TIBFIT: Trust Index Based Fault Tolerance for Arbitrary Data Faults in Sensor Networks

Mark Krasniewski, Padma Varadharajan, Bryan Rabeler, Saurabh Bagchi
Dependable Computing Systems Lab
School of Electrical and Computer Engineering,
Purdue University
Email:{mkrasnie,pvaradha,brabeler,sbagchi}@purdue.edu

Y.Charlie Hu
Distributed Systems & Networking Lab
School of Electrical and Computer Engineering,
Purdue University
Email: ychu@purdue.edu

Abstract

Since sensor data gathering is the primary functionality of sensor networks, it is important to provide a fault tolerant method for reasoning about sensed events in the face of arbitrary failures of nodes sending in the event reports. In this paper, we propose a protocol called TIBFIT to diagnose and mask arbitrary node failures in an event-driven wireless sensor network. In our system model, sensor nodes are organized into clusters with rotating cluster heads. The nodes, including the cluster head, can fail in an arbitrary manner generating missed event reports, false reports, or wrong location reports. Correct nodes are also allowed to make occasional natural errors. Each node is assigned a trust index to indicate its track record in reporting past events correctly. The cluster head analyzes the event reports using the trust index and makes event decisions. TIBFIT is analyzed and simulated using the network simulator ns-2 and its coverage evaluated with a varying number and varying intelligence of the malicious nodes. We show that once TIBFIT gathers enough system state, accurate event detection is possible even if more than 50% of the network nodes are compromised.

***Keywords**: Sensor networks, secure and intrusion tolerant systems, trust index, arbitrary data faults, event aggregation.*

1 Introduction

Recent innovations made in the fields of electronics and wireless communication have enabled the advent of sensor networks. These networks comprising of thousands of inexpensive sensor nodes can be set up with relative ease by placing the nodes in predefined locations manually or through the use of robots, as well as by random deployment of self-organizing nodes. A wide gamut of applications ranging from health, home, environmental to military and defense make use of sensor nodes for collection of appropriate data. The sensor nodes comprising of data collecting, processing, and transmitting units are very small in size and can be densely deployed owing to their low cost.

Sensor nodes have serious limitations in available resources, such as power, memory, and processing ability[2]. The sensor nodes and wireless links are prone to failure, while the network is also open to various malicious attacks. While significant research has been done in the areas of communication architecture, routing, and energy conservation in sensor networks, development of fault tolerance in this highly volatile scenario remains an interesting open research issue. Conventional fault tolerance and intrusion tolerance protocols do not translate well to the sensor network domain due to its large scale and the resource constraints on the sensor nodes.

In this paper we consider fault tolerance in an event driven model for sensing. An event driven model of behavior for sensing finds many applications in civilian, military as well as industrial scenarios. Examples could be seismic monitoring to detect and locate tremors in a given area, or military applications to sense any movement within a cordoned-off area. The inherent unreliability of sensor nodes makes fault tolerance in such an environment an important concern. The problem is essentially one of aggregating data from multiple sensor nodes to decide if an event has occurred and determining the location of the event, in the face of natural and malicious failures in both the sensing nodes as well as the aggregating nodes. In particular, our approach looks at arbitrary faults in the sensor networks, whether natural or malicious. Natural arbitrary faults may arise suddenly and intermittently in sensor networks, thereby causing a node to miss reporting an event (missed alarms) or falsely reporting an event that has not occurred (false alarms). Malicious faults occur when some nodes in the network have been compromised by an adversary. This adversary can make the nodes send out corrupt information intended to adversely affect the data gathering role of the network. These malicious nodes, depending on their level of intelligence, may have some knowledge of how the network functions and can to behave in a manner to escape detection.

The goal of the proposed TIBFIT protocol involves event detection and location determination in the presence of faulty sensor nodes, coupled with diagnosis and isolation of faulty or malicious nodes. The accuracy of the system is defined in terms of fraction of instances when an event occurrence is correctly detected, and its location determined within the given error bound.

The approach followed by the protocol is to maintain state of the sensing nodes in terms of the fidelity of their previous sensing actions, and use this information in making decisions involving those sensing nodes. Sensor nodes report the occurrence and location of events to a data sink, and remain silent otherwise. The data sink then decides on whether the event occurred and where based on the aggregated data. To determine the location of the event the data sink must aggregate all reports from nodes within the detection radius. The aggregation could be a simple voting scheme. However voting is a stateless approach and does not reflect on the past performances of the sensing nodes. TibFit introduces a new parameter called *trust index* for this purpose. The *Trust Index* (referred to as TI) of a node is a quantitative measure of the fidelity of previous event reports of that node as seen by the data sink. In a system comprised of sensing nodes, the data sink assigns and maintains a TI for each node in its domain, and does voting in a stateful manner. As the system runs over a longer time, more state is built up concerning the performance of the associated sensing nodes, and hence tolerance for faults also goes up accordingly. So while the simple voting approach falls apart when more than 50% of the nodes within detection range of the event are corrupted, TibFit can tolerate faults in a network with more than 50% of its nodes compromised *after* it has built up adequate state of the nodes.

To demonstrate the effectiveness of TibFit, we use an event-driven simulation with ns-2. All nodes are considered liable to fail, whether in a natural or a malicious manner. We group the nodes into four categories: a) non-faulty nodes that naturally fault some percentage of the time; b) faulty nodes that err randomly; c) malicious nodes working independently that err occasionally and attempt to subvert the system but also try to remain undetected; d) malicious nodes that collaborate and err occasionally and attempt to subvert the system but also try to remain undetected. We show through simulation that TibFit is capable of accurately detecting and determining locations of events even when more than 50% of the network is compromised. Finally we also simulate a system that has a gradually increasing number of malicious nodes and analyze the accuracy of the system.

The main contributions of this paper are the following:

1. TibFit tolerates nodes that fail both naturally and maliciously, and makes decisions on event occurrence as well as location**.** Under several scenarios, accurate event determination and localization can be done even with more than 50% of the network compromised. We also demonstrate diagnosis and limited recovery in the system.
2. No nodes are considered immune to failure, whether they are sensing nodes or the data sink.
3. We have come up with an adversary model with increasing levels of sophistication and demonstrate the effectiveness of the protocol in each case.
4. The protocol is generic and can be applied to any data sensing and aggregation application in sensor networks.

The rest of the paper is organized as follows. First, we discuss the parameters of our system model in Section 2, we discuss TibFit design in Section 3, the simulation implementation and results in Section 4, the analysis of TibFit in Section 5, related work in Section 6, and conclusions in Section 7.

2 System Model

All nodes in the network are identical and are arranged into disjoint clusters, each with a set of *cluster heads* (CHs), only one of which is active at any point in time. The CH serves as the data sink for its particular cluster. The nodes in a cluster are within one hop communication of the CH. The clusters themselves are formed randomly around the elected CHs. The CHs are rotated over time and CH election is based on energy-related parameters of the constituent nodes. In each cluster, the node that is chosen to be the CH knows the topology of the cluster. Nodes that are within the detection range of an event are called *event neighbors* for that event. This topology is illustrated in figure 1.

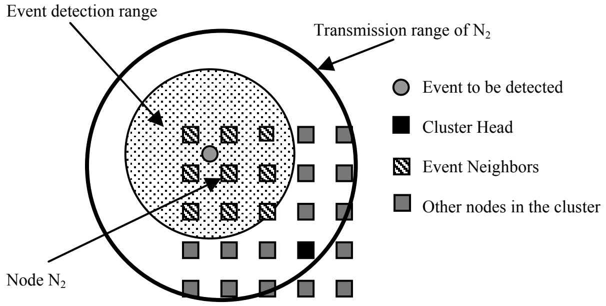

Figure 1: Event detection

When an event occurs, all the event neighbors are expected to report the occurrence of the event to the CH. The CH makes a decision on whether the event has occurred based on the reports received from the event neighbors and their *trust indices*. A detailed description of the TI model follows in Section 3.

The sensor network is deployed by placing the nodes randomly in the network. It is assumed that the nodes have the ability to determine their own locations. This can be accomplished through GPS mechanisms, deploying nodes with deterministic mobility in known locations and using triangulation methods to compute their positions as functions of time, etc. Further discussion is beyond the scope of this paper. The locations of the nodes at a given time are known to the CHs, but not necessarily to the non-CH nodes. The network could be stationary or mobile, as long as it is possible for the CH to estimate the positions of its cluster nodes during decision making. The sensor nodes function in an event-driven model, that is, they sense the environment for occurrence of a particular detection-level event and transmit data only if they sense such an event. We will assume that the event is typically detectable by multiple nodes, which makes our protocol practical. This assumption is not unreasonable for many practical sensor deployments.

We adopt the low energy, adaptive hierarchical clustering protocol (LEACH), for cluster formation as well as CH election [3],[4]. This protocol architecture aids in the formation of self-organizing clusters, with dynamically chosen CHs. Each node is assigned a probability of becoming a CH at the beginning of each round, which depends on the number of times it has been made CH previously and the energy available in the node. These properties help spread energy usage equally throughout the network. We have also incorporated the TI of the node as an additional parameter to be considered for CH election. The TI of the node has to be higher than a threshold value to ensure that only sufficiently trusted nodes can become CHs. This is not a property of the original LEACH protocol.

Each node independently decides if it wishes to be a CH. Once a node decides to become a CH, it broadcasts this information. Any node that receives advertisements from *n* different contending CHs, affiliates itself with a single CH based on the strength of the signal received. If a node's TI is below a certain threshold, the central base station will cancel this node's effort to become a CH and re-initiate CH election. A CH that reaches the end of its leadership period sends the aggregate TI information that it has gathered for all nodes in its cluster to the base station before ending its leadership. A newly CH elected for an existing cluster requests the base station for TI information for nodes in its cluster.

We group event detection into two categories – binary event detection and event detection with location determination. Binary event detection leads to the system recognizing the occurrence of the event with a binary decision about whether it happened or not and not being concerned with the location of the event. An example could be detection of a forest fire based on the temperature reaching a critical threshold. Location determination is when the coordinates of the event are also reported by the sensing node. In the forest fire example, the sensor can detect environmental changes such as wind and variation in light intensity in a direction and estimate the location of the oncoming fire.

2.1 Failure Model

The nodes in the network may fail due to accidental failures or may be compromised by an adversary and therefore exhibit failure due to malicious causes. Three types of failure scenarios are possible. A node may have a missed alarm where it does not report an event within its sensing radius to the data sink within a specified time. A node may provide a false alarm where it either reports an event outside of its sensing radius or reports an event within its sensing radius that did not occur. A node may exhibit a location faults where it reports an event but at the wrong location. Flooding based denial of service (DoS) attacks are not considered in this paper.

Four categories of sensing nodes are identified. *Correct nodes* are not assumed to be 100% accurate, but are expected to make errors within a specified bound referred to as *natural error rate. Faulty nodes* form the superset for nodes with natural or malicious failures. A faulty node can exhibit *naïve* behavior in terms of randomly sending out corrupt information following no specific pattern. The node lies arbitrarily, either in dropping an event report, falsely reporting an event, or reporting a faulty location (level 0). A *smart* faulty node is aware partially of the system model and tries to retain its TI at a reasonably high level where it estimates it will not be detected and isolated. If a malicious node's TI is reaching a level at which it will either be dropped from the network or its vote has too little influence on the event decision, then the node will stop lying until its TI is raised sufficiently. The *smart* faulty nodes may lie independently (level 1) or in collusion (level 2). The colluding nodes are assumed to be connected in a way that is undetectable by the reliable nodes in the network.

3 Basic Design

The goal of the TIBFIT protocol is to determine whether an event has occurred from analyzing reports from the event neighbors. To combat failures in the reporting nodes, each node is assigned a TI, maintained at the CH, to indicate its track record in reporting past events correctly. The TI is a real number between zero and one and is initially set to one. For each report a node makes that is deemed incorrect by the CH, the node's TI is decreased. Similarly, for each report a node makes that is deemed correct by the CH, the node's TI is increased, but not beyond one. Thus correctly functioning nodes will have a TI approaching one while faulty and malicious nodes will have a lower TI.

We assume that correct nodes are allowed to make occasional errors due to natural causes. The rate of these errors is denoted the natural error rate (NER). The TI is decremented exponentially. Nodes that make mistakes are penalized more for earlier mistakes, and find it more difficult to regain their previous trust levels. This is considered better than a linear model where a node that lies 50% of the time would still occasionally have the trust index value of one. If a node errs more frequently than its NER its index decreases, while if it errs less frequently then its index increases.

The TI is calculated as follows. Let the natural error rate be f_r (<1). A variable v is maintained for each node at the CH. Each time a node makes a report deemed faulty by the CH its v is incremented by the expression $1\text{-}f_r$. Each time a node makes a report deemed to be correct by the CH its v is decreased by f_r if v is larger than zero. The TI is calculated as

$$TI = e^{-\lambda v}$$

where λ is a proportionality constant that is application dependent. An uncompromised node's TI is expected to remain at the same value. It can be expected to suffer a fault at the rate of one per every $1/f_r$ events and the expected change in v is:

$$E[\Delta v] = (1 - f_r) - \left(\frac{1}{f_r} - 1\right) * f_r = 0$$

The design of the protocol is explained next by successively relaxing some simplifying assumptions.

3.1 Binary Events

Let us initially assume that event reports are binary in nature simply specifying whether the event has occurred or not. All the nodes in the cluster, say *k*, are event neighbors for any event detected by the cluster. A sensing node can detect the occurrence of an event perfectly for events that happen within a radius r_s surrounding the node. All the nodes within radius r_s of an event *E* are called event neighbors for *E*.

After the CH receives the first event report, it calculates the *k* event neighbors for the event. The CH then waits for a predefined interval of time T_{out} for event reports to be received from these nodes. After T_{out} has elapsed, the CH partitions the event neighbors into two sets *R* and *NR* based on whether they have reported the occurrence of the event or not, respectively. The trust indices of each group are summed and the group with the higher cumulative TI (CTI) wins out. The trust index values of nodes in the *winning* group are increased while the index values of nodes in the *losing* group are decreased according to the formula given above. It should be noted that a smaller group of reliable nodes can win the vote against a larger group of unreliable nodes based on higher TI for the individual reliable nodes earned over past events. This process provides detection, diagnosis, and masking of the fault.

It is evident that we do not need a TI model for a system with faulty nodes in the minority. A simple voting would suffice to mask the decision of the faulty nodes. However, consider a system where the density of faulty nodes increases over time. Examples could be batteries of the nodes dying out with time, or existing nodes being compromised by adversaries. The faulty nodes that have been in operation for a while would have had their TIs reduced to low values. Hence even when the total number of faulty nodes is in a majority, their CTI may still be lower than that of the correct nodes. Hence, TIBFIT can lead to correct aggregation as well as diagnosis even with more than 50% of the nodes compromised. It is obvious that if the initial condition consists of faulty nodes being in the majority, then the protocol will be unsuccessful in tolerating faults. After time, the system can identify a faulty node when its TI falls below a certain threshold. It can then be removed from the network.

3.2 Location Determination

In this section we build on the previous model by adding location details to the event reports. The event report consists of location in terms of (*r, Θ*) with respect to that node. The nodes do not sense the location of the event perfectly and the CH must determine the actual location of the event. One sensor network problem that can be solved through this extension is where a network is attempting to track a mobile sensor node that is transmitting a signal as it moves throughout the network.

Simplifying Assumptions: Let us assume there is a time difference of at least T_{out} between any two events to avoid overlapping event neighbors. A correct event report sent in by a sensing node reports the location of an event to within a radius r_{error} surrounding the event.

Once time T_{out} has elapsed after the first event report, let there be *k* other reports that have come in from the nodes in the cluster during this time. The CH performs a clustering algorithm based on K-Means which groups these *k* event reports into a number of *event clusters* based on the locations indicated by the reports [14]. Each event cluster represents a possible location where the event could have occurred, as indicated by the reports. The clustering algorithm is a heuristic based on K-Means, so as to minimize the sum of squares error.

Goal of the algorithm presented below is to organize the event reports into disjoint event clusters of radius $r_{error.}$ Let C be the set of all event clusters consisting of elements {C_1, C_2…C_r}. Let {c_1, c_2…c_r} be the centers around which the event clusters {C_1, C_2…C_r} are formed. Let d(x, y) denote the distance between two points x and y. d(c_i, c_j) > r_{error} ∀ C_i, C_j ∈ C. C_k.*cg* (Center of gravity) denotes the average location indicated by all event reports in cluster C_k.

Event clusters are created using the following procedure.

(1) The clustering algorithm is started once T_{out} has elapsed after the first event report. The set of all event reports in this time T_{out} is referred to as E. The distances between each pair of event reports are computed and sorted in a 2D array.

(2) Let E_1 and E_2 be event reports from the set E with the greatest distance between them. Event clusters C_1 and C_2 are created with E_1 and E_2 as centers, and C_1, C_2 are added to C.

(3) Condition for any event report E_k to form a separate event cluster is that d(E_k, c_i) > r_{error} ∀C_i ∈C. The set E is iterated through and the set of all cluster centers are identified, so that the remaining event reports are at a distance of less than r_{error} from at least one element in C, i.e., the remaining event reports cannot form separate event clusters.

(4) Once the initial set of clusters in C are formed, remaining event reports in E are added to one of the clusters in C based on which cluster center it is nearest to. C_k.*cg* for the clusters are updated appropriately.

(5) If the centers of two or more clusters lie within r_{error} of each other the clustering algorithm is repeated by forming a new cluster center at the weighted average of these centers. The rounds are executed until no change in cluster constituency takes place in a new round.

The final elements in C represent the set of all events. C_k.*cg* represents the location of the event *k.* The event neighbors can be determined for the location determined and a determination of whether an event has occurred is made based on the trust indices of the associated nodes as in Section 3.1. This design successfully throws out event reports from nodes that make a localization error of more than r_{error}.

3.3 Concurrent Events

Additions: In this section we build on the previous model by assuming that multiple events can occur within T_{out} of each other (referred to as concurrent events from here on). We however assume that concurrent events cannot occur closer than a distance of r_{error}.

(1) When the CH receives the first event report *E1*, a symbolic circle of radius r_{error} is drawn around it. A new timer $E1.T_{out}$ is started for the associated event reports from other event neighbors to come in. All subsequent events that lie within r_{error} of *E1* reported within time T_{out} are added to the same circle.

(2) If any subsequent event report E_k received lies outside this circle, a new circle of radius r_{error} is formed with this event report E_k as its center. Associated $E_k.T_{out}$ is started.

(3) Once time $E_k.T_{out}$ has passed from the reception of event report E_k that is the center of a circle, all the event reports inside this circle are put into a group and the clustering algorithm described in the previous section is performed on them to determine the location of the event.

(4) However if one or more other circles overlap with this circle, then the CH must wait until time T_{out} has elapsed for all such overlapping circles. The clustering algorithm is performed on the union of all event reports in all the overlapping circles to determine the event clusters and thus how many events have actually occurred.

3.4 Unreliable Cluster Heads

Though the CHs are chosen based on high TI values, it is still possible for a selected CH to fail. To combat this problem we assign two additional shadow cluster heads (SCH) to each cluster such that the SCHs can monitor all input and output traffic associated with the selected CH. The SCHs themselves may be considered to be reliable as they are chosen based on the fact that they have the highest trust indices among nodes within one hop of the CH. The SCHs listen in to the communication going in and out of the CH and perform all the functions as the CH except transmitting the aggregated event reports to the base station. On perceiving a wrong conclusion being drawn at the CH based on the input data, the SCHs also send the result of their own computations to the base station. The base station, on receiving data from all CHs in the cluster, does a simple voting to arrive at the right conclusion. It also prompts CH election in that cluster to pick a new CH and reduces the TI of the previous faulty CH. Thus, only a single CH failure can be tolerated.

TIBFIT can also be extended to scenarios where the sensing nodes are more than one hop away from the data sink. The data sink still needs to know the location of the constituent node and reliable data dissemination primitive needs to be introduced to ensure that the data sent out by the sensing nodes reliably reach the data sink without alteration [15],[16].

4 Simulation

The TIBFIT protocol is simulated using the *network simulator – ns-2* [6]. A sensing radius of 20 units is considered. Events are generated at regular time intervals by the *event generator,* using a uniform random variable to generate X and Y coordinates uniformly distributed in the network. The event generator informs the event neighbors of the event and its location.

We run three different experiments. In experiment 1 we show the accuracy of the binary event model versus percentage of the network compromised by level 0 faulty nodes. In experiment 2 we show the accuracy of the location event model versus percentage of the network compromised by level 0, 1, and 2 faulty nodes. In experiment 3 we show the accuracy of the location event model versus time, where the percentage of the network compromised increases linearly over time.

For each simulation we use either the TIBFIT system that uses the trust index, or we use the baseline system, which uses majority voting to make event decisions. Experiments are run with faulty nodes belonging to only one level for a given experiment. Nodes are stationary in all experiments.

4.1 Experiment 1 – Binary Events

A cluster of ten nodes is formed, and all nodes are considered event neighbors for every randomized event. Level 0 faulty nodes are used for the fault model, generating both missed alarms and false alarms. The CH makes a decision regarding occurrence of the event based on the data forwarded to it from the sensing nodes.

Type of Event	Binary Event Model
Independent Variable	Percentage Faulty Nodes: varied from 40%-90%
Correct Nodes NER	0, 1, and 5%
Faulty Nodes NER	Level 0:Missed Alarm 50% False alarm 0,10, and 75%
Size of network	10 sensing nodes, 1 CH
Number of Event neighbors	10
Events per simulation	100
λ	0.1
Fault rate (f_r)	Same as NER

Table 1: Parameters for Experiment 1

For this experiment we started simulations with 40% of the network compromised. As Section 5 shows, even for the baseline system, the probability of failure with less than 40% of the network compromised is very small, and therefore not simulated.

The results in figure 2 include only missed alarms. The most noteworthy result from this experiment is that the network can have 70% of its nodes compromised and still maintain over 85% accuracy. This result is superior to the analytical results shown in figure 10 in Section 5.

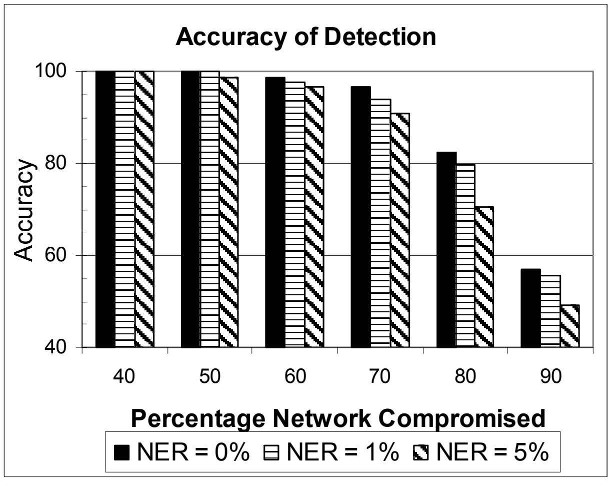

Figure 2: Experiment 1 – 50% accurate faulty Nodes, missed alarms only

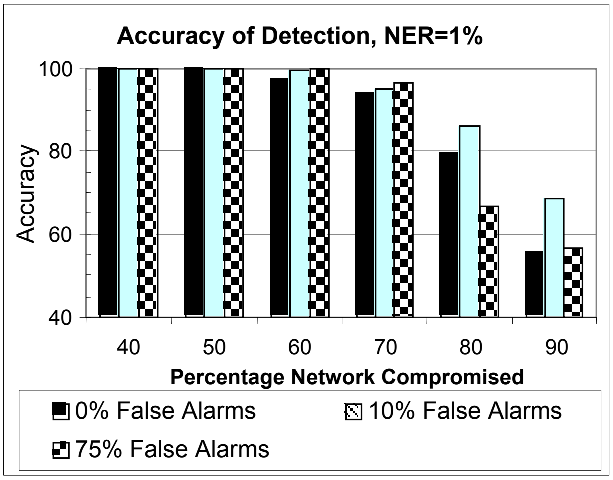

Figure 3: Experiment 1 – 50% accurate faulty nodes, missed alarms and false alarms

Figure 3 shows the simulation with both false alarms and missed alarms from faulty nodes. All correct nodes have 1% NER. Again, the network performance starts to degrade with 70% faulty nodes. The interesting results is that 75% false alarms shows the best accuracy when less than 80% of the network is compromised, indicating that the excessive false alarms lower faulty nodes' TIs and therefore increase system reliability. At 80% faulty nodes with 75% false alarms, accuracy falls dramatically, as the system is no longer able to tolerate the excessive false alarms. 10% false alarms maintains the highest accuracy at this point, indicating that occasional false alarms lower faulty nodes' trust indices enough to outperform 0% false alarms.

4.2 Experiment 2 – Location Determination Model

In the second type of simulation, 100 nodes are placed uniformly on a 100X100 grid. The CHs and event generator are two other entities present in the network. The CH decides on both the occurrence of the event as well as its location. The network is a single cluster, and the CH knows the positions of all 100 nodes. All nodes can reach the CH in a single hop. For location estimation r_{error} is 5 units. Table 2 shows various experimental parameters for this experiment. Due to the ns-2 wireless model, correct nodes' packets are naturally dropped less than 1% of the time.

A lower threshold (lowerTI) of 0.5 is used for level 1 and level 2 nodes to ensure their trust indices do not fall too low. If they reach the lower threshold they behave like a correct node until they reach an upper threshold (upperTI) of 0.8, after which they begin erring again. Each node reports an event with error in both the X and Y directions as dictated by a Gaussian random variable with standard deviation σ.

Type of Event	Location Determination Concurrent or single events
Independent variable	Percentage faulty nodes, varied from 10%-58%
Error rate for correct nodes	Location report has std. deviation of 1.6 or 2.0
Error rate for faulty nodes (levels 0, 1, and 2)	Location report has std. dev. of 4.25 or 6.0, drop packets 25% of the time
Size of network	100 sensing nodes, 5 CH
Number of event neighbors	Variable on location
λ	0.25
Fault rate (f_r)	0.1 (different from NER to compensate for wireless channel model losses)

Table 2: Parameters for Experiment 2

The error percentage indicated in Table 2 is calculated as the joint probability distribution of the two Gaussian rv's, which are Rayleigh distributed, and it indicates the probability a node reports an event more than 5 units away from the actual event location. The standard deviation for a correct node is much less than that for a faulty node. Level 1 nodes work independently, while level 2 nodes collude with each other and all either send the event report for the same location or do not send the event report.

This experiment initialized a network with a percentage of the network compromised by Level 0, 1, or 2 malicious nodes. 58% was the upper limit for the compromised network as past this point the system did not work with much accuracy. The output accuracy metric was the number of events detected by the CH within r_{error} of the actual event. Simulations are run with both concurrent and single events. The legend format for all the result figures from this point on is "*Lvl M W-Z [TibFit or Baseline]*", where *M* is the type of malicious node used, *W* is the standard deviation of the correct nodes, *Z* is the standard deviation of the malicious nodes, and the final parameter is whether the TibFit or the baseline model was used.

The results in figure 4 show that at low percentages of the network compromised, the TibFit system and the baseline system perform similarly. However, after 40% of the network is compromised, the TibFit model performs better than the baseline model by at least 7% percent, and by as much as 20% percent. More importantly, TibFit has accuracy near 80% even with faulty nodes having errors

70% of the time. A consequence of the execution of the network with TIBFIT is that the trust index values of the faulty nodes continue to decrease and once they reach the threshold, the nodes can be removed from the network, thus eliminating them from causing future damage.

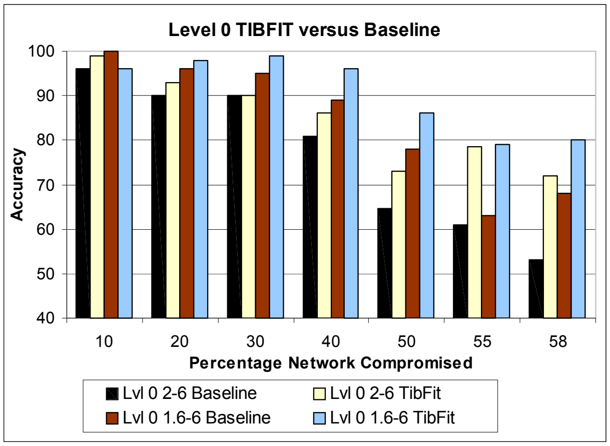

Figure 4: Experiment 2 – Level 0 faulty nodes

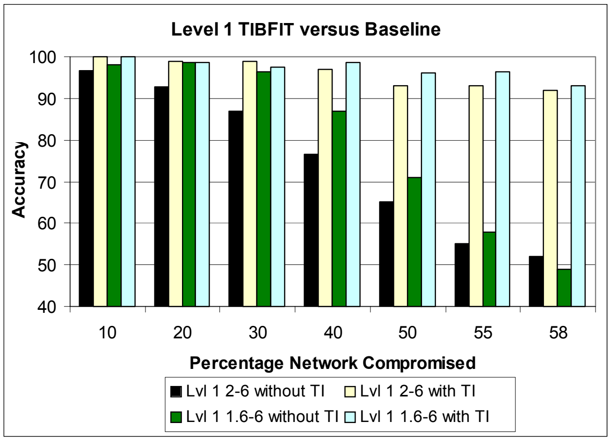

Figure 5: Experiment 2 – Level 1 faulty nodes

The second graph for location estimation, shown in figure 5, is for level 1 nodes. The result shows that even with 58% of the network compromised, TIBFIT's accuracy remains over 90%. In contrast, the baseline model falls well below that level once the network reaches 40% malicious nodes. The reason for this trend is that the level 1 nodes lie with intention to keep them from being detected. In effect, the trust index forces the malicious nodes to lie less frequently and therefore helps to improve the accuracy of the event determination.

Figure 6 shows results for level 2 malicious nodes. It shows that these nodes dramatically reduce the accuracy of the network, although the TIBFIT still outperforms the baseline model. It is clear from this figure that even the trust index has trouble tolerating level 2 type faults due to the collaborative nature of the nodes.

Figure 7 shows level 0 nodes with concurrent events compared to single events, both simulations using TIBFIT. The concurrent events occur with uniform distribution simultaneously, although never within r_{error} of each other. The graph indicates that tolerating concurrent events does not significantly alter the success of the nodes in accurate detection of events.

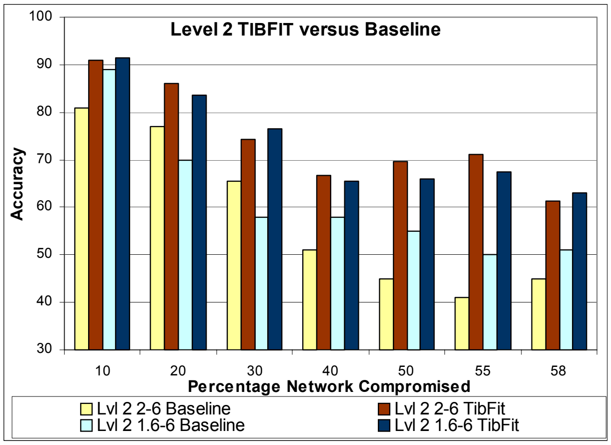

Figure 6: Experiment 2 – Level 2 faulty nodes

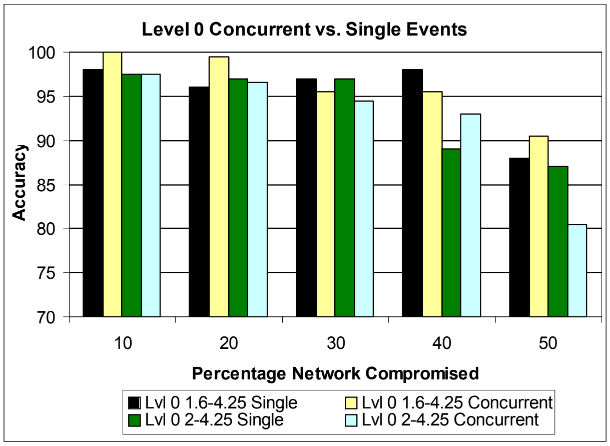

Figure 7. Experiment 2 – Single and Concurrent Events

4.3 Experiment 3 – Decay of Network

The next simulation increases the percentage of the network compromised by malicious nodes linearly over time. The network is initialized with 5% of the network compromised by level 0 faulty nodes. After every 50 events 5% more of the network is compromised until 75% of the network is compromised.

Figure 8 and figure 9 show that over time TIBFIT outperforms the baseline model in all cases. This occurs because the trust indices of the faulty nodes decrease over time and the system can then handle the transition of some correct nodes to faulty nodes. It is important to compare only the lines with the same standard deviation parameters, because for some time the baseline model with 1.6-4.25 outperforms the TIBFIT 2-4.25 case, although after a longer period of time the TIBFIT line does better, even though it has a higher fault rate in its correct nodes. What is also notable is that the TIBFIT network maintains nearly 80% accuracy even with 60% of the network compromised.

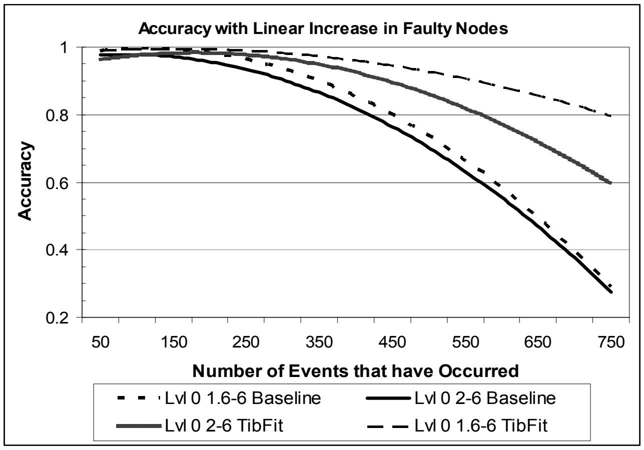

Figure 8: Experiment 3 – Linear increase in number of faulty nodes

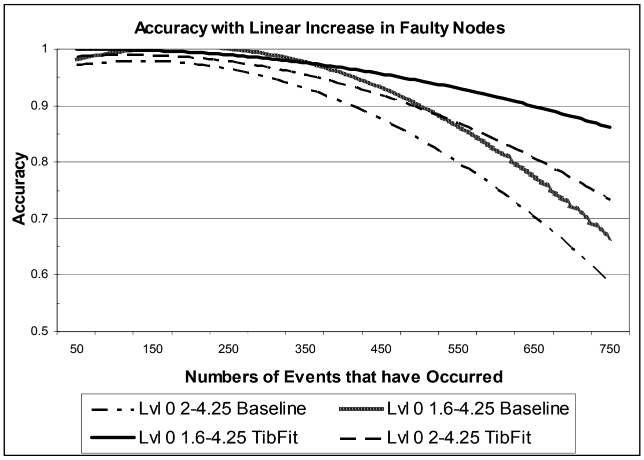

Figure 9: Experiment 3 – Linear increase in number of faulty nodes

5 Mathematical analysis

In this section we analyze the probability associated with the CH successfully identifying a binary event in the presence of faulty nodes.

Consider a baseline model with no trust indices assigned to the nodes. Let us assume that there are *N* event neighbors, of which *m* are faulty. The probability of a successful report from a correct node is *p*, and the probability of a successful report from a faulty node is *q*. Let **X** be the random variable that is the number of correct reports from correct nodes, and **Y** be the random variable indicating the same for the faulty nodes. They are defined:

$$P\{X = k\} = \begin{pmatrix} N-m \\ k \end{pmatrix} p^k \left(1-p\right)^{N-m-k}$$

$$P\{Y = k\} = \begin{pmatrix} m \\ k \end{pmatrix} q^k \left(1-q\right)^{m-k}$$

The probability that the *N-m* correct nodes make *k* or more correct reports is therefore the sum of the probabilities from *k* to *N-m*, and from *k* to *m* for faulty nodes. Define the random variable **Z**=**X**+**Y**. We wish to know the probability that **Z** has a majority of the *N* votes, which is the probability that the event is successfully identified. The expressions are shown in equations 1, 2, and 3. These expressions map to Figure 10 with *N*=10, *q*=0.5, and *p*=0.99, 0.95, 0.90, 0.85.

The accuracy begins to fall off steeply once fifty percent of the network is compromised. TIBFIT can tolerate both an increase in faulty nodes over time and more initial nodes being faulty, and will therefore outperform this baseline case. Next we will show how TIBFIT performs over time.

Consider the TIBFIT model. Assume the network initializes with *N* nodes with 1 faulty node and N-1 correct nodes. We will corrupt the nodes in the network at a constant rate of one after every *k* events and show how the system still functions with 100% accuracy till N-3 nodes are corrupted, thereby outperforming the baseline case which drops in accuracy once 50% of the nodes in the system are compromised. Without loss of generality, let us assume that *N* is odd. We also make the simplifying assumption that correct nodes are always correct and the faulty nodes always fail. Let $\text{CTI}_{\text{correct}}$ be the CTI of the set of correct nodes and $\text{CTI}_{\text{faulty}}$ be the CTI of the set of faulty nodes.

After every *k* events a good node is compromised. After *(N-2)*k* rounds, total number of correct nodes is 3, and faulty nodes is *N-3*. $\text{CTI}_{\text{correct}}$ is 3 as correct nodes are always correct and each has a TI of one. After the first faulty report, the TI of a node becomes $e^{(-\lambda).}$ Therefore after *k* rounds, the TI of the faulty node would be $e^{(-k\lambda)}$. So, $\text{CTI}_{\text{faulty}}$ for *(N–3)* faulty nodes when the newest addition to the faulty set has made k errors would be $e^{-k\lambda} + e^{-2k\lambda} + \ldots + e^{-(N-2)k\lambda}$.

$$P(success) = P\left(\overline{Z} \geq \left\lfloor \frac{N}{2} \right\rfloor + 1\right) = \sum_{j=1}^{\lceil N/2 \rceil} P\left(\overline{Z} = \left\lfloor \frac{N}{2} \right\rfloor + j\right), \text{now let } i = \left\lfloor \frac{N}{2} \right\rfloor + j - k \qquad (1)$$

$$P(success) = \sum_{j=1}^{\lceil N/2 \rceil} \left(\sum_{k=\left\lfloor \frac{N}{2} \right\rfloor + j - m}^{\min\left(\left\lfloor \frac{N}{2} \right\rfloor + j, N-m\right)} \left(\begin{pmatrix} N-m \\ k \end{pmatrix} * p^k \left(1-p\right)^{N-m-k} * \left(\begin{pmatrix} m \\ i \end{pmatrix} * q^i (1-q)^{m-i} \right) \right) \right) \quad m \leq N-m \ldots (2)$$

$$P(success) = \sum_{j=1}^{\lceil N/2 \rceil} \left(\sum_{k=\left\lfloor \frac{N}{2} \right\rfloor + j - (N-m)}^{\min\left(\left\lfloor \frac{N}{2} \right\rfloor + j, m\right)} \left(\begin{pmatrix} m \\ k \end{pmatrix} * q^k \left(1-q\right)^{m-k} * \left(\begin{pmatrix} N-m \\ i \end{pmatrix} * p^i (1-p)^{N-m-i} \right) \right) \right) \quad m > N-m \ldots (3)$$

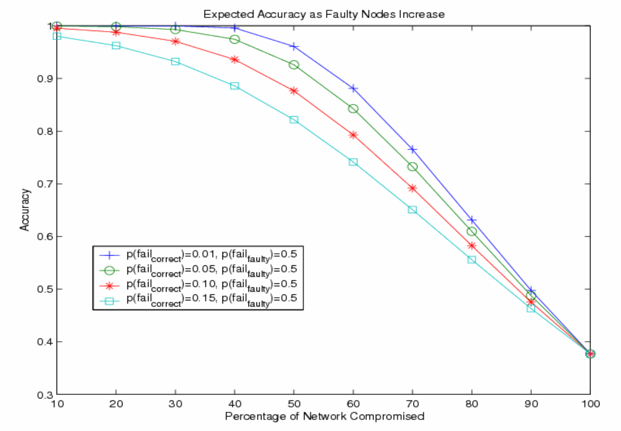

Figure 10. Expected accuracy of the network as the percentage of faulty nodes increases

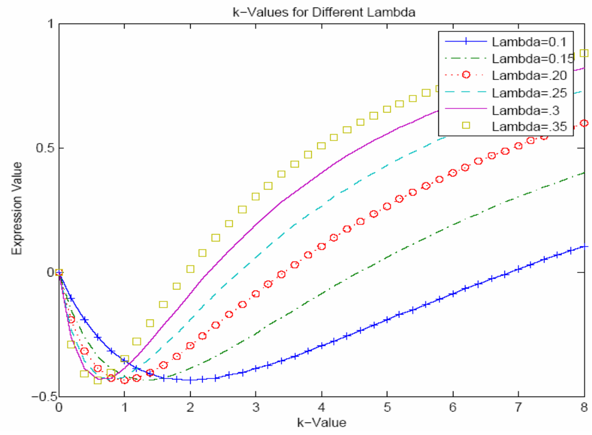

Figure 11. Variation of k with different λ values

For the system to be 100% accurate, CTI of correct nodes ($CTI_{correct}$) should always be greater than CTI of faulty nodes (CTI_{faulty}). For a correct node to be corrupted, CTI_{faulty} should be infinitesimally close to 1, so that $CTI_{correct}$ -1 > CTI_{faulty}+1 (a node is transferred from the good side to the bad side). We have the following expression:

$$3-1>1+e^{-k\lambda}+e^{-2k\lambda}+\ldots+e^{-(N-2)k\lambda}\quad,\quad \text{or}$$

$$2=\frac{1-e^{-(N-1)k\lambda}}{1-e^{-k\lambda}}\to 0=e^{-k\lambda(N-1)}-2e^{-k\lambda}+1,$$ which can be solved with Matlab.

Figure 11 shows this expression for several different λ values. Wherever a given line crosses the x-axis that is the value of k and the number of rounds after which a good node can be made into a faulty node. Expectedly as λ increases, the frequency of nodes failing that can be tolerated increases since the TI degrades more rapidly with failures. It is for this reason we chose λ=0.25 for our simulations, so that we could create a fair number of data points but without needing a very large number of events to show the beneficial effects of TIBFIT.

The upper limit on k is the k necessary to make three good nodes tolerate an additional failure. We stop the analysis at two because once the system has two good nodes left then the sum of the faulty nodes' trust indices must be less than zero to allow the addition of a bad node, which is impossible. When there are 3 good nodes left in the system, then 3 > CTI_{faulty}, where CTI_{faulty} = 3-ε, ε>0. After k_{max} rounds from this state, let us assume that one more correct node can be transferred to the faulty side. Therefore after k_{max} rounds the value of CTI_{faulty} should be = 1- ε before the transfer. Solving $3{*}e^{-k_{max}\lambda}$ =1-ε gives us $k_{\max}=\frac{1}{\lambda}\ln 3$ as $\varepsilon\to 0$. Hence, the maximum number of rounds needed to tolerate another faulty node is $\frac{1}{\lambda}\ln 3$.

6 Related Work

As in any sensor networks problems, we require a great deal of related material to ensure that our model accounts for the many challenges of creating a functioning wireless sensor network. For instance, [18] gives an algorithm that guarantees reliable and fairly accurate output from different types of sensors when at most k out of n sensors are faulty. [17] gives a fault tolerant way of averaging sensor data, and the author also gives a control process to deal with individual sensor failures. [19] deals with multi-sensor data fusion and assumes that the biggest loss in sensor network efficiency is from sensor readings. They propose a method of handling sensor failures through substitution of another on-board sensor. [20], [21], and [22] provide techniques of localization for finding node position, such as triangulation and lateration. Nodes within sensing range of this mobile node must be able to determine the location of this node. Location determination efforts with directional antennas can aid in finding the location of such a mobile node. In [13] it is shown that given signal strength and attenuation model one can estimate sensor location. Given enough fixed anchor nodes Bagchi *et al.* present a technique for finding an unknown node within some range of error [12].

There appears to be a dearth of existing work related to our specific topic of data fault tolerance in sensor networks. Schaeffer *et al.* discuss decision making concerned with propagating an alert through a network [7]. They set a threshold for event propagation, where if a node hears more than n nodes announce an alert then that node sounds the alert. They analyze the characteristics of this network with false alarms and missed alarms, where the evaluation is on whether the event notification reaches some data sink. They address natural faults exclusively and do not consider cases with faulty nodes colluding.

Wagner discusses aggregation of data in a sensor network with malicious intruders in [10]. The author presents a mathematical framework for analyzing the vulnerabilities of common aggregation functions and then presents the mathematical basis for secure aggregation functions, such as average with trimming. The work presented here can complement this by providing trimming of some failing nodes so that the aggregation can work on the remaining data set. However, the paper does not address the problem of in network aggregation, which is covered here through the analysis of failure prone CHs. It admits that the aggregation functions break down with more than half the network compromised. Also, the paper presents the case for aggregation with redundant deployments of cheap, crude sensor nodes.

Koo shows an upper bound on the tolerance of a broadcast decision process as approximately 1/π of the network being compromised [1]. This model is proven theoretically with arbitrarily powerful malicious nodes.

7 Conclusions

We present a protocol called TIBFIT that maintains state for event decisions in a sensor network. This protocol can handle both binary event detection and event location estimation with high accuracy in the face of natural and malicious node failures within the network. The protocol outperforms the standard voting scheme for event detection. We also define two types of intelligent malicious fault models that can disrupt a network, and find that using TIBFIT malicious nodes acting independently are successfully tolerated. However, the accuracy of TIBFIT in a system of colluding nodes is not as high though it outperforms the baseline voting scheme.

There still remains much work to be done with this protocol. We would like to further explore the impact of different system parameters on performance. We would also like to make TIBFIT more robust against level 2 malicious nodes. Another step would be to explore more types of intelligent models involving different levels of collusion and decision sharing amongst malicious nodes. We would also like to develop a more extensive theoretical model to demonstrate correctness and predict system reliability under given constraints. Ultimately, we would like to implement the protocol in our hardware testbed of Berkeley motes to measure the resource consumption.

8 References

[1] C-Y Koo. “Broadcast in Radio Networks Tolerating Byzantine Adversarial Behavior.” In PODC 2004.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. “A Survey on Sensor Networks.” IEEE Communications Magazine, pp. 102-114, Aug. 2002.

[3] W. Heinzelman, J. Kulik, and H. Balakrishnan. “Adaptive Protocols for Information Dissemination in Wireless Sensor Networks.” ACM Mobicom 99.

[4] W. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. “An Application-Specific Protocol Architecture for Wireless Microsensor Networks.” IEEE Transactions on Wireless Communications, Oct 2002.

[5] J. Lu, T. Suda. “Coverage-aware Self-scheduling in Sensor Networks.” IEEE Computer Communications Workshop, Oct 2003.

[6] http://www.isi.edu/nsnam/ns.

[7] S. E. Schaeffer, J. C. Clemens, P. Hamilton. “Decision Make in a Distributed Sensor Network.” In Proceedings of the Santa Fe Institute Complex Systems Summer School, Santa Fe, NM, USA, 2004. Santa Fe Institute. To appear.

[8] G. J. Pottie and W. J. Kaiser. “Wireless Integrated Network Sensors.” Communications of the ACM, vol. 43 no. 5, May 2000, pp. 51-58.

[9] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan. “Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks.” MobiCOM, 2001. pp. 272-287.

[10] D. Wagner. “Sensor networks: Resilient aggregation in sensor networks.” ACM Workshop on Security of ad hoc and sensor networks, 2004. pp. 78-87.

[11] Hoblos G., Staroswiecki M., Aitouche A. “Optimal Design of Fault Tolerant Sensor Networks” IIII Int’l Conf. Cont. Apps. Anchorage, AK, Sept 2000, pp. 467-72.

[12] S. Cabuk, N. Malhotra, L. Lin, S. Bagchi, and N. Shroff, “Analysis and evaluation of topological and application characteristics of unreliable mobile wireless ad-hoc network,” In Proceedings of the 10th Pacific Rim Dependable Computing Conference, March, 2004 (PRDC 04), March 2004.

[13] J. Hightower, R. Tower, and G. Borriello, “SpotON: An indoor 3d location sensing technology based on RF signal strength,” Technical Report of the University of Washington, Computer Science Department, February 2000.

[14] Sanjiv K. Bhatia, “Adaptive K-Means Clustering” In Proceedings of Florida Artificial Intelligence Research Symposium, 2004

[15] Gunjan Khanna, Saurabh Bagchi, and Yu-Sung Wu, "Fault Tolerant Energy Aware Data Dissemination Protocol in Sensor Network," In Proceedings of the IEEE Dependable Systems and Networks Conference (DSN 2004), pp. 739-748, June 28-July 1, 2004, Florence, Italy.

[16] "Design and Analysis of Hierarchical Key Management for Scalable and Energy Efficient Secure Communicationon Sensors," Issa Khalil, Ness Shroff, and Saurabh Bagchi. Submitted to AD HOC NETWORKS JOURNAL (ELSEVIER). Also available as CERIAS Tech Report TR-2003-33.

[17] Loren Schwiebert, Sandeep K.S. Gupta, “Research challenges in wireless networks of biomedical sensors” ACM Sigmobil 7/01 Rome Italy.

[18] K. Marzullo, “Tolerating failures of continuous valued sensors,” ACM Transactions on Computer Systems, vol. 8, no. 4, pp. 284-304, November 1990.

[19] F. Koushanfar, M. Potkonjak, A. Sangiovanni-Vincentell, “Fault tolerance techniques for ad-hoc sensor networks,” Proceedings of IEEE Sensors, vol. 2, pp. 1491-1496, June 2002.

[20] Jeffrey Hightower and Gaetano Borriello, “Location sensing techniques,” Technical Report of the University of Washington Computer Science Department, UW-CSE-01-07-01, July 2001.

[21] J. Hightower and G. Borriello, “Location systems for ubiquitous computing,” IEEE Computer, pages 57-66, August 2001.

[22] N. Bulusu, J. Heidemann, and D. Estrin, “GPS-less low cost outdoor localization for very small devices,” IEEE Personal Communications Magazine, pages 28-34, October 2000.