, P. A. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q. Cao, J. A. Stankovic, and T. F. Abdelzaher. Achieving real-time target tracking using wireless sensor networks. In *IEEE RTAS 2006*, San Jose, California, April 2006.

[14] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. In *SOSP*, New York, USA, Oct. 2001.

[15] W. R. Heizelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *HICSS*, Maui, Hawaii, Jan. 2000.

[16] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *IPSN*, pages 63–79, 2003.

[17] Q. Huang, C. Lu, and G.-C. Roman. Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints. In *IPSN*, Apr 2003.

[18] Q. Huang, C. Lu, and G.-C. Roman. Spatiotemporal multicast in sensor networks. In *Sensys*, Los Angeles, CA, Nov 2003.

[19] I. Hwang, H. Balakrishnan, K. Roy, J. Shin, L. Guibas, and C. Tomlin. Multiple target tracking and identity management. In *2nd IEEE Sensors*, pages 36–41, Toronto, Canada, Oct 2003.

[20] J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Infor. Theory*, 37:145–151, 1991.

[21] J. Liu, P. Cheung, F. Zhao, and L. Guibas. A dual-space approach to tracking and sensor management in wireless sensor networks. In *WSNA '02*, NY, USA, 2002. ACM Press.

[22] J. Liu, P. Cheung, F. Zhao, and L. Guibas. Apply geometric duality to energy efficient non-local phenomenon awareness using sensor networks. In *IEEE Wireless Communication Magazine, special issue on Wireless Sensor Networks: Theory and Systems*, Dec 2004.

[23] J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao. Distributed Group Management for Track Initiation and Maintenance in Target Localization Applications. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, Palo Alto, CA, Apr 2003.

[24] M.-Y.Nam, C.-G. Lee, K. Kim, and M. Caccamo. Time-parameterized sensing task model for real-time tracking. In *Proc. of IEEE RTSS*, Miami, FL, December 2005.

[25] S. Madden, M. J. Franklin, and J. M. Hellerstein. TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *OSDI*, Boston, Massachusetts, Dec 2002.

[26] N. Malpani, J. Welch, and N. Vaidya. Leader Election Algorithms for Mobile Ad Hoc Networks. In *Proc. Fourth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 2000.

[27] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *IPSN*, pages 80–95, Palo Alto, CA, 2003.

[28] G. J. Pottie and W. J. Kaiser. Embedding the internet: Wireless integrated network sensors. *Communications of the ACM*, 99(7):1–100, January 1999.

[29] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, (5), 2000.

[30] D. Scott. *Multivariate Density Estimation: Theory, Practice and Visualization*. Wiley & Sons, 1992.

[31] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online Outlier Detection in Sensor Data Using Non-Parametric Models. In *VLDB*, Seoul, Korea, Sept. 2006.

[32] Y. Xu, J. Winter, and W. Lee. Prediction-based strategies for energy saving in object tracking sensor networks. In *MDM'04*, Berkeley, CA, January 2004.

[33] H. Yand and B. Sikdar. A protocol for tracking mobile targets using sensor networks. In *IEEE International Workshop on Sensor Networks Protocols and Applications*, Palo Alto, CA, Apr 2003.

[34] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A Two-Tier Data Dissemination Model for Large-Scale Wireless Sensor Networks. In *MOBICOM*, Atlanta, GA, USA, 2002.

[35] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid,energy-efficient approach. In *IEEE INFOCOM*, Hong Kong, China, Mar. 2004.

[36] W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *IEEE INFOCOM*, Hong Kong, China, Mar. 2004.