SNAP: Fault Tolerant Event Location Estimation in Sensor Networks Using Binary Data

Michalis P. Michaelides, *Member*, *IEEE*, and Christos G. Panayiotou, *Senior Member*, *IEEE*

Abstract—This paper investigates the use of wireless sensor networks for estimating the location of an event that emits a signal that propagates over a large region. In this context, we assume that the sensors make binary observations and report the event (positive observations) if the measured signal at their location is above a threshold; otherwise, they remain silent (negative observations). Based on the sensor binary beliefs, a likelihood matrix is constructed whose maximum value points to the event location. The main contribution of this work is Subtract on Negative Add on Positive (SNAP), an estimation algorithm that provides an efficient way of constructing the likelihood matrix by simply adding ± 1 contributions from the sensor nodes depending on their alarm state (positive or negative). This simple estimation procedure provides very accurate results and turns out to be fault tolerant even when a large percentage of the sensor nodes report erroneous observations.

Index Terms—Wireless sensor networks, event localization, maximum likelihood estimation, binary data, fault tolerance.

1 Introduction

Recent advances in wireless communications and electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate untethered in short distances. These tiny and generally simple sensor nodes consist of sensing, data processing, and communicating components. A wireless sensor network (WSN) consists of a large number of such nodes that collaborate in order to accomplish difficult tasks. This paper investigates the use of a WSN that is densely deployed over a large area for estimating the location of an event source that releases a certain signal or substance in the environment. The proposed sensor network can deal with a number of environmental monitoring and tracking applications including acoustic source localization, toxic source identification, early detection of fires, and so on [1], [2], [3].

As already mentioned, a WSN consists of low-cost devices, which have limited resources (processing capabilities, memory, and power) and may fail frequently. Note that in this context, it makes sense to use only binary observations; binary decisions are "easier" problems for the sensors to solve and they also limit the communication cost (single bit transmissions). Moreover, binary decisions are less sensitive to calibration mismatches and varying sensor sensitivities.[1] With this in mind, the objective of this work is to develop a simple, fault-tolerant algorithm that can identify the event location using only binary data from the sensor nodes.

Subtract on Negative Add on Positive (SNAP) is one such algorithm and the main contribution of this paper. The main idea behind SNAP is to use the observations of *all* sensors (positive or negative) to construct a likelihood matrix by summing contributions of ± 1. In other words, sensors with positive observations add their contributions to the cells of the likelihood matrix that correspond to their Region of Coverage (to be defined in the sequel), while sensors with negative observations subtract their contributions. By bounding the contribution of each sensor to ± 1, we do not allow any sensor measurement to dominate the overall estimation result that constitutes the basic reason for the algorithm's fault-tolerant behavior.

SNAP is designed to address some of the major challenges outlined in [4] in the new research area of Collaborative Signal Information Processing (CSIP): it promotes the collaborative efforts of the sensor nodes and provides a simple, fault-tolerant way to combine their binary data to estimate the event location.[2]

The paper is organized as follows: First, in Section 2, we look at related work in target tracking and localization based on sensor networks. Then, in Section 3, we present the model we have adopted and the underlying assumptions. Section 4 describes the details of the proposed SNAP algorithm. In Section 5, we introduce three variants of the SNAP algorithm. Section 6 shows the connection between SNAP and maximum likelihood estimation. In Section 7, we describe other binary estimators that are also implemented for the performance evaluation of SNAP. Section 8 explains the fault tolerance of SNAP against other binary estimators. Section 9 presents several simulation results and compares the performance of SNAP with the performance of other

1. If a sensor has an offset $x > 0$, then *all* of its measurements include this error. On the other hand, if the sensor decision is binary, i.e., it has to decide whether its measurement z is less than a threshold Y, then its decision is correct for most measurements except only for the measurements in the range $Y - x < z < Y$.

• *The authors are with the KIOS Research Center for Intelligent Systems and Networks, Department of Electrical and Computer Engineering, School of Engineering, University of Cyprus, 75 Kallipoleos Street, PO Box 20537, 1678 Nicosia, Cyprus. E-mail: {michalism, christosp}@ucy.ac.cy.*

Manuscript received 13 May 2008; revised 19 Jan. 2009; accepted 28 Jan. 2009; published online 26 Mar. 2009.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2008-05-0214.
Digital Object Identifier no. 10.1109/TC.2009.60.

2. Note that for the purpose of this paper, the performance of SNAP is measured in terms of estimation accuracy and fault tolerance. Other performance metrics with respect to energy, bandwidth, and QoS are topics of ongoing research.

algorithms. Finally, the paper concludes in Section 10, where we also present our future research plans.

2 Related Work

The problem of localizing an event has been extensively studied in the last 20 years using arrays of sensors for radar, sonar, and acoustic target tracking applications (e.g., see [5], [6], [7] and references therein). Several techniques have been proposed to solve the localization problem. Such techniques can be classified into three main categories: 1) Direction of Arrival (DOA), 2) Time Difference of Arrival (TDOA), and 3) Energy-based. Out of the three categories, the energy-based techniques seem more appropriate for sensor networks since they require simpler hardware and do not assume any synchronization between the sensor nodes.

More recently, the use of binary sensors for localization has been proposed in the literature. In [8], the authors propose a maximum likelihood estimator that uses only binary readings that are communicated to a central processing unit to estimate the event position. The approach adopted in this paper is also based on maximum likelihood estimation but differs in the likelihood function used. As a result, the proposed estimation algorithm (SNAP) is significantly more fault tolerant compared to the approach in [8].

Nofsinger [9] proposes the use of binary sensors for a plume tracking application. First, tracks are created based on the correlation between the sensor nodes that have detected the target and then a likelihood function is constructed between them whose maximum points to the event location. A main difference between SNAP and this work is that we also consider the sensors that did not detect the event when constructing the likelihood function.

Fault tolerance is an important problem that has been studied for event detection [10]. Fault-tolerant event localization, however, is a topic that did not receive enough research attention and more work is still needed. Sensor nodes are simple devices that are very prone to failure as a result of energy depletion, environmental harsh conditions of operation, software problems, etc. These problems have been reported in real experiments and can result in erroneous, unexpected behavior [11]. Robustness to this kind of fault is essential for any estimation algorithm, so it can tolerate a number of misbehaving nodes. In [12], the authors propose a median detector to filter out extreme measurements followed by a centroid estimator to achieve fault-tolerant localization. Our results indicate that SNAP can achieve higher accuracy and is more fault tolerant compared to the centroid estimator. This paper significantly extends our work in [13] by providing a more in-depth analysis of the SNAP algorithm and its fault-tolerant behavior.

3 Model

For the sensor network that estimates the position of an event, we make the following assumptions:

1. A set of N sensor nodes is uniformly spread over a rectangular field of area A. The nodes are static. Their position is denoted by (x_n, y_n), $n = 1, \ldots, N$ and it is assumed that it is known (e.g., a small fraction of the sensor nodes uses GPS, while the rest estimate their location using localization algorithms).
2. A single source of the event is located at a position (x_s, y_s), which is generated by a uniform distribution inside A.
3. The source emits a continuous signal that propagates uniformly in all directions and there are no environmental changes throughout the propagation.
4. The sensors that detect the event (i.e., they are alarmed) send a packet to a base station (sink); otherwise, they remain silent.
5. The fusion center (sink) has correctly detected the event.

Assumptions 1, 2, and 4 are quite common and reasonable for sensor networks. Assumption 3 defines an event propagation model that may be accurate for sources that emit sound or electromagnetic waves, but it may not be very accurate for problems where an actual substance is released in the environment (for example, in problems where the wind pushes the substance toward some direction). We point out that extensions of SNAP to such problems are possible and it is the topic of the ongoing research. Regarding Assumption 5, readers are referred to [14] or [15], where the detection problem in the context of WSN is addressed.

In this paper, we assume that the measured signal at the source location is c, and as we move away from the source, the signal is attenuated inversely proportional to the distance from the source raised to some power $\alpha \in \mathbb{R}^+$, which depends on the environment. As a result, the tth sample measurement of any sensor n located at (x_n, y_n) is given by

$$z_{n,t} = s_n + w_{n,t}, \tag{1}$$

for $n = 1, \ldots, N$, $t = 1, \ldots, M$, where

$$s_n = \min\left\{V_{max}, \gamma \frac{c}{r_n^{\alpha}}\right\}. \tag{2}$$

In the above equation, V_{max} and γ are sensor-specific design parameters (the first reflects the maximum measurement that a sensor can register while the second is a scaling factor corresponding to the sensor gain). In addition, r_n is the radial distance from the source, i.e.,

$$r_n = \sqrt{(x_n - x_s)^2 + (y_n - y_s)^2}. \tag{3}$$

We point out that similar models have been extensively used in the literature (see [6], [7], [8], [12]) for acoustic source localization using sensor networks. For the purpose of this paper, $w_{n,t}$ is additive white Gaussian noise, i.e., $w_{n,t} \sim N(0, \sigma_w^2)$ for $n = 1, \ldots, N$ and $t = 1, \ldots, M$.

Then, we assume that the sensor nodes have been programmed with a common threshold T. Given T for any t, we define the following:

- *Alarmed Sensor*: Any sensor with $z_{n,t} \geq T$.
- *Nonalarmed Sensor*: Any sensor with $z_{n,t} < T$.

The threshold T is chosen large enough such that the probability of a sensor being falsely alarmed is very small (i.e., with high probability, the sensor nodes should become

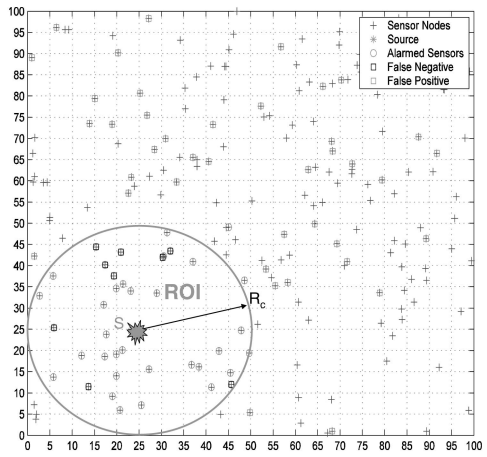


Fig. 1. A field with 200 randomly placed sensor nodes and a source placed at position (25,25). Alarmed sensors are indicated on the plot with red circles inside the disc around the source (ROI). Of the sensor nodes, 50 exhibit faulty behavior and are indicated on the plot as false positives (red squares outside disc) and false negatives (black squares inside disc).

alarmed as a result of the event signal and not because of the noise alone).

Next, we make the following definition with respect to the footprint of the event.

Definition 1. *Given the threshold T,* Region of Influence *(ROI) of a source is the area around the source location inside which a sensor node will be alarmed with high probability, at least 0.5.*[3]

For the uniform propagation model used in this paper ((1) and (2)), given that $w_{n,t}$ is Gaussian white noise, we see that any sensor placed on a circle centered around the source location with radius

$$R_c = \sqrt[\alpha]{\frac{\gamma c}{T}} \tag{4}$$

will become alarmed with probability 0.5. Thus, any point inside the circle corresponds to the ROI, as shown in Fig. 1.

Finally, to define the *Fault Model* that we adopt in this paper, we assume that the sensor nodes that exhibit erroneous behavior are randomly chosen and their original belief is simply reversed as shown in Fig. 1. When applying the fault model, some sensor nodes that fall outside the ROI of the source become alarmed as a result of a fault and are shown as *false positives*. Similarly, sensors that fall inside the ROI become nonalarmed as a result of a fault and are shown as *false negatives*.

4 SNAP Algorithm

The SNAP algorithm consists of four main components:

1. *Grid Formation:* The entire area is divided into a grid $\mathcal{G}$.
2. *Region of Coverage (ROC):* This is the area covered (monitored) by each sensor. Given $\mathcal{G}$, the ROC of a sensor is a neighborhood of grid cells around the sensor node location.
3. *Likelihood Matrix Construction:* Associated with $\mathcal{G}$ is a likelihood matrix $\mathcal{L}$. Inside the matrix cells that correspond to the sensor's ROC, each sensor adds a value based on its binary observation ($+1$ on positive observation and -1 on negative). We calculate the likelihood of a source found in each grid cell, by summing the corresponding contributions from all sensor nodes.
4. *Maximization:* The maximum of this likelihood matrix points to the estimated event location.

Next, we describe the algorithm in more detail.

4.1 Grid Formation

The area is divided into a grid $\mathcal{G}$ with $G \times G$ cells and grid resolution g, e.g., Fig. 1 shows a 100×100 field with $G = 20$ and a grid resolution $g = 5$. Let $C(i,j)$ for $i,j = 1,\ldots,G$, denote the centers of these cells in a matrix form. The number of cells is a trade-off between estimation accuracy and complexity that is further investigated in Section 9. Each sensor node is associated with a cell (i,j) based on its position (depending on the resolution, a cell may contain multiple sensors or no sensors at all). Given $\mathcal{G}$, we also define a $G \times G$ likelihood matrix $\mathcal{L}$.

4.2 Region of Coverage

Given a threshold T, ROC is a region around a sensor node where if a source is located, it will be detected by the sensor with high probability (at least 0.5). For the uniform propagation model considered in this paper, we can determine the ROC of a sensor using the following result the proof of which is included in the appendix.

Lemma 1. *Under conditions of no noise and no faults, assuming 1) a uniform propagation model (as in (2)) and 2) the presence of at least one sensor node inside the source ROI, the maximum of the likelihood matrix constructed by SNAP is $L_{max} > 0$ and the set of cells of the matrix that attain this value, denoted by A_{max}, always includes the true source location iff $ROC \equiv ROI$.*

Lemma assumption 2 is rather technical and is only used to exclude cases of events that could not have been detected. The interpretation of Lemma 1 is that for the uniform propagation model used, SNAP can achieve the highest accuracy when one sets the ROC of the sensor equal to the ROI of the source. Note that for nonuniform propagation models, as in the case of environmental pollution where the wind may push the pollutant in some direction, this result does not hold. In fact, the ROC used in SNAP should be the "mirror image" of the source ROI. This is a subject of ongoing research.

4.3 Likelihood Matrix $\mathcal{L}$

Each alarmed sensor adds a positive one ($+1$) contribution to the elements of $\mathcal{L}$ that correspond to the cells inside its ROC. On the other hand, every nonalarmed sensor adds a negative one (-1) contribution to all elements of $\mathcal{L}$ that

3. Using 0.5 is a convenient way to define the ROI when dealing with noise that has symmetric pdf (e.g., Gaussian) because the ROI size becomes independent of the noise variance.

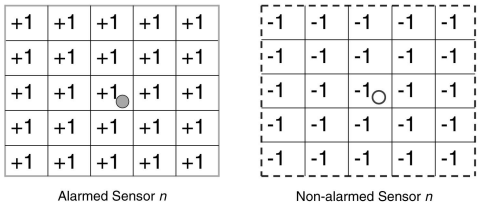


Fig. 2. Region of Coverage.

correspond to its ROC. Thus, the elements of the likelihood matrix are obtained by

$$L(i,j) = \sum_{n=1}^{N} \sum_{t=1}^{M} b_{n,t}(i,j), \quad \text{for } i,j = 1,\ldots,G, \tag{5}$$

where

$$b_{n,t}(i,j) = \begin{cases} +1, & \text{if } z_{n,t} \geq T \quad \text{AND} \quad (i,j) \in ROC_n, \\ -1, & \text{if } z_{n,t} < T \quad \text{AND} \quad (i,j) \in ROC_n, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

and ROC_n represents the region of coverage of sensor node n.

4.4 Maximization

Let (i^*, j^*) be the element of $\mathcal{L}$ with the maximum value, i.e.,

$$L(i^*, j^*) \geq L(i,j) \quad \forall i,j = 1,\ldots,G,$$

then the estimated event location is $C(i^*, j^*)$, i.e., the center of the corresponding cell in the grid $\mathcal{G}$. In cases where more than one elements of the $\mathcal{L}$ matrix have the same maximum value, the estimated event position is the centroid of the corresponding cell centers.

4.5 SNAP Example

To illustrate the SNAP algorithm, we provide a simple example using a square ROC. In the example, the ROC of sensor node n is the set of cells that fall in a square of 5×5 cells around cell (i,j), where sensor n is located, as shown in Fig. 2. For the construction of the likelihood matrix $\mathcal{L}$, each alarmed sensor adds a positive one $(+1)$ contribution to the elements of $\mathcal{L}$ that correspond to the cells that are inside its ROC, as shown in Fig. 2. On the other hand, every nonalarmed sensor adds a negative one (-1) contribution to all elements of $\mathcal{L}$ that correspond to its ROC. The resulting likelihood matrix after adding and subtracting the contributions of eight such sensors is shown in Fig. 3. The event in the example is correctly localized in the grid cell with the maximum value $+3$.

5 SNAP Variants

In this section, we introduce three variants of the SNAP algorithm. First, *SNAPm*, we attempt to conserve energy by first calculating the mean of M measurements *locally* at each sensor node before sending the information to the fusion center (sink). Second, *SNAPe*, we provide a heuristic for estimating the ROC using *only* the fraction of alarmed sensor nodes in the field. Third, *AP* is a variant of SNAP that does not consider the nonalarmed sensor nodes in estimating the source location.

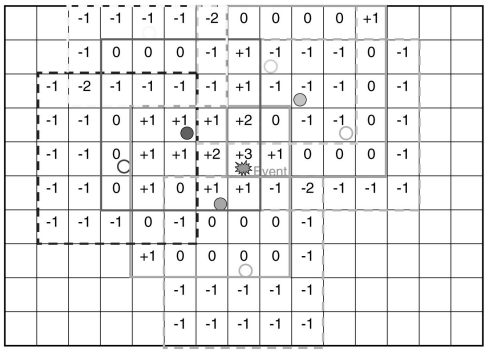


Fig. 3. $\mathcal{L}$ resulting from SNAP with eight sensor nodes, three of which are alarmed and shown in solid color. The event is correctly localized in the grid cell with the maximum value +3.

5.1 SNAPm

Each sensor node n first computes the mean of M measurements and compares the result to the threshold T to decide whether to become alarmed and communicate its observation to the sink, i.e.,

- *Alarmed Sensor*: Any sensor with $\overline{z}_n \geq T$;
- *Nonalarmed Sensor*: Any sensor with $\overline{z}_n < T$;

where $\overline{z}_n = \frac{1}{M}\sum_{t=1}^{M} z_{n,t}$. At the sink, the elements of the likelihood matrix are obtained as before by using (5) and (6) and dropping the time index t:

$$L(i,j) = \sum_{n=1}^{N} b_n(i,j), \quad \text{for } i,j = 1,\ldots,G, \tag{7}$$

where

$$b_n(i,j) = \begin{cases} +1, & \text{if } \overline{z}_n \geq T \quad \text{AND} \quad (i,j) \in ROC_n, \\ -1, & \text{if } \overline{z}_n < T \quad \text{AND} \quad (i,j) \in ROC_n, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

SNAPm results in further energy savings compared to SNAP since it requires a *single* message from each sensor node that is alarmed on the "average." In this way, it avoids congesting the network with repeated messages from sensor nodes that have correctly detected the target and eliminates occasional messages from sensor nodes that were having a false alarm. Since communication is the most expensive operation in terms of energy, SNAPm cannot only improve the bandwidth utilization but it can also prolong the lifetime of the sensor network. As mentioned in Section 9, SNAPm achieves similar estimation accuracy as SNAP. On the other hand, depending on the sampling rate, SNAPm may increase the estimation delay since the sink will have to wait until all M samples are collected.

5.2 SNAPe

By Lemma 1, estimating the ROC is equivalent to estimating the ROI. To calculate the ROI, we need to have information about the signal amplitude at the event location c and the environmental parameter α. In many real-time target tracking scenarios, however, these parameters may be unknown. Under these circumstances, it is still important to estimate the ROI of a source in order to perform the event localization using SNAP. Thus, in this section, we present a

possible heuristic that one can use for this estimation, even though it is worth pointing out that other heuristics are also possible. Neglecting boundary conditions, for densely deployed sensor networks, the fraction of the ROI compared to the overall area of the field A should be approximately equal to the fraction of alarmed sensors, in other words, we can write

$$\frac{\pi \hat{R}_c^{\,2}}{A} = \frac{N_{alarmed}}{N}. \tag{9}$$

This result follows from the assumption that the source is uniformly distributed, making all points in area A equally probable [16]. From this equation, R_c can be estimated from knowing only the fraction of alarmed sensor nodes in the field. The advantage of SNAPe is that the sink is not required to know a priory the signal strength c. We point out that this is just a simple heuristic. Another possibility is to obtain an estimate of the diameter of the ROC from the maximum distance $R_{\max}$ between any two alarmed sensors, e.g., set $2 \cdot R_c = \lambda \cdot R_{\max}$, where λ is an appropriate scaling factor that depends on the density of the network.

5.3 Add Positive (AP)

The Add Positive algorithm is similar to SNAP in the sense that it defines a likelihood matrix similar to SNAP, but in this matrix, only positive contributions (+1) from the alarmed sensors inside the ROI of the source are added. The purpose of this algorithm is to demonstrate the value of using the negative information from nonalarmed sensors.

6 SNAP: A Maximum Likelihood Estimator

In this section, we show that SNAP is actually a maximum likelihood estimator. First, let us define the indicator function for $n = 1, \ldots, N$ and $t = 1, \ldots, M$:

$$I_{n,t} = \begin{cases} 0, & \text{if } z_{n,t} < T, \\ 1, & \text{if } z_{n,t} \geq T, \end{cases} \tag{10}$$

thus, the sensor data can be represented as $\mathbf{I} = \{I_{n,t}: n = 1, \ldots, N, t = 1, \ldots, M\}$. The goal is to estimate the source location $\theta = [x_s, y_s]$ using the collected data $\mathbf{I}$.

6.1 Maximum Likelihood

The Maximum Likelihood Estimator has the form

$$\hat{\theta}_{ML} = \max_{\theta} \log p(\mathbf{I} \mid \theta), \tag{11}$$

where the log-likelihood function is given by

$$\begin{aligned} \log p(\mathbf{I} \mid \theta) = \sum_{n=1}^{N} \sum_{t=1}^{M} & I_{n,t} \times \log\left[Q\left(\frac{T - s_n(\theta)}{\sigma_w}\right)\right] \\ & + (1 - I_{n,t}) \times \log\left[1 - Q\left(\frac{T - s_n(\theta)}{\sigma_w}\right)\right], \end{aligned} \tag{12}$$

and $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{t^2}{2}} dt$ is the complementary distribution function of the standard Gaussian distribution. Also, $s_n(\theta)$ is the signal that would have been measured by sensor n if the source was at location θ and there was no noise (given by (2)). The derivation details for the case where $\sigma_w^2 = 1$, $\forall n, \forall t$ can be found in [8].

6.2 SNAP

Given there is a source at a position θ, let us only consider the data from the sensors that fall inside the source ROI (defined in Section 3), and let us assume that $0 < K \leq N$ such sensors exist. The data from these sensors can be represented as $\mathbf{I_K} \subseteq \mathbf{I} = \{I_{k,t} : r_k \leq R_c\}$ where r_k is the euclidean distance between the sensor node $k = 1, \ldots, K$ and the source position θ. The joint likelihood function is given by

$$\begin{aligned} p(\mathbf{I_K} \mid \theta) = \prod_{k=1}^{K} \prod_{t=1}^{M} & \left[Q\left(\frac{T - s_k(\theta)}{\sigma_w}\right)\right]^{I_{k,t}} \\ & \times \left[1 - Q\left(\frac{T - s_k(\theta)}{\sigma_w}\right)\right]^{(1 - I_{k,t})}. \end{aligned}$$

The only difference from (12) is that we only use the sensors inside the ROI to construct this likelihood function. For the selected sensors, it is expected that most will be alarmed, thus, we propose the following arbitrary probability assignment for their indicator function $I_{k,t}$:

$$\begin{aligned} \Pr\{I_{k,t} = 1 \mid \theta\} &= Q\left(\frac{T - s_k(\theta)}{\sigma_w}\right) = 0.99, \\ \Pr\{I_{k,t} = 0 \mid \theta\} &= 1 - Q\left(\frac{T - s_k(\theta)}{\sigma_w}\right) = 0.01. \end{aligned}$$

Next, consider the modified likelihood function $p'(\mathbf{I_K} \mid \theta) = 10^{2KM} p(\mathbf{I_K} \mid \theta)$. Taking the logarithm of the modified likelihood function, we get

$$\begin{aligned} \log p'(\mathbf{I_K} \mid \theta) &= \sum_{k=1}^{K} \sum_{t=1}^{M} I_{k,t} \times \log(9.9) \\ &\quad + (1 - I_{k,t}) \times \log(0.1) \\ &\approx \sum_{k=1}^{K} \sum_{t=1}^{M} I_{k,t} \times (+1) \\ &\quad + (1 - I_{k,t}) \times (-1). \end{aligned}$$

Therefore, in order to find the likelihood of each source location θ, we simply add one from all alarmed sensor nodes and subtract one from all nonalarmed sensor nodes that are sufficiently close to θ (i.e., inside the source ROI). If the sensor is far away from θ, then its contribution to the likelihood function is zero. The SNAP estimator is therefore the following:

$$\hat{\theta}_{SNAP} = \max_{\theta} \log p'(\mathbf{I_K} \mid \theta). \tag{13}$$

So the SNAP estimator is in fact a maximum likelihood estimator. It has two major differences though from the traditional ML estimator presented in the previous section. First, to construct the likelihood function at a location θ it uses only "local" information in the sense that it uses only the data from the sensors that are inside the ROI. Second, and most important, the way the likelihood function is created is very simple and it turns out to be extremely fault tolerant as shown in the sequel.

7 Binary Estimators

For comparison purpose, in addition to the SNAP algorithm, we also implement the following two algorithms: Centroid Estimator (CE) and Maximum Likelihood (ML) the details of which are explained below.

7.1 Centroid Estimator

The centroid of a finite set of points can be computed as the arithmetic mean of each coordinate of the points. Let (x_n, y_n), $n = 1, \ldots, P$ $(P \leq N)$ denote the positions of all alarmed sensor nodes. Then, the event location estimated by CE is the centroid of these positions:

$$\hat{\theta}_{CE} = [\hat{x}_s, \hat{y}_s] = \left[\frac{1}{P}\sum_{n=1}^{P} x_n, \frac{1}{P}\sum_{n=1}^{P} y_n\right]. \quad (14)$$

7.2 Maximum Likelihood

For the purpose of this paper, we use a discrete version of the algorithm described in [8] to compare its performance to SNAP. Specifically, we divide the area into $G \times G$ equal size grid-cells and evaluate the log-likelihood function G^2 times using (12), assuming the source is located at the center of each cell $C(i, j)$. The maximum of the resulting matrix points to the event location. In the rest of the paper, we will refer to this discrete version of the algorithm as simply ML.

8 Fault Tolerance of SNAP

In this section, we investigate the behavior of the three proposed estimators in the presence of faults: CE, ML, and SNAP. To gain some further insight into the behavior of these estimators, we first present a simple example.

8.1 Test Case

Consider the 1D scenario displayed in Fig. 4, where the line is divided into 20 equal cells. The event is located at position $(x_s = 9.5)$ and we try to estimate its position using four sensor nodes that are located as shown in the figure. Two of the sensor nodes fall inside the source ROI and are alarmed $(x_1 = 13.5, x_2 = 5.5)$, the other two fall outside and are nonalarmed $(x_3 = 14.5, x_4 = 4.5)$. According to the SNAP algorithm described in Section 4, the alarmed sensor nodes provide +1 contribution in the cells inside their ROC, while the nonalarmed sensors provide -1 contribution. The sum of the contributions in each cell i gives the likelihood $L(i)$ of the source occurring in that cell and is plotted over the corresponding cells in Fig. 4a above the sensor locations. The maximum of this likelihood plot corresponds to the SNAP-estimated event location. In the absence of faults, all the three estimation algorithms correctly estimate the event position.

Now consider a fifth sensor node that is faulty and behaves according to the fault model described in Section 3 (i.e., it is nonalarmed when positioned inside the source ROI and alarmed everywhere else). Fig. 4b shows the estimated source location when the three different algorithms are used as a function of the position of the faulty sensor. From the plot, it is evident that only SNAP displays a fault-tolerant behavior for all positions of the faulty sensor node; in Fig. 4a, it is clear that the maximum of $L(i)$ cannot

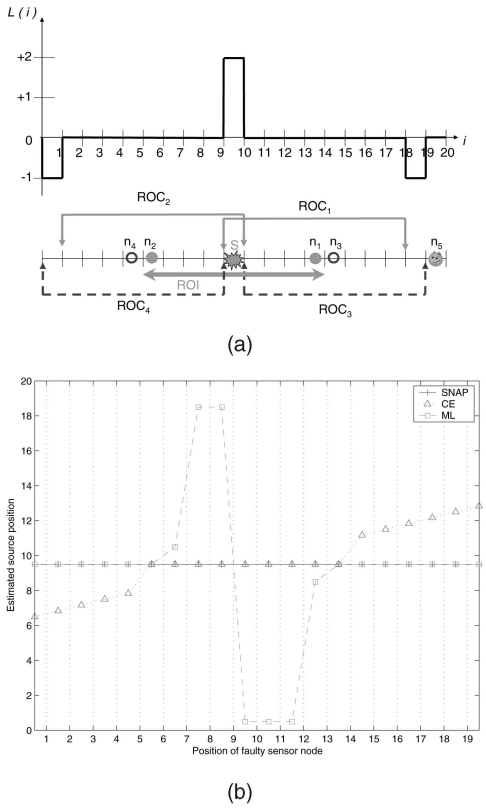


Fig. 4. 1D example with five sensor nodes, one of which is faulty (n_5). (a) Source and sensor positions and the resulting likelihood function. (b) Estimated source location using the three different algorithms, SNAP, CE and ML, as we vary the position of the faulty sensor node.

change no matter where the faulty sensor node is placed. ML fails to estimate the source location when the faulty sensor node is close to the event (false negative), while CE fails to estimate the event location when the faulty sensor node is far away (false positive). Next, we further analyze the strengths and weaknesses of each algorithm in the presence of a fault.

8.2 Centroid Estimator

CE treats all alarmed sensor nodes with equal weight when calculating the centroid of the position of the alarmed sensor node (see (14)). This fact makes the algorithm fairly sensitive to the presence of false positives, especially the ones that occur far away from the true event location. Such faults distort the estimated source position toward the location of the faulty sensor. On the other hand, CE is robust to a large percentage of false negatives. Since it does not consider information from nonalarmed sensor nodes, it can estimate the event location even with a single alarmed sensor node following a "closest point approach."

8.3 Maximum Likelihood

ML is extremely sensitive to false negatives. Even a single faulty sensor node inside the ROI of the source can completely throw off the estimation results. This is a direct result of the construction of the likelihood matrix of ML using (12). Note that for source positions θ close to the sensor, the term $Q(\frac{T-s_n(\theta)}{\sigma_w}) \to 1$, and as a result, $\log[1 - Q(\frac{T-s_n(\theta)}{\sigma_w})] \to -\infty$. If for some reason (e.g., due to a fault), a

TABLE 1
Default Parameter Values

Parameter	Symbol	Default Value
Number of sensor nodes	N	200
Number of measurements	M	1
Saturation voltage	$V_{\max}$	3000
Source amplitude	c	3000
Noise variance	σ_w^2	1
Threshold	T	5
Grid resolution	g	1

sensor fails to detect an event that is very close to it (false negative), then $(1 - I_{n,t}) = 1$, and as a result, the faulty sensor has a *very* large negative contribution to the likelihood function at all points near itself, which (by assumption) is also the point where the source is located. On the other hand, a healthy sensor near the source contributes only $I_{n,t} \times \log Q(\frac{T - s_n(\theta)}{\sigma_w}) \to 0$, which is not sufficiently large to counteract the negative contribution of the faulty sensor. In fact, since the negative contribution of the faulty sensor is unbounded, even several well behaving sensors cannot correct the error.

8.4 SNAP

The fault-tolerant behavior of SNAP is a result of two basic reasons: First, to construct the likelihood function at a source location θ, it uses only "local" information in the sense that it uses only the data from the sensors that are inside the ROI of θ. Therefore, false positives away from the source location have no influence on the estimation results. Second, and most important, it bounds the "damage" that a faulty sensor can cause to the likelihood function by allowing a sensor to subtract *at most* one from the corresponding cells in the likelihood matrix. Furthermore, unlike ML, a *single* healthy sensor close to the source can correct the error in the likelihood function caused by the faulty sensor.

9 Results

For all subsequent experiments, we use a square 100×100 sensor field, where the sensor readings are given by

$$z_{n,t} = \min\left\{V_{\max}, \frac{c}{r_n^2}\right\} + w_{n,t}, \tag{15}$$

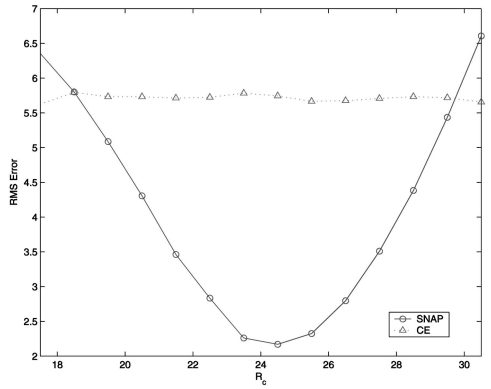


Fig. 5. Experimental ROC calculation.

TABLE 2
Optimal ROC Calculation for Different Percentages of Alarmed Sensor Nodes

c	%Nal	R_c	$\hat{R}_c$	$R_c(exp)$
1000	7	14.1	14.9	14.5
2000	14	20.0	21.1	20.5
3000	21	24.5	25.8	24.5
4000	28	28.3	29.8	28.5
5000	35	31.6	33.4	31.5

%Nal denotes the percentage of alarmed sensors, R_c and $\hat{R}_c$ are calculated using (16) and (9), respectively, while $R_c(exp)$ is obtained experimentally.

for $n = 1, \ldots, N, t = 1, \ldots, M$. Also for the parameters used in the experiments, we use the default values shown in Table 1, unless otherwise stated in the experiment.

The RMS Error reported is the average over B experiments, where we assume that the source is randomly placed at points $(x_{s,b}, y_{s,b}) \in A$ and we solve the problem B times to obtain $(\hat{x}_{s,b}, \hat{y}_{s,b}), b = 1, \ldots, B$ for each estimator considered. In other words, the RMS Error shown in our results is given by

$$RMS\ Error = \frac{1}{B}\sum_{k=1}^{B}\sqrt{(x_{s,k} - \hat{x}_{s,k})^2 + (y_{s,k} - \hat{y}_{s,k})^2}.$$

At the beginning of these B experiments, we randomly initialize the sensor field but it remains fixed for all B experiments. For the following experiments, we used Matlab and assumed that $B = 500$.

9.1 ROC

In the first set of experiments, we investigate the Region of Coverage, which is a main ingredient of the SNAP algorithm. As mentioned before, given that $w_{n,t}$ is white noise and using (15), the ROC of a sensor is a circular disc with radius

$$R_c = \sqrt{\frac{c}{T}}. \tag{16}$$

Thus, we investigate the validity of (16) when calculating the optimal ROC. In Fig. 5, we plot the RMS error versus R_c. CE performance is also indicated on the same plot for comparison purpose. For the scenario investigated, the optimal value for R_c is 24.5.

Table 2 shows the optimal R_c results for different percentages of alarmed sensor nodes. For obtaining the analytical results, we used (16), and for obtaining the experimental results, we varied R_c as in the first experiment and recorded the value that minimized the RMS Error. As seen in the table, for all cases tested, the experimental results for R_c are very much in agreement with the analytical ones.

9.2 Performance Evaluation

In this section, we test the performance of SNAP against the other estimators in terms of accuracy and time complexity. Fig. 6 shows the results for various grid resolutions ($g = 1, 2, 5, 10, 20$), while Fig. 7 shows the results as a function of the threshold T. We varied threshold T from 1 (80 percent of sensors alarmed) to 20 (5 percent of the

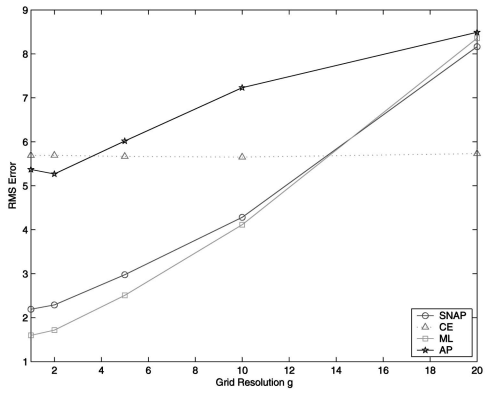


Fig. 6. Estimator performance for various grid resolutions g.

sensors alarmed). In both plots, it is evident that SNAP and ML exhibit a similar performance with ML being the most accurate. This is attributed partly to the fact that ML uses the exact sensor positions, while for SNAP, the sensor positions are quantized. Also, ML calculates the probabilities $\Pr\{I_{k,t} \mid \theta\}$ while SNAP has "arbitrarily" assigned some values. AP has worse performance than SNAP for all grid resolutions and thresholds tested; this is a direct result of not including the nonalarmed sensor nodes in the estimation procedure.

Furthermore, there is a trade-off between accuracy and complexity of the different algorithms that we need to consider. Some typical results are shown in Table 3. CE completes essentially in zero time and is not shown in the table. SNAP (or AP) only require simple additions to estimate the event position and they complete relatively fast even for the smallest grid resolution tested ($g = 1$). ML on the other hand, requires numerical integration and the time for completion is considerably larger. In Fig. 6 and Table 3, it is evident that one should use the smallest g allowed by the available computational budget.

9.3 Sensor Lifetime

In this section, we study the influence of sensor lifetime on SNAP and the other estimators considered. Fig. 8 shows the performance of each estimator as the number of alive sensor nodes in the field is decreased from 200 to 50 (e.g., due to node failures). From the plot, it is evident

TABLE 3
Complexity Analysis in Elapsed Time (Second)

Algorithm	$g = 1$	$g = 2$	$g = 5$	$g = 10$
SNAP / AP	0.07	0.03	0.02	0.02
ML	2.55	0.66	0.13	0.05

that the performance of SNAP does not deteriorate even when a large number of sensor nodes die in the field.

9.4 Fault Tolerance

For the fault tolerance analysis, we vary c to obtain different percentages of alarmed sensor nodes in the field. The number of faulty sensor nodes are chosen randomly at the beginning of each experiment and their original beliefs are simply reversed (see Fig. 1).

Fig. 9 displays the results for the fault tolerance analysis of the different estimators. Although in the absence of faults, the performance of ML and SNAP was very similar, the same cannot be stated here. In fact, both CE and ML are very sensitive to sensor faults and loose accuracy continuously as the number of faults increases. This is especially evident for ML in situations with higher percentages of alarmed sensor nodes (see Figs. 9c and 9d). SNAP, however, as it can be observed from these plots, displays a fault-tolerant behavior and looses very little in accuracy even when 50 out of the 200 sensor nodes exhibit erroneous behavior. In fact, its fault tolerance improves when we increase the percentage of alarmed sensor nodes in the field. The intelligent construction of the likelihood function makes individual sensor faults unimportant in estimating the correct result. Finally, AP displays a similar fault-tolerant behavior with SNAP though the estimation error is larger than SNAP. Note, however, that for a range of faulty sensors from 10 to 80, AP exhibits better performance than both ML and CE.

9.4.1 Dropped Packets

In this section, we investigate the performance of the four algorithms if packets are dropped by the network. Recall that for the network we investigate, alarmed sensors send a packet to the sink while nonalarmed sensors remain silent. Thus, if the sink does not receive a packet from a node, it

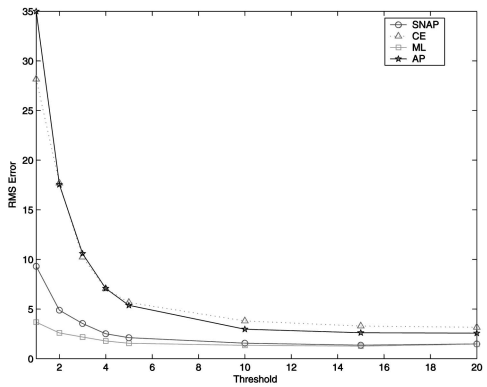


Fig. 7. Estimator performance for different values of threshold T.

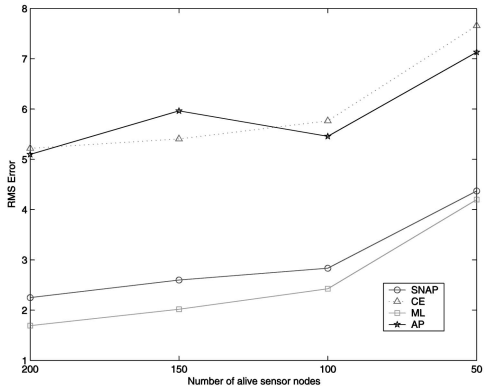


Fig. 8. Estimator performance versus number of alive sensor nodes.

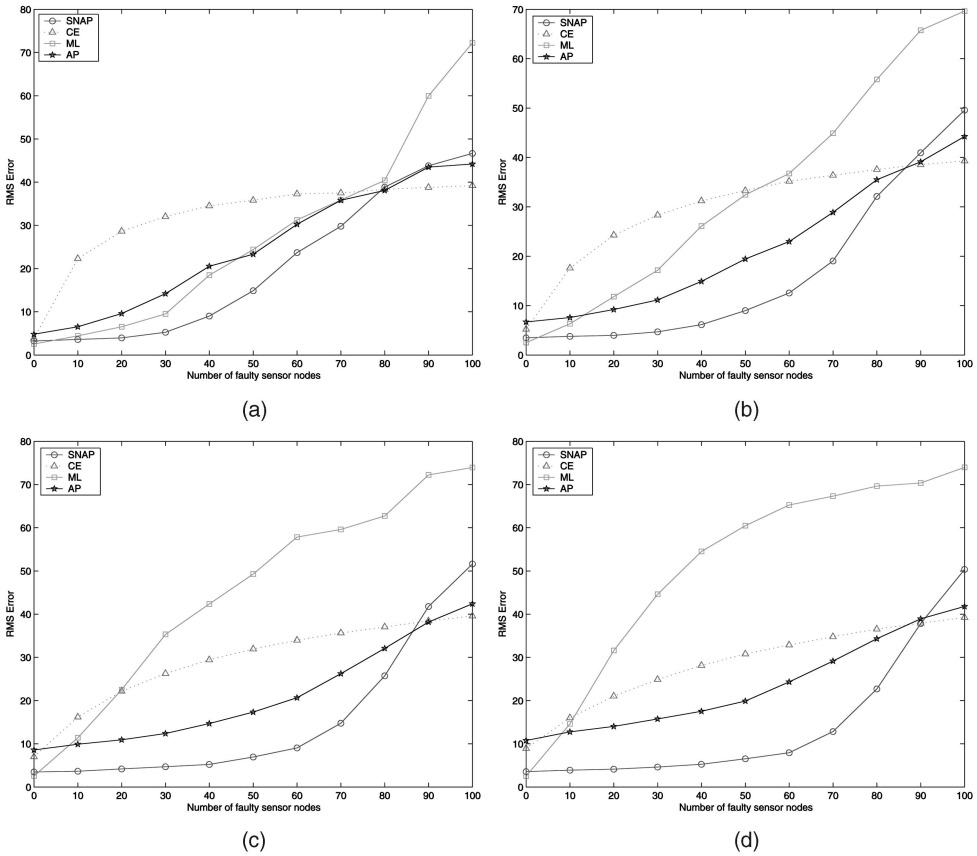


Fig. 9. Estimator performance for different percentages of alarmed sensor nodes as we vary the number of faulty sensor nodes in the field. (a) 7 percent alarmed, (b) 14 percent alarmed, (c) 21 percent alarmed, and (d) 28 percent alarmed.

assumes that the node is in the nonalarmed state. Therefore, to investigate the effect of dropped packets, we change the fault model to allow *only* alarmed sensors to randomly flip their state (false negatives) with probability P_d, which corresponds to the probability of dropping a packet.[4]

In Fig. 10, we investigate the performance of the estimators in the presence of false negatives. ML, as expected from the analysis in Section 8, looses accuracy immediately in the presence of false negatives. SNAP is the best estimator for values of $P_d \leq 0.3$. For higher percentages of dropped packets, CE becomes the better option. This is due to the increasing number of false negatives that counteract the small number of correctly alarmed sensor nodes left in the field when using SNAP. AP does not have this problem, so it displays a similar robust behavior to false negatives as CE. For CE and AP, even one correctly alarmed sensor can localize the source at its own location since both algorithms completely neglect the nonalarmed sensor nodes in estimating the event location.

9.4.2 Board Overheating

For symmetry, in this section, we also investigate the effect of the board overheating, which, as reported in [11], caused the nodes to report false events. In Fig. 11, we investigate the performance of the estimators in the presence of false positives. We simulate the presence of false positives by varying the probability P_o that a nonalarmed sensor node produces a positive observation. CE displays the worst performance in the presence of false positives; this is expected from the analysis in Section 8. SNAP on the other hand, displays the most robust behavior against these overheating faults. The importance of the nonalarmed sensor nodes for SNAP in correcting false positives is also evident by looking at the large difference in performance between SNAP and AP. ML steadily looses performance, and for values of $P_o \geq 0.4$, it becomes even worse than AP.

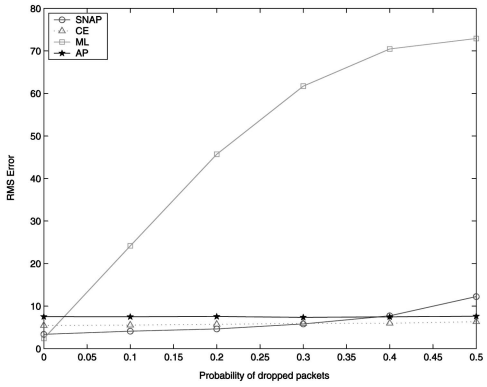


Fig. 10. Estimator performance versus probability of dropped packets P_d.

4. For simplicity, we assume that each packet has equal probability of being dropped irrespective of the number of hops that it has to travel before it reaches the sink. However, this assumption is not expected to affect the fault tolerance analysis.

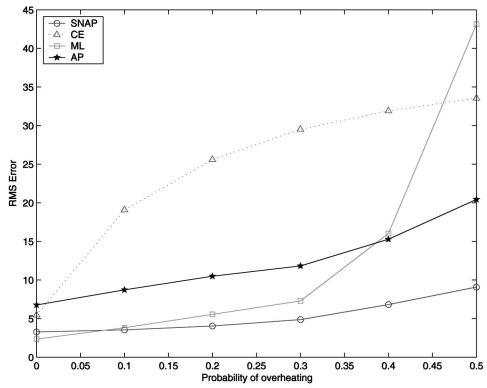


Fig. 11. Estimator performance versus probability of overheating P_o.

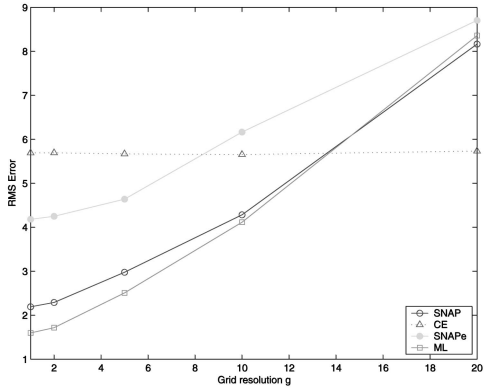


Fig. 13. SNAPe performance for various grid resolutions g.

9.5 SNAPm

In this section, we investigate SNAPm, the version of SNAP, where each sensor node first computes the mean of the M measurements to decide whether to become alarmed. Fig. 12 shows the results for both algorithms as we vary the noise variance for different values of M. Note that in the period of collecting the M observations, it is assumed that the event source is static and has a fixed amplitude. From the plot, it is evident that both algorithms exhibit a similar performance, with SNAP performing slightly better than SNAPm for smaller noise variance conditions. SNAPm, however, is more energy efficient since it requires a single transmission from each sensor node and this benefit increases as we increase M. On the other hand, SNAPm is not expected to be as robust as SNAP when faults occur (e.g., dropped packets) since the sink will have less information to counteract such errors. Thus, when a sensor has a large number of measurements M, a possibility is to group them into h groups with M/h measurements in each group and send up to h packets that will correspond to the event that the sensor is alarmed or not "on the average."

9.6 SNAPe

In this section, we investigate SNAPe, the version of SNAP that does not require knowledge of the source amplitude c or the parameter α. Recall that for SNAPe, R_c is estimated using (9) based on the fraction of alarmed sensor nodes.

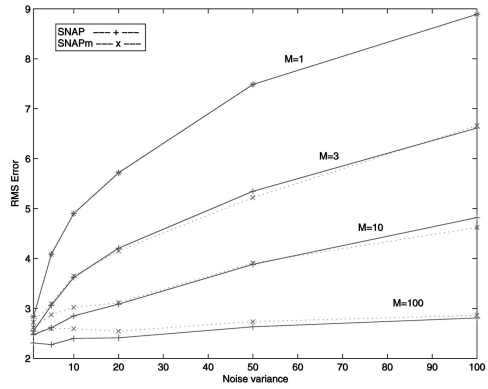


Fig. 12. SNAP performance for various M, σ_w^2.

First, we investigate the validity of (9) for computing the optimal R_c for SNAPe. Table 2 shows the results for different percentages of alarmed sensor nodes. The percentage of alarmed sensors (percent Nal) shown in the table, is the average obtained by varying the event amplitude c and using only nonboundary sources in the simulations. R_c shows an almost linear relationship to the percent of alarmed sensor nodes in the field. For all cases tested, (9) is very good at estimating the analytical values of R_c obtained by (16). The same experiment was also performed with $N = 100$ with almost identical results. This shows that R_c is not very sensitive to the total number of sensors deployed.

Then, in Fig. 13, we investigate the performance of SNAPe. There is an evident loss in performance compared to SNAP—the case when we use the analytical R_c calculated from (16). This is mainly due to the sources on the boundaries, where the number of alarmed sensor nodes does not produce the correct ROC for estimating R_c. The performance of SNAPe is still better than CE for $g = 1, 2, 5$.

Finally, Fig. 14 displays the fault tolerance results for SNAPe. CE and SNAP are also shown on the same plot for comparison purpose. For low percentages of alarmed sensor nodes (see Fig. 14a), SNAPe is very sensitive to sensor faults. As we increase the percentage of alarmed sensor nodes in the field, however, SNAPe continuously improves in performance. In fact, for higher percentages of alarmed sensor nodes (see Fig. 14d), it approximates the same fault-tolerant behavior as SNAP. This result shows the robustness of SNAP for applications, where we have a large number of alarmed sensor nodes. Using only information about the fraction of alarmed sensors in the field and binary information from the sensor nodes, it can localize the event position even when as many as 25 percent of the sensor nodes are faulty.

9.7 Uncertainty in the Position of the Sensor Nodes

In this section, we relax Assumption 1 by assuming that the position of each sensor node is not known exactly but it is estimated imprecisely. We model the errors in the estimated sensor node positions with normal Gaussian noise $N(0, \sigma_p^2)$. Fig. 15 shows the results as we vary σ_p^2 for four different values of the source amplitude ($c = 1{,}000, 2{,}000, 3{,}000, 4{,}000$). From the plot, it is evident that SNAP looses very little in accuracy when we have errors in the estimated sensor node positions.

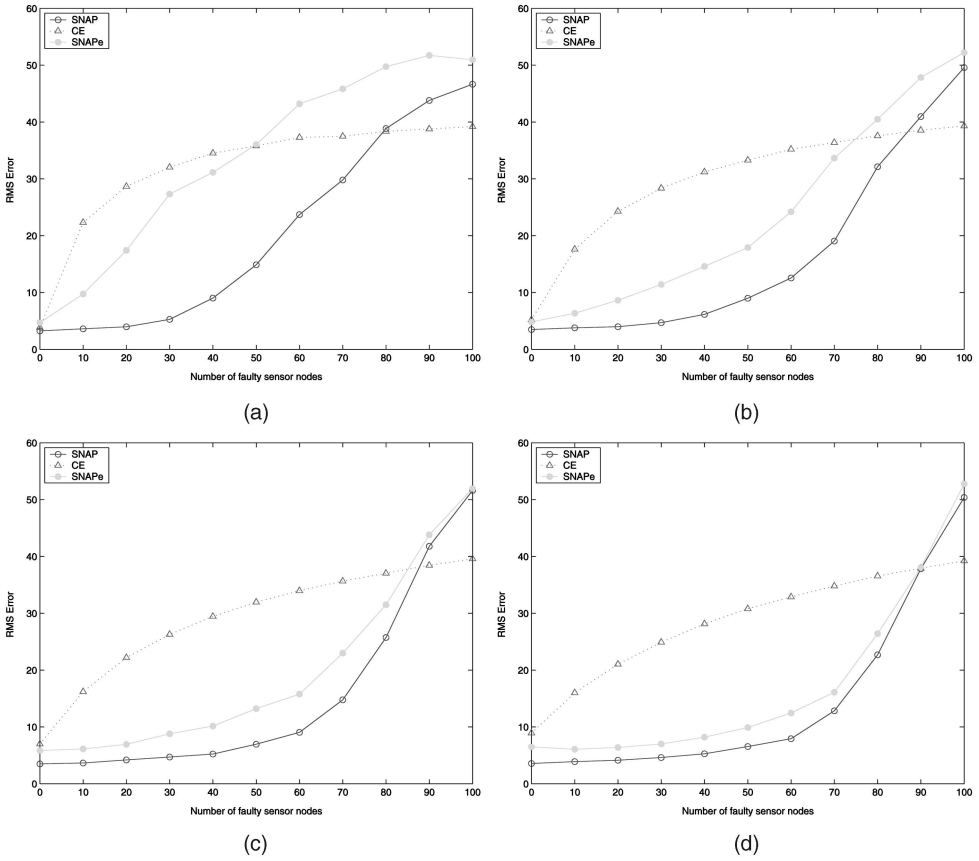


Fig. 14. SNAPe performance for different percentages of alarmed sensor nodes as we vary the number of faulty sensor nodes in the field. (a) 7 percent alarmed, (b) 14 percent alarmed, (c) 21 percent alarmed, and (d) 28 percent alarmed.

This robust behavior to sensor position errors is extremely important for sensor networks, since it is usually impossible to obtain accurate positions for hundreds of nodes using localization algorithms.

10 Conclusions and Future Work

SNAP is a simple, efficient, fault-tolerant algorithm, which can be applied in time-critical applications for estimating the position of an event given only binary data from the sensor nodes. Compared to the maximum likelihood estimation, SNAP is slightly less accurate but is computationally less demanding and can achieve accurate event localizations even when a large percentage of the sensor nodes report erroneous observations. Moreover, for the construction of the likelihood matrix, we only need "local" information, which is something that can be exploited in order to derive a distributed version of SNAP. This will allow the optimization of routing and querying for the signal processing task of estimating the event position. For our future work, we also plan to study the performance of SNAP with respect to energy, bandwidth, and QoS. Furthermore, we will look at different propagation models, where an actual substance is released in the environment (for example, in problems of environmental pollution). Finally, we will try to extend our results to situations where we have multiple sources.

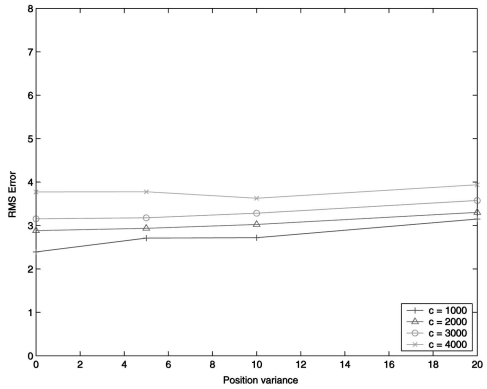


Fig. 15. SNAP performance for various σ_p^2.

Appendix

Proof of Lemma 1

For the forward direction of the proof, we use induction over the number of sensors n. Without loss of generality, assume that the source is placed at point $(0,0)$ and its ROI is a circle with radius r_{ROI}. For $n = 1$, (single sensor case) by assumption 2, this has to be placed inside the ROI of the source, and given the ideal conditions, it will be alarmed. Thus, $L_{max} = 1 > 0$ and A_{max} is the ROC of the sensor. Thus, a circular ROC with radius $r_{ROC} \geq r_{ROI}$ guarantees that $(0,0) \in A_{max}$ and the forward statement of the lemma holds for $n = 1$. Then, let us assume that the lemma statement

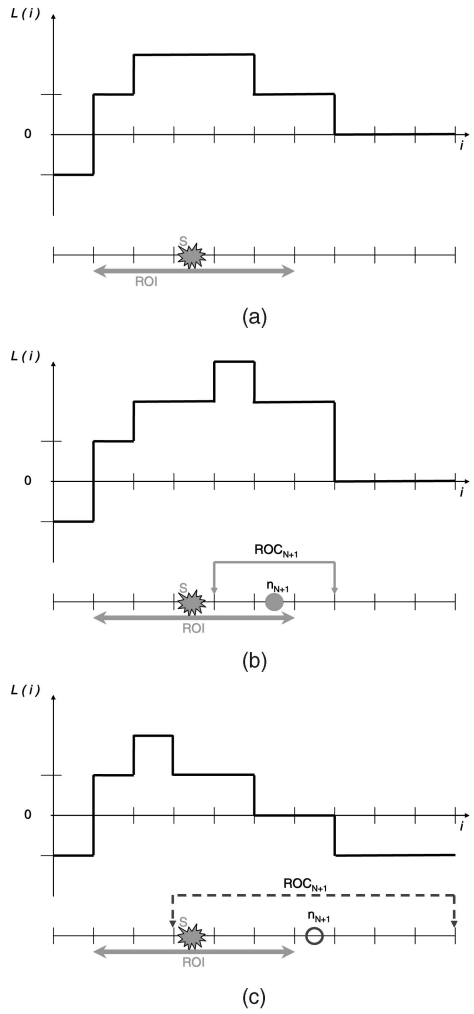


Fig. 16. SNAP Likelihood matrix along the line between the source and the $(n+1)$th sensor. (a) L function after contribution of n sensors. (b) $r_{ROC} < r_{ROI}$. (c) $r_{ROC} > r_{ROI}$.

holds for any sensor field with n sensors. We need to show that this also holds for a field with $n+1$ sensors.

Fig. 16a shows a typical likelihood function due to n sensors along the line that connects the source with the $(n+1)$th sensor. According to the lemma statement, the source position is included in $A_{max}(n)$ (the set of cells with maximum value when n sensors are used). If this sensor is placed inside the ROI of the source, it will be alarmed, and it will add $+1$ to all cells of the likelihood matrix that fall inside its ROC. If $r_{ROC} \geq r_{ROI}$, the ROC of the sensor will also include the true source position and it is guaranteed that the source will remain in the $A_{max}(n+1)$. If the sensor is placed outside the source ROI, then it will be nonalarmed, and thus, will add a -1 contribution to all cells in its ROC. Thus, if $r_{ROC} \leq r_{ROI}$, the -1 contribution will not affect the cell with the true source position, thus it is guaranteed that the source will remain in $A_{max}(n+1)$. Thus, to guarantee that the source will always remain in the $A_{max}(n+1)$, we need $r_{ROC} = r_{ROI}$.

If $r_{ROC} \neq r_{ROI}$, one can easily construct examples where $A_{max}(n+1)$ does not include the true source position. If $r_{ROC} < r_{ROI}$, one can place the $(n+1)$th sensor just inside the ROI, so it will be alarmed but its ROC will not include $(0,0)$. Thus, it may add $+1$ to some of the cells in $A_{max}(n)$, which may constitute an $A_{max}(n+1)$ that does not include the true source position (see Fig. 16b). If $r_{ROC} > r_{ROI}$, one can place the $(n+1)$th sensor just outside the ROI, so it will be nonalarmed but its ROC will include $(0,0)$. By subtracting one from the cell with the true source position, it is possible that some other cells in $A_{max}(n)$ but further away from the $(n+1)$th sensor are not affected. As a result, they may constitute an $A_{max}(n+1)$ that once more does not include $(0,0)$, the true source position (see Fig. 16c).

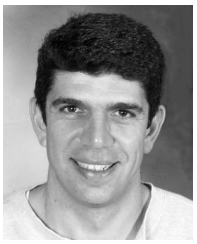
ACKNOWLEDGMENTS

This work is partly supported by the Cyprus Research Promotion Foundation under contracts ΠΛΗΡΟ/1104/10 and ΕΡΥΔΙ/0105/01.

REFERENCES

[1] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach.* Morgan Kaufmann, 2004.
[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Magazine,* vol. 40, no. 8, pp. 102-114, Aug. 2002.
[3] C. Chong and S. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proc. IEEE,* vol. 91, no. 8, pp. 1247-1256, Aug. 2003.
[4] S. Kumar, F. Zhao, and D. Shepherd, "Collaborative Signal and Information Processing in Microsensor Networks," *IEEE Signal Processing Magazine,* vol. 19, no. 2, pp. 13-14, Mar. 2002.
[5] J. Chen, R. Hudson, and K. Yao, "Maximum-Likelihood Source Localization and Unknown Sensor Location Estimation for Wideband Signals in the Near-Field," *IEEE Trans. Signal Processing,* vol. 50, no. 8, pp. 1843-1854, Aug. 2002.
[6] X. Sheng and Y. Hu, "Energy Based Acoustic Source Localization," *Proc. Second Int'l Workshop Information Processing in Sensor Networks (IPSN),* pp. 286-300, Apr. 2003.
[7] D. Li, K. Wong, Y. Hu, and A. Sayeed, "Detection, Classification, and Tracking of Targets," *IEEE Signal Processing Magazine,* vol. 19, no. 3, pp. 17-29, Mar. 2002.
[8] R. Niu and P. Varshney, "Target Location Estimation in Wireless Sensor Networks Using Binary Data," *Proc. 38th Ann. Conf. Information Sciences and Systems,* Mar. 2004.
[9] G. Nofsinger, "Tracking Based Plume Detection," PhD thesis, Dartmouth College, Hanover, 2006.
[10] B. Krishnamachari and S. Iyengar, "Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks," *IEEE Trans. Computers,* vol. 53, no. 3, pp. 241-250, Mar. 2004.
[11] A. Arora et al. "A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification and Tracking," *Computer Networks,* vol. 46, pp. 605-634, 2004.
[12] M. Ding, F. Liu, A. Thaeler, D. Chen, and X. Cheng, "Fault-Tolerant Target Localization in Sensor Networks," *EURASIP J. Wireless Comm. and Networking,* vol. 2007, no. 1, p. 19, 2007.
[13] M.P. Michaelides and C.G. Panayiotou, "Subtract on Negative Add on Positive (SNAP) Estimation Algorithm for Sensor Networks," *Proc. Seventh IEEE Int'l Symp. Signal Processing and Information Technology,* Dec. 2007.
[14] M.P. Michaelides and C.G. Panayiotou, "Event Detection Using Sensor Networks," *Proc. 45th IEEE Conf. Decision and Control,* pp. 6784-6789, Dec. 2006.
[15] R. Niu, P. Varshney, M. Moore, and D. Klamer, "Decision Fusion in a Wireless Sensor Network with a Large Number of Sensors," *Proc. Seventh IEEE Int'l Conf. Information Fusion (ICIF '04),* June 2004.
[16] S. Ross, *Introduction to Probability Models,* eighth ed. Academic Press, 2003.

Michalis P. Michaelides received the BS degree in 1997 and the MS degree in electrical engineering in 1998 from Purdue University, Indiana. From 1999 to 2002, he worked as a medical service engineer for Medvision, and from 2002 to 2006, as an IT officer for the Nicosia Race Club. In September 2004, he was accepted as a part-time PhD student in the Electrical and Computer Engineering Department at the University of Cyprus. Since January 2007, his status is that of a full-time PhD special scientist. His research interests include wireless sensor networks, distributed detection and estimation, collaborative signal information processing, and fault tolerance. He is a member of the IEEE.



Christos G. Panayiotou received the BSc and PhD degrees in electrical and computer engineering from the University of Massachusetts at Amherst in 1994 and 1999, respectively, and the MBA degree from the Isenberg School of Management of the same university in 1999. From 1999 to 2002, he was a research associate at the Center for Information and System Engineering (CISE) and the Manufacturing Engineering Department at Boston University. During 2002 to 2003, he was a visiting lecturer in the Electrical and Computer Engineering Department at the University of Cyprus (UCY). During 2003-2008, he was an assistant professor and currently is an associate professor both in the Electrical and Computer Engineering Department at the UCY. He is a founding member of the KIOS Research Center for Intelligent Systems and Networks at the UCY and currently its interim deputy director. His research interests include wireless, ad hoc, and sensor networks, computer communication networks, quality of service (QoS) provisioning, distributed control systems, optimization and control of discrete-event systems, resource allocation, and simulation. He is also a reviewer of various conferences and journals and has served in the organizing and program committees of various international conferences. He is an associate editor of the Conference Editorial Board of the IEEE Control Systems Society. He is a senior member of the IEEE.