Resilient and Energy Efficient Tracking in Sensor Networks

Maria Halkidi, Dimitris Papadopoulos, Vana Kalogeraki, Dimitrios Gunopulos

Department of Computer Science & Engineering, University of California, Riverside, CA 92521, USA, E-mail: {mhalkidi, dimitris, vana, dg}@cs.ucr.edu

Abstract: One of the main applications of sensor networks is to detect and monitor transient events of interest, or objects as they move or spread through an area. To track such objects efficiently, and accurately we need distributed mechanisms that allow the cooperation of many sensors and the exchange of real-time data. This cooperation must also be achieved in the presence of possible failures and within the constraints of the sensor nodes and the established network. In this paper we present a new distributed mechanism for tracking moving objects that addresses these desiderata. Our mechanism provides efficient setup and cooperation of the sensors within the network, while providing fault tolerant characteristics through replication. We also provide an algorithm for predicting, with high probability, the future location of an object based on the past observations of many sensors. We empirically evaluate the performance of our approach and our simulations demonstrate its efficiency and accuracy.

1 INTRODUCTION

Advances in wireless communication technologies and micro-electronic-mechanics have enabled the deployment of large-scale sensor networks. *Sensor networks* are systems of many small and simple wireless devices (further referred to as sensor nodes) distributed over a vast field in an attempt to sense and monitor events of interest or track objects as they move through the field. These sensor nodes are equipped with sensing, communicating and data processing units, which allow them to collect, exchange and process information about the monitored events in their environments. These characteristics of sensor nodes make them attractive and applicable in various domains such as target tracking, surveillance, battlefields, environmental control and security management [33].

The problem of distributed tracking of moving objects is fundamental for several of these applications. In such domains, possible scenarios of interesting events could be the movement of enemy or vehicles in battlefields or the movement of wildlife in forests. For example, in habitat monitoring, the requirement is that sensors are able to monitor the occupancy of the wildlife and collect data from the physical environment where the wildlife resides, such as tem-

perature, light and pressure. To achieve this, distributed tracking requires collaboration among the sensor nodes (typically the exchange of a few bits of data) to gather information from other sensors within a region of interest. These will be used to estimate the location of an event, predict its trajectory and warn appropriate sensors as the event is approaching.

There are a number of fundamental issues that we need to address in developing such a tracking system: (1) sensor nodes must be positioned strategically to detect the event of interest, (2) having detected the event, its location must be estimated, (3) a distributed tracking algorithm must be employed to predict the trajectory of the event accurately based on multiple sensor measurements detecting the same event, and (4) information must be generated and propagated dynamically and in real-time to other sensor nodes to alert them about the approaching event.

Solving this problem dynamically and in a completely distributed fashion creates additional challenges:

- **Energy Savings.** Sensors are typically small devices with limited resource capabilities, including communication bandwidth, CPU capacity and battery lifetime. A key issue in designing the tracking system is to reduce energy consumption. Frequently sending and receiving messages among many nodes can consume a lot of power and lead to increased collisions in the network. To save energy, distributed data aggregation at multiple locations in the sensor network is essential. This will reduce transmission, improve resource savings and distribute the load across multiple nodes.

- **Resilience.** In a sensor network, nodes can fail due to energy depletion, they can temporarily become unreachable or their readings may drift and lose calibration due to environmental interference. Therefore, we cannot rely on measurements made by a single node as this may lead to inconsistencies in determining the location or movement of an event. This requires us to exploit some amount of redundancy, so that we improve reliability and increase the accuracy of the esti-

mates.

- **Power conservation and quality of monitoring.** An area can be monitored perfectly if it is entirely covered with a set of sensor node. However the sensors have limited power and thus the quality of monitoring becomes inversely proportional to the life time of the network. Hence, the addition of a warning mechanism in the target tracking becomes important. This implies a mechanism that can accurately predict the next monitoring area, so that we can wake up all the nodes surrounding the target to participate in tracking the moving object.

- **Presence of obstacles.** In addition, the presence of obstacles in the area (such as the location of bridges or rivers in tracking a vehicle in a field) could influence the direction of the moving event, hinder the communication among the sensors and affect the behavior of the tracking algorithm. Thus, the presence of such obstacles have to be taken into account in the tracking and warning algorithms.

In this paper we propose a two-level approach to address the resilient and efficient tracking in sensor networks. The two-layers consist of:
(a) a *local low-level loop* executed at individual sensors whose functionality is to detect the presence of a mobile target (an event) and estimate its trajectory based on previous history and using local information only, and
(b) a *global high-level loop* executed across multiple sensor nodes to combine individual estimates made at single sensors and predict with high probability its trajectory across the system.

The system uses a grid structure divided into cells; each cell has a sensor, with a special role, called *leader* that is responsible to collect tracking data from all the sensors in its cell and communicate with other leaders. The global high-level loop of the system is executed by the leaders. To conserve energy, we let non-leader nodes to sleep; these will subsequently be activated by the leaders. By collecting tracking data from its cell, the leader estimates the trajectory of the event and decides dynamically which subsequent

cells have to be activated in order to carry on the tracking process. Based on this decision, the leaders of the affected cells are notified to continue the tracking. To save energy and accommodate sensor faults, the leader is not fixed. Instead, different nodes in the cell take the role of the leader. This is done in order to balance the consumption of energy, particularly because the leader, apart from the tracking task, is also responsible for doing local computations in order to determine what is the next monitoring area. To predict locally the location of a target, we implement a well-known tracking technique (Kalman filters[18]) locally at each sensor node. Kalman filters are efficient prediction techniques when past observations are used to estimate the location of the target, even under conditions of noise. The Kalman filter estimations made by individual sensors are then collected by the leader of the corresponding cell(s). Note, that, to capture objects with irregular moving behaviors, multiple possible trajectories of the object need to be taken into account. Thus, to accurately predict the future location of an object, we use the individual sensor predictions to compute a probability distribution function for the detected target direction and location. This is done by using Kernel function estimators, which can efficiently approximate the probability density distribution using a relatively small number of samples. In our approach we essentially use the individual sensor predictions as samples of the distribution that describes the probability with which the object will be at a specific location in the future.

Our general approach has the following several advantages:

- The technique uses local computation: each sensor computes an estimate of the future location of an object using its own past readings. As a result, the communication requirements are minimized.

- Combining the estimates is done efficiently, and fault-tolerance properties are built into the system: when a sensor fails, the remaining sensors can still provide an accurate estimate.

- The leader does not need to keep any state (other than knowing the neighboring leaders). That is,

a leader estimates the future location of the object using the current sensor estimates only. As a result, even if a leader fails, tracking can be resumed immediately after a new leader node is selected.

- The tracking mechanism is not sensitive to sensor inaccuracy: each sensor employs Kalman filters to smooth out the tracking of the object, and the predictions of the tracking sensors are used to compute a probability density function for the object location.

- The warning mechanism addresses issues related to the trade-off between power conservation and quality of monitoring in target tracking sensor networks. The density distribution of the next target's location is approximated using Kernel functions. This results in a good approximation of the monitoring region that the target is going to traverse and as a consequence the set of the sensors that have to be put on alarm can be defined.

Furthermore, we give an outline of how to use our framework in the more general case that obstacles are present. Obstacles can block the surveillance area of a sensor, the communication between sensors, and the movement of an object. To handle these cases, our framework assumes that if an obstacle appears inside a cell, then the cell is split in such a way, so that sensors in each part are able to communicate directly.

The paper is organized as follows. In Section 2 we discuss the sensor network topology. Section 3 describes the tracking framework. The experimental evaluation is presented in Section 4, while in Section 5 we describe related work. We conclude the paper and provide paths for future work in Section 6.

2 SENSOR NETWORK COMMUNICATION

To efficiently handle the communication between sensors, the sensors network is partitioned into grid cells. This can be done using existing techniques (e.g. [29], [31]). We use a grid structure for the fol-
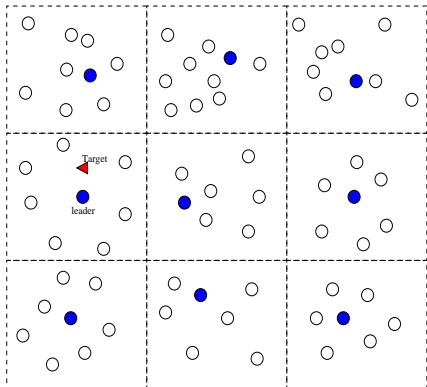
Figure 1: Defining grid cells in a sensor network

lowing reasons: a) the communication costs required for tracking can be reduced, while at the same time the collaboration mechanism between sensors can be simplified, and b) tracking can be performed in a more robust way, since data from many nodes (in the grid cell) can be taken into consideration. To simplify the presentation, we assume that a sensor field spans in a two dimensional plane. The partition is done in such a way, that sensors in each cell are able to communicate efficiently with each other. If the transmission range of each sensor is R, then the side of each cell is $a \cdot R$, where $0 < a \leq 1$. In each cell there is a *leader* node. Such a structure of a sensor network is presented in Figure 1.

We assume that each sensor knows its location, and that the leader at least knows the geographic boundaries of its cell. Also the leader is aware for the sensors in its cell while it keeps information for *the leaders in its neighboring cells* and a *map of the obstacles* in its vicinity. Moreover, each leader is responsible to collect the tracking data from all nodes in its cell. Then based on the tracking information the leaders can decide which subsequent cells have to be activated in order to carry on the tracking.

To select the leader we adopt an energy-efficient scheme that periodically assigns the leadership to a different sensor in each cell. This is achieved by randomly selecting nodes as leaders and rotating the selection, so that the high-energy dissipation experienced by the leader nodes in communicating with

sensors or leader nodes in lower tiers is spread across all nodes in the cell. This makes sure that no single sensor node uses up its energy because it has to play the role of the leader for too long. Thus at specific time periods the leader election algorithm is performed so that a new leader is elected. Note that for the new leader to be able to continue the work of the old one (i.e. tracking task and figuring out the next monitoring area), it suffices to receive the approximation model used by the old leader.

As far as the leader election procedure is concerned, several approaches have been proposed [21]. For example, Liu *et al* [19] implement a geographically-based group management scheme that uses time-stamped messages to solve the contentions and elect a single leader in a region. In their scheme, the node that sends the message with the earliest time stamp is declared the leader. Another approach is introduced in [14], which uses randomization to distribute the energy load evenly among the nodes in the network. Sensors elect themselves to become a leader(cluster-head) at a given time with a certain probability (i.e. based on the suggested percentage of leaders and the number of times the node has been a leader so far).

However, our approach is orthogonal to the leader election process and we consider that any of the approaches could be used. As soon as a leader has been defined in a cell, it sends a message to the sensors in its cell to inform them for its location. Also a similar message is sent to the leaders of its neighboring cells.

To accommodate sensor faults, a *backup-leader* is assigned by the leader whose responsibility is to periodically communicate with the leader to determine if it is still alive. This can be done through the use of *Hello* messages. If the backup-leader detects that the leader has died, it becomes the main leader and notifies all the sensors in the cell and the leaders in its neighboring cells. Then, it randomly chooses another sensor to become the backup-leader.

2.1 Communication issues

According to the proposed framework, direct communication paths can be defined between any pair of sensors in a cell or between the leaders of neighboring cells. Then the communication types in a sensor
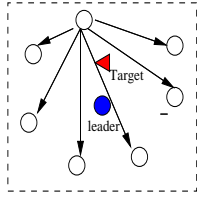
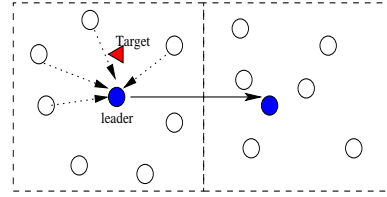Figure 2: Sensor-to-sensor

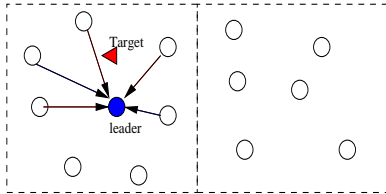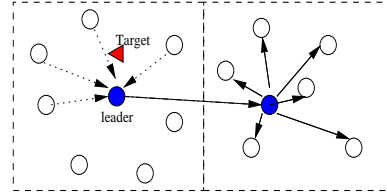Figure 4: Wake-up message: Leader-to-Leader

Figure 3: Sensor-to-leader

Figure 5: Wake-up message: Leader-to-Sensors

network can be summarized to the following ones:

- *Sensor-to-Sensor*. When a sensor detects the presence of a target, it broadcasts a detected message to the others in its cell (see Figure 2). The message contains information for the originating sensor and its distance estimate from the target.

- *Sensor-to-Leader*. Since the tracking procedure has finished at the level of sensors, each sensor that detects the target has an estimation of the target's next probable location. Then a data announcement message is sent from each sensor to the leader of the cell (see Figure 3). The message provides information for the originator (e.g. sensor id), the current and the predicted location of the target.

- *Leader-to-Leader*. This type refers to communication between cells and it takes place at the warning step. The leader of the current cell, where the target has been detected based on the observations of the sensors in its cell, makes a prediction of the next monitoring region that the target is likely to traverse. Then it sends a wakeup (warning) message to the leaders of its neighboring cells (see Figure 4) that intersect the predicted monitoring region. It also forwards to them the information they need to continue the tracking. A wakeup message contains information about the originating leader and the

predicted motion direction of the target (i.e. location, velocity and slope of its direction).

- *Leader-to-Sensor*. When the leader of a cell receives a warning message, it puts on alert all the sensors in its cell sending a warning message to them (see Figure 5). Also it forwards a message with the prediction information to the sensors enabling them to continue the tracking and prediction procedure.

3 TRACKING FRAMEWORK

In this section we give an overview of the proposed framework discussing also the main steps of the tracking and warning procedure. In general terms the framework involves a two loop algorithm for *tracking* and *warning* in sensor networks which can be summarized as follows:

1. The *low level loop* is executed in each sensor. Its purpose is to improve the prediction accuracy of the sensor. Each sensor estimates the location of the target that it detects (Subsection 3.1). Then it uses Kalman Filters and a linear motion description as described in Subsection 3.2 to predict the next location of the target.

2. The *high level loop* is done at the "cell leader"

5

level. The purpose of this loop is to identify the cells of sensors to be awaken. The leader of a cell collects the observations of all the sensors, and computes the probability that a neighboring cell will be visited by the target. Herein, the Kernel estimators are used in order to approximate the underlying probability distribution of the target's motion direction. The whole procedure is discussed in detail in Subsection 3.3.

3.1 Target Detection

When a target enters the detection region of the sensor network, sensors that are awaken and close to the target can detect it. Each sensor is able to detect the target that enters its field and estimate its distance from the target. Also the velocity of the object can be estimated. As soon as a sensor detects the presence of a target, it broadcasts a detected message to the other sensors in its cell (*Sensor to Sensor communication* - Figure 2). When a sensor node receives the message, it stores the coordinates of the originator and the target's distance.

3.2 Tracking

Having detected a target, its location has to be estimated. Herein we use the approach proposed in [6] to estimate the location of a target. This implies that the measurements of at least three sensors are used to apply the commonly used triangulation method. Any sensor node that detects itself the target and receives a detected message from two different neighbors, can compute a location estimate of the target via the triangulation method.

At the sensor level also a prediction of the next probable location of the target is made. The sensors in the cell, where the target has been detected, maintain the prediction model based on which the next location of the target is estimated. Then each sensor forwards its prediction for the next location of the target to the cell leader (*Sensor to Leader Communication* - Figure 3).

As we have already noted above the leaders of the cells are not static but they change periodically, in order to increase the robustness of the network.

3.2.1 Eliminating tracking by multiple leaders

It is possible that a target may be sensed by sensors belonging to different neighboring cells. Reporting their estimates for the target position to their respective leaders may lead to unsuccessful prediction of the next monitoring area, since the same object is tracked by different leaders.

A possible solution to this problem will be the leaders to exchange their predictions. We assume that the leader that tracks a given object is the leader of the cell that the object falls in. When a leader tracks an object, it does not only communicate with the sensors in its cell, but with the leaders of the neighboring cells as well. Hence, if sensors in neighboring cells track an object, they report it to their leader, which aggregates their information and makes an estimation on where the object is. As we have discussed in Section 2 we assume that each sensor knows its location and the leaders at least know the geographic boundaries of their cell and their neighboring leaders. Based on this information the leaders can estimate if the detected target falls in their cell. Then, once a leader detects an object o, it checks if its position estimate for o is within the boundaries of its cell. If o does not belong to the same cell as the leader that detected it, the leader sends its track to the neighboring leader which is responsible to track the object because it falls in its cell. The responsible leader uses the track of the neighboring leader as if it came from just another sensor.

For instance, assume that an object o traverses the area that is close to the boundaries of two neighboring cells, as Figure 6 shows. Then sensors of both cells (e.g. s_1, s_2) will detect o since it traverses an area that is in their vicinity. According to our approach, once the sensors s_1 and s_2 detect the object, they send a message with its estimation to the leaders of their cells, i.e. *leader 1* and *leader 2* respectively (as Figure 6a shows). However *leader 2* detects that o falls out of its cell and hence based on its knowledge of its cell's boundaries and its neighboring leaders sends a message with its estimation to the responsible leader for this object tracking, i.e. *leader 1* (leader-to leader communication as Figure 6b depicts). Hence
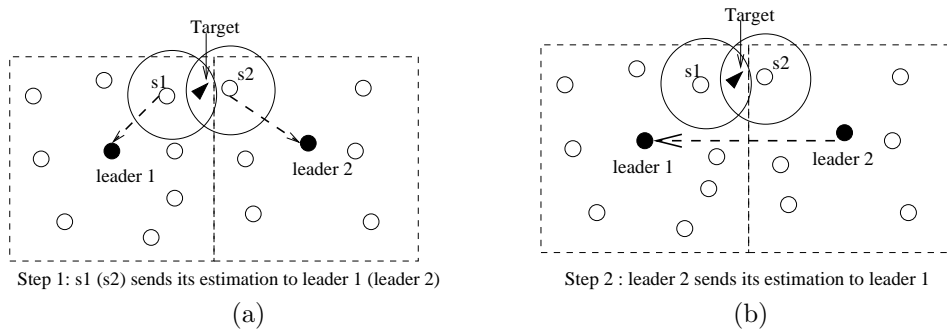
Figure 6: Eliminating tracking by multiple leaders

each time an object is detected, only the leader of the cell that the object actually traverses will track the object and make a prediction of its trajectory.

3.2.2 Prediction procedure

Considering a linear motion target direction we predict the expected location of the target. We compare the expected position of the object with the one measured based on the triangulation approach, also referred to as *actual position* of the target, and the prediction model is updated to take into account the error. Herein a Kalman filter is added to the track estimation in order to correct the predicted position and thus reduce the track divergence. The whole process of tracking and prediction is performed by the sensors that have detected the target.

A brief description of the Kalman filter algorithm that we adopt in our approach follows. The Kalman filter is a set of mathematical equations that provides an efficient computation means to estimate the state of a process in a way that minimizes the mean of squared error. The filter estimates the process state at some time and then feedback in the form of noisy measurements.

In this work, the equations for the Kalman filter aim at computing an *a posteriori* location estimate $\hat{x}_k$ of the target as a linear combination of an a priori estimate $\hat{x}_k^-$ (we assume a linear trajectory of the target) and a weighted difference between an actual measurement z_k of its location and a measurement

prediction $H\hat{x}_k^-$. In general terms, the Kalman equations fall into two categories [18]:

- *Time update* equations, which are responsible for projecting forward the current location and estimating the error covariance to obtain an a priori estimation of the next time location.

- *Measurement update* equations, which are responsible for the feedback. They incorporate a new measurement into the a priori estimate to obtain an improved *a posteriori* estimate of the target's location.

3.3 Alerting

After estimating the trajectory of an object, the sensors that lie near the predicted trajectory of the target are put on alert. When we wake up the sensors, we also forward the prediction model, so that the new sensors can use the previous observations and predict the next location of the target.

3.3.1 Alerting based on different trajectory predictions

Given that different prediction models can be defined based on different location estimates from the sensors which detect an object at a specific time period, we may have different potential estimated trajectories for the object under consideration. Then a question concerning the trajectory that the target is expected to follow arises.
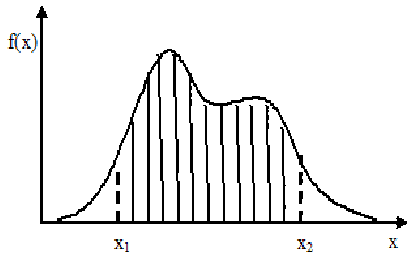
7

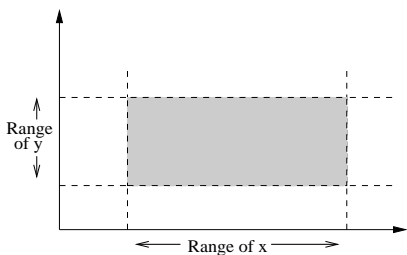Figure 7: Kernel function for the x coordinate

Figure 8: The predicted locations of the target

At the level of the cell, the leader considers the direction estimates as they have been defined at the tracking step and uses kernel functions to compute a probability density function for the direction and the location of the target. Alternatively we can use histograms. The goal is to approximate with high probability, the future location of an object based on the past observations and the predictions of many sensors. The leader of the cell where the target is detected considers the different predictions for the target's next location estimates (as defined by the sensors in its cells) and exploiting statistical techniques, such as kernel estimators, tries to define the range of the targets motion direction.

For a target o, let $T = \{tr_i = (x_i, y_i) | i = 1, \ldots, m\}$ be a set of its predicted location as defined in the next T secs by a set of m sensors, where x_i and y_i are the coordinates of the i-th prediction for the target's location. Actually, this is the information that each sensor sends to the leader of its cell when the track-

ing and prediction procedure have been completed. Assuming that there exists a function $k(x_i)$, the *Kernel function*, with the property $\int_0^1 k(x_i)dx_i = 1$, the approximation of the underlying probability distribution of the target's motion direction is

$$f(x) = \frac{1}{n} \sum_{tr_i \epsilon Tr} k(x - tr_i) \qquad (1)$$

It has been shown that the shape of the kernel function does not affect the approximation substantially. It is the standard deviation, or bandwidth, of the function that is important. Therefore, we choose a kernel function that it is easy to integrate. The Epanechnikov kernel function has this property [8]. The 1-dimensional Epanechnikov kernel function centered at 0 is

$$k(x) = (\frac{3}{4})\frac{1}{B}(1 - (\frac{x}{B})^2 \qquad (2)$$

if $|\frac{x}{B}| < 1$, and 0 otherwise.

To get an initial estimate for the bandwidth we use Scott's rule [23]: $B = \sqrt{5} \, s \, |S|^{-\frac{1}{5}}$, where s is the standard deviation of the sample. This rule is derived under the assumption that the data distribution is a normal and so it over-smoothes the function.

In order to address the boundary problem in estimating the kernels, we project the parts of the kernel function that lie outside $[0, 1]$ back into the data space. Herein we consider that a density estimation function is defined for each of the variables (i.e. x_i and y_i) that define the location of the target o. By aggregating the integrals of the defined $f(x)$ we have an approximation of the variables range in the next T secs, while a proper combination of these ranges gives an approximation of the area that the target is expected to traverse the time period of T secs. Then, the leader defines the bounds of 99% probability, and uses these to define a set of wedges that specify the locus of the probable locations of the target. These wedges specify the next monitoring region that the target is going to traverse.

The following example describes how this monitoring area can be approximated based on Kernel estimators. Assume that the density distribution of the target's location is approximated by kernel functions,
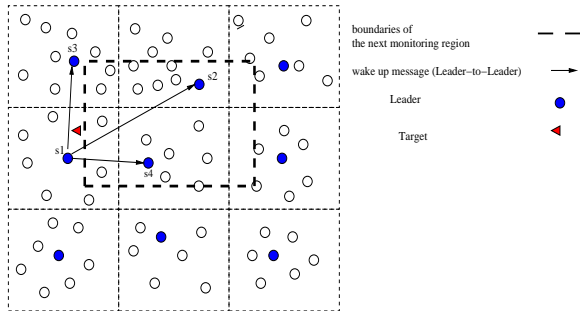
8

Figure 9: Alerting and Trajectory transfer

as shown in Figure 7 for the x-coordinate (similarly for the y). Then the shading areas (representing the integrals of the respective functions) in the figure show the proportion of the kernel function that contributes to the definition of the next monitoring region. In other words they represent the estimated range of each of the target's direction variables in the next T secs. An aggregation of these areas gives the region, R, which the target under consideration is likely to traverse the next T secs. The grey region (rectangle) in Figure 8 represents the boundaries of this region. Then, we claim that based on the estimates of the sensors at the current location of the target, the target is most probable to traverse R in the next T secs. The next step is to define the cells that have to be put on alert. The leader finds the neighboring cells that are intersected by the rectangle, which represents the next monitoring region. Then it sends wakeup messages to the respective leaders of this region.

Such an alerting scenario is presented in Figure 9. Let the sensors in the cell having as leader sensor s_1 detect a target. Each of them estimates the target's location and based on the prediction model under concern they predict the next location of the target. This information (predicted locations of target) is then forwarded to the leader, s_1 which using Kernel functions estimates the monitoring region that the target is likely to traverse the next T secs. This region is mapped to the sensor network plane so that we find the cells that have to be informed. Assuming that the dot lines in Figure 9 illustrate the boundaries of this

region, the neighboring cells with leaders s_2, s_3 and s_4 are those that are intersected with the next predicted monitoring area and hence s_1 sends wakeup messages to each of these sensors. These warning message contains the location of the originator and the prediction model. Then the leaders are responsible to warn the sensors in their cells (*Leader to Sensor Communication* - Figure 5) and the tracking process is repeated at this point.

3.3.2 Tracking and Warning Algorithm

Let $S = \{s_i\}$ be the initial set of sensors that detect the target o at the time point t_1 and let s_L be the leader of the cell where the target is detected for the first time. Based on the framework discussed in the previous sections, the main steps of the two-level tracking and warning algorithm can be summarized as follows:

I. *Low level Steps.* The following processes are performed at the node level.

- For each $s_i \epsilon S$
  if s_i receives a message from at least two of its closest neighboring sensors 0.

  1. s_i estimates the location of o,
  2. s_i uses Kalman filters and predicts the motion direction of o in the next T secs,
  3. s_i sends a message containing the predicted location of o (as defined in step 2) to its leader s_L.

- s_L defines the set of predicted motion directions for o
$$Tr = \{(x_i, y_i)\}$$

II. *High level Steps.* In the leader s_L the following processes are performed:

- The Kernel function is applied to Tr and the probability distribution of target's direction, P, is estimated.
- The next monitoring region, R, that the target is going to traverse is estimated based on P.

9

- The neighboring cells of s_L are defined :

$$Neigh_{s_L} = \{n^i_{s_L}\}$$

- For $n^i_{s_L} \epsilon Neigh_{s_L}$
  if $n^i_{s_L}$ intersects with R then
  s_L sends a wakeup message to the leader of $n^i_{s_L}$

3.4 Handling obstacles in sensor networks

In real-world applications the presence of obstacles in sensor networks is one of the most important problems that we have to tackle. Since the obstacles could hinder the motion direction of a target, it is important to take their presence into account during both the prediction of a target's trajectory and the warning procedure in sensor networks. Also the presence of an obstacle in a sensor network may affect the communication range of the sensors in such a way that sensors that are close to each other are not able to communicate directly. To handle these cases, the proposed framework modifies the cells wherever the initial regular partitioning is intersected by the obstacles. It also considers that the leader of each cell has a local map of the obstacles in its vicinity. Then if the leader estimates that, based on the predicted target's motion direction, the target is likely to meet an obstacle, it redefines the neighboring cells to which is going to send a warning message. Regarding the motion direction of the target in the presence of an obstacle, each leader has to take into account the following cases in order to warn the appropriate sensors (see Figure 10): i) the target can be reflected by an obstacle, ii) follows an obstacle to go around it, and iii) change its direction according to the direction of the obstacle.

To figure out where the object is likely to go we have to use geometric properties. In the context of this paper we don't investigate the issue of estimating alternative directions of an object given the presence of obstacles. However, this is an important issue and we consider it as future work direction.

4 EXPERIMENTAL EVALUATION

In this section we present our empirical results that show the performance of our technique.

4.1 Experimental Testbed

We have implemented our technique on top of TAG [20]. TAG provides functionality for a given sensor to broadcast queries to all other sensors, and functionality to aggregate the sensor replies to the broadcast query. We used TAG's basic communication modules to perform the communication between leaders and sensors in each cell. However we had to implement a new module of communication that allows neighboring leaders to communicate with each other.

We assume that each sensor is able to provide a spatial location for the tracked object (and by combining sequential observations, a current velocity vector and a predicted future location). We use this model for the experiments without loss of generality since, as it has been shown, simpler sensors can also collaborate in small groups and together provide this information.

The simulation scenarios model a target moving on a terrain containing cells. The default size of the terrain in our experiments was set to 100x100. By fixing the terrain size and modifying the length of the side of each cell, we impose different topologies on the terrain. We vary the length of the cell's side from 4 up to 25, thus varying the total number of cells 625 to 16, given the fixed 100x100 terrain. Furthermore, different cell sizes imply different node capacities per cell. A cell having side of length a, has a maximum capacity of a^2 nodes.

In our experiments we assumed that the object is moving in a piece-wise linear trajectory. That is, at any given moment the velocity vector is constant, and changes with a probability α to a new vector. To make the movement realistic we do not allow arbitrary changes in the speed and the direction (direction changes are within an angle $\theta < 30^o$ in our experiments, and the speed can change within a factor of $\beta = 0.1$).
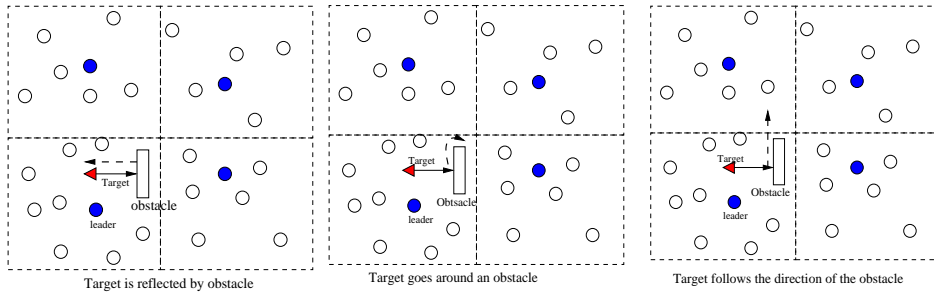
10

Figure 10: Different cases of a target's motion directions given the presence of an obstacle

4.2 Evaluation metrics

We used the following metrics to evaluate the performance of the technique:

1. **Accuracy:** There are two possible errors that can affect the accuracy of the tracking algorithm:

   (a) A leader warns a neighboring leader that the object will enter its area, but the object does not.

   (b) An object exits the area of coverage. This may happen either because the coverage is not dense, or because one or more sensors fail and thus a target is allowed to pass undetected through their area of coverage.

   We record the number of errors of each type that the tracking algorithm makes over a range of parameter settings.

2. **Efficiency:** To evaluate the efficiency of the algorithm we count the number of messages (between sensor to leader, leader to leader and leader to sensor) that get exchanged during each tracking run, for different parameter settings.

4.3 Evaluation Parameters

1. **Cell size and Sensor Detection Range:** We run a set of experiments to determine if changing the cell size affects the performance when sensor detection range is held constant.

2. **Sensor Accuracy:** To estimate the effect of individual sensor accuracy to the accuracy of the whole algorithm we add Gaussian noise to the readings of each sensor, and vary the standard deviation of the noise distribution.

3. **Sensor Failure Rate:** We allow sensors (including leaders) to fail with a small probability in each time step.

4.4 Experimental Results

Evaluating the Efficiency
In the first set of experiments, we evaluated the efficiency of the technique by counting the number of messages sent during the entire tracking of an object. In Figure 11 we plot the total number of messages sent, while varying the length of the cell's side and the sensor detection range. We observe that, fixing the cell size, the number of messages increases as the detection range gets larger. This happens because as the detection range gets larger, more sensors have to report their tracking data to their leader. On the other hand, fixing the size of the detection range, the number of messages decreases as the cells get larger and the grid imposed by the cells gets less dense. Low number of cells implies less messages exchanged due to leader-to-leader communication.

Evaluating the Accuracy
In Figure 12 we plot the number of errors that affect the accuracy, for different lengths of the side of
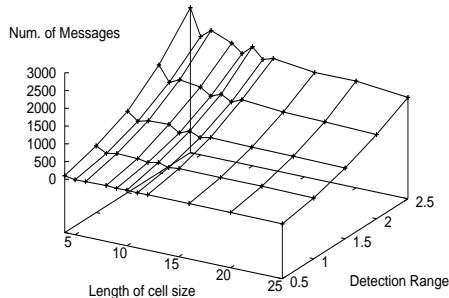
11

Figure 11:
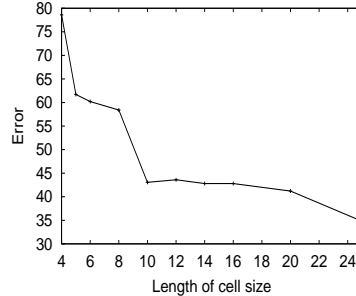Number of messages sent vs. sensor detection range, and the length of the cell's side

Figure 12:
Number of errors vs the length of the cell's side (the terrain size was kept constant to 100x100)
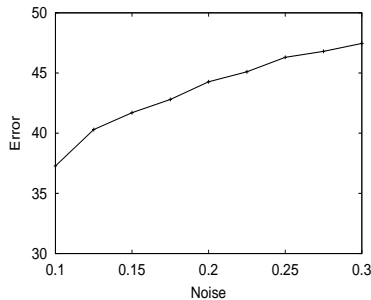

Figure 13:
Number of errors vs the Gaussian noise (radius of detection range: 1)
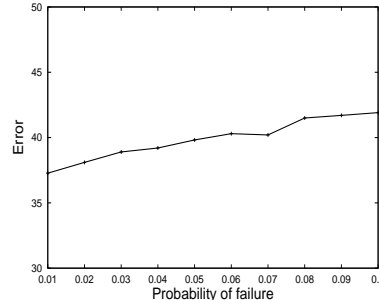

Figure 14:
Number of errors vs probability of failure (radius of detection range: 1)

each cell. We consider two types of errors: when the target passes undetected, and when the prediction of the leader is mistaken, regarding the cell that the target is about to cross to. We observe that the error rate decreases as the size of the cell increases. This happens because the leader in a large cell has more sensors in its cell, and so it can compute more accurate estimates. Also, when the individual cell size increases, the number of cells decreases, and so there are fewer hand-offs from leader to leader.

Accuracy results when sensors are not accurate
In order to evaluate the accuracy of our system, we added Gaussian noise to the readings of each sensor. Figure 13 presents how the error is affected when the standard deviation of the noise distribution is varied. We note that there is small degradation of the accuracy numbers, even for large amounts of noise. This is due to the smoothing effect that kernels have on the estimated distribution.

Evaluating the effects of sensor failures
We also ran experiments where we introduced random failures into the network. At any time instant a node may fail, with a fixed probability. Figure 14 presents how the error is affected with respect to the probability of failure. Again we notice that our

12

method for combining the sensor estimates is robust against sensor failures.

4.5 Discussion

Our experiments show that the error rate and the efficiency of the sensor network improve when the cell size increases. However there are many reasons to keep the cell size small. These include the time and energy it takes to send messages from a sensor to a leader and from a leader to a leader. Once the average distance between leaders and sensors becomes too large, messages have to go across multiple hops to reach their destination, making difficult to predict delays and therefore difficult to combine estimates from different sensors. Also, the time it takes to send messages from a sensor to a leader and from a leader to a leader imposes an upper bound on the speed of the target that can be tracked.

In addition, leaders of very large cells have heavier computational requirements because they have to coordinate many sensors. This is especially important in our scheme where ordinary sensors become leaders in a round-robin fashion. For these reasons we conclude that the optimal cell size is the largest practical. In the presence of noise in the sensor readings, as well as when sensor failures are introduced, our method for combining the sensor estimates is robust. This is due to the smoothing effect that kernels have on the estimated distribution, and also to the fault-tolerance properties that are built into the system.

5 RELATED WORK

Target tracking has applications in various domains, therefore it attracts the interest of researchers from different fields such as computer vision, sensor networks, air traffic control, surveillance, robotics, security and first response to emergencies. As a consequence several methods for target tracking have been proposed. They can be categorized into two major areas: one containing tracking techniques *per se*, and one encompassing methods and/or protocols related to general communication mechanisms (e.g. group management, clustering etc) which facilitate the task of target tracking. Needless to say, there is great overlap over those two areas.

Starting with the methods that fall in the first category, in [4] a collaborative signal processing (CSP) approach for target classification and tracking in distributed sensor networks is presented. It uses location-aware data routing that limits the scope of CSP to relevant subset of nodes conserving network resources. The location of a target is estimated based on linear regression and trigonometry methods. Then, given a local detection information, i.e. a list of tracks, data association is required to map the detection to a track. As the tracking continues, a new cell is defined that encloses the region the target is likely to traverse. In the same work an approach for target classification is proposed. It is based on the measurements of the nodes within a cell. Both data fusion and decision fusion approaches are discussed in the context of a classification approach, which combines the different measurements regarding the event, in order to define the category into which it can be classified. A restriction of this method is that CSP techniques rely on prior statistical information about the signals. Also the cells formed during the tracking are pre-defined and their definition relies on the velocity of target.

A binary model for tracking a moving object in sensor networks is presented by Aslam et al. [1]. According to this model each sensor network node detects one bit of information and broadcasts it to a base station. The sensor's bit denotes whether an object is approaching it or moving away from it. The authors propose a filtering style algorithm for target tracking. Even though this approach seems to give an accurate prediction of the target's location using only one bit information, the main drawback is its centralized computational structure.

In [7] an approach for *acoustic* target tracking was presented. The network architecture is cluster based. The cluster head calculates the target location based in the signal reading of the nodes in its cluster (slave nodes).

Hwang et al. [17] propose techniques for simultaneously tracking and maintaining identities of multiple targets. Motivated by an air-traffic control scenario, the algorithm presented therein considers

the problem of associating measurements with targets and tracking them, and it is based on stochastic approaches. In [24], Shin et al. address the issue of multiple-target identity management. They introduce the novel *identity belief matrix*, a doubly stochastic matrix, which forms a description of the identity information of each target. Also, they present a distributed algorithm for computing and updating this matrix.

We now present methods and techniques whose main characteristic is that they employ some communication framework in order to achieve their goal of target tracking:

The *Dynamic Convoy Tree-based Collaboration*(DCTC) framework was proposed in [33] to facilitate sensor nodes collaborating in detecting and tracking a mobile target. DCTC relies on a tree structure called *convoy tree*, which includes sensor nodes around the moving target. The tree dynamically evolves by adding some nodes and pruning others, as the target moves. The sensing data are aggregated to a node close to the target, i.e. to the current root of the convoy tree. A node in the convoy tree only sends data to its parent. As the target moves, many nodes in the convoy tree may end up being far away from the root, and hence a large amount of energy may be wasted when they send their sensing data to the root. In this case, a new root should be elected to replace the old root, and the tree should be reconfigured accordingly. In [33] the tree reconfiguration problem is formalized as finding a min-cost tree sequence.

A cluster based approach for predictive tracking in sensor networks is proposed in [30]. The main idea is to predict the target's future location based on known previous locations. The cluster head aggregates the information that three of the sensors in its cluster send to it in order to define the current location of the target. Then it predicts the target's next location considering that it obeys a two-dimensional Gaussian distribution. In the same context, in [28] a *prediction-based* energy saving scheme is proposed that aims to reduce the energy consumption for object tracking under acceptable conditions. The proposed prediction model is based on the assumption that the object's movement usually remains constant

for a certain period of time. Also three heuristics are discussed for a wake-up mechanism. The first heuristic considers that only the destination node is informed. The second one suggests that all the nodes on the route from the current node to the destination node are informed, while the third one wakes up all the nodes along the predicted route, as well as their neighbors. However, our prediction and warning approaches are more general, considering that the sensors collaborate to track and predict the target's movement. Both filtering and probabilistic methods are used in order to efficiently correct the errors in the estimation of the target's movement and most accurately define the sensors that have to be notified.

In [15] and [16] a multicast protocol that takes into account spatio-temporal constraints was presented. Also a data dissemination approach that takes into account sink mobility was proposed in [31]. In [2] and [32] an algorithm and a protocol are proposed, respectively, for clustering sensors into groups, while aiming at minimizing the network's energy consumption.

A group management method for track initiation and maintenance in target tracking application is discussed in [19]. It is a leader-based tracking approach. The leader is selected based on the time stamp of the detection message that the sensors send to each other upon detection of the target. Since the selected leader is responsible to maintain the collaborative group, the system uses a suppression message to notify all nodes that detect the target to abandon detection and join the group of the current leader.

The quality of surveillance issues in target tracking sensor networks are discussed in [13]. Therein, they propose a model for quantifying metrics for the quality of surveillance. A sleeping-awake protocol is developed, which provides high quality of surveillance. Also, deployment guidelines for sensor node in target tracking applications are given. Contrary to our approach, however, they do not study issues related to the efficient tracking and prediction of the target's movement, so that the next monitoring region is accurately defined.

In [29] Xu et al. introduced the notion of adaptive fidelity into the realm of ad hoc wireless networks, by proposing the GAF algorithm. GAF runs inde-

pendently of the underlying routing protocol, and it addresses the issue of preserving the nodes' energy by identifying which of them are equivalent from a routing perspective, and by turning off unnecessary nodes. The GAF algorithm adapts sleep times based on node density, scaling back node duty cycles (and so reducing routing *fidelity*) when many interchangeable nodes are present. This allows it to substantially increase the network lifetime [5]. It makes use of application- and system-information to turn off node radios for extended periods of time. It also employs the nodes' deployment density to adaptively adjust routing fidelity. Location information (e.g. via GPS) and active node communication is used to determine node density and redundancy. Each GAF node uses this information to associate itself with a *virtual grid*, where all nodes in a particular grid square are equivalent with respect to forwarding packets. Nodes in the same grid then coordinate with each other to determine who will sleep and how long. This decision is determined by application and system information. Nodes periodically wake up and switch roles in order to achieve load balancing. A variety of protocols that rely on geometric routing and extend or improve GAF have been proposed in the literature, such as [25], [26], and [27].

In [22] Fang et al. propose a protocol for distributed aggregate management (DAM) and algorithms for activity monitoring in sensor-nets (EBAM and EMLAM). The DAM protocol divides the sensors into clusters and provides means for leader election. It provides a framework to cluster the sensors according to their sensed signal strength, so that there is one signal peak per cluster. The *energy-based activity monitoring* algorithm (EBAM) addresses the under-counting problem, i.e. how to accurately estimate the number of targets within each cluster. The *expectation-maximization like activity monitoring* algorithm (EMLAM) focuses on the issue of how to distinguish between multiple targets which enter the field from different locations, then gather together, and finally split apart. The techniques presented in [22] address issues arising in target enumeration, rather than target tracking and trajectory prediction.

Several other protocols, algorithms or techniques have been proposed for general routing related issues in sensor-nets, as well for information aggregation and in-network storage. For instance, in [9], Fang et al. introduced techniques for overcoming the problems that arise when routing holes appear in a sensor-net. The work in[10] proposes a novel group communication scheme, named *roamingcast*. It enables symmetric multicast routing in a fixed, densely deployed wireless sensor network. They deal with cases in which processes migrate from one node to another following physical events being tracked. They describe the *roaming hub based architecture* (ROAMHBA) that supports such agent processes. Gao et al. [11] address the problem of distributed information aggregation and storage in a sensor network. They advocate the idea that a sensor should know a fraction of the information from distant parts of the network, in an exponentially decaying fashion by distance. They propose a distributed mechanism for storing data in a sensor network that allows range queries injected anywhere in the network to be served. Biswas et al. [3] have recently proposed a method for probabilistic inference, in the presence of incomplete and noisy information. In that work, a Bayesian network models the sensor network, while Markov Chain Monte Carlo sampling is used to perform approximate inference. Gao et al. [12] give a distributed algorithm for constructing a kinetic data structure (KDS), motivated by the need to cluster *mobile nodes* in an ad-hoc network. This work presents a thorough study of and a solution to the problem of maintaining a clustering of a set of N moving points in the plane.

6 CONCLUSIONS

We presented a new two-level approach to perform resilient and power efficient tracking in sensor networks.

The low-level process is executed at individual sensors whose functionality is to detect the presence of a mobile target (an event) and to estimate its trajectory based on previous observations. One the other hand, the high-level one is executed across multiple

sensor nodes, in order to combine individual estimates made at single sensors and predict, with high probability, the target's trajectory across the system. Filtering approaches are used to increase the accuracy of the target's trajectory that each detecting sensor predicts. Moreover, in order to eliminate the potential errors in the estimation of target's location, its motion direction is predicted taking into account the readings of all the sensors that detect it. Then, the leader sensor decides which subsequent cells have to be activated in order to carry on the tracking. Our approach uses Kernel functions to compute the probability density function for the location of the target and to identify the neighboring cells of the sensors to be awaken.

The experimental results show the accuracy and efficiency of our approach to track a moving object in sensor networks and predict its next location.

A path for future work is the study of issues related to tracking multiple objects, as well as objects that spread through an area (e.g. clouds, poisonous gas etc). Furthermore, the handling of obstacles in sensor networks and their influence in tracking is another open research issue. In this work we briefly discuss this problem, while we give some directions for addressing it. However, we are planning to study it further, as well as to thoroughly evaluate the initial approach outlined in Section 3.4.

ACKNOWLEDGEMENTS

REFERENCES

1 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.

2 Seema Bandyopadhyay and Edward J. Coyle. An energy efficient hierarchical clustering algorithm for wireless sensor networks. In *IEEE INFOCOM*, 2003.

3 R. Biswas, S. Thrun, and L.Guibas. A probabilistic approach to inference with limited information in sensor networks. In *3rd Int. Conf. Information Processing in Sensor Networks (IPSN)*, pages 269–276, 2004.

4 Richard Brooks, Parameswaran Ramanathan, and Akbar Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, August 2003.

5 Nirupama Bulusu, Deborah Estrin, Lewis Girod, and John Heidemann. Scalable coordination for wireless sensor networks: Self-configuring localization systems. In *Proceedings of the Sixth International Symposium on Communication Theory and Applications (ISCTA '01)*, July 15-20th 2001.

6 J. Caffery and Jr. A New Approach to the Geometry of TOA Location. In *IEEE Vehicular Technology Conference (VTC)*, pages 1943–1949, sep 2000.

7 Wei-Peng Chen, Jennifer C. Hou, and Lui Sha. Dynamic clustering for acoustic target tracking in wireless sensor networks. In *ICNP*, 2003.

8 N. A. C Cressie. *Statistics For Spatial Data*. Wiley and Sons, Inc., 1993.

9 Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *23rd Conference of the IEEE Communications Society*, 2004.

10 Q. Fang, J. Liu, L. Guibas, and F. Zhao. Roamhba: Maintaining group connectivity in sensor networks. In *3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 151–160, 2004.

11 J. Gao, L. J. Guibas, J. Hershberger, and L. Zhang. Fractionally cascaded information in a sensor network. In *3rd International Symposium on Information Processing in Sensor Networks*, pages 311–319, April 2004.

12 J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. *Discrete and Computational Geometry*, 30(1):45–65, 2003.

13 C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In *Proceedings of the Int. Conf. on Mobile Computing and Networking (MOBICOM)*, 2004.

14 W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnam. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Int. Conf. on System Sciences*, 2000.

15 Qingfeng Huang, Chenyang Lu, and Gruia-Catalin Roman. Mobicast: Just-in-time multicast for sensor networks under spatiotemporal constraints. In *IPSN*, 2003.

16 Qingfeng Huang, Chenyang Lu, and Gruia-Catalin Roman. Spatiotemporal multicast in sensor networks. In *SenSys*, Los Angeles, CA, November 2003.

17 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.

18 R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME– Journal of Basic Engineering*, 82(Series D):35–45, 1960.

19 J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao. Distributed group management for track initiation and maintenance in target localization applications. In *Information Processing in Sensor Networks, Second International Workshop*, Palo Alto, CA, April 2003.

20 S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *OSDI 2002*, Boston, MA, December 2002.

21 N. Malpani, J. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks. In *Proceedings of the International Workshop on Discrete algorithms and methods for mobile computing and communications*, Boston, Massachusetts, 2000.

22 Q.Fang, F.Zhao, and L.Guibas. Lightweight sensing and communication protocols for target enumeration and aggregation. In *4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 165–176, 2003.

23 D. Scott. *Multivariate Density Estimation: Theory, Practice and Visualization.* Wiley and Sons, Inc., 1923.

24 J. Shin, L. Guibas, and F. Zhao. Distributed identity management algorithm in wireless ad-hoc sensor network. In *2nd Int'l Workshop on Information Processing in Sensor Networks (IPSN)*, pages 223–238, 2003.

25 Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

26 Guoliang Xing, Chenyang Lu, Robert Pless, and Qingfeng Huang. On greedy geographic routing algorithms in sensing-covered networks. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.

27 Guoliang Xing, Chenyang Lu, Robert Pless, and Joseph A. O'Sullivan. Co-grid: and efficient coverage maintenance protocol for distributed sensor networks. In *International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004.

28 Y. Xu, J. Winter, and W. Lee. Prediction-based strategies for energy saving in object tracking sensor networks. In *Proceedings of the IEEE Int. Conf. on Mobile Data Management (MDM'04)*, January 2004.

29 Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 70–84, Rome, Italy, July 2001. ACM.

30 H. Yand and B. Sikdar. A protocol for tracking mobile targets using sensor networks. In *IEEE International Workshop on Sensor Networks Protocols and Applications*, 2003.

31 Fan Ye, Haiyun Luo, Jerry Cheng, Songwu Lu, and Lixia Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *MOBICOM*, 2002.

32 Ossama Younis and Sonia Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid,energy-efficient approach. In *IEEE INFOCOM*, Hong Kong, China, March 2004.

33 Wensheng Zhang and Guohong Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *IEEE INFOCOM*, Hong Kong, China, March 2004.

BIOGRAPHICAL NOTES

Maria Halkidi received her B.Sc. degree in Informatics in 1997 from the University of Piraeus, Greece. She graduated with M.Sc. and Ph.D. degrees from Dept of Informatics of Athens University of Economics & Business(AUEB) in 1999 and 2003, respectively. Since 1999, she has been member of the DB-NET research group in AUEB participating in National and European-funded projects. In 2004, she was awarded the Marie Curie International Fellowship. She visited Univ. of California

at Riverside where she conducted research in Database lab under the supervision of Dr. Gunopulos. Now she is a postdoctoral researcher at AUEB. Her research interests include Data Mining techniques, Web Content Management, data analysis in sensor networks.

Dimitris Papadopoulos received the B.Sc. in Computer Science in 1998 from the Univ. of Ioannina, Greece, and graduated with the MSc. and PhD degrees from Univ. California, Riverside in 2003 and 2005, respectively. His research interests are in the areas of multidimensional indexing, data mining and sensor networks.

Vana Kalogeraki is an Assistant Professor at the Department of Computer Science and Engineering at the Univ. of California, Riverside. Her research interests include sensor networks, peer-to-peer systems, distributed and real-time systems and resource management. She received her Ph.D. from UC Santa Barbara in 2000. During 2001-2002 she held a Research Scientist Position at Hewlett-Packard Labs in Palo Alto, CA. Dr. Kalogeraki is a member of the Editoral Board of the Ad hoc Networks Journal. She has delivered tutorials on peer-to-peer computing at the ACM/ USENIX/IFIP International Middleware Conference (Middleware 2003) and the International Symposium on Distributed Objects and Applications (DOA.02). She is the Program co-Chair of the "IEEE International Conference on Pervasive Services (ICPS) 2005", the "13th International Workshop on Parallel and Distributed Real-Time Systems (WP-DRTS) at IPDPS 2005", the "International Workshop on Databases, Information Systems and Peer-to-Peer Computing" at VLDB'2003. She has published many technical papers, including co-authoring the Object Management Group (OMG) CORBA Dynamic Scheduling Standard. Her research is supported by NSF.

Dimitrios Gunopulos is an Associate Professor at the Computer Science and Engineering Department, in the Universrity of California Riverside. His research is in the areas of Data Mining and Knowledge Discovery in Databases, Databases, and Algorithms. Dr. Gunopulos has held positions at the IBM Almaden Research Center (1996-1998) and at the Max-Planck-Institut for Informatics (1995-1996). He completed his undergraduate studies at the Univ. of Patras, Greece (1990) and graduated with M.A. and Ph.D. degrees from Princeton Univ. (1992 and 1995 respectively). His research has been supported by NSF (including an NSF CAREER award), the DoD, the Institute of Museum and Library Services, the Tobacco Related Disease Research Program, and a gift from ATT. He is currently an Associate Editor in IEEE TKDE and in the Editorial Advisory Board of the Elsevier Information Systems Journal.