Fault Tolerant Localization and Tracking of Multiple Sources in WSNs using Binary Data

Michalis P. Michaeilides, Member, IEEE, Christos Laoudias, Student Member, IEEE, and Christos G. Panayiotou, Senior Member, IEEE

Abstract—This paper investigates the use of a Wireless Sensor Network for localizing and tracking multiple event sources (targets) using only binary data. Due to the simple nature of the sensor nodes, sensing can be tampered (accidentally or maliciously), resulting in a significant number of sensor nodes reporting erroneous observations. Therefore, it is essential that any event tracking algorithm used in Wireless Sensor Networks (WSNs) exhibits fault tolerant behavior in order to tolerate misbehaving nodes. The main contribution of this paper is the development and analysis of a low-complexity, distributed, real-time algorithm that uses the binary observations of the sensors for identifying, localizing and tracking multiple targets in a fault tolerant way. Specifically, our results indicate that the proposed algorithm retains its performance in tracking accuracy in the presence of noise and faults, even when a large percentage of sensor nodes (25%) report erroneous observations.

Index Terms—Wireless Sensor Networks, multiple sources, localization and tracking, binary data, fault tolerance.

I. INTRODUCTION

Wireless Sensor Networks (WSNs) have been proposed for monitoring large areas against the presence of event sources1. The events can be intruders, enemy vehicles, pollutant sources or fires depending on the application [1]. In all situations, each event source emits a signal or substance that attenuates inside the area under observation and is measured by some of the sensor nodes based on their location relative to the events. We assume that the sensor nodes are static and binary in nature; they become alarmed when their received signal exceeds a certain threshold. Based on the sensor binary observations, the main goal of this work is to localize all event sources and track their movements in real-time and in a fault tolerant way.

Sensor nodes are often envisioned to operate unattended for long periods of time in hostile environments, e.g., airdropped inside an area for localizing potential targets and intruders. From a security point of view, there is an increased probability that the sensor readings are affected by a malicious attacker. For instance in [2], the authors present a scenario with malicious actuator nodes deployed to perturb or distort the readings of neighboring sensor nodes within their actuation radius. In addition, sensor nodes can provide erroneous observations for a variety of other reasons: noise, energy depletion, environmental harsh conditions of operation and software problems. For example, the authors in [3] report situations where a node would constantly report the presence of an event (false positive) when its processing board was overheated, or scenarios in which sensor nodes exhibited unexpected behavior due to software bugs that were not detected. Since sensing can be tampered (accidentally or maliciously), resulting in a significant number of sensor nodes reporting erroneous observations, it is essential that any tracking algorithm used in WSNs exhibits fault tolerant behavior.

In this paper, we develop a fault tolerant localization and tracking algorithm for multiple event sources in WSNs using only binary sensor observations. The proposed algorithm, FTLT (Fault Tolerant Localization and Tracking), has the following three main phases: (i) Identification, (ii) Localization, and (iii) Smoothing. In the Identification phase, all alarmed sensor nodes broadcast their status in their own neighborhood. Then, based on the received information and using a distributed leader election protocol a subset of the alarmed sensor nodes are self-appointed as leaders. After the first phase, the number of elected leaders should correspond to the number of sources present in the field. In the Localization phase, each leader runs a distributed localization algorithm (dSNAP) to determine the location of the source by only contacting the sensor nodes in its own neighborhood that are relevant to the estimation problem. The location estimates are further refined in the Smoothing phase, where each leader uses a smoothing algorithm to better approximate the path of the source and in order to predict the next leader responsible for tracking it.

The FTLT algorithm, which constitutes the main contribution of this work, features low complexity, distributed implementation, real-time application, and fault tolerance in all three phases of its implementation. The algorithms used in both the Identification and the Localization phase require only simple operations (i.e., additions of $(\pm 1)$). Furthermore, the Kalman filter used in the Smoothing phase was found to be the most efficient tracking method in terms of memory and computation out of the different Bayes filters tested in [4]; however, we point out that for certain applications (e.g., maneuvering targets), other filtering techniques may provide more accurate tracking at the expense of computational complexity. In addition, the proposed approach is distributed in nature, which

1In the sequel the terms event sources and targets will be used interchangeably.
allows for real-time tracking and scalable implementation. The most important quality of FTLT is fault tolerance, i.e., its ability to withstand a large percentage of erroneous data coming from the faulty or compromised sensors. In particular, our results indicate that FTLT can accurately identify, localize and track all targets in the field, avoiding at the same time the creation of spurious tracks, even when as many as 25% of the sensor nodes report erroneous observations.

The paper is organized as follows. In Section II, we review related work on event localization and target tracking. Next, in Section III, we present the model we have adopted, the underlying assumptions and definitions. The next three sections present and analyze the different phases of the proposed FTLT algorithm: source identification in Section IV, distributed localization in Section V the target tracking in Section VI. In Section VII, we provide extensive simulation results with emphasis on fault tolerance. We conclude with Section VIII.

II. RELATED WORK

A. Localization

Localization has been studied extensively in the context of wireless cellular networks, using signal strength and angle or time of arrival measurements, to support location-based services; see [5] for an overview. However, WSNs come with unique features and constraints that make the development of position determination algorithms for such distributed environments a challenging task. In the last 20 years, event localization techniques have been proposed in the literature using arrays of sensors for radar, sonar and acoustic target tracking applications; see [6], [7], [8] and references therein. Sensor nodes are expected to be low-cost, simple devices with limited resources (processing capabilities, memory, and power). For a positioning algorithm to be implementable in the context of WSNs, it has to be simple enough so it can be performed by any sensor node in a distributed fashion, energy-efficient and fault-tolerant. Fault tolerance has been outlined among the major challenges in WSNs in the new research area of CSIP (Collaborative Signal Information Processing) [9]. Classical estimation techniques like least squares and maximum likelihood usually give accurate results at the expense of heavy message exchanges and computations. Therefore, it is not practical for a sensor node to implement these methods in a distributed fashion. Moreover, the fault tolerance of these methods has not been demonstrated.

B. Binary Estimators

For this paper we focus on estimators that use binary data to localize the event. Binary decisions are simple problems a sensor node can solve by just comparing its received signal to a predefined threshold. Binary decisions are also less sensitive to calibration mismatches and varying sensor sensitivities. Moreover, using binary observations limits the bandwidth usage and conserves energy; only single-bit information needs to be transmitted. For event localization in WSNs using binary data several methods have been proposed. The Centroid Estimator (CE) [10] which simply takes the centroid of the positions of all alarmed sensor nodes as the estimated event location. Although sub-optimal, this simple method works quite well under conditions of dense sensor deployment and when events are not located close to the field boundaries. Another approach is based on the classical maximum likelihood (ML) estimator which is shown to be optimal in [11] when enough sensor measurements are used. Both of these methods however, can yield significant estimation errors when the sensor nodes start failing; CE is sensitive to false positives, i.e., sensor nodes far away from the source becoming falsely alarmed. On the other hand, ML is extremely sensitive to false negatives, i.e., when a node located close to the event does not become alarmed. More recently, we have developed a fault-tolerant maximum likelihood (FTML) estimator [12] and its performance is closely approximated by the SNAP (Subtract on Negative Add on Positive) algorithm [13]. SNAP as proposed in our earlier work, is a centralized algorithm in the sense that it constructs a scoring matrix over the entire field utilizing the binary information from all sensor nodes in the field. Moreover, it assumes a single static target in the field. Motivated by SNAP, in this paper we present a distributed localization algorithm where a node that is elected as cluster leader uses only locally available data for locating a nearby source. Furthermore, multiple leaders can be elected in order to identify and locate multiple sources. A different approach for achieving robust target localization using binary decisions is proposed in [14]. Instead of using the original binary decisions, the authors first utilize a local vote decision fusion (LVDF) scheme [15] for correcting the original decisions based on the majority of the neighboring sensors’ decisions followed by numerical optimization techniques, including direct methods, maximum likelihood (ML) and the expectation maximization (EM) algorithm. Compared to this approach, FTLT has lower complexity because it does not require a pre-processing step (which makes the decisions of the sensors correlated) and thus avoids using nonlinear optimization techniques. In fact, the scoring matrix utilized by the dSNAP algorithm for localizing the target, can be constructed by merely using additions of ±1.

C. Tracking

Tracking techniques rely on Bayesian filtering variants [4], such as Kalman Filter or Particle Filter, to mitigate the effect of measurement noise and alleviate high positioning errors that do not reflect the target’s mobility pattern. The problem of tracking multiple targets using WSNs has been explored by many prior references in a variety of different contexts (e.g., [16], [8], [17], [18], [19], [20], [21]). However, none of the aforementioned references considers fault tolerance. In [22], the authors consider a 1-D scenario in order to prove some fundamental limits of the binary sensor network in discriminating multiple targets while the 2-D case is considered in a more recent publication [23]. Although similar in concept to ours, i.e both methods utilize the information from non-alarmed sensors for localizing the targets, the method in [23] employs purely geometric techniques for localizing the targets, is used for off-line tracking and it requires the solution of the maximal independent set problem (an NP complete
problem in the 2-D case). On the other hand, FTLT is designed for on-line tracking, is computationally very efficient (mostly linear and at most polynomial complexity) and is based on a probabilistic framework which makes it significantly more fault tolerant. Some preliminary analysis and simulation results of our proposed tracking solution have appeared in our earlier work [24].

D. Source Identification

In [19], the authors suggest a distributed group management algorithm for electing a single leader and resolving contention via message exchange. They use a time stamp to elect the leader that first witnessed the event. In [20], the sensor nodes are aggregated into collaborative groups for a target counting task. For each distinct target a single leader is elected via one hop information exchange by identifying the sensor node that received the highest signal power. In [21], the authors extend these results to multiple sources.

III. MODEL & DEFINITIONS

In this section we present the modeling assumptions and some important definitions that will be used in the sequel.

Assumptions and Notations:

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 $(x_n, y_n)$, $n = 1, \cdots , N$ and is assumed known.
2) Each sensor has a limited communication range $R_s$.
3) A set of $K$ point event sources are moving inside $A$ according to a piecewise linear pattern. At time $t$, each source $k$ is located at position $(x_k(t), y_k(t))$ inside $A$. Each source emits a continuous signal that attenuates uniformly in all directions and can be measured by some of the sensor nodes, depending on their location relative to the source.

All assumptions are quite common and reasonable for sensor networks. We point out that an extension of our results to problems with non uniform propagation pattern is possible by appropriately estimating the influence fields of the sources (see for example [26]).

In the adopted model, $z_n(t)$ is the received signal of sensor $n$ located at $(x_n, y_n)$ and is given by the sum of the signals from all sources at the sensor location:

$$z_n(t) = \min \left\{ V_{\text{max}} \sum_{k=1}^{K} s_{n,k}(t) + w_n(t) \right\},$$

for $n = 1, \cdots , N$ and $t = 1, \cdots , T$. $V_{\text{max}}$ is a sensor specific parameter that reflects the maximum measurement that a sensor can register. In addition, the signal from each source $k = 1, \cdots , K$ attenuates inversely proportional to the distance from the source location raised to some power $\alpha \in \mathbb{R}^+$ which depends on the environment. In other words,

$$s_{n,k}(t) = \frac{c_k}{r_{n,k}(t)^\alpha}$$

where $c_k$ is some constant characterizing the $k$-th source and $r_{n,k}(t)$ is the radial distance of sensor node $n$ from source $k$ at time $t$, i.e.,

$$r_{n,k}(t) = \sqrt{(x_n - x_k(t))^2 + (y_n - y_k(t))^2}.$$

Also, $w_n(t)$ is additive white Gaussian noise, $w_n(t) \sim \mathcal{N}(0, \sigma_n^2)$ for $n = 1, \cdots , N$ and $t = 1, \cdots , T$. We should point out that the results presented in this paper do not depend on the particular model adopted. In fact, any model where the signal attenuates with the distance from the source (i.e. exponential) could also be used.

Next, we assume that the sensor nodes have been programmed with a common threshold $\gamma$ which depends on the acceptable false alarm rate [27]. Given $\gamma$, at any time instant $t$, the sensors can be in one of the following two states:

- **Alarmed**: If $z_n(t) \geq \gamma$ (positive observation)
- **Non-Alarmed**: If $z_n(t) < \gamma$ (negative observation)

The value of $\gamma$ determines the alarm status of the sensor nodes in the estimation procedure. In the sequel, we assume that $\gamma$ is chosen large enough so that the probability of falsely alarmed sensors (due to noise) is smaller than 0.01. Note that such an assumption is reasonable since users will very likely abandon a system with frequent false alarms.

![Fig. 1. Schematic representation of the definitions of the various regions that will be used in the sequel: 1. Region of Influence (ROI) 2. Region of Coverage (ROC$_n$) and 3. Region of Subscription (ROS$_n$). Alarmed sensor nodes with positive observations are shown in solid color.](image)
Next, we define the footprint of the events:

**Definition 1: Region of Influence (ROI)** is the area around the source location inside which a sensor node will be alarmed with high probability (at least 0.5)\(^2\).

For the model of (2), the ROI of a single source with \(c_k = c\) is a disc centered at the source location with radius \(R_I = \sqrt{c/\gamma}\) (demonstrated in Fig. 1). Note that for multiple sources, the shape and size of the ROI depends on the separation distances between the sources. For two sources, the following result holds.

**Lemma 1:** For any two sources, their ROI becomes connected if and only if their separation distance \(d\)

\[
d \leq \frac{1}{\sqrt{\gamma}} \left( \alpha \sqrt{c_1} + \sqrt{c_2} \right)^{\alpha + 1}.
\]

**Proof:** Without loss of generality suppose that source 1 is located at the origin and source 2 at point \(d\) along the \(x\)-axis. The total signal (neglecting noise) at any point \(x\) in the interval \([0,d]\) along the \(x\)-axis is given by

\[
s(x) = \frac{c_1}{x^\alpha} + \frac{c_2}{(d-x)^\alpha},
\]

which has a minimum at point

\[
x_{min} = \frac{d \sqrt{c_1}}{\alpha \sqrt{c_1} + \sqrt{c_2}}.
\]

The signal strength at \(x_{min}\) is given by

\[
s(x_{min}) = \frac{1}{d^\alpha} \left( \alpha \sqrt{c_1} + \sqrt{c_2} \right)^{\alpha + 1}.
\]

The ROI of the two sources remains connected as long as \(s(x_{min}) \geq \gamma\). Solving this inequality we obtain the lemma result.

**Corollary 1:** If the two sources are identical, i.e., \(c_1 = c_2 = c\), then their ROI is connected if and only if

\[
d \leq 2 \sqrt{2} \sqrt{c/\gamma} = 2 \sqrt{2} R_I
\]

where \(R_I = \sqrt{c/\gamma}\) is the radius of the ROI of a single source. The factor \(\sqrt{2}\) represents the effect of the superposition of the two sources which goes to 1 as \(\alpha\) increases. For simplicity, for the rest of this paper we will be assuming identical sources with \(c_1 = c_2 = c\) and \(\alpha = 2\).

From the sensor node perspective we define two more regions also demonstrated in Fig. 1 for the single source case.

**Definition 2: Region of Coverage (ROC)\(_n\)** of sensor node \(n\) is the area around a sensor node location inside which if a source is present, then it will be detected with high probability (at least 0.5).

For the single source case, \(ROC\(_n\)\), by symmetry with the \(ROI\), becomes a disc centered at the alarmed sensor node \(n\) location with area equal to that of the \(ROI\). Using the model of (2), the radius of \(ROC\(_n\)\) becomes \(R_c = \sqrt{c/\gamma}\) for all \(n\). If two or more sources are present, then the \(ROC\(_n\)\) remains a disc since a single sensor cannot differentiate between the signals from the different sources. Note that in this case, the value of the radius \(R_c\) becomes a function of the relative positions of the sources and can no longer be computed analytically. In general, using the same \(R_c\) for the multiple sources case as the one computed for the single source case (i.e., \(R_c = \sqrt{c/\gamma}\)) yields the most accurate results on average (see Section VII).

**Definition 3: Region of Subscription (ROS)\(_n\)** of sensor node \(n\), is the area around the sensor node \(n\) location inside which the sensor node needs to subscribe for information from all other sensor nodes which are relevant to the application considered. We also refer to this region as the neighborhood of sensor node \(n\).

We will be demonstrating in the sequel, that the \(ROS\(_n\)\) size should be selected based on the application and the relative size of the \(ROC\(_n\)\).

Finally, we define a Fault Model where each sensor node can exhibit erroneous behavior with probability \(p_f\) and in this case its original belief is reversed. When applying the fault model, some sensor nodes that fall outside the ROI become alarmed as a result of a fault and are shown as false negatives. Similarly, sensors that fall inside the ROI become non-alarmed as a result of a fault and are shown as false positives. This fault model is chosen to reflect real world conditions where sensor nodes may provide erroneous and often quite unpredictable observations due to different reasons such as noise, energy depletion, environmental harsh conditions of operation and software problems [3].

In the next three sections we present the main components of the binary tracking approach (FTLT), source identification, localization and smoothing.

### IV. Source Identification

During the identification phase a number of the sensor nodes are elected as leaders. The number of elected leaders corresponds to the number of sources identified in the field. Sensor node \(n\) can become a leader in two possible ways:

1. **Node** \(n\), receives a message from a neighboring node which is currently a leader, indicating that a source is moving closer to \(n\) and thus it should become the new leader.

2. **Through** a distributed election protocol where node \(n\) did not hear any neighbor becoming a leader and the majority of its neighbors are in the alarmed state.

A current leader can determine whether a source has moved closer to one of its neighbors through the smoothing algorithm (e.g. the Kalman filter) and thus it can inform its neighbors that it should become the next leader. More details on this approach are presented in Section VI. The distributed fault tolerant leader election protocol (D-FTLEP) is presented next and is also demonstrated in Fig. 2.

Initially, each alarmed sensor node broadcasts an **ALARM** message inside its ROS. Next, each alarmed sensor \(n\) computes the following function \(F_n\) using the received information,

\[
F_n = \sum_{m \in ROC_n} b_{mn}
\]
Algorithm 1: D-FTLEP: Distributed Fault Tolerant Leader Election Protocol

Input: Set of neighboring alarmed sensor nodes $A$.
Output: Elected leader status.
1: All alarmed sensors broadcast an ALARM message.
2: Node $n$ calculates the function $F_{n}$ using the received ALARM messages from its neighbors.
3: If $F_{n} > 0$ then continue with next step else STOP.
4: Wait for a period $h(1/F_{n})$.
5: If during the waiting period a LEADER message with value $f \geq F_{n}$ is received STOP.
6: Broadcast LEADER message with value $F_{n}$ and assume leadership role.

where,

$$b_{m} = \begin{cases} +1 & \text{if } m \text{ is alarmed}, \\ -1 & \text{otherwise}, \end{cases}$$  \hspace{1cm} (5)

Note that for the sensor nodes inside the ROS of node $n$ for which no message is received, a non-alarm status is implied. If $F_{n} > 0$ for at least one alarmed sensor node then we have Detection of at least one source. The use of the value $F_{n}$ is an essential part of the leader election algorithm for the following two reasons. First, it causes the leader node to be elected close to the actual source location since on average sensor nodes closer to the source have more alarmed neighbors. Keep in mind that this is achieved using only binary information from the sensors. Second, by using a bounded $\pm 1$ contribution from the sensors it can handle a percentage of erroneous observations (false positives) as well as dropped packets (false negatives) coming from the sensor nodes.

Next, sensor node $n$ with $F_{n} > 0$ waits for a period that is given by a strictly increasing function $h(x)$. If during this period, $n$ does not receive a LEADER message with value $f \geq F_{n}$ it implies that $n$ is the node with the most alarmed neighbors and is likely located close to the source, thus it becomes a leader and broadcasts a LEADER message to its neighbors.

If node $n$ receives a LEADER message with value $f \geq F_{n}$, then it is “suppressed” meaning that for a certain period it cannot become a leader itself. Note that according to the algorithm a sensor $m$ outside the ROS of an elected leader with $F_{m} > 0$ may also become leader if it has the maximum value among its neighbors (excluding the ones in the ROS of the already elected leader which are suppressed). This turns out to be an important feature of the algorithm for correctly counting sources that are located close to each other (e.g. targets with crossing trajectories). The accuracy of the detection, however, depends on the relative size of the ROS (compared to the ROC) and the separation distance between the sources. Also note that for every source the number of messages required for the leader election protocol is linear with respect to the number of neighbors of the leader.

A leader may initiate and maintain a track as described above. A leader should also terminate a track when a source moves away and it is not clear who the next leader is going to be. Therefore, a leader $n$ who did not determine the next leader and for three consecutive steps computes a value $F_{n} \leq 0$ simply terminates the track.

V. DISTRIBUTED EVENT SOURCE LOCALIZATION ALGORITHM

The event source localization algorithm is motivated by the SNAP algorithm that is in fact a maximum likelihood estimator as shown in [13]. For completeness, the mathematical relationship between SNAP and maximum likelihood estimation is also provided in the Appendix. The version presented in this section (dSNAP), however, is adapted for a distributed environment. Furthermore, the analysis presented in this section involves multiple event sources while [13] addressed only the single source case. The algorithm assumes a quantized field, thus the entire area is divided in a grid $G$ with $G \times G$ cells and grid resolution $g$. Depending on the resolution a cell may contain multiple sensors or no sensors at all. The position index of each node $n = 1, \cdots, N$ is denoted by $(X_{n}, Y_{n})$, where $X_{n}, Y_{n} \in \{1, 2, \cdots, G\}$ are the discrete position indices related to the real-valued position of the node $(x_{n}, y_{n})$ by $X_{n} = \left\lfloor \frac{x_{n}}{g} \right\rfloor, Y_{n} = \left\lfloor \frac{y_{n}}{g} \right\rfloor$.

The algorithm is run by the leader node $l$ which needs to collect the alarm status of all sensors inside its Region of Subscription ($ROS_{l}$). Note that this information is already available from the identification phase. Using this information, the leader constructs the scoring matrix $L_{l}$ over a sub-grid $G_{l}$ around its location. The maximum value of $L_{l}$ points to the estimated position of the source. Next we present the details of the algorithm.

The leader node $l$, $l \in \{1, \cdots, N\}$ is associated with $G_{l}$, a sub-grid of $G$ with $G_{l} \times G_{l}$ cells, centered around its location.
\((X_l, Y_l)\). The size of the sub-grid \(G_l = \left\lfloor \frac{R_s}{g} \right\rfloor + 1\) depends on the size of the ROS of the grid resolution \(g\). Furthermore, node \(l\) defines a \(G_l \times G_l\) Scoring matrix \(L_l\) where each element \((i,j)\) of \(L_l\) corresponds to a cell \((u,v)\) of \(G_l\). This relation is given by a mapping \(M_l : G_l \rightarrow L_l\), thus

\[
M_l([u,v]^T) = \left[ u - X_l + \frac{G_l}{2}, v - Y_l + \frac{G_l}{2} \right]^T
\]

where \(u, v \in \{1, \cdots, G_l\}\) and \(i, j \in \{1, \cdots, G_l\}\). For every element of \(L_l\), the leader adds the contribution of each sensor that has the corresponding cell in its ROC. The contributions can be \(\pm 1\) depending on the sensor’s state: \(+1\) on alarmed (positive observation) and \(-1\) on non-alarmed (negative observation). More specifically, the leader updates every element \((i,j)\) of \(L_l\) using

\[
L_l(i,j) = \sum_{m \in \text{ROS}_l} b_m(i,j), \quad i, j \in \{1, \cdots, G_l\}
\]

where

\[
b_m(i,j) = \begin{cases} 
+1 & \text{if } \text{m alarmed AND } M_l^{-1}(i,j) \in \text{ROC}_m \\
-1 & \text{if } \text{m non-alarmed AND } M_l^{-1}(i,j) \notin \text{ROC}_m \\
0 & \text{otherwise}
\end{cases}
\]

and \(\text{ROC}_m\) is the set of all grid cells that are covered by the ROC of sensor \(m\). In other words, the leader in a distributed fashion subtracts on Negative observations and adds on positive (dSNAP). The maximum of the scoring matrix points to the estimated position of the event source which is taken to be the center of the corresponding cell of \(G_l\). Let \(L_l(i^*, j^*) = \max_{i,j} L_l(i,j)\), then the estimated source position is the center of \(M_l^{-1}(i^*, j^*)\). If two or more elements have the same maximum value, the estimated position is the centroid of the corresponding cell centers. The complexity of the algorithm is linear with respect to the number of elements of the \(L_l\) matrix, or \(O(G_l^2)\) which is significantly more efficient than the centralized approach where the number of messages is proportional to the total number of nodes in the field (not just the neighbors) and the search for the maximum is over a scoring matrix that covers the entire field.

Fig. 3 outlines the algorithm used by the leader node for constructing the scoring matrix. Fig. 4 illustrates how the algorithm is used through a simple test case scenario assuming a square ROC.

A. Region of Subscription (ROS)

In order to estimate the event location, the leader node \(l\) needs to collect information from sensor nodes located inside its neighborhood (ROS). Since energy efficiency is a major consideration in wireless sensor networks and communication is the most expensive operation in terms of energy, we would like to find the smallest \(\text{ROS}_l\) that achieves the desired objectives in terms of estimation accuracy. This is given by the following Lemma.

**Lemma 2:** Assuming the leader is correctly alarmed (i.e., it falls in the ROI of the source) then its \(\text{ROS}_l\) radius \(R_s = 2R_c\).

**Proof:** If the leader is correctly alarmed (it falls in the source ROI) then the maximum of the scoring matrix should occur in the elements that correspond to the ROC of the leader (\(\text{ROS}_l\)). By definition, \(\text{ROS}_l\) is a disc of radius \(R_c\) centered at the leader location. Since, we do not know the exact location of the source, \(\text{ROS}_l\) should include all sensor nodes relevant to the estimation problem. The ROC of a sensor that is within a distance \(2R_c\) from the leader can directly influence the results of the scoring matrix so we need to have \(R_s \geq 2R_c\). On the other hand, the ROC of any sensor located further away than \(2R_c\) has no overlap with \(\text{ROS}_l\) and therefore it cannot affect the computation of the element with the maximum. So \(R_s \leq 2R_c\). This completes the proof.

B. Multiple Sources

Next, we present a bound on the performance of the distributed localization algorithm in the presence of multiple sources.

**Lemma 3:** In the one-dimensional scenario, assuming a dense deployment (i.e., where each cell includes a sensor), and under the assumptions made in Section III, the dSNAP localization algorithm cannot distinguish any two sources with connected ROI.

**Proof:** For the proof, we assume no-fault conditions and a uniform and dense deployed sensor network in 1-D such that each location on the line is adequately covered. Without loss of generality, consider two identical sources located less than \(d \leq 2\sqrt{2}R_c\) apart as shown in Fig. 5(a) (i.e., their ROI
is connected). Then, according to the localization algorithm, there exists a continuous area $A_{\text{max}}$ between the two sources whose corresponding scoring value attains the maximum value $L_{\text{max}}$. Note, that in the area between the two sources, it is impossible to get any correctly non-alarmed sensors since their ROI is connected and thus only a single continuous peak is identified. So for this case, the algorithm would estimate a single source at the centroid of $A_{\text{max}}$. Now, consider the case where their ROI is not connected demonstrated in Fig 5(b). In this case, given a dense deployment, there exists at least one non-alarmed sensor node between the two sources. The negative contribution of the non-alarmed sensor allows $A_{\text{max}}$ to have two distinct peaks on the constructed scoring matrix $L_i$, identifying two distinct targets.

**Proof:** For the proof, we assume a uniform and dense deployed sensor network in 2-D (e.g., grid topology) such that each location on the field is adequately covered. Without loss of generality, consider two identical sources at locations (0,0) and (d,0) with $d \leq 2R_c$. As before, what we need to show is that $A_{\text{max}}$ (i.e., the set of cells with maximum scoring value) is a continuous area between the two sources and thus the localization algorithm would estimate a single source at the centroid of $A_{\text{max}}$. For this we need to show that sensors located within $R_c$ from $A_{\text{max}}$ are all alarmed making it impossible to get any negative contributions inside this area. For the proof, it is enough to consider the status of the sensor that receives the minimum total signal from the sources located $R_c$ units away from $A_{\text{max}}$. If this sensor is shown to be alarmed then any other sensor located within $R_c$ units from $A_{\text{max}}$ would also be alarmed. Specifically, we consider a sensor node $n$ located at $(d/2, R_c)$ since it can be shown that the minimum of the total signal in the x-direction occurs midway between the two sources (i.e., on the line $x = d/2$). Let $r$ denote the radial distance of node $n$ from either of the sources. Using basic trigonometry, $r \leq \sqrt{2}R_c \Rightarrow s_n = 2r/\pi c \geq \pi c/\pi c = \gamma$, thus node $n$ should be alarmed. This completes the forward direction of the proof. For proving the other direction of the lemma we use contradiction. Specifically, consider the case where $d > 2R_c$. In this case, given a dense deployment, there exists at least one sensor node $n$ at location $(d/2, R_c)$ which is non-alarmed since $r > \sqrt{2}R_c \Rightarrow s_n = 2r/\pi c < \gamma$. The negative contribution of the non-alarmed sensor breaks the continuity of $A_{\text{max}}$ at point $(d/2, 0)$ creating two distinct peaks on the constructed scoring matrix $L_i$, thus identifying two distinct targets.

Based on the results of Lemma 4, it becomes evident that in two dimensions the dSNAP algorithm can correctly localize two sources even with connected ROI by using the contributions of non-alarmed sensors located on the center-line which is perpendicular to the line that connects the two sources. Recall from Corollary 1, that the ROI of the two sources becomes connected for separation distances $d \leq 2\sqrt{3}R_t = 2\sqrt{2}R_c$. We should also point out that even though the above lemmas indicate that the localization algorithm cannot distinguish between two sources that are sufficiently close to each other, this result only holds for dense deployments. In non-dense deployments, it is possible to have some sensors appropriately positioned such that the scoring matrix may include two or more non-connected sets of cells that take a local maximum value which can identify two different sources. For example, in Fig. 5(a), it is possible to have just two sensors located at cells 2 and 9. In this case, $A_{\text{max}}$ will include two non-connected sets of cells indicating that two sources exist.

**VI. SMOOTHING AND TRACKING ALGORITHM**

Once the leader estimates the position of the source, it also runs a smoothing algorithm in order to improve the location estimate and to predict the next leader that should continue the track. For the smoothing algorithm we have adopted the Kalman filter due to its simplicity, however, we point out that other algorithms could also be used. Using the estimated
source velocity and the positions of all neighboring nodes, a leader determines if one of its neighbors should become the next leader.

Since we have no a priori information about the movement of the targets, we assume the following random force model which gives the position and velocity of each target \( k \) at every instant \( t \):

\[
x_k(t) = \begin{bmatrix} I_2 & \tau t I_2 \\ 0 & I_2 \end{bmatrix} x_k(t-1) + \begin{bmatrix} \tau t^2 I_2 \\ \tau t I_2 \end{bmatrix} q(t-1)
\]

where, \( x_k(t) = [x^r_k(t) \ y^r_k(t) \ u^r_k(t) \ u^s_k(t)]^T \) represents the process state (\([u^r_k(t) \ u^s_k(t)]^T \) is the source velocity), \( I_2 \) is the \( 2 \times 2 \) identity matrix and \( \tau \) is the sampling interval. The two dimensional process noise \( q(t) \sim N(0, Q) \) is assumed to be drawn from a zero mean normal distribution with covariance matrix \( Q = \sigma Q^2 I_2 \).

The observations, i.e., position estimates provided by dSNAP, are assumed to be normally distributed according to the following equation:

\[
y_k(t) = \begin{bmatrix} I_2 & 0 \end{bmatrix} x_k(t) + r(t)
\]

where, \( y_k(t) \) only measures the source current position but not the velocity and the noise vector \( r(t) \) represents the uncertainty in the position estimates. The noise \( r(t) \sim N(0, R) \) is assumed to be zero mean normally distributed with covariance matrix \( R = \sigma R^2 I_2 \). The \( \sigma Q \) parameter in the process noise covariance \( Q \) is related to the target dynamics and an appropriate value may be selected based on a generic estimation of the expected target mobility behavior. In general, \( \sigma Q \) should be set to a low value for a target that is not expected to accelerate or decelerate rapidly and is moving at almost constant speed (e.g., an army vehicle), while a higher value should be preferred for a target that changes its mobility sharply (e.g., a missile). On the other hand, \( \sigma R \) can be set equal to the mean positioning error attained by dSNAP based on offline tracking tests.

Below is the Kalman filter recursive algorithm that is used in order to estimate the source position \( x_k(t) \). The predicted state vector and error covariance matrix, without correction based on the observations \( y_k(t) \), are denoted as \( \hat{x}_k(t) \) and \( P_k(t) \), respectively while the equivalent filtered quantities are denoted as \( \tilde{x}_k(t) \) and \( \tilde{P}_k(t) \).

\[
\hat{x}_k(t) = \Phi \hat{x}_k(t-1)
\]
\[
\tilde{P}_k(t) = \Phi \tilde{P}_k(t-1) \Phi^T + \Gamma Q \Gamma^T
\]
\[
K_k(t) = \tilde{P}_k(t) \Delta \left( \Delta \tilde{P}_k(t) \Delta^T + R \right)^{-1}
\]
\[
\tilde{x}_k(t) = \hat{x}_k(t) + K_k(t) (y_k(t) - \Delta \hat{x}_k(t))
\]
\[
\tilde{P}_k(t) = \left( I - K_k(t) \Delta \right) \tilde{P}_k(t)
\]

where \( \Phi, \Gamma \) and \( \Delta \) are given in (6) and (7), while \( Q \) and \( R \) are the covariance matrices of the noise in the process \( q(t) \) and the noise in the position estimates \( r(t) \) respectively. The initial estimated value of the state vector \( \hat{x}_k(0) \) is determined as \( \hat{x}_k(0) = [x^r_k(0) \ y^r_k(0) \ 0 \ 0]^T \) where, \([x^r_k(0) \ y^r_k(0)]^T \) is the first estimated position using dSNAP, while the initial velocity is assumed zero. Also, the error covariance matrix is initialized at

\[
\tilde{P}_k(0) = \begin{bmatrix} R & 0 \\ 0 & \nu^2 I_2 \end{bmatrix}
\]

where \( \nu \) is the upper bound for the variance of the initial velocity.

At every step, the Kalman Filter produces a prediction of the next state of the target (i.e., the position and velocity). If the predicted position is closer to a neighbor \( n \) of the current leader \( l \), then the leader sends a message to \( n \) with the current state of the filter so that \( n \) will be the new leader to continue the track. At this point, the new leader will also broadcast a LEADER message in order to suppress all of its neighbors from becoming leaders.

**Corollary 2:** In order to guarantee that a set of consecutive leaders can continuously track a moving target, the distance between any two consecutive leader nodes \( d(l_i, l_{i+1}) \) should be less than \( 2R_c \).

_Proof:_ If the two leaders \( l_1(t) \) and \( l_2(t) \) are located at a distance greater than \( 2R_c \), their ROCs do not overlap. When the source is about to exit from the ROC of \( l_1 \), \( l_1 \) informs \( l_2 \) that it will become the new leader. However, at this point, it is also possible that the source may change direction and as a result will never enter the ROC of \( l_2 \). As a result \( l_2 \) will lose track of the source, as shown in Fig. 6.

![Fig. 6. Target tracking scenario where next elected leader loses target.](image)

In situations where the distance between two consecutive leaders is more than \( 2R_c \), or when the next source position is “wrongly” computed, then the next leader might be elected outside of the ROI of the source and as a result the tracking is falsely abandoned (e.g., when the majority of the nodes inside the leader neighborhood become non-alarmed). At this point a new track will be initiated by a newly elected leader following the D-FTLEP algorithm of Section IV. Another situation where a track is falsely terminated is when the paths of two sources cross each other and their separation distance is \( d \leq 2R_c \). By Lemma 4, the localization algorithm “sees” only a single source and consequently only a single target can be tracked, thus the other is abandoned. When the two sources move sufficiently apart, the D-FTLEP can once again detect the source that has been abandoned and a new track is re-initiated.

**VII. SIMULATION RESULTS**

For all subsequent experiments we use a square 100 \times 100 sensor field with \( N = 625 \) randomly deployed nodes unless
otherwise stated in the experiment, where the sensor readings are given by:

\[
z_n(t) = \min\left\{ 5000, \sum_{k=1}^{K} \frac{5000}{r_{n,k}(t)} + w_n(t) \right\}
\]

for \( n = 1, \ldots, N \). In the above expression, \( K \) is the total number of sources and \( r_{n,k} \) is the Euclidean distance of sensor node \( n \) from source \( k \) given in (3). Furthermore, we assume \( w_n(t) \) to be white Gaussian noise \( \mathcal{N}(0,1) \). For the distributed localization algorithm DSNAP, we use grid resolution \( g = 1 \) and ROC radius \( R_c = 10 \). Finally, the mean values reported are the average over 100 Monte-Carlo simulations (each with a randomly deployed sensor field). All experiments were performed using Matlab.

### A. Two Sources Identification

In the first set of experiments we evaluate the performance of the identification algorithm (D-FTLEP) described in Section IV using a simulation scenario with two sources located in the middle of the sensor field as we increase the separation distance between them \( d \) from 5 to 50m. The identification algorithm should correctly estimate the number of sources in the field by electing a leader for each source present. We are particularly interested in exploiting the limits of the proposed algorithm as the two sources move closer together - i.e., the source ROI becomes connected - and as the number of faulty sensor nodes in the field is increased. The performance of the D-FTLEP algorithm is compared against another “naive” leader election algorithm referred to as D-NLEP which allows any alarmed sensor node to become a leader irrespective of the value of \( F_n \), whereas for D-FTLEP only alarmed sensor nodes with \( F_n > 0 \) are eligible for becoming leaders. Other than this D-FTLEP and D-NLEP follow exactly the same protocol for estimating the number of targets in the field.

1) Effect of the distance between the sources: In Fig. 7 we evaluate the performance of the identification algorithms in terms of the mean number of leaders elected and the average distance of the elected leaders from the actual source positions as we vary the separation distance \( d \) between the 2 sources. The error bars on the plots indicate the standard deviation. For the simulations we used 100 experiments and 10 repetitions for each value of \( d \) producing 1000 results for each tested scenario. From these plots it becomes evident that D-FTLEP has very good performance and on average it correctly counts the leaders even in situations where the two sources are located almost next to each other \( (d = 5) \). The reason lies in the distributed implementation. Even though the closely spaced sources produce a single connected ROI, the proposed algorithm can still separate the two sources because the superposition of the signals from the two sources becomes strong enough so it can be detected outside the ROS of the leader sensor, thus a second leader is also correctly elected.

2) Effect of the ROS: In the simulations so far we have been using \( R_s = 1.5R_c \). Recall that \( R_c \) is the radius of ROS which defines the neighborhood where a sensor node can subscribe for information for calculating the \( F_n \) value according to the D-FTLEP. Moreover, ROS is the neighborhood where a leader node would send the suppression message in order to avoid the election of other leaders. Since, ROS plays such an important role for the algorithm, in the following set of experiments we investigate its performance as we vary the ROS size (through the radius \( R_s \)) for the 2 sources identification scenario. The ROC size is kept fixed at \( R_c = 10 \).

Fig. 7. Performance evaluation of identification algorithms as we vary the separation distance \( d \) between the two sources. Plot (a) shows the average number of elected leaders corresponding to the number of sources identified. Plot (b) shows the mean distance of the elected leaders from the actual source locations.

![Figure 7](image_url)

Fig. 8. Performance evaluation of the D-FTLEP algorithm as we vary the ROS size (through the radius \( R_s \)) for the 2 sources identification scenario. The ROC size is kept fixed at \( R_c = 10 \).

![Figure 8](image_url)
tested. Based on these results, for all subsequent experiments we continue to use $R_s = 1.5R_c$.

B. Multiple Sources Identification

In this section, we consider the more general case where we have $K$ randomly deployed sources in the field and evaluate the performance of the two identification algorithms (D-FTLEP and D-NLEP) by varying the number of faults and the node density in the field.

1) Effect of sensor faults: In the first set of experiments, we investigate the performance of the identification algorithms for $K = 2, 3$ and 5 sources as we increase the fault probability from 0 to 0.4. Fig. 9 shows the results for the leader count and the leader distance from the actual source locations. From Fig. 9(a), it becomes evident that in the no fault case ($p_f = 0$) the D-FTLEP algorithm is able to accurately count the number of sources for the cases $K = 2, 3$ while for $K = 5$ the leader count is 5.8. This result is expected, because by increasing the number of randomly deployed sources in the field we also increase the probability that 2 or more sources are located very close to each other. As shown in the 2 source simulation scenario, this situation can result in additional leaders becoming elected. As we increase the probability of faults (see Fig. 9(a)), D-FTLEP is able to retain its accuracy for all different numbers of sources tested by electing the same number of leaders as in the fault-free case. In fact, the fault tolerance of D-FTLEP is evident even when as many as 20% of the sensor nodes give erroneous observations. At the same time, the D-NLEP algorithm (dotted lines) fails immediately when faults are introduced and ends up electing a large number of leaders for all different numbers of sources tested. Note that the number of elected leaders using the D-NLEP algorithm exceeds 15 even for $K = 2$ sources and $p_f = 0.05$. The increased number of faults in the system causes the elected leaders for D-FTLEP to be on average further away from the true source locations (see Fig. 9(b)), while for the D-NLEP algorithm the leaders are always elected far (more than 6m) from the actual source locations even in the absence of faults.

2) Effect of sensor density: In the second set of experiments, we investigate the performance of the identification algorithms using $K = 5$ sources as we increase the number of sensor nodes in the field from 100 to 1000. Fig. 10 shows the leader count and leader distance results for different values of the fault probability $p_f$ from 0 to 0.3. Fig. 10(a) shows that the performance of D-FTLEP is robust with respect to the number of sensor nodes in the field. Fig. 10(b) shows that for the D-FTLEP algorithm, reducing the number of sensor nodes in the field
results in the elected leaders to be on average further away from the true source locations while the D-NLEP algorithm elects the leaders far (more than 7m) from the actual source locations independent of the underlying node density.

C. Tracking Performance for Multiple Targets with Non-overlapping Trajectories

![Fig. 11. Simulation scenario with 3 targets with parallel tracks and non-connected ROI. Target 1 enters the field at $t = 1$ at location (1,80) and leaves the field at $t = 60$ at location (60,80). Target 2 enters the field at $t = 20$ at location (20,50) and leaves the field at $t = 80$ at location (80,50). Target 3 enters the field at $t = 40$ at location (40,20) and leaves the field at $t = 100$ at location (100,20).](image)

In this section we evaluate the performance of the proposed distributed target tracking algorithm for multiple sources with non-overlapping trajectories using a scenario with three sources that traverse the field in a horizontal direction as shown in Fig. 11. The actual target trajectories, together with the filtered estimates and the elected leaders are also depicted in the figure. For this scenario, sensors that are "correctly" alarmed - i.e., based on the signal received from the target and not because of noise or a fault - can be uniquely associated with one of the three sources. The purpose of these simulations is twofold: First, we want to demonstrate as a proof of concept that the distributed target tracking algorithm proposed in this paper actually works. For this we need to show that the sources are successfully tracked at all times. Since the three sources enter and leave the field at different times as shown in Fig. 11, successful tracking implies dynamically creating and terminating continuous tracks for the 3 sources only, avoiding the creation of spurious tracks or multiple tracks for the same source. Second, we want to test the fault tolerance of the proposed algorithm. Specifically we are interested in demonstrating its ability to handle a large number of faulty sensor nodes in the field and the importance of using a fault tolerant localization algorithm such as dSNAP.

Fig. 12 shows the simulation results as we vary the fault probability from 0-0.4 using four different localization approaches as part of our tracking algorithm: 1. The proposed approach from Section V referred to as dSNAP, 2. The Centroid Estimator (CE) 3. The Closest Point Approach (CPA) and 4. The Geometric Estimator (GE). For the CE, the source is estimated as the centroid of the positions of the alarmed sensor nodes in the leader’s neighborhood, while in the CPA the estimated source location corresponds to the leader position. The GE is a geometric based approach that relies on the overlap of the sensing areas of the sensors depending on their alarm status for localizing the targets (see [22],[23] for more details). For the implementation of GE we employ the non-ideal 2-disc sensing model with parameters (0.5, 1.5). This specific set of parameters was selected out of the ones suggested in [23] because it resulted in the best possible performance for GE for the type of scenarios we are considering. The sources in our simulations are considered to be successfully tracked if there exists at least one track within a distance $R_c$ from the source’s actual position. For the percentage of time that the sources are successfully tracked, Fig. 12(a) shows the mean positioning error of the estimated to the actual track position. When multiple tracks exist within $R_c$, the closest one is chosen for the calculations. Fig. 12(b) shows the average percentage of time that the sources were not tracked. Finally, Fig. 12(c) demonstrates the average number of valid tracks created at the end of the simulation scenario. In order to correctly assess the performance of the proposed tracking solution one has to consider the combined results of the three aforementioned plots. When we have no faulty sensors ($f_l = 0$) the proposed tracking solution correctly creates 3 tracks for the 3 simulated target trajectories. Using dSNAP we are able to track the targets 99% of the time with a mean positioning error of $1m$, while all other localization algorithms achieve inferior performance. As we increase the number of faulty sensor nodes in the field, the proposed tracking algorithm is able to retain its performance, even when 25% of the sensor nodes become faulty ($f_l = 0.25$). At this point, the number of valid tracks is still 3 (with a larger standard deviation though) and using dSNAP we are still able to track the targets 99% of the time with a mean positioning error of about $2m$.

1) Effect of the noise variance: In the next set of experiments we investigate the robustness of the proposed algorithm as we vary the noise variance in the field ($\sigma_w^2$) from 1-10,000. Fig. 13 shows the simulation results for fault probability $f_l = 0.25$. From the plots, it becomes evident that the proposed algorithm is quite robust to the effects of noise. Even when 25% of the sensors in the field are already faulty, it can still accommodate a noise variance of $\sigma_w^2 = 100$ without any significant performance degradation compared to the noise-free condition ($\sigma_w^2 = 1$). In fact, according to our simulations, in the absence of other faults ($f_l = 0$) the maximum noise variance that the algorithm can tolerate increases to $\sigma_w^2 = 1000$. Note that all other parameters for these experiments are kept fixed at their default values, so we are using an ROC radius $R_c = 10$ and a common detection threshold of $\gamma = 50$ for all sensor nodes in the field. In other words, increasing the noise variance, increases the probability of false alarms which can be computed based on the test statistic and the fusion rule used for detection (see [27]). For example, assuming the simple threshold detection scheme of this paper, the probability of false alarms for each sensor node is given by $p_{FA} = Q \left( \frac{\mu}{\sigma^2} \right) = 0.3$ where $Q(x) = \frac{1}{\sigma^2} \int^\infty_{-\infty} \exp (-\frac{1}{2}y^2)dy$ is the right-tail probability of a Gaussian random variable $\mathcal{N}(0,1)$. This corresponds to a false alarm probability of $p_{FA} = 0.3$ for $\sigma_w^2 = 10000$ which demonstrates the robustness of the
proposed tracking solution to the effects of noise.

2) Effect of the sensor density: Next, we evaluate the performance of our tracking solution as we vary the number of sensor nodes in the field from 100-1000. The simulation results are portrayed in Fig. 14 for \( p_f = 0.25 \). From the plots we observe that out of the four localization solutions, dSNAP achieves the best results for all different numbers of sensors tested. It is interesting to note that when using dSNAP, increasing the number of sensors can improve the performance of the tracking solution in the presence of faults, while the same cannot be stated for GE. The reason is that geometric-based solutions (GE) begin to fail when the density of the faulty sensors in the field exceeds a certain threshold. This result is irrespective of the density of healthy sensors.

D. Tracking Performance for Multiple Targets with Crossing Trajectories

In this section we investigate the performance of the proposed tracking algorithm when we have multiple crossing targets. For our simulation setup, we assume that all \( K \) targets enter simultaneously from the left side of the field (spread out in equally spaced positions), choose a random heading so as to traverse the full length of the field in a straight line and depart from the right side of the field. The chosen scenario is meant to be a “worst case” scenario in the sense that all targets have a strong probability of crossing their trajectories in the middle of the field. This creates the most unfavorable conditions for testing the performance of our algorithm. We should emphasize at this point that one can easily create simulation scenarios with multiple crossing targets where only two targets cross at any given time.

Fig. 15 and Fig. 16 show the simulation results for three and five targets respectively as we vary the fault probability from 0-0.4. We compare the same four localization approaches as part of our tracking solution, as in Section VII-C. From the figures, it becomes evident, that dSNAP not only outperforms the other localization approaches in all tested categories but more importantly is able to retain its performance, even when 25% of the sensor nodes become faulty. More specifically, from Fig. 15 we see that for \( p_f \leq 0.25 \) dSNAP is able to successfully track the three targets more than 95% of the time with about 3m positioning error creating on average 5 tracks, while all other localization solutions achieve inferior performance. The benefit of using dSNAP is even more evident for five targets as displayed in Fig. 16. When multiple targets are close together, the ROI of the targets becomes connected for achieving more accurate results is a subject we
Fig. 14. Performance evaluation of the tracking algorithms for the 3 source simulation scenario as we vary the number of sensors \((N)\) for \(p_f = 0.25\). Plot (a) shows the mean positioning error between the estimated and actual tracks. Plot (b) shows the % of time that the targets were not tracked. Plot (c) shows the total number of valid tracks created.

Fig. 15. Performance evaluation of the tracking algorithms for 3 sources with crossing trajectories simulation scenario as we vary the probability of faults \((p_f)\). Plot (a) shows the mean positioning error between the estimated and actual tracks. Plot (b) shows the % of time that the targets were not tracked. Plot (c) shows the total number of valid tracks created.

are currently investigating.

VIII. CONCLUSIONS

In real world applications of WSNs, sensors often fail and report erroneous observations for various reasons, thus compromising the trust of people towards WSNs technologies. This work is targeted towards providing fault tolerant algorithms and solutions to this important problem. In this paper, we investigate the use of a sensor network for detecting, identifying and tracking multiple moving sources using binary data. We present FTLT, a fault tolerant localization and tracking algorithm which is a low complexity, distributed method suitable for real-time applications in WSNs. We verify the efficiency of our tracking method through simulations, even when a large percentage (25%) of the nodes report erroneous observations due to various reasons, such as random sensor faults.

In our future work, we plan to investigate methods that can correctly identify closely spaced targets (e.g., by exploiting the leader election protocol). This is expected to improve our tracking accuracy, especially for the case of targets with crossing trajectories. Furthermore, we would like to test the performance of the proposed tracking solution with different event propagation models, target mobility models and noise distributions.

REFERENCES

Fig. 16. Performance evaluation of the tracking algorithms for 5 sources with crossing trajectories simulation scenario as we vary the probability of faults ($p_f$). Plot (a) shows the mean positioning error between the estimated and actual tracks. Plot (b) shows the % of time that the targets were not tracked. Plot (c) shows the total number of valid tracks created.


where $Q$ is the complementary distribution function of the standard Gaussian distribution. Also, $s_n(t)$ is the measurement of sensor node $n$ at time instant $t$. The joint likelihood function is given by:

$$P(I_n, t) = \prod_{n=1}^{N} \prod_{t=1}^{T} \left[ Q \left( \frac{T - s_n(t)}{\sigma_w} \right) \right]^{I_{n,t}} \times \left[ 1 - Q \left( \frac{T - s_n(t)}{\sigma_w} \right) \right]^{1 - I_{n,t}}$$

or the selected sensors, it is expected that most will be alarmed, thus we propose the following “arbitrary” probability assignment for their indicator function $I_{k,t}$:

Pr$[I_{k,t} = 1 | \theta] = Q \left( \frac{T - s_k(\theta)}{\sigma_w} \right) = 0.99$

Pr$[I_{k,t} = 0 | \theta] = 1 - Q \left( \frac{T - s_k(\theta)}{\sigma_w} \right) = 0.01$

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

$$\log p'(I_K | \theta) = \sum_{k=1}^{K} \sum_{t=1}^{T} I_{k,t} \times \log(9.9) + (1 - I_{k,t}) \times \log(0.1)$$

$$\approx \sum_{k=1}^{K} \sum_{t=1}^{T} I_{k,t} \times (+1) + (1 - I_{k,t}) \times (-1)$$

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

$$\hat{\theta}_{SNAP} = \max_{\theta} \log p'(I_K | \theta)$$

This proves that the SNAP estimator is in fact a maximum likelihood estimator.