A

Detecting Faulty Nodes with Data Errors for Wireless Sensor Networks

Shuo Guo, University of Minnesota, Twin Cities
Heng Zhang, Zhejiang University
Ziguo Zhong, University of Nebraska, Lincoln
Jiming Chen, Zhejiang University
Qing Cao, University of Tennessee, Knoxville
Tian He, University of Minnesota, Twin Cities

Wireless Sensor Networks (WSN) promise researchers a powerful instrument for observing sizable phenomena with fine granularity over long periods. Since the accuracy of data is important to the whole system's performance, detecting nodes with faulty readings is an essential issue in network management. As a complementary solution to detecting nodes with functionnal faults, this paper proposes FIND, a novel method to detect nodes with data faults that neither assumes a particular sensing model nor requires costly event injections. After the nodes in a network detect a natural event, FIND ranks the nodes based on their sensing readings as well as their physical distances from the event. FIND works for systems where the measured signal attenuates with distance. A node is considered faulty if there is a significant mismatch between the sensor data rank and the distance rank Theoretically, we show that average ranking difference is a provable indicator of possible data faults. FIND is extensively evaluated in simulations and two test bed experiments with up to 25 MicaZ nodes. Evaluation shows that FIND has a less than 5% miss detection rate and false alarm rate in most noisy environments.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Operations

General Terms: Algorithms, Design, Management

Additional Key Words and Phrases: Wireless Sensor Networks, Data Fault Detection

ACM Reference Format:
Guo, S., Zhang, H., Zhong, Z., Chen, J., Cao, Q., He, T. 2013. Detecting Faulty Nodes with Data Errors for Wireless Sensor Networks. *ACM Trans. Sensor Netw.* V, N, Article A (January YYYY), 25 pages.
DOI : http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

1.1. Background

Wireless Sensor Networks (WSNs) have been widely used in many application fields such as habitat monitoring [Szewczyk et al. 2004], infrastructure protection [Xu et al. 2004], and scientific exploration [Tolle et al. 2005]. The accuracy of individual nodes' readings is crucial in these applications, e.g., in a surveillance network [He and et. al. 2006], the readings of sensor nodes must be accurate to avoid false alarms and missed detections. Although some applications are designed to be fault tolerant to some extent, removing nodes with faulty readings from a system with some redundancy or replacing them with good ones can still significantly improve the whole system's performance and at the same time prolong the lifetime of the network [Banerjee et al. 2012; Teymoori et al. 2012]. To conduct such after-deployment maintenance (e.g., remove and replace), it is essential to investigate methods for detecting faulty nodes.

Research was supported in part by NSF grant CNS 1117438, CNS 1117384 and 111 Program under grant B07031, NCET-11-0445.

DOI : http://dx.doi.org/10.1145/0000000.0000000

In general, wireless sensor nodes may experience two types of faults that would lead to the degradation of performance. One type is *function fault*, which typically results in the crash of individual nodes, packet loss, routing failure or network partition. This type of problem has been extensively studied and addressed by either distributed approaches through neighbor coordination [Marti et al. 2000] or centralized approaches through status updates [Ramanathan et al. 2005; Staddon et al. 2002]. The other type of error is *data fault*, in which a node behaves normally in all aspects except for its sensing results, leading to either significant biased or random errors [Elnahrawy and Nath 2003; Takruri and Challa 2007; Takruri et al. 2009]. These errors can result in data inaccuracy, and may seriously impact system performance and lead to faulty decision for critical tasks. [Elnahrawy and Nath 2003] illustrated the impact of these errors by bacteria growth monitoring experiment. More examples can be found in [Elnahrawy and Nath 2003; Takruri and Challa 2007; Takruri et al. 2009]. Although constant biased errors can be compensated for by after-deployment calibration methods, random and irregular biased errors can not be rectified by a simple calibration function.

One could argue that random and biased sensing errors can be addressed with outlier detection, a conventional technique for identifying readings that are statistically distant from the rest of the readings [Zhang et al. 2010]. However, the correctness of most outlier detection relies on the premise that data follow the same distribution. This holds true for readings such as temperatures, which are considered normally uniform over space. However, many other sensing readings (e.g., acoustic volume and thermal radiation) in sensor networks attenuate over distance, a property that invalidates the basis of existing outlier-based detection methods [Ding et al. 2005].

1.2. Related work

In general, faults in a sensor network can be classified into two types. One is *function fault* [Cao et al. 2008a; Yang et al. 2007a; Khan et al. 2008a; 2010; Liu et al. 2010], in which abnormal behaviors lead to the breakdown of a node or even a network as a whole. To improve our ability to observe abnormal behaviors at individual nodes, a rich set of tools have been proposed. Notable ones include Clairvoyant [Yang et al. 2007b], Declarative Tracepoints [Cao et al. 2008b], NodeMD [Krunic et al. 2007], Safe TinyOS [Cooprider et al. 2007], and JTAG [Atmel Corporation]. As a step further, methods to observe network-wide function faults also have been proposed. Marti et al. [Marti et al. 2000] suggest the use of behavior watchdogs so that a node can change its route when its neighbor fails. A similar distributive approach is proposed in [Hsin and Liu 2002], where malfunction nodes are detected through neighbor coordination. Having realized the constraints of distributed approaches with limited local information, researchers have recently begun to focus on centralized approaches, taking advantage of the comprehensive view of a network. Sympathy [Ramanathan et al. 2005] provides an effective sink-based method to localize node failures within a network based on a root-cause-analysis decision tree. SNMS [Tolle and Culler 2005] provides network infrastructure for the logging and retrieval of runtime states, which can be used for later diagnosis. Dustminer [Khan et al. 2008b] uses frequent discriminative pattern mining to reveal the causes of failure based on the events collected from a network. Tanachaiwiwat et al. [Tanachaiwiwat et al. 2003] propose that base stations can launch marked packets to sensor nodes,identifying malfunction nodes based on their response. Liu et al. [Liu et al. 2010] employ marking scheme for diagnosing the fault of wireless sensor networks and introduce a probabilistic inference model that encodes internal dependencies among different network elements for online diagnosis of an operational sensor network system. Miao et al. [Miao et al. 2011] study silent failures which are unknown beforehand and account for a large fraction of network performance degradation, and propose Agnostic Diagnosis(AD) approach to find silent failures in wireless sensor networks. In [Liu et al. 2011], a self-diagnosis approach is proposed for large scale wireless sensor networks. Sakib [Sakib 2012] proposes an asynchronous failed sensor node detection (AFSD) method which aims at minimising energy and control overheads, while detecting all the failed nodes.

The other type of fault is *data fault*, in which a node behaves as a normal node in the network but generates erroneous sensing readings. This kind of faulty node is difficult to identify by previous methods because all its behaviors are normal except the sensor readings it produces. This fault

Faulty readings would degrade the performance of the network significantly [Elnahrawy and Nath 2003; Takruri and Challa 2007; Takruri et al. 2009], so it is important to correct or remove them from the network. One way to solve this problem is after-deployment calibration such that a mapping function (mostly linear) [Feng et al. 2003; Balzano and Nowak 2007; Bychkovskiy et al. 2003; Miluzzo et al. 2008; Ramanathan et al. 2006; Whitehouse and Culler 2002] is developed to map faulty readings into correct ones. Although the parameters of the mapping function are obtained in different ways, additional assumptions such as sensing model [Feng et al. 2003; Whitehouse and Culler 2002], dense deployment [Bychkovskiy et al. 2003; Feng et al. 2003], similarity of readings among neighbors [Bychkovskiy et al. 2003; Miluzzo et al. 2008], and availability of ground truth result (highly accurate nodes) [Miluzzo et al. 2008; Ramanathan et al. 2006; Panda and Khilar 2011] are much needed. Thus, the performance of existing calibration methods not only highly depends on the correctness of the proposed model but also exhibits significant degradation in a real-world system in which too-specific additional assumptions no long hold. Outlier detection is a conventional method for identifying readings that depart from the norm. For example, Ding [Ding et al. 2005] proposes detecting faulty nodes by determining if the difference between a node's reading and its neighbors' is above a threshold. However, its correctness is based on the assumption that neighboring nodes have similar readings. For many phenomena of interest (e.g., thermal radiation and acoustic signals) in sensor networks, such an assumption does not hold, because these signals attenuate over space.

This paper proposes a faulty node detection method that can be generically and effectively applied as long as the sensor readings roughly reflect the corresponding distance. By removing or correcting faulty nodes detected by the proposed method, the performance of a networking system can be significantly improved.

1.3. Contributions

This paper proposes FIND, a novel sequence-based detection approach for discovering data faults in sensor networks, assuming no knowledge about the distribution of readings. In particular, we are interested in Byzantine data faults with either biased or random errors, since simpler fail-stop data faults have been addressed sufficiently by existing approaches, such as Sympathy [Ramanathan et al. 2005]. Without employing the assumptions of event or sensing models, detection is accomplished by identifying *ranking violations in node sequences*, a sequence obtained by ordering IDs of nodes according to their readings of a particular event.

The objective of FIND is to provide a blacklist containing all possible faulty nodes (with either biased or random error), in order of likelihood. With such a list, further recovery processes become possible, including (i) correcting faulty readings, (ii) replacing malfunctioning sensors with good ones, or (iii) simply removing faulty nodes from a network that has sufficient redundancy. As a result, the performance of the whole system is improved. Specifically, the main contribution of this paper can be summarized as follows:

— This is the first faulty node detection method that assumes no a priori knowledge about the underlying distribution of sensed events/phenomena. The faulty nodes are detected based on their violation of the distance monotonicity property in sensing, which is quantified by the metric of ranking differences.
— FIND imposes no extra cost in a network where readings are gathered as the output of the routine tasks of a network. The design can be generically used in applications with any format of physical sensing modality, such as heat/RF radiation and acoustic/sematic wave, as long as the magnitude of their readings roughly monotonically changes over the distance a signal travels.
— We theoretically demonstrate that the ranking difference of a node is a *provable* indicator of data faults and that if the ranking difference of a node exceeds a specified bound, it is a faulty node.
— We extend the basic design with three practical considerations. First, we propose a robust method to accommodate noisy environments where distance monotonicity properties do not hold well; second, we propose a data pre-processing technique to eliminate measurements from simultaneous

multiple events; and third, we reduce the computation complexity of the main design using *node subsequence*.

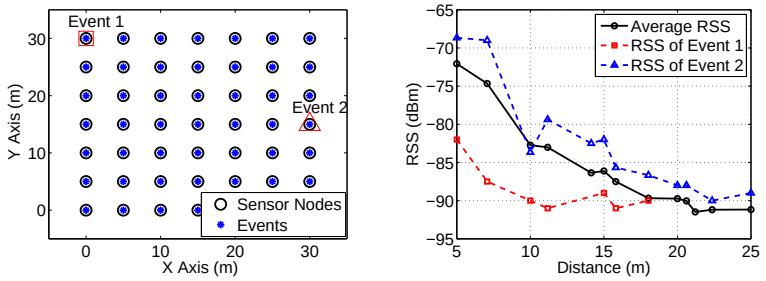
The rest of the paper is organized as follows. Section 2 introduces the model and assumptions of this paper. Then the detailed main design is presented in Section 3, and the subsequence detection is shown in Section 4. Section 5 provides additional techniques to deal with several practical issues. Sections 6 and 7 present the performance evaluation results from both test bed implementation and simulations. Section 8 concludes the paper.

2. MODEL AND ASSUMPTIONS

We consider a network model where N sensor nodes are deployed in an area of interest. Nodes are localized [Zhong and He 2009; Chang et al. 2008; Yedavalli and Krishnamachari 2008; Liu et al. 2006; He et al. 2012]and their positions are available at a base station. In this paper, we consider an event-driven model in which sensor nodes are roughly global synchronized to detect incoming events nearby and obtain corresponding sensing readings. Similar to recent centralized approaches for network fault detection [Ramanathan et al. 2005; Tolle and Culler 2005; Khan et al. 2008b; Panda and Khilar 2011], we assume the sensing readings are collected to the base station. If too many nodes in the network are faulty, the network will be paralysis and there is no need for 'fault detection'. Thus we also assume that the faulty nodes are minority. To be generic, we also introduce our design conceptually independent of the type of event used.

2.1. Assumption on Monotonicity

Many recent studies [Hwang et al. 2006; Zhou et al. 2004; Srinivasan et al. 2008; Liu et al. 2011] indicate that the environment is a dominating factor that affects the sensing and communication characteristics in sensor networks. It is therefore unrealistic to assume a particular mathematical model that describes the relationship between the sensing reading attenuation and the distance a signal travels. In this work, we take a much weaker assumption that the readings can generally reflect the *relative* distance from the nodes to the event. In other words, normally the sensing readings monotonically change as the distance becomes further. Although this assumption can be violated locally with environment noise, the general trend holds.

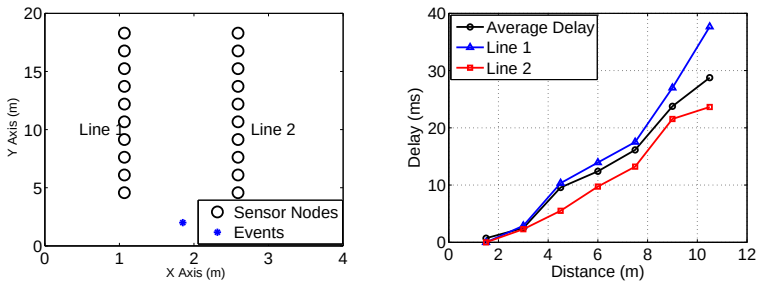


(a) Network Layout (b) RSS vs. Distance

Fig. 1. RSS of Radio Signals

To check the validity of our assumption, we conducted two outdoor experiments: one on radio signals and the other on acoustic signals. In the first experiment, 49 nodes are placed on a parking lot as shown in Fig.1(a). Each node generates an event by broadcasting a packet 100 times with 0dBm sending power, and all the other nodes recorded the received signal strength (RSS) for this event upon receiving this packet. In Fig.1(b), which plots the relationship between RSS and distance, the solid line shows the average RSS under different distances based on all event-node pairs. Each dashed line is the RSS of a single event (depicted by the triangle and circle symbols in Fig.1(a)) measured by different nodes. It can be seen from the figure that for a fixed distance, the RSS of different events have a large variance compared with the average RSS. For example, at distance 7m,

the RSS difference of the two events is nearly 20dBm, which causes a more than 10m ranging error: a big error given that the communication range is only 25m. As a result, even if a mathematical model can be derived to closely match the solid curve of average RSS, the behavior of individual events still has a large variance that makes this model ineffective. For each dashed line, on the other hand, the RSS decreases as the distance increases except for only a few points, which means the monotonicity of the RSS of a single event generally holds.



(a) Network Layout (b) Delay vs. Distance

Fig. 2. Propagation Time of Acoustic Signals

In the second experiment, 20 sensor nodes are placed in two lines and an acoustic signal is generated as shown in Fig.2(a). Fig.2(b) plots the delay of the nodes that receive the acoustic signal; to simplify the comparison, the delay is normalized such that the first node receives the signal with delay 0. Similar to experiment one, the delay of different lines for the same distance has a large variance, especially when distance increases, but the monotonicity still holds for both lines: a longer distance experiences a longer delay.

Based on the analysis above, we conclude that the monotonicity assumption is more accommodating to real world environments than the assumption based on a more specific model.

3. MAIN DESIGN

The main idea of FIND is that *it considers a node faulty if its readings violate the distance monotonicity significantly. The significance of violation is quantified by the metric of ranking difference between a detected sequence and a distance sequence, as will be defined later.*

(1) The first stage is **map division**, in which the map is divided into a number of subareas named *faces* based on the topology of the network as shown in Fig.3. Each face is uniquely identified by a *distance sequence*, which is denoted as a sequence of sensor node IDs (e.g., 2-1-3-4-5 in Fig.3). Within a distance sequence, the IDs are sorted in order of nodes' distances from an arbitrary point within this face. Map division can be pre-computed before detecting faulty nodes such that a number of distance sequences can be obtained and considered as the ground truth for the events taken place in corresponding subareas.

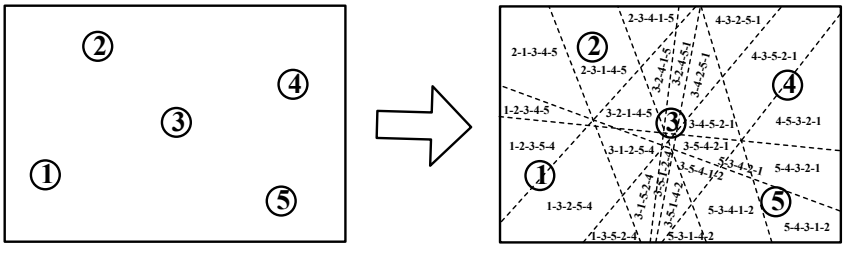


Fig. 3. First Stage: Map Division

(2) The second stage is **detection sequence mapping**. As shown in Fig.4, a number of events appear in the monitored region and are detected by the sensor nodes. For a single event, the sensing result of each node (e.g., the received signal strength or time-of-arrival) varies depending on how far the node is away from the event. A *detected sequence* is then obtained by ordering the sensing results of all the sensor nodes. Without knowing the location of the event, this detected sequence is mapped with one of the distance sequences corresponding to the face in which the event most likely takes place, yielding an *estimated sequence*. As shown in Fig.4, the same mapping process is repeated for all the events such that an estimated sequence is obtained for every detected sequence after this stage.

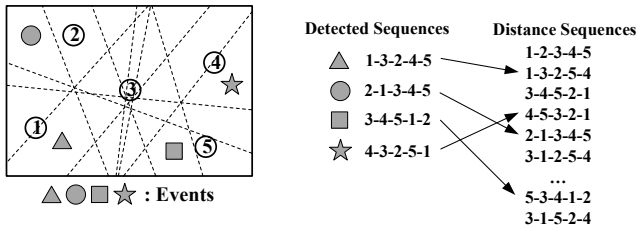


Fig. 4. Second Stage: Detection Sequence Mapping

(3) The third stage is **fault detection**. As shown in Fig.5., after a sufficient number of mappings becomes available, a blacklist is then obtained by analyzing the inconsistencies between the detected sequences and the estimated sequences.

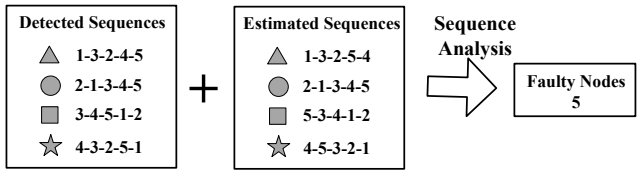


Fig. 5. Third Stage: Fault Detection

The detailed design of each stage is presented next.

3.1. Map Division

The goal of the first stage is to divide the map into a number of subareas, named *faces*, each of which can be identified by a node sequence indicating the distance between this face and all sensor nodes.

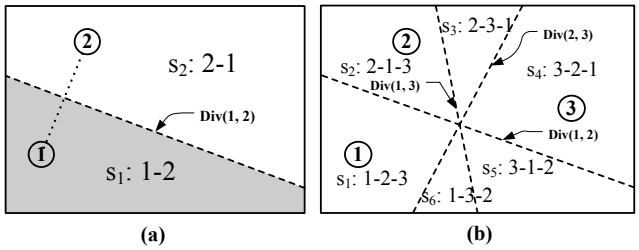


Fig. 6. Examples of Map Division

As shown in Fig.6(a), for two sensor nodes 1 and 2, the perpendicular bisector $Div(1,2)$ divides the area into two subareas. For any position point below $Div(1,2)$, node 1 is closer than node 2, so the distance sequence is 1-2. For any position point above $Div(1,2)$, node 2 is closer than node 1, so the distance sequence is 2-1. We call the subarea consisting of all position points with identical distance sequences a *face*.

Since crossing a perpendicular bisector reverses the order of two node IDs, each face is identified by the unique distance sequences. Also, according to the geometry study [de Bery et al. 2008], for a network of N sensor nodes, there are $C_N^2 = \frac{N(N-1)}{2}$ perpendicular bisectors that divide the graph into $O(N^4)$ faces. This is a much smaller number than the number of all possible node sequences $n!$. Due to the effect of faulty nodes and environment noise, a detected sequence with faulty readings would violate the physical geometry constraint that is imposed on distance sequences. As a result, a detected sequence can be arbitrary sequenced among $n!$ sequences. Thus it is essential to map the detected sequence reliably with one of the distance sequences, which is shown next.

3.2. Detection Sequence Mapping

To detect the violation of distance monotonicity, we need to first obtain the ground truth about the distance relationship. If the location of an event is known, such a ground truth can be trivially obtained by calculating the distances between the event and the nodes that detect it. Unfortunately, the locations of natural events are normally unavailable and it is necessary to obtain the ground truth using the sensing results. Formally, given the detected sequence of an event, the objective of the second stage is to estimate the face where an event takes place. This is essentially a mapping process between a detected sequence to the distance sequence that reflects the ground truth about the distances' relationship. With this ground truth, we can further quantify the severity of the violation in Section 3.3.

Let the N sensor nodes divide the map into M faces, identified by a set of distance sequences $V = \{s_1, s_2, \cdots, s_M\}$. Suppose $\bar{s}$ is a detected sequence from a single event and s is the distance sequence corresponding to the face where the event takes place. It is clear that without knowing where the event is, s is a random variable whose sample space is V. If $\{A_1, A_2, \cdots, A_M\}$ further denotes the size of the M faces, the prior distribution of s (i.e., the probability that an event takes places within the ith face) can be computed as the following:

$$Pr(s = s_i) = Pr(s_i) = \frac{A_i}{\sum_{j=1}^{M} A_j}, 1 \leq i \leq M \tag{1}$$

Eq.1 can be interpreted as: without any further information, the probability that an event appears within a face is in proportion to the size of that face. Then, s can be estimated by the method of Maximum A Posteriori (MAP) estimation as

$$\begin{aligned} \hat{s}_{MAP}(\bar{s}) &= arg\max_{s} Pr(s|\bar{s}) \\ &= arg\max_{s_i \in V} \frac{Pr(\bar{s}|s_i)Pr(s_i)}{\sum_{k=1}^{M} Pr(\bar{s}|s_k)Pr(s_k)} \\ &= arg\max_{s_i \in V} Pr(\bar{s}|s_i)Pr(s_i) \end{aligned} \tag{2}$$

where the second equality comes from Bayes' Theorem and the third equality holds because the denominator does not depend on s so that it plays no role in optimization.

Given $V = \{s_1, s_2, \cdots, s_M\}$, the new objective is to find a distance sequence $s_i \in V$ that maximizes Eq.2. For any $s_i \in V$, $Pr(s_i)$ is already given in Eq.1. The only unsolved problem is how to compute the conditional probability $Pr(\bar{s}|s_i)$, which is covered by the rest of this section.

3.2.1. Recursive Computation of $Pr(\bar{s}|s_i)$. Suppose α denotes the defective rate of the sensor nodes. Given N-node sequences $\bar{s}$ and s_i, we compare whether the nodes of the same ranking in $\bar{s}$

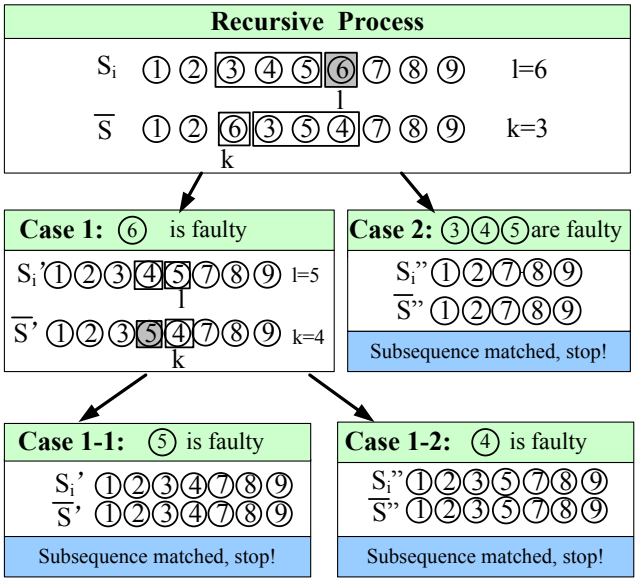


Fig. 7. The Conditional Probability Computation

and s_i are identical. Suppose $\bar{s}[k]$ and $s_i[k]$ denote the kth node in $\bar{s}$ and s_i respectively. Then, we compare if $\bar{s}[k] = s_i[k]$ holds for any $1 \leq k \leq N$.

When $\bar{s}[k] = s_i[k]$ holds for all $1 \leq k \leq N$, $\bar{s}$ and s_i are completely the same. In this case, there are two possibilities: all the nodes are normal or w faulty nodes ($1 \leq w \leq N$) still exist but "luckily" did not change the sequence for this particular case. Then for the sake of simplicity the conditional probability can be estimated as

$$\begin{aligned} Pr(\bar{s}|s_i = \bar{s}) &\approx (1-\alpha)^N + \sum_{w=1}^{N} C_N^w (\frac{\alpha}{N})^w (1-\alpha)^{(N-w)} \\ &= (1 - \alpha + \frac{1}{N}\alpha)^N \end{aligned} \tag{3}$$

where $(1-\alpha)^N$ is the probability that all nodes are normal. $C_N^w(\frac{\alpha}{N})^w(1-\alpha)^{(N-w)}$ is the probability that w nodes are faulty nodes. It is worth noting that there is a coefficient $\frac{1}{N}$ along with α. This is because a faulty node may shift to anywhere with approximate N possible outcomes of equal probability without additional information and the detected sequence $\bar{s}$ is just one of the N possible outcomes for any faulty node.[1] For example, faulty node 3 may cause the distance sequence 1-2-3 to become the detected sequences 3-1-2,1-3-2 or 1-2-3. Then for each of them the probability is $\frac{1}{3}\alpha(1-\alpha)^2$, given that node 3 is faulty.

When $\bar{s}$ and s_i are identical, the conditional probability $Pr(\bar{s}|s_i)$ can be computed by Eq.3. When $\bar{s}$ and s_i are not the same, a recursive method is used. The basic idea is to convert the two sequences into identical sequences whose conditional probability can be computed by Eq.3 after removing possible faulty nodes.

To illustrate this process, we use a running example shown in Fig.7 Suppose $\bar{s}[1..k-1] = s_i[1..k-1]$ and $\bar{s}[k] \neq s_i[k]$ are the first *unmatched* nodes. Also suppose the kth node in $\bar{s}$ is the lth node in s_i. As shown in Fig.7, k is 3 and l is 6. Then, there are two possibilities.

[1] Accurate probability is $\frac{\alpha}{N} \cdot \frac{\alpha}{N-1} \cdots \frac{\alpha}{N-w+1}$. Since we assume that the faulty nodes are minority, i.e., α is very small, this accurate probability can be approximated by $(\frac{\alpha}{N})^w$.

(1) **Case 1:** The kth node in $\bar{s}$ (which is also the lth node in s_i) is a faulty node, i.e., node 6 is a faulty node in the example. A pair of new sequences can be obtained by removing the faulty node (node 6) from the original sequences. In this example, s_i' is 1-2-3-4-5-7-8-9 and $\bar{s}'$ is 1-2-3-5-4-7-8-9. Given that node 6 is faulty, the original conditional probability $Pr(\bar{s}|s_i)$ depends on the new conditional probability $Pr(\bar{s}'|s_i')$.
Formally, define subsequence s_i' as the original distance sequence without the lth node $(s_i[1..(l-1)] + s_i[l+1..N])$. Also, define subsequence $\bar{s}'$ as the detected sequence without the kth node $(\bar{s}[1..k-1] + \bar{s}[k+1..N])$. Then, the conditional probability $Pr(\bar{s}|s_i)$ can be computed by the following equation:

$$Pr(\bar{s}|s_i, \bar{s}[k]\text{is faulty}) = \frac{1}{N} Pr(\bar{s}'|s_i') \tag{4}$$

where, the coefficient $\frac{1}{N}$ indicates that $\bar{s}$ is one of the N possible outcomes that node $\bar{s}[k]$ is faulty.

(2) **Case 2:** The kth node in $\bar{s}$ is a normal node. In this case, the nodes in s_i from k to $l-1$ must be faulty nodes. In the example in Fig.7, if node 6 is normal, nodes 3, 4 and 5 must be faulty nodes. Given $\bar{s}[k]$ is faulty, a new pair of sequences can be obtained by removing the nodes that are determined to be either faulty or normal from the original sequences. In the example, nodes 3, 4, 5, and 6 are all removed and the new sequences s_i'' and $\bar{s}''$ are both 1-2-7-8-9. Then, the original conditional probability $Pr(\bar{s}|s_i)$ depends on the new conditional probability $Pr(\bar{s''}|s_i'')$ as well as the probability that $s_i[k..l-1]$ are faulty nodes.
Define subsequence s_i'' as the original distance sequence without the nodes from k to l, $(s_i[1..k-1] + s_i[l+1..N])$. Also, define subsequence $\bar{s}''$ as the original detected sequences without the corresponding $l-k+1$ nodes. Similarly, the conditional probability $Pr(\bar{s}|s_i)$ can be computed by the following equation:

$$Pr(\bar{s}|s_i, \bar{s}[k]\text{is normal}) = (\frac{1}{N}\alpha)^{l-k} Pr(\bar{s}''|s_i'') \tag{5}$$

Based on the law of total probability, the final conditional probability is computed by the following equation:

$$\begin{aligned} Pr(\bar{s}|s_i) &= Pr(\bar{s}|s_i, \bar{s}[k]\text{is normal}) Pr(\bar{s}[k]\text{is normal}) \\ &\quad + Pr(\bar{s}|s_i, \bar{s}[k]\text{is faulty}) Pr(\bar{s}[k]\text{is faulty}) \\ &= (4) \cdot \alpha + (5) \cdot (1-\alpha) \end{aligned} \tag{6}$$

Eq.6 is recursive. The recursive process continues until the two sequences are identical. As shown in the example in Fig.7, the original sequences first generate two pairs of subsequences. The second pair has identical sequences $\bar{s}''$ and s_i'' whose conditional probability can be computed by Eq.3 and stops generating new subsequences. The first pair, however, does not have identical $\bar{s}'$ and s_i' so that it generates two pairs of new sequences. The process stops because both of the two new pairs have identical sequences and the conditional probability can be computed by Eq.3.

3.2.2. A Note on α Sensitivity. The MAP estimation uses the defective rate α as an input parameter. Although α can be estimated from the manufacturer's product specification sheet (e.g., Mean Time Between Failure (MTBF)) or the statistical testing result of samples, α would be affected by the in-situ factors such as severe weather and physical damage. Therefore it is important to investigate whether the correctness of MAP estimation is sensitive to the inaccuracy of the α value. When $\bar{s}$ and s_i are identical, from Eq.3 we can see the conditional probability $Pr(\bar{s}|s_i)$ is approximately $(1-\alpha)^N$ which is close to 1 when α is small and N is large. For each pair of unmatched nodes, there is at least one faulty node so that at least one $(1-\alpha)$ is replaced by $\frac{\alpha}{N}$ for both Case 1 and Case 2 according to Eq.4, Eq.5 and Eq.6. Since $\frac{\alpha}{N} << (1-\alpha)$, the distance sequence that has the least unmatched nodes with $\bar{s}$ is most likely to maximize $Pr(\bar{s}|s_i)$ and becomes the estimated sequence of the MAP estimation. In brief, an inaccurate α value does not affect the correctness of

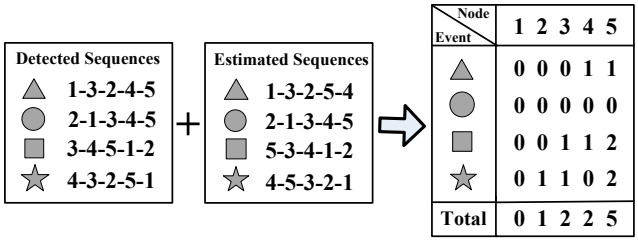


Fig. 8. A Simple Example of Ranking Difference

the mapping as long as $\frac{\alpha}{N} << (1-\alpha)$ holds. We note α is normally small (e.g., HMC1002 magnetic sensors used in the mica series have MTBF of 50000 hours [Honeywell 2007]), and thus the inequality $\frac{\alpha}{N} << (1-\alpha)$ actually holds very well in practice.

3.3. Detection Using Ranking Differences

Given the detected sequences from the measurement of a number of events and their corresponding estimated sequences, the objective of the third stage is to find a list of nodes in descending order of likelihood of being faulty nodes. We define *ranking difference* as the difference a node ranks in a detected sequence and its corresponding estimated sequence. In Section 3.3.1, we develop two theorems to state why we can use the *average ranking difference* as a provable indicator of possible data faults. In Section 3.3.2, we present how to develop a detection algorithm based on the theoretical analysis in Section 3.3.1 to finally find the blacklist.

3.3.1. Average Ranking Difference. Here we show that by using the average ranking difference, we can effectively identify faulty nodes. We mainly prove the following two statements:

(1) A node with a larger average ranking difference has a higher probability of being a faulty node.
(2) The majority of faulty nodes can be obtained by selecting nodes whose ranking differences are above a lower bound.

Suppose n is the number of detected sequences available at the base station. In Section 3.2, the MAP estimation method maps these detected sequences $\{\bar{s}_1, \bar{s}_2, \cdots, \bar{s}_n\}$ with the distance sequences in V as the estimations of where the events take place. Suppose the estimated sequences are denoted as $\{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_n\}$ ($\hat{s}_i$ is short for $\hat{s}_{iMAP}$). We call each $\bar{s}_i$ and the corresponding $\hat{s}_i$ a *sample*. n is also named the *sample size*. Again, without data faults, the detected sequence $\bar{s}_i$ and the corresponding estimated sequence $\hat{s}_i$ are identical for any sample. In this case, there is no ranking difference for any node since the rankings of any node in $\bar{s}_i$ and corresponding $\hat{s}_i$ are the same. When faulty nodes are present, these two sequences are no longer identical and the rankings of some nodes in the two sequences are no longer the same, leading to non-zero ranking differences.

We start with the simple example in Fig.8 to see how average ranking difference works, while formal definitions and theoretical proof will be shown later. In Fig.8, four detected sequences are mapped with their estimated sequences and the ranking differences can be computed by comparing the rankings of each node in the two sequences. Let us take the square event as an example. For the square event, the detected sequence is 3-4-5-1-2 and the estimated sequence is 5-3-4-1-2, where nodes 1 and 2 have the same rankings such that their ranking differences are 0. Node 3 and 4 both shift by one and their ranking differences are 1. Node 5 ranks the first in estimated sequence and the third in detected sequence, and thus its ranking difference is 2. The ranking differences of all the nodes in all events are shown in Fig.8. In this example, node 5 is a faulty node, with all its readings lower than expected. The figure offers a couple of insights. First, as a faulty node, node 5 has a non-zero ranking difference in most events due to its faulty reading. Second, node 5 also changes some normal nodes' rankings. However, *a faulty node at most changes normal nodes' rankings*

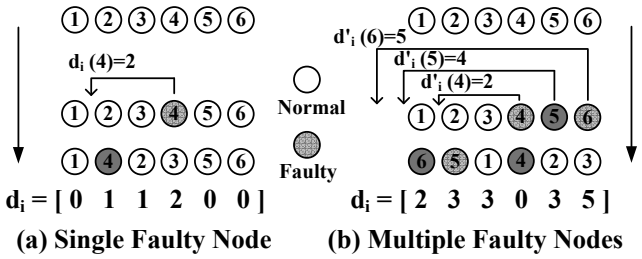


Fig. 9. Ranking Differences Affected By Faulty Nodes

by 1. Third, for different events, the sets of normal nodes whose rankings are changed by node 5 are different. From these insights, it is not difficult to explain that the total (or average) ranking difference of node 5 is the largest, as shown in Fig.8.

Formally, we define the ranking difference and average ranking difference as follows:

Definition 3.1. The **ranking difference** $d_i(k)$ for node k in sample i is given by the following equation

$$d_i(k) = |R(\hat{s}_i, k) - R(\bar{s}_i, k)|, 1 \leq k \leq N, \tag{7}$$

where $R(*, k)$ denotes the ranking of node k in sequence $*$. The **average ranking difference** of node k in the n samples (denoted as $D(k)$) is computed by averaging $d_i(k)$:

$$D(k) = \frac{1}{n} \sum_{i=1}^{n} d_i(k). \tag{8}$$

$D(k)$ is also known as the sample mean of the ranking differences. Based on these definitions the first theorem is developed as follows:

THEOREM 3.2. ***A node*** q ***is faulty if its average ranking difference*** $D(q)$ ***is greater than a bound*** B ***given by***

$$B = \frac{N_e}{N-1}(\mu_e + N_e - 1) \tag{9}$$

where N_e is the number of faulty nodes in an N-node network, μ_e is the arithmetic mean of the average ranking difference of faulty nodes, and $D(q)$ is calculated under a sufficient large sample size.

The detailed proof of Theorem 3.2 can be found in appendix A.1.

Theorem 3.2 provides a sufficient condition for node q being a faulty node (i.e., $D(q) \geq B$ implies q is a faulty node). At the same time, it also provides a necessary condition for node p being a normal node (i.e., if p is normal, $D(q) \leq B$ holds). It is worth noting that none of these conditions are both sufficient and necessary, which is a tricky part in the proof of the next theorem.

Based on Theorem 3.2, we develop Theorem 3.3 showing how the entries in D correlated with the probability that the corresponding nodes are faulty.

THEOREM 3.3. *Given the network in Theorem 3.2 and D as the ranking difference vector, suppose $Pr(k = \text{“}F\text{”})$ denotes the probability that node k is a faulty node. Then, for any node p and q satisfying $D(p) < D(q)$:*

$$Pr(p = \text{“}F\text{”}) \leq Pr(q = \text{“}F\text{”}) \tag{10}$$

The conclusion of this theorem is straightforward. We leave the detailed proof to appendix A.2.

ALGORITHM 1: Detection Algorithm

Input: Sorted node sequence n_1-$\cdots$-n_N, average ranking differences D
Output: The $Blacklist$ $\{n_1, \cdots, n_k\}$
1: $Blacklist \leftarrow \emptyset$
2: $\mu_e \leftarrow 0$,$B \leftarrow 0$,$N_e \leftarrow 1$
3: **for** $j \leftarrow 1$ to N **do**
4: **if** $D(n_j) \leq B$ **then**
5: Break
6: **else**
7: $\mu_e \leftarrow \frac{\mu_e N_e + D(n_j)}{N_e}$,$N_e \leftarrow N_e + 1$
8: $B \leftarrow \frac{N_e}{N-1}(\mu_e + N_e - 1)$
9: $Blacklist \leftarrow Blacklist + \{n_j\}$
10: **end if**
11: **end for**
12: Return $Blacklist$

From Theorem 3.2 and 3.3, it is clear that if we order all the nodes by their corresponding average ranking differences in D, we obtain a list of nodes in descending order of their probability of being faulty nodes.

3.3.2. Detection Algorithm. In the previous section we concluded that ordering the nodes by their average ranking differences yields a node sequence in order of the likelihood of their being faulty nodes. Then, if the number of faulty nodes N_e is known, the top N_e nodes on the list are the best estimation of faulty nodes according to Theorem 3.3. If the node sequence is denoted as n_1-n_2-$\cdots$-n_N, then $\{n_1, n_2, \cdots, n_{N_e}\}$ are best estimations of faulty nodes.

However, N_e *is normally unknown a priori*. In other words, we do not know how many faulty nodes there are in a network until we detect them. Therefore, we need to design a detection algorithm in this section to estimate what N_e is; i.e., given n_1-n_2-$\cdots$-n_N which consists of all nodes, we want to find a cutting point k such that $\{n_1, n_2, \cdots, n_k\}$ are the estimation of faulty nodes. The detection algorithm is designed based on Theorem 3.2 and 3.3.

The basic idea is to find all the nodes whose ranking differences are above B and consider them faulty. Specifically, we want to find a node n_k such that the ranking differences of all the nodes from n_{k+1} to n_N are no greater than B and all the nodes from n_1 to n_k are greater than B. Then, $\{n_1, n_2, \cdots, n_k\}$ forms the estimation of faulty nodes.

Formally, n_k is a node that satisfies $D(n_{k+1}) \leq B < D(n_k)$ where $B = \frac{N_e}{N-1}(\mu_e + N_e - 1)$, N_e is estimated by k and μ_e is estimated by $\frac{\sum_{i=1}^{k} D(n_i)}{k}$.

The pseudo-code of the detection algorithm is shown in Algorithm 1. This is a greedy selection algorithm. Initially the blacklist is empty (Line 1),μ_e and B are set to 0, and N_e is set to 1(Line 2). Starting from n_1 (which is the node with the largest average ranking difference) and check the nodes one by one (Line 3) until the ranking difference of a node is no greater than B (Line 4 and 5). Specifically, a node can be added into the blacklist if and only if its ranking difference is greater than B. After adding a new node into the blacklist, N_e, μ_e and B are updated (Line 7 to 9). Then an estimation of faulty nodes $\{n_1, \cdots, n_k\}$ can be obtained if this loop breaks at node n_{k+1}.

Example: Retrospecting to the example which has shown in Fig.4 and Fig.5 at the beginning of this section, we illustrate the running process of Algorithm 1. Initially, we sort the node sequence n_1-n_2-n_3-n_4-n_5 as 5-4-3-2-1 and compute the corresponding average difference $D = (\frac{5}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, 0)$. Then the blacklist is set to $\emptyset$, μ_e and B are set to 0 (Line 2), and N_e is set to 1. For $j = 1$, since $D(n_1) = \frac{5}{4} > B = 0$, we have $\mu_e = \frac{5}{4}$, $N_e = 2$, $B = \frac{9}{16}$, and $Blacklist = \{n_1\}$. For $j = 2$, we have $D(n_2) = \frac{1}{2} < B = \frac{9}{16}$. Then the 'for-end' process breaks and the final $Blacklist = \{n_1\}$ is obtained. Thus the fault node 5 has been detected.

Uniqueness: An important feature of this detection algorithm is its uniqueness. Given a node sequence and the corresponding average ranking differences, there is only one cutting point k. This is proved in Appendix A.3. Then, by using Algorithm 1, a unique blacklist consisting of all the nodes $\{n_1, n_2, \cdots, n_k\}$ whose ranking difference is above B can be constructed.

4. SUBSEQUENCE DETECTION

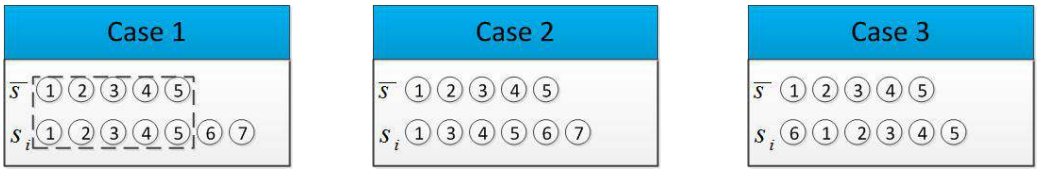
By computing conditional probability for every possible $s_i \in V$ using Eq.3, the MAP estimation can be finally calculated by Eq.2. It is worth noting that the complexity of computing $Pr(\bar{s}|s_i)$ is exponential ($O(2^N)$) to the length of the sequence N. However, in reality an event can not be detected by all sensor nodes in the network. The length of the detected sequence is no greater than the number of sensor nodes within sensing range, which is much smaller than N. Also, the length of detected sequence can be controlled below a certain bound L (e.g., 10) by selecting the top L nodes in the detected sequence so that the complexity is upper bounded by 2^L (1024 if $L = 10$). Therefore we should pay more attention on the issue of subsequence estimation to reduce the complexity.

4.1. Truncated sequence

When the network size N is large, a single event can be detected only by the sensor nodes within the sensing range. As a result, the length of the detected sequence $\bar{s}$ is less than N most of the time. On the other hand, even if a full detected sequence (of length N) is available, a truncated $\bar{s}$ is still needed due to the exponential complexity of Eq.6.

Suppose L is set as the upper bound of the length of detected sequences. Then any detected sequence $\bar{s}$ is either of a length no greater than L or is truncated by selecting only the first L nodes so that the complexity of computing the conditional probability is upper bounded by 2^L.

Given $\bar{s}$,the distance sequences s_i is also truncated accordingly. In our design, the distance sequences are truncated to length $2L$ (later we will see why $2L$ is a better choice than L).



(a) case 1 (b) case 2 (c) case 3

Fig. 10. Comparing subsequence $\bar{s}$ with s_i

4.2. Modified recursive process

The estimation for subsequences is very similar to the estimation for full sequences as presented in Section 3.2 where the objective is to find an $s_i \in V$ that maximizes Eq.2. Again, the conditional probability $Pr(\bar{s}|s_i)$ needs to be computed, the only difference being that $\bar{s}$ and s_i are no longer of the same length as they were in Section 3.2. Previously, the recursion in Eq.6 stops when current $\bar{s}$ and s_i are identical and their conditional probability can be computed by Eq.3. If they are not identical, the first unmatched nodes $\bar{s}[k] \neq s_i[k]$ are found and possible faulty nodes (either $\bar{s}[k]$ in case 1 or $s_i[k..l]$ in case 2) are removed from both sequences so that they are closer to identical. For subsequences, however, there are two new problems. First, $\bar{s}$ and s_i are not identical most of the time since initially they are of different lengths. Second, given the first unmatched node $\bar{s}[k] \neq s_i[k]$, there is no guarantee that there exists an $s_i[l]$ (as in full sequence case) such that $\bar{s}[k] = s_i[l]$ since s_i is also truncated. Based on these observations, we make the following changes for the recursive computation process for $Pr(\bar{s}|s_i)$:

(1) The condition for terminating the recursion has been relaxed to that s_i can be truncated to $\bar{s}$ after removing a number of nodes from the back, in contrast to the previous requirement that s_i

and $\bar{s}$ are completely identical. Then the corresponding conditional probability is computed by Eq.3 with N replaced by the length of $\bar{s}$. Examples are shown in Fig. 10. If $\bar{s}$ is 1-2-3-4-5 and s_i is 1-2-3-4-5-6-7, the recursion stops since s_i can be further truncated to 1-2-3-4-5 and the conditional probability can be computed by Eq.3, replacing N with 5. However, if $\bar{s}$ remains 1-2-3-4-5 but s_i is 1-3-4-5-6-7 or 6-1-2-3-4-5, the recursion does not stop.

(2) For each pair of unmatched nodes $\bar{s}[k] \neq s_i[k]$, if $\bar{s}[k]$ does not exist in s_i, it is considered faulty directly as in case 1, in contrast to the previous method where both cases are considered.
Suppose $\bar{s}[k] \neq s_i[k]$ are the first pair of unmatched nodes. k is no greater than L since the length of $\bar{s}$ is upper bounded by L. If the node $\bar{s}[k]$ exists in s_i as in the previous full sequence case, the computation process remains the same. If it does not exist in s_i, its ranking in the s_i before truncation (denoted as l) must be larger than the length of truncated s_i, which is $2L$. Then the ranking difference of this node is $l - k$ which is greater than L. Again for node $\bar{s}[k]$, there are two possibilities: faulty or normal. Consider the case when it is normal. Then, $\bar{s}[k]$ is a normal node with ranking difference of $l - k > L$, which indicates that there are at least L faulty nodes (since a faulty node can cause a normal node's ranking change by at most 1). This case can be ignored because the probability that there are more than L faulty nodes is small, especially when α is small. As a result, truncating s_i to length of $2L$ allows us to only consider the case $\bar{s}[k]$ is faulty when $\bar{s}[k]$ does not exist in s_i.

By making these two changes, the method presented in Section 3.2 can be easily applied to subsequence estimation.

5. PRACTICAL ISSUES

In this section we discuss and address some practical issues that compensate to the basic design of FIND. In Section 5.1 we develop a technique to reduce the effect of high level noise on detection algorithm and Section 5.2 discusses how to eliminate data measured from multiple events to preserve the good performance of FIND.

5.1. Detection in Noisy Environments

In Section 3.3.2, a detection algorithm is designed under the assumption of a low noise level: Given a node sequence n_1-n_2-$\cdots$-n_N in descending order of their ranking differences in D, a cutting point k is found so that $D(n_{k+1}) \leq B < D(n_k)$ and $\{n_1$-n_2-$\cdots$-$n_k\}$ forms an estimation of faulty nodes.

This algorithm works well when the environment has a low noise level, especially when the defective rate α is small. As discussed before, the smaller α is, the further B is away from μ_e and the less chance that a faulty node can be miss detected. However, one problem arises when applying this algorithm in an environment with a high noise level. In such an environment, a normal node may sometimes behave as a faulty node due to noise, resulting in a larger ranking difference than previously. Especially when α is small, B is small such that the ranking differences of some normal nodes may exceed B. To address this issue, we use a simple statistical method to filter out some possible normal nodes in estimation $\{n_1$-n_2-$\cdots$-$n_k\}$.

Given the estimation $\{n_1$-n_2-$\cdots$-$n_k\}$, all the rest nodes $\{n_{k+1}$-$\cdots$-$n_N\}$ are considered normal nodes. Then, the sample mean and the sample variance of the ranking difference of normal nodes can be computed as

$$\hat{\mu} = \frac{\sum_{i=k+1}^{N} D(n_i)}{N-k}, \quad \hat{\sigma}^2 = \frac{\sum_{i=k+1}^{N} (D(n_i) - \hat{\mu})^2}{N-k-1} \tag{11}$$

Then, the nodes whose ranking differences are not in the interval $(\hat{\mu} \pm 3\hat{\sigma})$ are removed from the blacklist.This statistical technology comes from 3σ *principle*.

5.2. Simultaneous Events Elimination

We present the main design assuming that a node detects one event at a time. However, since we cannot control natural events, it would be the case that multiple events are detected by sensor nodes at the same time. In this case, the resulting node sequences are no longer correlated to any distance sequence most of the time. For example, for radio signals, the RSS measured by a sensor node will be equal to the sum of the RSS of two single events; for acoustic signals, the delay measured by a sensor node will be equal to the delay of the event that arrives earlier. Including such sequences into FIND will bring detection error since ranking difference is no longer a good metric to identify faulty nodes. As a result, eliminating data from multiple events is very important to preserve high confidence in detection.

A pre-processing of data is designed where Longest Common Subsequences (LCS) is used to identify multiple events. (If two sequences are 1-2-3-4-5 and 4-1-3-5-2, the LCS is 1-3-5.) When noise is absent, the measured sequence $\bar{s}$ is just one of the distance sequences with αN faulty nodes shifts to left or right. No matter how they shift, the rest $(1-\alpha)N$ normal nodes still keep their original order thus the LCS between $\bar{s}$ and its corresponding distance sequence is at least $(1-\alpha)N$ long. By computing the LCS between $\bar{s}$ and all distance sequences we get the longest LCS of $\bar{s}$. Then, the node sequences measured from multiple events can be filtered out if the length of their longest LCS is shorter than $(1-\alpha)N$. In practice, this bound can be lower due to the presence of noise. We evaluate the pre-processing design in Section 7.

6. SYSTEM EVALUATION

We evaluate FIND using data from real world experiments. In the first experiment, we detect data faults for radio signal in RSS measurements while for the second experiment, we detect synchronization errors that causes inaccurate delay measurements for acoustic signals.

6.1. Performance Evaluation

We evaluate the performance of FIND in two scenarios. In the first scenario, an accurate α is assumed so that the first αN nodes with the largest ranking differences are selected as faulty nodes without using the detection algorithm in Algorithm 1. This scenario is used as the performance upper bound for testing the effectiveness of our ranking difference based design. In the second scenario, accurate α is unavailable and Algorithm 1 is used for selecting faulty nodes whose ranking differences are greater than B (α is still used for computing $Pr(\bar{s}|s)$ but it can be less accurate). Also, the statistical method presented in Section 5.1 is used for further refining the blacklist. To distinguish between these two scenarios, we name the first ideal scenario α-based detection (or α-detection for short), the second one B-based detection (or B-detection for short).

Two metrics are used for evaluating FIND: false negative rate and false positive rate. The former is defined as the proportion of faulty nodes that are reported as normal, which is also known as miss detection rate. The latter is defined as the proportion of normal nodes that are reported as faulty, which is also known as the false alarm rate.

6.2. On Radio Signal

We deploy 25 MicaZ nodes in grids (5×5) on a parking lot, as shown in Fig.11(a). The distance between each row and column is 5m. Broadcasting events, identified by their unique event ids, are generated one by one. In the experiment, the sending power of each event is adjusted to 0dBm such that the communication range is around 25m. Upon receiving a broadcasting packet, MicaZ nodes measure the RSS and record it together with event id. The number of events generated varies from 19 to 49. After recording the RSS of all the events, we randomly select 1 to 5 nodes and inject errors (either biased or random) into their readings, corresponding to a 4% to 20% defective rate. According to Fig.1(b), where the RSS from the events of the same distance may have a difference of as large as 20dBm, the injected errors are at least 20dBm above or below the normal readings. Fig.12 and Fig.13 plot the false negative rate and false positive rate for α- and B-detection given

(a) On Radio Signals (b) On Acoustic Signals

Fig. 11. Outdoor Experiments Using Two Types of Events: Radio and Acoustic Signals

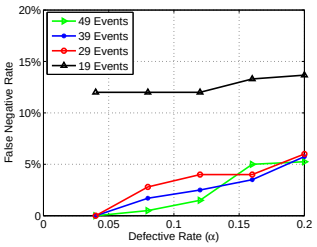
different number of events. It can be seen from the figure that except for 19 events, the false negative rate of both α- and B-detection are below 6%. For false positive rate, B-detection is below 5% for most cases, a little higher than that of α-detection, which is below 3%. Noting that α-detection always selects αN nodes into the blacklist, its false positive rate is closely correlated to its false negative rate. B-detection, however, selects nodes based solely on the detection algorithm. Its false positive rate curves are more irregular than those of α-detection. Since the network size is only 25, a 5% false positive rate means fewer than 1.3 normal nodes are inaccurately reported faulty on average, which is still a good performance. We also observe that the performance of 19 events has a significant degradation compared with the other three scenarios, especially for the false negative rate. This is because with 19 events, only 19 truncated detected sequences are available so that a faulty node may appear in only a small portion of them, leading to a low ranking difference. With the effect of noise, such a faulty node is difficult to be detected since it does not have a high enough ranking difference due to limited number of events. When the number of events becomes 29 or higher, the false negative rate decreases since the faulty nodes appear enough of times in detected sequences.

Fig.14(a) and (b) plot the examples of average ranking differences of all the 25 nodes with an 8% defective rate when 19 events and 49 events are given. In (a), the faulty nodes are nodes 12 and 23, while in (b), the faulty nodes are 12 and 24. It can be seen from the figures that these faulty nodes can be easily identified by their average ranking differences since in these two examples, the two faulty nodes have the highest ranking difference. Also, with more events, the detection becomes easier. The horizontal line in both figures show the average ranking difference of normal nodes. Comparing (a) and (b), we observe that when there are 19 events, the normal nodes have an average ranking difference of about 2.2, which is larger than that when there are 49 events, which is 1.8. This is because with more events, more detected sequences become available and more faulty nodes appear in these truncated sequences. Thus, it is easier for the detection algorithm to avoid the effect of noise.

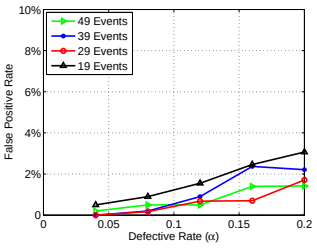
6.3. On Acoustic Signal

We deploy 20 MicaZ nodes in grids (4×5) as shown in Fig.11(b). The distance between each row and column are set to 1.5m. All nodes are equipped with microphones to receive 4KHz acoustic signals generated by a speaker and record corresponding timestamps. Before generating the acoustic signal, all the nodes are synchronized.

After recording the timestamps for 18 events, we inject errors into 1 to 4 nodes, corresponding to a 5% to 20% defective rate. The errors are at least 10ms, which correspond to a distance of about 3.5 meters. Fig.15(a) plots the false negative rate of α- and B-detection, and for α below 0.1, the false negative rate is less than 5%. For α greater than 0.1, the false negative rate is about 10% higher than that of the first experiment using 29 or more events, but is close to that using 19 events. In Fig.15(b), B-detection has a false positive rate of around 8%. Again, for this small-scale network with 20 nodes, this indicates that about 1.6 normal nodes are inaccurately considered faulty on average, which is tolerable. α-detection still has a low false positive rate as expected.

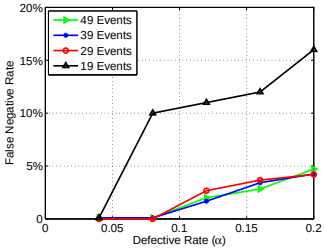


(a) False Negative Rate

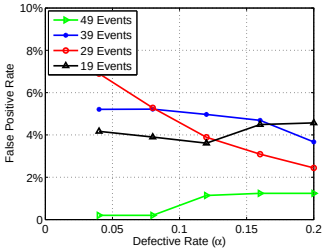


(b) False Positive Rate

Fig. 12. α-Detection

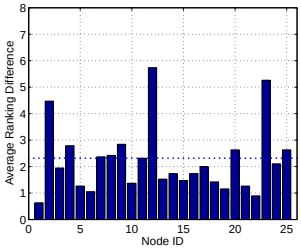


(a) False Negative Rate

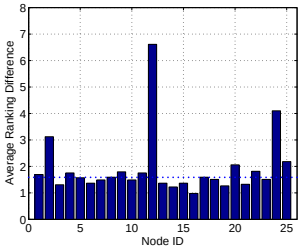


(b) False Positive Rate

Fig. 13. B-Detection

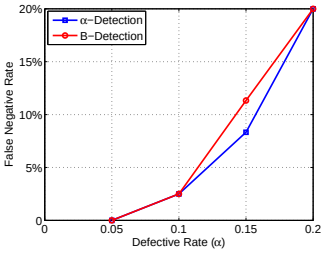


(a) 19 Events

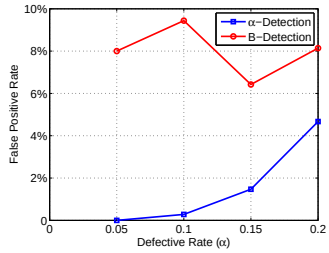


(b) 49 Events

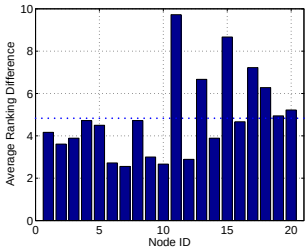
Fig. 14. Ranking Difference



(a) False Negative Rate



(b) False Positive Rate



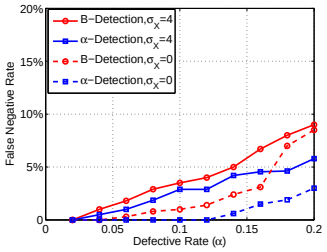
(c) Ranking Differences

Fig. 15. Experiment Results on Acoustic Events

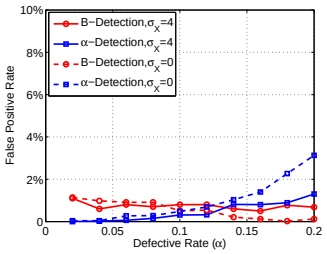
Fig.15(c) shows the average ranking differences of the 20 nodes where node 11 and 15 are faulty nodes. From this figure we can see the average ranking differences of normal nodes is nearly 5, which is much larger than that in the first experiment, indicating a higher noise level. The noise in this experiment is higher due to the limited quality of microphones equipped on MicaZ nodes. Also, the limited power of the speaker causes many nodes to miss the signal even if they are nearby. However, even with severe noise, FIND performs well especially when α is below 0.1.

7. LARGE SCALE SIMULATION STUDY

The results of the system evaluation indicate that FIND can be efficiently and effectively applied to real-world systems with very good performance. Two implementations on radio and acoustic events indicate FIND is generic and compatible with different event modalities. However, physical test beds can only investigate a limited design space. In order to understand the performance of FIND under diversified settings, in this section, we provide extensive simulation results as a complementary study.

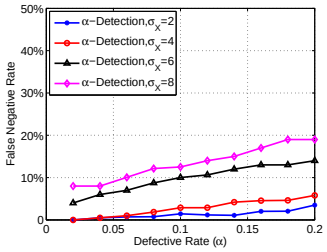


(a) False Negative Rate

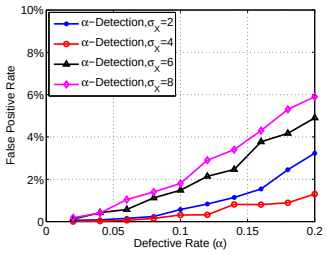


(b) False Positive Rate

Fig. 16. α- vs. B-Detection

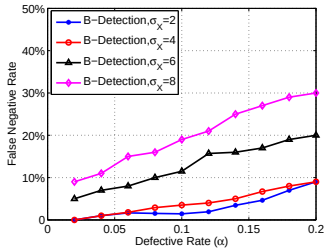


(a) False Negative Rate

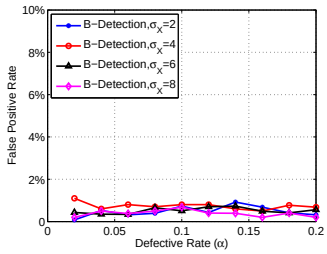


(b) False Positive Rate

Fig. 17. α-Detection vs. Noise



(a) False Negative Rate



(b) False Positive Rate

Fig. 18. B-Detection vs. Noise

7.1. Simulation Setup

In the simulation, both the sensor nodes and events are randomly generated on the map. If not specified, 100 nodes are randomly deployed on a 250m $\times$ 250m map. The sensing range is 25m and the sample size (the number of generated events) is 50. L is set to 10, and thus the complexity of computing $Pr(\bar{s}|s_i)$ is bounded by 10^3. All the data are based on 100 runs.

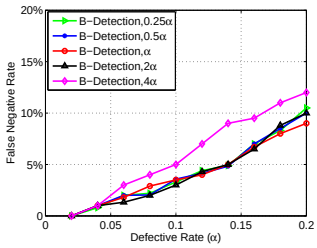
We use the logarithmic distance path loss model [Lord et al. 1980; Rappaport 1996] to simulate received signal strength, and the received signal strength of the ith node S_i can be formulated as

$$S_i \propto -10\beta log(\frac{d_i}{d_0}) + X_i \tag{12}$$

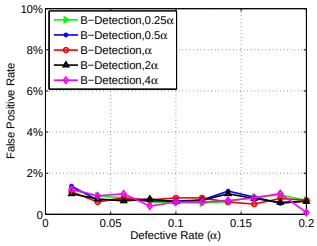
where d_i is the distance between node i and event. d_0 is the reference distance and is set to 1m. β is the signal fading factor and is set to 4 as in [Zhong et al. 2009]. X_i is a random noise whose unit is dB and follows a 0-mean normal distribution with variance σ_X^2. In the simulation, σ_X is changed from 0 to 8 to evaluate FIND in different environments, and the default value is 4. For faulty nodes, faulty readings are at least $3\sigma_X$ away from their normal readings.

7.2. Comparison of α- and B-detection

The false negative rate and false positive rate of α- and B-Detection, with or without noise, are shown in Fig.16. In Fig.16(a), the false negative rate (miss detection rate) of all curves are below 10%. When the defective rate is below 10% (which is always the case in a well-maintained system), the miss detection rate of all curves are below 4%. Thus, both α- and B-detection can effectively detect faulty nodes. Comparing the two solid curves (or the two dashed curves) we can see α-detection always has a lower false negative rate (thus a higher detection ratio) than B-detection. This is not difficult to explain: the only difference between the two detections is that α-detection utilizes a known α and selects the top αN nodes directly based on average ranking differences, while B-detection uses Algorithm 1 to figure out this number. As a result, α-detection has a better performance since it has more information. Also, comparing the two curves with squares (or the two curves with circles), we can see that with the presence of noise, the performance slightly degrades.

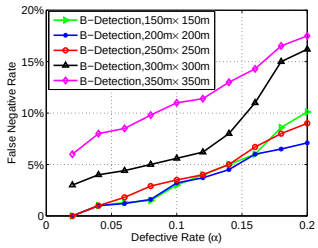


(a) False Negative Rate

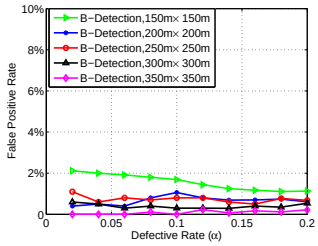


(b) False Positive Rate

Fig. 19. Impact of Inaccurate α

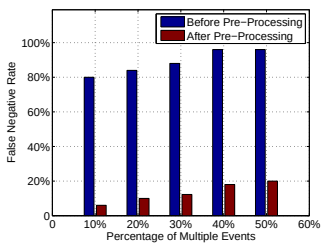


(a) False Negative Rate

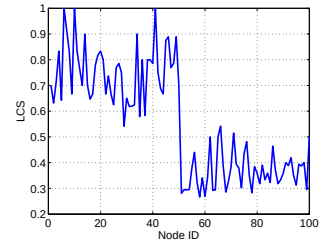


(b) False Positive Rate

Fig. 20. B-Detection vs. Density



(a) False Negative Rate



(b) Length of LCS

Fig. 21. Impact of Multiple Events

However, both α- and B-detection can still detect more than 90% of the faulty nodes in a noisy environment. We will study the impact of noise in the next subsection.

Fig.16(b) shows the false positive rate (false alarm rate) corresponding to previous 4 scenarios. From this figure we can see most of the false positive rates are around 1%, meaning that only a small portion of normal nodes are inaccurately determined faulty, which is a good performance.

7.3. Impact of Noise

We study how noise affects α- and B-detection. In simulation, the standard deviation of noise σ_X is changed from 2 to 8. It is worth noting that the unit of σ_X is dB. As a result, a small increase in σ_X is a significant increase in noise. For example, $\sigma_X = 4$ indicates that more than 30% normal nodes' readings are either greater than 2.5 times or less than 40% of the original readings, which is already a high noise level in sensor networks in which the transmission power is low. $\sigma_X = 8$ indicates that more than 30% of normal nodes' readings are either greater than 6.3 times or less than 15% of original readings, which is an extremely high noise level.

Fig.17 and Fig.18 are the simulation results of α-detection and B-detection with different noise levels. It can be seen from Fig.17(a) and Fig.18(a) that the false negative rate (miss detection rate) increases as noise level increases, as expected. When the defective rate α is below 10% (which is always the case in a well-maintained system), the false negative rate for $\sigma_X = 2$ and $\sigma_X = 4$ are both below 5%. As the noise level increases, the false negative rate is still less than 20% even in extremely noisy environments ($\sigma_X = 8$).

The corresponding false positive rate (false alarm rate) are plotted in Fig.17(b) and Fig.18(b). It can be seen from the figures that the false positive rate of α-detection increases as α increases and the false negative rate increases. This is because in α-detection, the αN nodes with the largest average ranking differences are determined to be faulty nodes so that the number of false alarmed nodes is correlated with the number of miss detected nodes. On the other hand, the false positive rate of B-detection is always below 1%. This is because B-detection detects faulty nodes by the detection algorithm, and thus the number of miss detected nodes does not correlate with the false positive rate.

In summary, both α- and B- detection have low false negative rate and low false positive rate (less than 5% for $\alpha \leq 0.1$) in environments with moderate noise. When the network experiences

extremely high noise, B-detection based on detection algorithm is a better method even when α is accurately given. This is because B-detection has a less than 1% false positive rate, which means that almost all the nodes that are detected as faulty are actually faulty. The miss detected nodes may also be detected after correcting or removing the already detected faulty nodes and applying the whole detection method again. We leave this topic as future work.

7.4. Impact of Inaccurate α

In a system in the real world, it is always impossible to get the true defective rate (i.e., accurate α) of the system. In this subsection we evaluate the performance of FIND with an estimated and inaccurate α. Since the α-detection method simply considers the αN nodes with the largest average ranking differences as faulty nodes, it no longer provides good performance when α is inaccurate: with a halved α, the false negative rate is at least halved; with a doubled α, the false positive rate is at least doubled. As a result, we evaluate only how inaccurate α affects B-detection.

Fig.19(a) and (b) plot the false negative rate and false positive rate for using different estimated α values from 25% to 4 times of the true α in B-detection. It can be seen from Fig.19(a) that except for the curve of using 4α as an estimated defective rate, all the other curves show similar performance. Also, the 4α curve has a higher false negative rate (miss detection rate) than all the other curves, especially when $\alpha > 0.1$. This is because when the estimated α is four times of that of true α, a node may become more likely to be faulty instead of normal. For example, if $\alpha = 0.2$, a node has a probability of 0.2 to be faulty. But if 0.8 is used, a node has a probability of 0.8 to be faulty. (In reality, no one will estimate $\alpha = 0.8$, but from this simulation we can get the performance of FIND in very extreme cases.) From this figure we conclude that a reasonable estimated α does not degrade the performance of FIND. Fig.19(b) shows the false positive rate from which we can see the false positive rate of all curves are all around 1%. Based on this simulation we conclude that the performance of B-detection is insensitive to the inaccurate estimation of α. When no information of α in a networking system is available, choosing α ranging from 2% to 5% will not affect the performance.

7.5. Impact of Network Density

The performance of B-detection in networks with different densities are shown in Fig.20. The map size is changed from $150m \times 150m$ to $350m \times 350m$ while the network size remains 100. It can be seen from Fig.20(a) that when the density is reduced from $300m \times 300m$ to $350m \times 350m$, the false negative rate increases by around 5%. When the density is reduced from $250m \times 250m$ to $300m \times 300m$, the false positive rate also increases by around 2%. This is because with lower density, the number of nodes within the sensing range of the event becomes smaller and the node sequences are shorter. As a result, sequence estimation becomes less accurate because less information (shorter sequences) is available. Also, when the density increases from $200m \times 200m$ to $150m \times 150m$, the false negative rate does not increase. This is because when the density is higher, more nodes are within sensing range and more nodes have similar distances from the events. Due to the presence of noise, the probability that a normal node's ranking is changed by noise also becomes higher because there are more nodes nearby with similar readings. As a result, the false negative rate is lowest in a network with moderate density, as in $200m \times 200m$ and $250m \times 250m$. The false positive rate, as shown in Fig.20(b) is for most cases around 2%.

7.6. Impact of Simultaneous Events

We study how simultaneous events affect the performance of FIND. The total number of events is 100 and the percentage of simultaneous events (for which the sensing results are the sum of several individual events) are changed from 10% to 50%. For each sample, if the length of the longest LCS is less than 75% of its original length, it is discarded. Fig.21(a) shows the false negative rate of our design with or without data pre-processing. It can be seen that without pre-processing, the performance degrades significantly, with a more than 80% false negative rate. After pre-processing, the false negative rate reduces to less than 20%. The fewer the simultaneous events, the better the

performance because more valid samples can be used without being discarded. Fig.21(b) shows the longest LCS (normalized by original length) of the 100 samples where the first 50 samples are from a single event and the last 50 samples are from simultaneous multiple events. It is clear that the length of LCS of the first 50 events is nearly twice that of the last 50, showing the effectiveness of LCS-based data pre-processing.

8. CONCLUSIONS

In this paper we proposed FIND, a faulty node detection method for wireless sensor networks. Without assuming any communication model, FIND detects nodes with faulty readings based only on their relative sensing results, i.e., node sequences. Given detected node sequences, we first proposed an approach to estimate where the events take place and what the original sequences are. Then we theoretically proved that the average ranking differences of nodes in detected sequences and original sequences can be used as an effective indicator for faulty nodes. Based on the theoretical study, a detection algorithm is developed for finally obtaining a blacklist when an accurate defective rate is unavailable. Based on extensive simulations and test bed experiment results, FIND is shown to achieve both a low false negative rate and a low false positive rate in various network settings. In the future, we will extend our FIND method to the distributed wireless sensor networks.

REFERENCES

Atmel Corporation. *Mature AVR JTAG ICE*. Atmel Corporation. Available at http://www.atmel.com/dyn/products/toolscard.asp?tool-id=2737.

L. Balzano and R. Nowak. 2007. Blind Calibration of Sensor Networks. In *IPSN'07*.

Sujogya Banerjee, Shahrzad Shirazipourazad, and Arunabha Sen. 2012. On region-based fault tolerant design of distributed file storage in networks. In *Proceedings of INFOCOM*. IEEE, 2806–2810.

V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak. 2003. A Collaborative Approach to In-Place Sensor Calibration. In *IPSN'03*.

Q. Cao, T. Abdelzaher, J. Stankovic, K. Whitehouse, and L. Luo. 2008a. Declarative tracepoints: a programmable and application independent debugging system for wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 85–98.

Qing Cao, Tarek Abdelzaher, John Stankovic, Kamin Whitehouse, and Liqian Luo. 2008b. Declarative Tracepoints: A Programmable and Application Independent Debugging System for Wireless Sensor Networks. In *SenSys '08*.

Ho-lin Chang, Jr-ben Tian, Tsung-Te Lai, Hao-Hua Chu, and Polly Huang. 2008. Spinning Beacons for Precise Indoor Localization. In *SenSys '08*.

Nathan Cooprider, Will Archer, Eric Eide, David Gay, and John Regehr. 2007. Efficient Memory Safety for TinyOS. In *SenSys '07*.

M. de Bery, O. Cheong, M. van Krevald, and et al. 2008. *Computational Geometry: Algorithms and Applications* . Springer-Verlag,3rd Edition.

Min Ding, Dechang Chen, Kai Xing, and Xiuzhen Cheng. 2005. Localized Fault-Tolerant Event Boundary Detection in Sensor Networks. In *INFOCOM*.

Eiman Elnahrawy and Badri Nath. 2003. Cleaning and querying noisy sensors. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. ACM, 78–87.

J. Feng, S. Megerian, and M. Potkonjak. 2003. Model-Based Calibration for Sensor Networks. In *IEEE Sensors*.

Shuo Guo, Ziguo Zhong, and Tian He. 2009. FIND: Faulty Node Detection For Wireless Sensor Networks. In *SenSys'09*.

S. He, J. Chen, P. Chen, Y. Gu, T. He, and Y. Sun. 2012. Maintaining Quality of Sensing with Actors in Wireless Sensor Networks. (2012).

T. He and et. al. 2006. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *ACM Transactions on Sensor Networks* Vol. 2, No. 1 (February 2006), Page 1 – 38.

Honeywell 2007. *1- and 2-Axis Magnetic Sensor HMC1001 / 1002*. Honeywell. Available at http://www.ssec.honeywell.com/magnetic/datasheets/hmc1001-2_1021-2.pdf.

Chihfan Hsin and Mingyan Liu. 2002. A Distributed Monitoring Mechanism for Wireless Sensor Networks. In *3rd workshop on Wireless Security*.

Joengmin Hwang, Tian He, and Yongdae Kim. 2006. Achieving Realistic Sensing Area Modeling. In *SenSys'06*.

Mohammad Khan, Hieu Khac Le, Hossein Ahmadi, Tarek Abdelzaher, and Jiawei Han. 2008a. DustMiner: Troubleshooting Interactive Complexity Bugs in Sensor Networks. In *SenSys'08*.

M.M.H. Khan, K.H. Le, H. Ahmadi, T.F. Abdelzaher, and J. Han. 2010. Troubleshooting interactive complexity bugs in wireless sensor networks using data mining techniques. (2010).

Mohammad Maifi Hasan Khan, Hieu Khac Le, Hossein Ahmadi, Tarek F. Abdelzaher, and Jiawei Han. 2008b. Dustminer: Troubleshooting Interactive Complexity Bugs in Sensor Networks. In *SenSys '08*.

Veljko Krunic, Eric Trumpler, and Richard Han. 2007. NodeMD: Diagnosing Node-Level Faults in Remote Wireless Sensor Systems. In *MobiSys '07*.

Juan Liu, Ying Zhang, and Feng Zhao. 2006. Robust Distributed Node Localization with Error Management. In *MobiHoc '06*. ACM, 250–261.

K. Liu, Q. Ma, X. Zhao, and Y. Liu. 2011. Self-diagnosis for large scale wireless sensor networks. In *INFOCOM, 2011 Proceedings IEEE*. IEEE, 1539–1547.

Y. Liu, K. Liu, and M. Li. 2010. Passive diagnosis for wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)* 18, 4 (2010), 1132–1144.

H. Lord, W.S. Gatley, and H.A. Evensen. 1980. *Noise Control For Engineers*. McGraw Hill Book Co.

S. Marti, T.J. Giuli, K. Lai, and M. Baker. 2000. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *MOBICOM*.

X. Miao, K. Liu, Y. He, Y. Liu, and D. Papadias. 2011. Agnostic diagnosis: Discovering silent failures in wireless sensor networks. In *INFOCOM, 2011 Proceedings IEEE*. IEEE, 1548–1556.

E. Miluzzo, N. Lane, A. Campbell, and R. Olfati-Saber. 2008. CaliBree: A Self-Calibration System for Mobile Sensor Networks. In *DCOSS*.

M. Panda and PM Khilar. 2011. An Efficient Fault Detection Algorithm in Wireless Sensor Network. *Contemporary Computing* (2011), 279–288.

N. Ramanathan, L. Balzano, M. Burt, D. Estrin, T. Harmon, C. Harvey, J. Jay, E. Kohler, S. Rothenberg, and M. Srivastava. 2006. Rapid Deployment with Confidence: Calibration and Fault Detection in Environmental Sensor Networks. In *Technical Report CENS-TR-62, Center for Embedded Networked Sensing*.

NIthya Ramanathan, Kevin Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, and Deborah Estrin. 2005. Sympathy for the Sensor Network Debugger. In *3rd Embedded Networked Sensor Systems*.

T.S. Rappaport. 1996. *Wireless Communications, Principles and Practice*. Prentice Hall.

K. Sakib. 2012. Asynchronous failed sensor node detection method for sensor networks. *International Journal of Network Management* (2012).

Kannan Srinivasan, Maria A. Kazandjieva, Saatvik Agarwal, and Philip Levis. 2008. The Beta Factor: Measuring Wireless Link Burstiness. In *SenSys'08*.

Jessica Staddon, Dirk Balfanz, and Glenn Durfee. 2002. Efficient Tracing of Failed Nodes in Sensor Networks. In *1st ACM International Workshop on Wireless Sensor Networks and Applications*.

R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. 2004. An Analysis of a Large Scale Habit Monitoring Application. In *SenSys'04*.

Maen Takruri and Subhash Challa. 2007. Drift aware wireless sensor networks. In *International Conference on Information Fusion*. IEEE, 1–7.

Maen Takruri, Subhash Challa, and Ramah Yunis. 2009. Data fusion techniques for auto calibration in wireless sensor networks. In *International Conference on Information Fusion*. IEEE, 132–139.

S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy. 2003. Secure Locations: Routing on Trust and Isolating Compromised Sensors in Location-Aware Sensor Networks. In *SenSys'03*.

Peyman Teymoori, Mehdi Kargahi, and Nasser Yazdani. 2012. A real-time data aggregation method for fault-tolerant wireless sensor networks. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. ACM, 605–612.

G. Tolle and D. Culler. 2005. Design of an Application-Cooperative Management System for Wireless Sensor Networks. In *EWSN*.

G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, S. Burgess, D. Gay, P. Buonadonna, W. Hong, T. Dawson, and David Culler. 2005. A Macroscope in the Redwoods. In *SenSys'05*.

K. Whitehouse and D. Culler. 2002. Calibration as Parameter Estimation in Sensor Networks. In *WSNA*.

N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. 2004. A Wireless Sensor Network for Structural Monitoring. In *SenSys'04*.

Jing Yang, M.L. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant. 2007a. A Comprehensive Source-Level Debugger for Wireless Sensor Networks. In *SenSys'07*.

Jing Yang, Mary Lou Soffa, Leo Selavo, and Kamin Whitehouse. 2007b. Clairvoyant: A Comprehensive Source-Level Debugger for Wireless Sensor Networks. In *SenSys'07*. ACM, New York, NY, USA, 189–203. DOI:http://dx.doi.org/10.1145/1322263.1322282

Kiran Yedavalli and Bhaskar Krishnamachari. 2008. Sequence-Based Localization in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing* 7, 1 (2008).

Y. Zhang, N. Meratnia, and P. Havinga. 2010. Outlier detection techniques for wireless sensor networks: A survey. *Communications Surveys & Tutorials, IEEE* 12, 2 (2010), 159–170.

Ziguo Zhong and Tian He. 2009. Achieving Range-Free Localization Beyond Connectivity. In *SenSys'09*.

Ziguo Zhong, Ting Zhu, Dan Wang, and Tian He. 2009. Tracking with Unreliable Node Sequences. In *INFOCOM*.

G. Zhou, T. He, and J. A. Stankovic. 2004. Impact of Radio Irregularity on Wireless Sensor Networks. In *MobiSys'04*.

A. APPENDIX:

A.1. The proof for Theorem 3.2

PROOF. For simplicity, suppose the nodes labeled from 1 to N_e are faulty nodes and all the rest of the nodes are normal nodes. The ranking difference of the kth node in any sample is a random variable and can be denoted as $d(k)$.

First, consider the case when there is only one faulty node. We observe that a single faulty node k, with ranking difference $d(k)$, may make at most $d(k)$ nodes' ranking change by one. As the example shown in Fig.9(a), faulty node 4 originally shifts to its left by 2 and changes two nodes' rankings (node 2 and 3) by 1. Since node 4 is the only faulty node, its final ranking difference $d(4)$ is 2, which is equal to its original shift in this case.

When there are another $N_e - 1$ faulty nodes present, a faulty node's final ranking difference can be further changed by other faulty nodes, while the number of nodes it affected is still equal to its original shift. Fig.9(b) shows an example to illustrate the maximum possible number of nodes whose ranking differences are affected by a faulty node with ranking difference $d(k)$. In this example, faulty node 4 originally shifts to its left and makes itself a ranking difference of 2 (this is its original shift $d'(4) = 2$). The number of nodes it affected is two (node 2 and 3), which is equal to $d'(4)$. Then nodes 5 and 6, both of which are also faulty nodes, shift to the very left of the sequence and reduce node 4's ranking difference to 0 (this is node 4's final ranking difference $d(4) = 0$). However, it can be seen from the figure that node 2 and 3's ranking differences are 3, which are the results caused by all the three faulty nodes. This indicates that node 4's effect on node 2 and 3's ranking differences still remains. Thus, the number of nodes affected by node 4 is still equal to its original shift $d'(4)$ instead of its final ranking difference $d(4)$. Given the final ranking difference $d(k)$ of any node k, its maximum possible original shift $d'(k)$ is $d(k) + N_e - 1$, which happens when all other faulty nodes affect k's ranking in the same direction that is opposite to k's original shift direction just as shown in this example. As a result, the maximum number of nodes affected by node k is $d(k) + N_e - 1$.

Based on this analysis, the probability that a node's ranking is changed by node k is upper bounded by $\frac{d(k)+N_e-1}{N-1}$. Also, a normal node never changes its ranking itself. Then, given $d(k)$, the expected ranking difference of a normal node p caused by node k, denoted as $d^k(p)$, can be computed as

$$\begin{aligned} E(d^k(p)|d(k)) &\leq 0 \cdot (1 - \frac{d(k) + N_e - 1}{N-1}) + 1 \cdot \frac{d(k) + N_e - 1}{N-1} \\ &= \frac{d(k) + N_e - 1}{N-1}, \forall (N_e + 1) \leq p \leq N \end{aligned} \tag{13}$$

Taking expectations on both sides yields

$$E(d^k(p)) \leq \frac{E(d(k)) + N_e - 1}{N-1}, \quad \forall (N_e + 1) \leq p \leq N \tag{14}$$

Then, the expected ranking difference of any normal node p caused by all the N_e faulty nodes is upper bounded by

$$E(d(p)) \leq \sum_{k=1}^{N_e} E(d^k(p)) = \frac{\sum_{k=1}^{N_e}(E(d(k)) + N_e - 1)}{N-1} = \frac{N_e(\mu_e + N_e - 1)}{N-1}, \quad \forall (N_e+1) \leq p \leq N \tag{15}$$

where μ_e is the mean of average ranking difference of all faulty nodes and is given by

$$\mu_e = E(\frac{\sum_{k=1}^{N_e} d(k)}{N_e}) = \frac{\sum_{k=1}^{N_e} E(d(k))}{N_e} \tag{16}$$

The first inequality in Eq.15 holds since the maximum total ranking change caused by N_e faulty nodes happens when all the faulty nodes make node p move to the same direction.

When the sample size is sufficiently large, sample mean becomes a good estimator for the expectations in Eq.15. By replacing the expectations with sample mean we get

$$D(p) \leq \frac{N_e}{N-1}(\mu_e + N_e - 1), \quad \forall (N_e+1) \leq p \leq N \tag{17}$$

Let B denote the upper bound $\frac{N_e}{N-1}(\mu_e + N_e - 1)$ on the right hand side of Eq.17. Then, the average ranking difference of any normal node p is upper bounded by B. Also, if a node q has an average ranking difference $D(q) > B$, it must be a faulty node. □

A.2. The proof for Theorem 3.3

PROOF. For any node p and node q satisfying $D(p) < D(q)$, they must belong to one of the following cases:

(1) $B < D(p) < D(q)$: According to Theorem 3.2, for any node k, the probability that k is a faulty node when $D(k) > B$ is 1, regardless of what $D(k)$ is (as long as it is greater than B). Then, both p and q are faulty nodes for sure in this case. $Pr(p = \text{“}F\text{”}) = Pr(q = \text{“}F\text{”}) = 1$.
(2) $D(p) \leq B < D(q)$: According to Theorem 3.2, q is a faulty node for sure since $D(q) > B$. The probability that p is a faulty node is no greater than 1. Thus, $Pr(p = \text{“}F\text{”}) \leq Pr(q = \text{“}F\text{”}) = 1$.
(3) $D(p) < D(q) \leq B$: Without additional information, for any node k satisfying $D(k) \leq B$, the probability that k is a faulty node (denoted by β) is equal to the percentage of faulty nodes among all the nodes whose ranking differences are upper bounded by B, regardless of the value of $D(k)$. Thus, $Pr(p = \text{“}F\text{”}) = Pr(q = \text{“}F\text{”}) = \beta$ in this case.

Summarizing all the three cases, it is not difficult to get $Pr(p = \text{“}F\text{”}) \leq Pr(q = \text{“}F\text{”})$ when $D(p) < D(q)$. □

A.3. The Uniqueness of Detection Algorithm

The uniqueness of detection algorithm is given by Theorem A.1:

THEOREM A.1. *Given node sequence n_1-n_2-$\cdots$-n_N which is in descending order of their ranking differences, if $\{n_1, n_2, \cdots, n_k\}$ is the estimation of the blacklist such that n_k satisfies $D(n_{k+1}) \leq B < D(n_k)$, then this estimation is unique. In other words, there does not exist another estimation $\{n_1, n_2, \cdots, n_{k'}\}$ where $k' \neq k$ but satisfies $D(n_{k'+1}) \leq B' < D(n_{k'})$.*

PROOF. The proof is based on contradiction. Suppose $\{n_1, n_2, \cdots, n_{k'}\}$ exists. Without loss of generality, assume $k > k'$. Based on definition, $\mu_e N_e$ is the sum of the ranking differences

of $n_1, \cdots, n_k$ and $\mu'_e N'_e$ is the sum of the ranking differences of $n_1, \cdots, n_{k'}$. According to B's definition:

$$\begin{aligned} B &= \frac{N_e}{N-1}(\mu_e + N_e - 1) = \frac{N_e\mu_e + N_e^2 - N_e}{N-1} \\ &= \frac{D(n_1) + D(n_2) + \cdots + D(n_{k'}) + \cdots + D(n_k) + N_e^2 - N_e}{N-1} \\ &> \frac{D(n_1) + D(n_2) + \cdots + D(n_{k'}) + N_e'^2 - N'_e}{N-1} = B' \end{aligned} \tag{18}$$

where the inequality is based on $N_e > N'_e$ and $D(n_{k'+1})$,$\cdots$,$D(n_k)$ are all non-negative.

Eq.18 shows that B can be considered as a strictly increasing function of k. On the other hand, $D(n_k)$ is decreasing as k increases (n_1-n_2-$\cdots$-n_N are already sorted in descending order of their average ranking differences). Both functions are discrete, but they can be treated as samples of continuous functions. As $D(n_{k+1}) \leq B < D(n_k)$, k is essentially the nearest integer to the left of the point at which the two functions intersects. Such intersection is unique given that B is an strictly increasing function of k and $D(n_k)$ is decreasing.

Mathematically, $D(n_k) \leq D(n_{k'+1})$ (since $k > k'$ implies $k \geq k' + 1$). Also, based on the estimation method, n_k and $n_{k'}$ satisfy $D(n_{k+1}) \leq B < D(n_k)$ and $D(n_{k'+1}) \leq B' < D(n_{k'})$. As a result, $B < D(n_k) \leq D(n_{k'+1}) \leq B'$, which contradicts to Eq.18 that $B > B'$. Thus, n_k is unique and the estimation based on the method of finding all the nodes whose ranking differences are no less than B is unique. □