Assessing Diagnosis Approaches for Wireless Sensor Networks: Concepts and Analysis

Rui Li[1](李瑞), Kebin Liu[2](刘克彬), *Member ACM, IEEE*, Xiang-Yang Li[2],[3](李向阳), *Member ACM, Senior Member IEEE*, Yuan He[2](何源), *Member ACM, IEEE*, Wei Xi[1*](惠维), *Member ACM, IEEE, CCF*, Zhi Wang[1](王志), Jizhong Zhao[1](赵季中), *Member ACM, IEEE, CCF*, Meng Wan[4](万猛), *Member ACM, IEEE*

[1]School of Electronic and Information Engineering, Xi'an Jiaotong University, China

[2]School of Software and TNList, Tsinghua University, China

[3]Department of Computer Science, Illinois Institute of Technology, USA

[4]Center of Science and Technology Development, Ministry of Education, China

Email:{rli, zhiwang, zjz}@mail.xjtu.edu.cn, {kebin, he}@greenorbs.com, xli@cs.iit.edu, weixi.cs@gmail.com, wanmeng@cutech.edu.cn

Received month day, year

Abstract Diagnosis is of great importance to wireless sensor networks due to the nature of error prone sensor nodes and unreliable wireless links. The state-of-the-art diagnostic tools focus on certain types of faults, and their performances are highly correlated with the networks they work with. The network administrators feel difficult on measuring the effectiveness of their diagnostic approaches and choosing appropriate tools so as to meet the reliability demand. In this work, we introduce the D-vector to characterize the property of a diagnosis approach. The D-vector has five dimensions, namely the *Degree of Coupling*, the *Granularity*, the *Overhead*, the *Tool Reliability* and the *Network Reliability*, quantifying and evaluating the effectiveness of current diagnostic tools in certain networks. We employ a skyline query algorithm to find out the most effective diagnosis approaches, i.e., skyline points (SPs), from five dimensions of all potential D-vectors. The selected skyline D-vector points can further guide the design of various diagnosis approaches. In our trace-driven simulations, we design and select tailored diagnostic tools for GreenOrbs, achieving high performance with relatively low overhead.

Keywords Diagnosis Approach, Analysis and Measurement, Wireless Sensor Network

1 Introduction

The nature of error prone sensor nodes and dynamic wireless links make fault diagnosis a crucial task in wireless sensor networks (WSNs). With proliferation of the sensor network applications in the wild [1, 2, 3, 4], this trend has accelerated as the diagnosis is even harder caused by com-

* Wei Xi is the corresponding author.

This work is supported by the National Natural Science Foundation of China (NSFC) under Grant Nos. 61190112, 61325013, 61228202, and 61170216; Natural Science Foundation of U.S (NSF) under Grant Nos. CNS-0832120, CNS-1035894, ECCS-1247944, ECCS-1343306; the Fundamental Research Funds for the Central Universities of China under Project No. 2012jdgz02 (Xi' an Jiaotong University), and the Research Fund for the Doctoral Program of Higher Education under project No. 20130201120016.

plex topography and dynamic environmental factors [5, 6, 7]. Various debugging and diagnostic tools have been proposed, aiming at detecting different types of network faults, for instance, Declarative Tracepoints [8] and Clairvoyant [9] focus on debugging software bugs. Leveraging the periodically collected network state information, Sympathy [10] and PAD [11] deduce failures in sensor networks with rule-based and inference-based approach respectively. Dustminer [12] tries to uncover failures resulting from interactions between different components.

Diagnosis approaches are different from each other in many aspects, such as, the types of failures they tackle, the information they use in fault deduction process, the reliability they own under varying system settings, and the like. For example, some diagnostic tools deliver diagnosis information using initial network protocols, so network failures lead to incomplete information for the diagnosis engine and thus inaccurate judgments. These approaches work well in a reliable network. They, however, may experience significant performance degradation when more failures occur. In contrast, diagnostic tools using out-bound information can avoid such problems. A comprehensive understanding of the diagnosis approaches is indeed necessary for evaluating and selecting diagnosis tools as well as guiding the future design.

In this work, we propose a framework for modeling the features of diagnosis approaches in wireless sensor networks. We present a D-vector to specify the property of a diagnostic tool. The D-vector includes five dimensions, the *Degree of Coupling*, the *Granularity*, the *Overhead*, the *Tool Reliability* and the *Network Reliability*. Each dimension characterizes the diagnosis tools from one angle. Besides, different dimensions are inner-correlated. Under this framework, each diagnostic tool can be denoted by a D-vector, and all D-vectors form a set of diagnosis approaches. For example, the diagnosis approach which can detect all types of failures with zero overhead cannot be achieved based on current techniques. A following question is whether we can find all potential diagnosis solutions based on the existing efforts in this field, in other words, all kinds of diagnosis tools (represented by D-vectors) that can be achieved by appropriate selection and combination of current schemes.

We further propose to derive constraints by mining the correlations among different dimensions. According to these constraints, we find the set of D-vectors corresponding to the properties of all potential diagnosis approaches. We conduct a skyline search on this set for the skyline points. The skyline points indicate the best properties a diagnosis tool can achieve. That is, all the other points are dominated by at least one skyline point which means the diagnosis tool corresponding to the skyline point is no worse than the tool correspond to a normal point in all dimensions. This result gives a guidance in the diagnostic tool selection. D-vectors which violate constraints denote the properties of diagnosis tools that cannot be achieved at present. These D-vectors figure out the direction of future designs. The main contributions of this work are summarized as follows.

1. To the best of our knowledge, this is the first work on exploring the principal factors and inner-correlations to characterize various diagnostic tools. And the first step towards better understanding diagnosis approaches in terms of the *Degree of Coupling*, the *Granularity*, the *Overhead*, the *Network Reliability* and the *Tool Reliability*.

2. We introduce D-vectors to model the properties of different diagnosis approaches and analyze the cause-and-effect diagram, so as

to give the potential D-vector points in the space of diagnosis approaches.

3. By employing a skyline query algorithm, called *NNS*, and real trace-driven simulations, we can figure out the most effective diagnosis approaches, and thus give the future design guidance to diagnosis issues.

The remainder of this paper is organized as follows. Related works are illustrated in Section 2 and motivations of this work in Section 3. In Section 4, we define a D-vector, and quantify five principal factors that can form the D-vector. Section 5 analyzes the constraints of a D-vector by exploring the correlations of five factors. Section 6 contains skyline query algorithms to search the skyline points so as to find the most effective diagnosis approaches and Section 7 gives experimental results and analyzes the future design of diagnosis tools. Section 8 concludes this paper and gives future works we may further study.

2 Related Works

Most debugging tools [13, 14] target finding software bugs in sensor nodes, are considered as pre-deployment diagnosis. Declarative Tracepoints [8] reports a debugging system, which could automatically watch program states to detect bugs. Clairvoyant [9] enables the code-level debugging for WSNs which allows users remotely execute debugging commands such as step and breakpoint. Debugging tools are effective at finding network failures, however, in cost of incurring huge control message, they can bring relatively high overhead.

Operating period diagnosis attracts many efforts. MintRoute [15] visualize the network topology by collecting neighbor tables from sensor nodes. SNMS [16] constructs network infrastructure for logging and retrieving state information at runtime and EmStar [17] supports simulation, emulation and visualization of operational sensor networks. Sympathy [10] actively collects metrics from sensor nodes and determines the root-causes based on a tree-like fault decision scheme. PAD [11] reports the concept of passive diagnosis which leverages a packet marking strategy to derive network state and deduces the faults with a probabilistic inference model. TinyD2 [18] is a self-diagnosis tool, which combine the view of the node itself to the diagnosis process.

Post-deployment diagnosis are usually log-based analysis, e.g., Dustminer [12] focuses on troubleshooting interactive bugs by mining discriminant patterns from the logs on sensor nodes. Powertracing [19] uses current patterns to classify bugs into various types, however, it is an independent diagnostic tool that do not need network statistical data and loosely coupled with the network itself.

Skyline query is considered as a promising technique in multi-criteria optimization process in database community. The intuitive method, such as BNL [20], to computes the skyline is to compare each point q with other points, if q is not dominated by other points, then q is part of the skyline. Bitmap [21] technique encodes in bitmaps all the information required to decide whether a point is in the skyline. NN [22] and BBS [23] are both skyline query algorithms based on nearest neighbor search strategies. Since we do not care about the I/O performance and the CPU usage, we choose a nearest neighbor based skyline query algorithm for efficient to implement.

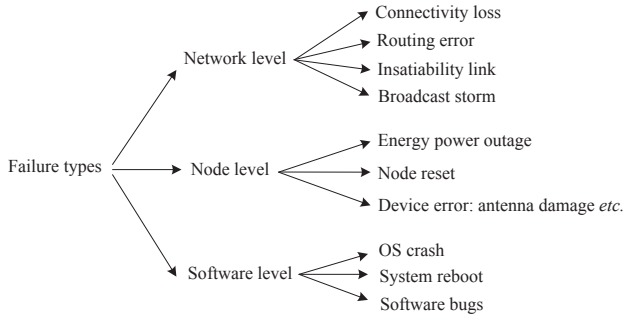


Fig. 1: **Three types of failures that mainly appear in GreenOrbs system.**

3 Motivation

To reveal the factors that influence the diagnosis performances, we introduce some basic observations in GreenOrbs [24] system. Fig. 1 classifies the failure types that appear in the system. For network diagnosis, many approaches have been developed to tackle a certain type of failures. However, if without domain knowledge, how can we choose diagnostic tools for network diagnosis? We further give our observations and summarize the principal factors that influence the choice of diagnostic tools.

Observation 1. We can distinguish node failure and network failure by data values changes and data quantity changes, node failure can only influences the data values, but network failure can result in changing the number of packets. As shown in Fig. 2, the number of transmissions varies from each other on different node, that may indicate the node failures appear in the network. However, Fig. 3 illustrates the packets received by sink during 48 sample period, that the severe network failures occur at sample point 3, 8 and 9.

Observation 2. Many symptoms are necessary conditions for the failure causes but not sufficient conditions. As shown in Fig. 4, packet loss is a necessary condition for three types of failures causes, therefore, the combination of several necessary conditions may form the sufficient conditions of a certain failure.

From above observations, we discuss the main factors that influence the diagnosis results that comes from the intrinsic nature of different diagnosis approaches. We classify the root causes that can influence the diagnosis results into five main factors. And the five factors are inner-correlated with each other as shown in Fig. 5, which are established with cause-and-effect diagram [25].

The network reliability and diagnostic tool reliability influences each other. If the network reliability is high enough, we do not need more effective diagnostic tools. And if the network reliability is low, we cannot rely on the network data, so we need an independent diagnostic tool and the diagnosis approach should not collect more information from the network itself. Fig. 5 elaborates that *Tool Reliability* and *Degree of Coupling* both influence the *Network Reliability*. However, the *Tool Reliability* only affected by the *Network Reliability*.

Since the effective diagnosis approaches is restricted by the information types that collected from sensor networks, the effective diagnosis information *Overhead* is affected by *Granularity* and *Degree of Coupling*.

The *Degree of Coupling* is affected by the *Network Reliability*, they influence each other, since the *Network Reliability* decides the reliability of the data collected from the network, without reliable data, the diagnosis approach is useless, so if the *Network Reliability* is low, we do not need the diagnostic tool high coupled with the network.

The *Granularity* influences the diagnosis results, but it is not affected by other factors. Since that the *Granularity* determines the capability of diagnosis approaches, with more fine-grained granularity of diagnostic information, the diagnosis results can be more accurate.

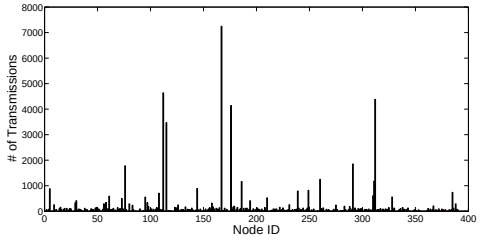


Fig. 2: **Number of Transmissions on different sensor nodes in GreenOrbs system**

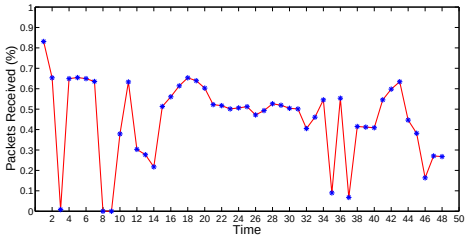


Fig. 3: **Packets received over continuous sample period during five days**

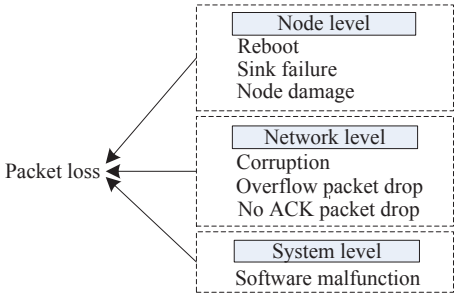


Fig. 4: **Causes of packet loss.** Packet loss is a necessary condition for three types of failures, however, the combination of several necessary conditions may form the sufficient conditions of packet losses.

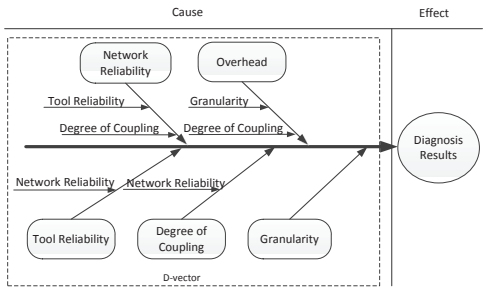


Fig. 5: **Cause-and-effect diagram of diagnosis results**. It shows five factors that influence the diagnosis results in WSNs. Smaller arrows connect the sub-causes to major causes, and reflect the inner-correlations among five factors.

4 Quantification of D-vector

The five main factors that may affect the diagnosis result of a diagnosis approach, and they are correlated with each other since the intrinsic nature of diagnosis approaches. In this section, we aim to quantify the main factors as to indicate various diagnosis approaches effectively.

4.1 Definition of a D-vector

We define a D-vector to specify the property of a diagnosis approach, which contains five principal dimensions. The five dimensions are the five factors that influence the diagnosis result of a certain application, and Fig. 5 describes the cause-and-effect diagram. A D-vector is formed to represent a certain diagnosis approach, however, not each D-vector corresponds to an existing diagnosis approach.

We further analyze the five factors that influence the property of a diagnosis approach. In order to elaborate the relationship between sensor networks and diagnostic tools, we introduce *Degree of Coupling* to indicate the extent, to which a diagnostic tool and a sensor network are coupled, namely the amount of effective information a diagnostic tool needs to collect from the sensor networks. We then use entropy to quantify the effective information that the diagnostic tool collects from the sensor network for diagnosis.

4.2 Degree of Coupling

Degree of Coupling is introduced as a first step to quantify the information collected by diagnostic tools from sensor networks. Since the interactive effects between sensor networks and diagnostic tools are complicated, the difficulties exist in quantifying the correlation between two parts. As a result, we introduce *Degree of Coupling* to measure the interaction between diagnostic tools and sensor networks. For ease of expression, we use *D-C* to denote *Degree of Coupling* in the rest of this paper.

The diagnostic tools always need to collect information as input to uncover faults. Different diagnostic tools need different information types. For example, source level debugging tools [8, 9] do not need any operational period network information, but need global state information as input. The approach for each tool to get diagnostic information is quite different. How much effective information a diagnostic tool can get from the sensor network and how much overhead does a tool needs to collect diagnostic information? We need to find a quantitative metric for all the tools to have a unified criterion and calculate the effective information that a diagnostic tool needs.

We introduce mutual information to measure how much information a tool collects. Note that different diagnostic tools collect different types of information. Inputs of the diagnostic tools bound the diagnosis granularity. We then use entropy to illustrate the uncertainty with the diagnostic information. If the diagnostic tools collect information without noise; the information can be calculated as the entropy of the source information, which is defined by,

$$H(X) = -\sum_{i} P(x_i) \log P(x_i) \qquad (1)$$

$P(x_i)$ is the probability of the i-th variable that source information may hold. However, $H(X)$ is entropy of the source information. The source information usually contains noise, and we use $I(X;Y)$ to denote the information that the diagnostic tools can get from source information, the $I(X;Y)$ is the mutual information and can be calculated below,

$$I(X;Y) = H(X) - H(X|Y) \qquad (2)$$

Where, $H(X|Y)$ is conditional entropy, for the

Symbol	**Definition**
Degree of Coupling	the amount of effective information a diagnostic tool needs to collect from the sensor network
R_{sensor}	network reliability that can be measured by network yield
R_{diag}	diagnostic tool reliability relies on the true positive and true negative diagnosis results
Granularity	the diagnostic information that diagnostic tool collected
Overhead	calculated by the diagnostic information traffic over total traffic during a sample period
$H(X)$	Entropy of source information X
$N_s = \{N_{s1}, N_{s2}, \ldots, N_{sm}\}$	information source statistical set, where N_{si} is the traffic of different features during observation period
$N_d = \{N_{d1}, N_{d2}, \ldots, N_{dk}\}$	the destination of all information set for diagnosis purpose
$N_p = \{N_{p1}, N_{p2}, \ldots, N_{pl}\}$	specified statistical set of diagnostic tools during the observation period
t_pos	number of true positives in diagnosis results
t_neg	number of true negatives in diagnosis results
D-vector	(DC, R_{sensor}, R_{diag}, Granularity, Overhead) quintuple to indicate a specific diagnosis approach

Table 1: Useful Notations

discrete information the $H(X|Y)$ can be illustrated as,

$$H(X|Y) = -\sum_i \sum_j P(x_i, y_j) \log P(x_i|y_j) \quad (3)$$

We address that $I(X;Y)$ has three properties,

- Non-negative property, i.e. $I(X;Y) \geq 0$, where the equality holds if and only if the sending information and receiving information are independent.
- Mutual information is no greater than the source information entropy, i.e. $I(X;Y) \leq H(X)$, when there is no noise in channel, the $I(X;Y)$ equals to $H(X)$ in numerical value.
- Symmetry property, that means the mutual information is equal to source information and destination information.

For the continuous information of $I(X;Y)$, it can be calculated through the generation of discrete information, and hold the same property with the discrete situation,

$$H(X) = -\int P(x) \log P(x) dx \quad (4)$$

$$I(X;Y) = \iint P(x,y) \log \frac{P(x,y)}{P(x)P(y)} dxdy \quad (5)$$

Definition 1. *For the entire network system, we can model the source information statistical set as* $N_s = \{N_{s1}, N_{s2}, \ldots, N_{sm}\}$, *where* N_{si} *is the network traffic of different features during the observation period, the destination of all information set for diagnostic purpose is* $N_d = \{N_{d1}, N_{d2}, \ldots, N_{dk}\}$, *where* N_{di} *is the traffic of all information collected from the sensor network.* $N_p = \{N_{p1}, N_{p2}, \ldots, N_{pl}\}$, *is the specified statistical set of diagnostic information, where* N_{pi} *is the traffic of different diagnostic tools during the observation period.*

From definition 1 and equation 1, we can describe the feature entropy of above three statistical information sets as follows,

$$H(N_s) = -\sum_{i=1}^{m} (\frac{N_{si}}{S_1}) \log(\frac{N_{si}}{S_1}) \quad (6)$$

$$H(N_d) = -\sum_{i=1}^{k} (\frac{N_{di}}{S_2}) \log(\frac{N_{di}}{S_2}) \quad (7)$$

$$H(N_p) = -\sum_{i=1}^{l} (\frac{N_{pi}}{S_3}) \log(\frac{N_{pi}}{S_3}) \quad (8)$$

Where, $S_1 = \sum_{i=1}^{m} n_{si}, S_2 = \sum_{i=1}^{k} n_{di}, S_3 = \sum_{i=1}^{l} n_{pi}$ illustrates the collected information traffic during the observation period. Equations 6, 7 and 8 show the entropy of source information observed from sensor networks, the entropy of sink side information, and the entropy of diagnostic tools' input, respectively. The value of feature entropy lies in the range $[0, logN]$. When n_i is closely the same event the value is approximate 0, when distribution of the feature is maximized, the value is approximate to $logN$, i.e. $n_1 = n_2, \ldots, = n_i$.

Therefore, we can calculate the distribution of entropy for each diagnostic tool in order to get DC. We use feature entropy of each diagnostic tool as to quantify DC, when the feature entropy is large, that means the diagnostic tool need more effective information from sensor networks. i.e. the DC is large, and vice versa. Then DC can be calculated as follows,

$$DC = \frac{H(N_p) - H(N_p|N_d)}{H(N_s) - H(N_s|N_d)} \quad (9)$$

where DC is the ratio of the effective input information for the diagnostic tools to all information collected from the sensor networks.

4.3 Network Reliability

The network reliability is one of the major factors that influence the selection of diagnosis ap-

proaches. The sensor networks suffer from different types of failures during the deploy period. If the network reliability is high enough, we do not need to care about the DC of the diagnostic tool. Then, how to measure the reliability of sensor networks?

Yield [26] is the metric of data quality that collected from sensor networks. The *node yield* measures the quality of each node, while the *network yield* measures the quantity of the entire sensor networks. The *node yield* can be calculated as follows,

$$Yield_i = \frac{\#\text{ of packets received by sink from } i \text{ during } p}{\#\text{ of packets sent by } i \text{ during } p} \tag{10}$$

Where i is the node ID, and p is a observation period. The *network yield* is calculated by,

$$Yield = \frac{\#\text{ of packets received by sink during } p}{\#\text{ of packets sent by all nodes during } p} \tag{11}$$

We use *yield* to measure the network reliability, and we adopt *network yield* as a key component of network reliability. We can calculate different *network yield* for various applications. In our case studies, we will adopt *network yield* combined with *node yield* as our metric to evaluate the network reliability. Therefore, network reliability, denoted by R_{sensor}, is influenced by the *network yield* and *node yield*. The threshold $R_{sensor0}$ is specified in different sensor network applications as tailored to diverse purposes.

In most sensor networks the value of R_{sensor} is equal to the value of *network yield*. However, in a small portion of sensor network applications, all nodes are one hop to sink node since such that the fidelity of data can be guaranteed. And for this kind of sensor networks, the *network yield* can usually be easily guaranteed. So we adopt average node yield instead of *network yield* to represent R_{sensor}. R_{sensor} can be calculated by,

$$R_{sensor} = \frac{1}{n}\sum_{i=1}^{n} Yield_i \tag{12}$$

Where $Yield_i$ is the *node yield* of each node.

4.4 Tool Reliability

For all kinds of sensor network applications, we can obtain the diagnostic information through many ways, like through the sensor network itself, the extra field from the data packets, or get information from sniffer nodes that can overhear the network status. Whatever methods that diagnosis approaches adopt, they all need information as input to unveiling the faults that occurred in sensor networks.

There are three main approaches that used in diagnostic tools. The first approach is inference of the faults with various inference algorithms, using data collected from sensor networks. The second choice is rule based diagnosis using decision trees to decide where the faults locate. The third way, or the most frequent way, to deal with performance degradation in every sensor network is the administrators to monitor the network and use domain knowledge to judge if there were faults happened in sensor networks. Manually detection is laborious but effective, since many faults are hidden or hard to detect by one or two specific diagnostic tools but easy to detect by administrators.

As we have addressed, the capabilities of diagnostic tools are bounded due to the constraints on information each tool adopt in diagnosis, which is shown in Table 2.

Nevertheless, we need to illustrate the tool reliability. For instance, the tool reliability, such as Sympathy and Declarative Tracepoints, are detecting different faults. The Sympathy may be of a high reliability using the information collected from the se nsor networks. However, the Declara-

Fault Types	Information Types	Recovery Methods	Diagnostic Tools
Node, Link	Out-network	Node reboot, Node replacement	PAD, Powertracing
State	In-network	Network or node reconfiguration	Sympathy, TinyD2
Source code	Global	Code correction or rewrite	Declarative Tracepoints, Clairvoyant

Table 2: Typical diagnostic tools that can detect certain types of faults.

tive Tracepoints detect faults in code lines, which cannot use the diagnostic information parsed from the common data packets.

Therefore, a unified metric to measure tool reliability is even harder to establish with a same set of information for the specified sensor networks they get. We just simply choose the reliability of each tool under the same condition of different granularity and the reliability is calculated by,

$$R_{diag} = \frac{t\_pos + t\_neg}{pos + neg} \tag{13}$$

Where t_pos is the number of true positives in diagnosis results, and t_neg is the number of true negatives. The pos is the number of positive results of the diagnostic tools and neg is the number of negative results of the diagnostic tools. Equation 13 denotes the accuracy of diagnosis results in sensor networks and we adopt this accuracy as the tool reliability, since the accuracy indicates the diagnosis results effectively.

4.5 Granularity & Overhead

Granularity measures the capability of a diagnosis approach. For different diagnosis approaches, the diagnosis granularity is bounded by the input information types, i.e. the granularity basically determines the capability of each diagnostic tool. As Table 2 describes, we can derive three levels of information granularity, namely in-network statistical data, out-network statistical data, and global information data. How to classify each diagnosis approach into different levels?

We formalize the three levels into three discrete values, and the granularity is the nature of each diagnosis approach, since the information collected for each tool is already determined by the approach itself. For different diagnostic tools and granularity they have, we obtain the results in Table 3.

The overhead usually defined as the amount of control packets for diagnosis. Nevertheless, the cost of diagnosis approach is considered as the traffic overhead combine the hardware cost. As most diagnosis approaches are relatively high coupled with sensor networks, and do not need any extra hardware, we simplify the overhead contains traffic overhead only. If a diagnostic tool contains extra hardware equipment, we ignore the traffic

*The Powertracing holds a DC of 0, since the diagnostic tool is an independent system and it does not need statistical data from sensor nodes.

†Some of these tools are debugging tools and they are used before sensor networks are deployed. They rely on exchange messages with sensor networks, that may cause more overhead.

Granularity	***DC***	**Diagnostic Tool**
Out-network (3)	Relative high	PAD, Powertracing*
In-network (2)	Relative fair	Sympathy, TinyD2
Global (1)	Relative low	Declarative Tracepoints, Clairvoyant†

Table 3: Granularity of each diagnostic tool is determined by the information types they collected from the sensor networks

overhead and simply set the traffic overhead equals to 1.

In this study, we define the overhead is calculated by the diagnostic information traffic over total traffic during a sample period, and it ranges from 0 to 1.

5 D-vector as a Property Set

In this section, we explore the five factors as a whole property set (D-vector) for indicating diagnosis approaches. Furthermore, we illustrate the potential D-vector points, that are restricted by the inner-correlations among the five factors.

5.1 Impact factors of D-vector

In this section, we explore the correlations among the five factors that restrict the potential D-vector points. In section 3, note that the *Granularity* and the *DC* both influence the *Overhead*. Since the granularity determines the capability of a certain diagnosis approach, when granularity is more fine-grained, more diagnostic information types are needed, that cause relatively high overhead. On the contrary, if the *Overhead* is relatively high, we cannot derive the *Granularity* is more fine-grained. If the *DC* is relatively high, the diagnosis overhead must be relatively low, and vice versa. However, the *Overhead* cannot influence the *DC*.

The *Network Reliability* is influenced by the *DC* and the *Tool Reliability*. The *DC* has a negative impact on the *Network Reliability*, while the *Tool Reliability* has a positive impact on the *Network Reliability*.

A D-vector denotes the property of a diagnostic tool. If there exists a chosen diagnosis approach, the candidate D-vector point refers to the best diagnosis approach we can achieve at present, and the future designs accompany with the selected diagnosis approach of a specific D-vector that can dominate all existing tools.

Table 3 shows the *Granularity* and *DC* of typical diagnostic tools. We can find that PAD is less effective than other diagnostic tools. But that is not always the fact. PAD has relatively low *DC* than other diagnostic tools. That means PAD slightly relies on the statistical information collected from the sensor network. It is more tolerant to faults than other diagnostic tools, which heavily rely on the network statistical data.

We claim that D-vector is pragmatic for characterizes the properties of diagnosis approaches, and of real usability for design and selection of diagnostic tools. To better understand D-vector, we give our trace-driven simulations and evaluations of different diagnostic tools in real prototype in Section 7. Although most existing works focus on designing and developing diagnostic tools, guiding design of diagnostic tools are of importance to all applications.

5.2 Constraints of DC and Granularity

From Table 3 and analysis, we can get two important constraints.

Constraint 1: The *DC* indicates how diagnostic tools are coupled with sensor networks. When the value of *DC* gets larger, the diagnostic tools become less effective.

The constraint 1 is easy to validate. If diagnostic tools rely more on network information, the higher uncertainty of the diagnosis results they will be. Since the diagnostic information is not always available due to error-prone nature of sensor nodes and dynamic characteristics of network environment. If the reliability of sensor networks is high, we can select diagnostic tools with high *DC*, since it could meet the requirements we set (the parameters are specified by the administrators or the users) and have relatively low overhead. (When *DC* gets larger, the *Overhead* of most diagnostic tools becoming smaller.)

Constraint 2: The capabilities of diagnostic tools are restricted by the *Granularity* they own, and the *Granularity* is of importance to coarse-grained selection of diagnostic tools.

The constraint 2 can be inferred from the diagnostic tools. Firstly, the capabilities of diagnostic tools are restricted since the *Granularity* is determined by the input of diagnostic information. However, each diagnostic tool aimed at unique failure due to information type constraints. Secondly, the *Granularity* can help coarse-grained selection of diagnostic tools. Without enough information types, the specific faults cannot be detected. Henceforth, the *Granularity* also restricts the capabilities of diagnostic tools, we cannot uncover source code level faults, without argument knowledge of current running status in sensor node memory. Therefore, if we know the granularity of information, the detectable types of faults can be determined. As a result, the granularity can help with diagnostic tools design and selection.

6 Design Guidance Using Skyline Query

Design a tailored diagnostic tool for network diagnosis is urged, as faults may lead to severe performance degradation of the entire system. The D-vector is introduced to characterize the property of diagnosis approach, so as to guide the design of diagnosis approaches. A straightforward question is, how to guide design or selection of diagnostic tools given different D-vectors? Since diagnostic tool design is deemed as a multi-criteria decision process, we need to consider tradeoffs among different factors for making the decision.

Through the analysis in Section 4 and 5, we derive five principal dimensions that may affect the decision. In this section, we are motivated by skyline query, to select most effective diagnostic tools consider tradeoffs of the five dimensions. With the selected skyline points (SPs), we can get a set of diagnostic approaches that can dominate the other diagnosis approaches, and the SPs can be a guidance for design of diagnostic tools.

6.1 Skyline Query Principles

The skyline query deals with a multi-criteria optimization problem. Given a set of objects p_1, p_2, ..., p_N, the operator returns all objects p_i so that p_i is not dominated by another object p_j. Consider an example of choosing diagnostic tools. The Fig.6 shows two factors that influence the choose of diagnostic tools, the *Degree of Coupling* (x axis) and the *Overhead* (y axis). The most interesting tools are a, k and n, which dominate other approaches on both dimensions.

We consider points in an n-dimensional numeric space $D = (D_1, \ldots, D_n)$. The dominance relation is built on the preferences on attributes $D_1, \ldots, D_n$. Without loss of generality, we as-

sume that, on $D_1, \ldots, D_n$, smaller values are more preferable.

Definition 2 (Dominance). *For two points u and v, u is said to dominate v, denoted by $u \prec v$, if for every dimension D_i $(1 \leq i \leq n)$, $u.D_i \leq v.D_i$, and there exists a dimension D_{i_0} $(1 \leq i_0 \leq n)$ such that $u.D_{i_0} < v.D_{i_0}$.*

Given a set of points S, a point u is a skyline point if there does not exists another point $v \in S$ such that v dominates u. The skyline on S is the set of all skyline points. Henceforth, a D-vector is considered to represent the property of a diagnosis approach such that we can obtain a set of diagnosis approaches.

Therefore, we need to tackle two questions as follows.

1) How do we form the space of potential D-vector points? Does all these points compose a complete set of all diagnosis approaches?

2) How to find skyline points from the space of potential D-vector points?

To answer the first question, we form the space of potential D-vector points based on combine and select existing diagnosis approaches under the influences made by the five principal factors. And it is a set of all potential diagnosis approaches that can be achieved using state-of-the-art techniques. With appropriate combination of different approaches, we can get all potential tools with expected capabilities.

And for the second question, we propose an algorithm, called *NNS*, to search the most effective diagnosis approach based on nearest neighbor query. The detailed *NNS* algorithm are described in Section 6.2.

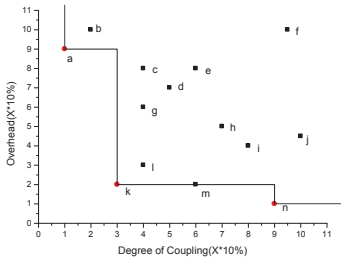


Fig. 6: **Example of skyline query using selection of diagnostic tools on two dimensions.**

6.2 Skyline Query Algorithm

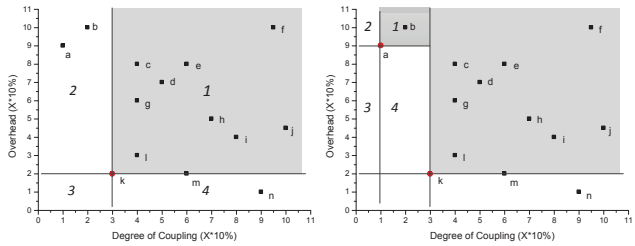
There exists many skyline query algorithms, such as Block Nested Loop (BNL), Divide and Conquer (*D&C*), Bitmap, Index, etc. And we take Nearest Neighbor based Skyline query algorithm (*NNS*), since it is effective and efficient.

NNS uses a nearest neighbor query (such as [22], which is described in Algorithm 1 in the space of diagnosis approaches, to get the minimum distance from the origin of the axes. Without loss of generality, we assume that the distance is computed according to L_1-norm.

Consider the example of choosing diagnostic tools. The first nearest neighbor is k (minimum distance from 0), it shows that k is part of the skyline. Then, all the points in the dominance region of k can be pruned, and the remaining space is divided into two parts based on the coordinates (k_x, k_y): $[0, k_x)$, $[0, \infty)$ and $[0, \infty)$, $[0, k_y)$. It is shown in Fig.7a that first partition contains region 2, 3 and the second partition contains region 3 and 4. The partition has been done after the SP is found and inserted into a *to-do* list. When the *to-do* list is not empty, *NNS* recursively do the same procedure. For example, point a is the nearest neighbor in partition $[0, k_x)$, $[0, \infty)$, which causes the insertion of $[0, a_x)$, $[0, \infty)$ and $[0, k_x)$, $[0, a_y)$ in the *to-do* list (as is described in Fig.7b). If the partition is empty, then we do not further divide and stop *NNS*. Finally, we obtain all the

SPs until all the subdivisions are finished. For N dimensional space, find each SP cost N times recursively executions of *NNS*. The detailed *NNS* is described in Algorithm 2.

A D-vector contains five dimensions, such that we need to do *NNS* five times to obtain a SP, and until all the partitions are empty, then the *NNS* stops.



(a) Obtain diagnostic tool k through nearest neighbor search (b) Get the query result of diagnostic tool a on subdivision

Fig. 7: **Example of *NNS* using two impact factors of diagnostic tools.** (a) obtains tool k as part of the skyline query, while (b) get the query result of tool a.

Algorithm 1: Nearest Neighbor Query

Input: the space of diagnosis approaches $S = (p_1, p_2, \ldots, p_N)$ with d dimensions.
Output: nearest neighbor p_i of S

```
foreach points in S do
    calculate the p_i according to L_1-norm;
    /* min Dist = min ||·||_{L_1}                    */
    partition space into d subdivisions according to
    coordinate of p_i ;
    prune the dominance region of p_i;
    add d subdivisions to to-do list;
    obtain p_i;
end
```

Algorithm 2: *NNS*

Input: nearest neighbors in each partition p_i
Output: SPs in S, which is denoted by Q (q_1, q_2, $\ldots$, q_M).

```
foreach subdivision p_i do
    if the partition is not empty in to-do list then
        do Nearest Neighbor Query;
        add points to Q;
        else
            the to-do list is empty;
        end
    end
    return stop;
end
```

7 Evaluations

7.1 Experimental Setup

We deploy a 61 nodes prototype to evaluate the effectiveness of various diagnosis approaches. The network physical topology is described in Fig.8. The prototype platform contains Telos-B motes with a MSP430 processor and a CC2420 transceiver. We apply the TinyOS 2.1 as our software development platform and have evaluated different diagnosis approaches with information collected from our prototype.

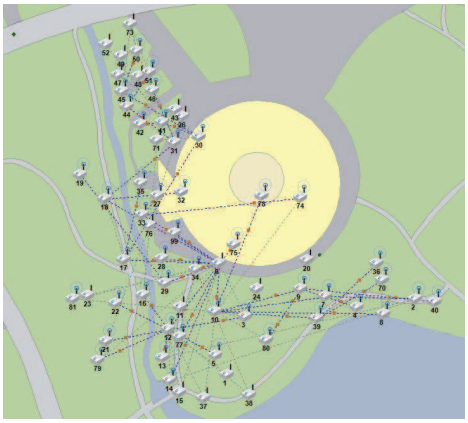


Fig. 8: **Physical topology of our prototype.** Our prototype contains 61 TelosB motes and the physical topology shows the data retrieval process. The blue links represent the ongoing data collection period, and the gray links show once two nodes have been communicated.

We collected three types of packets, and the content of each packet contains sensing information (96 bytes), neighbor information (96 bytes), and network statistical information (101 bytes). We use part of three packets' information for different diagnostic tools, and some of the approaches cannot be implemented in our system, such as the Powertracing [19], since we do not have the power meter supply to set up such an independent diagnostic system. Some typical diagnosis approaches in represent of D-vectors are illustrated in Table 4.

The goal of the evaluation is two-fold: Firstly, we evaluate the D-vector points in represent of typical diagnostic tools to validate the effectiveness of our approach. Secondly, we guide the design of diagnostic tools in GreenOrbs with selected skyline points, which shows the efficiency of D-vectors in design approaches for sensor networks.

7.2 Methodology

We implement diagnostic tools with two different granularity levels. One is out-network information level, we compare two typical diagnostic tools, called PAD [11] and Powertracing [19]. The other is in-network information level, we compare Sympathy [10] and TinyD2 [18] on this level. Since the diagnostic tools of global information level usually contain code level debugging techniques, we cannot employ this change after the network is established. So the results do not contain such tools, and even if the code level debugging tools are used, the system may still face faults coming from interactions of non-faulty components. We set the duty cycle as 10 minutes, so as to take different diagnosis approaches.

The metrics of each diagnostic tool we choose are shown in Table 5. For different diagnostic tools, we compare the D-vectors and the detected fault types. For implementation of skyline query, we generate potential D-vectors based on combinations of existing diagnosis approaches.

7.3 Fault Types and Skyline Query Results

Table 6 shows the detected fault types of typical diagnostic tools. Although TinyD2 is a more efficient diagnostic tool, the *DC* also stays high according to Table 4. That is to say, even though the TinyD2 can detect more failures, the tool still meet a high risk of unreliable diagnosis. Since it heavily relies on the statistical data that collected from sensor nodes.

Fig.9 shows the composition pattern of detected fault types with different diagnostic tools. We can see that Sympathy and TinyD2 receive a similar compositional pattern of the detected fault types. However, PAD is a little different from the former tools. And we cannot derive the results of Powertracing either in our prototype nor in [19]. The results of the fault compositional pattern illustrate that the diagnostic tools which hold similar capability is coming from the same granularity level.

Diagnostic Tools	D-vector (*DC*, Granularity, Overhead, Network Reliability, Tool Reliability)
PAD	(0.3122, 3, 0.05, 0.82, 0.89)
Powertracing	(0, 3, 1, 0.82, -)
Sympathy	(0.6938, 2, 0.12, 0.82, 0.85)
TinyD2	(0.8920, 2, 0.03, 0.82, 0.94)
Clairvoyant	(- , 1, 0.42, 0.82, -)

Table 4: **D-vector points in represent of typical diagnosis approaches.** Due to the implementation constraints, some values are missing.

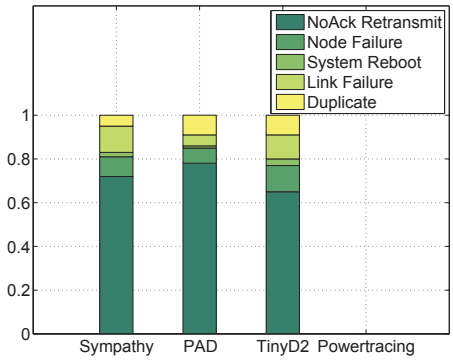


Fig. 9: **Fault compositional patterns of typical diagnostic tools.** The capability of each tool is constrained by the granularity level. Sympathy and TinyD2 hold a similar compositional pattern, since they are on the same granularity level.

The SPs are the diagnosis approaches which contain more delighted features that can be found through combination of existing diagnosis approaches. Table 7 shows the search results of all SPs in the space of potential D-vector points. The Powertracing is easy to validate, since it is an independent diagnosis approach, holds a *DC* of 0, and on the out-network level, no other points can dominate this point. TinyD2 is a low overhead diagnosis approach, since it stores the diagnosis reports on the nodes and transmits to sink if the report is required. On the in-network level, there exists no other points than TinyD2, that can dominate its overhead.

7.4 Impact of Overhead

The average overhead is used in evaluating different diagnostic tools, and the calculated results are described in Table 7. Overhead is an important factor that influences the design of diagnostic tools. Some sensor network users may not find this metric is important, they deem the diagnostic tools must be effective, whatever overhead they take. However, If they take more overhead than the application code itself, is this meaningful to establish such an sensor network application?

We need to choose the most effective tool under an acceptable overhead. As we can see from Powertracing, the overhead to establish such a system is unpredictable, and it is not scalable for large scale applications.

7.5 Design Guidance for GreenOrbs

GreenOrbs is a large scale working system contains up to 500 TelosB motes. In the operational period, we can get three types of packets to

Diagnostic Tools	**Metrics**	
Sympathy	Connectivity Metrics	Routing Table, Neighbor List.
	Flow Metrics	Packet Transmitted, Packet Received, Sink Packet Transmitted, Sink Packet Received, Sink Last Timestamp.
	Node Metrics	Uptime, Good Packet Received, Bad Packet Received.
TinyD2	Connectivity Metrics	Routing Table, Neighbor List, RSSI, ETX, LQI.
	Flow Metrics	Packet Transmitted, Packet Received, Packet Retransmit.
	Node Metrics	Packet Transmitted, Packet Received, Self-Transmitted, Parent Change Times.
PAD	Network Topology	
Powertracing	Current Pattern (measured by power meter)	

Table 5: **Metrics we take to evaluate different diagnostic tools.** The Powertracing is a power meter aided method, we cannot get the power meter supply to implement the evaluation, so we get the results from [19].

Typical Diagnostic Tool	**Fault Types Detected**
Sympathy	Node Crash, Node Reboot, No Neighbors, No Route, Bad Path to Node, Bad Node Transmit, Bad Path to Sink
PAD	Physical Damage, Software Crashes, Network Congestion, Environmental Interference, Application Flaws
Powertracing	Router Failure, Antenna Failure, OS Crash, Power Outage, Short Circuit, System Reboot
TinyD2	Node Crash, Node Reboot, No Neighbors, No Route, Bad Path to Node, Bad Node Transmit, Bad Path to Sink, Parent Change, Duplicate Packet

Table 6: **Fault types detected of various diagnostic tools.** The TinyD2 can detect more failures than other diagnostic tools in our experiment.

Diagnostic Tool	***DC***	**Granularity**	**Avg. Overhead**	**SPs**
Sympathy	0.6938	In-network	0.12	no
TinyD2	0.8920	In-network	0.03	Yes
PAD	0.3122	Out-network	0.05	no
Powertracing	0.0000	Out-network	1.00	Yes

Table 7: **Evaluation of typical diagnostic tools in our prototype.**

Diagnostic Tool	TinyD2
DC	0.5819
Granularity	2
Metrics	Neighbor list, Routing list, Packet send/receive time, # of packets transmitted/retransmitted, Radio on times, # of duplicate packets, # of parent changes
Fault Types Detected	Node failure, Link failure, Routing failure, Ingress drop

Table 8: The chosen methods for GreenOrbs, which can meet the requirements of most effective tools under the granularity level of 2.

judge if there are faults happened during the deploy period. The procedure of how to use our proposed method to choose diagnostic tool is shown in Fig.10.

Firstly, we classified the granularity level of candidate diagnosis approaches into in-network level, and then consider a D-vector is probably as (DC, 2, $Overhead$, R_{sensor}, R_{diag}). Secondly, we find the SP of granularity level equals to 2, and TinyD2 is selected as the best approach that can be achieved. Since the granularity level and SPs are considered, the diagnosis approaches can be selected efficiently. In order to guide the design of efficient approaches, we recommend the diagnostic tool has a D-vector as (DC, 2, 0.03, 0.82, R_{diag}), with $DC \leq DC_{TinyD2}$ and $R_{diag} \geq R_{diag0}$.

8 Conclusion and Future Works

This paper presents D-vectors to characterize various diagnostic tools. Combined with a skyline query algorithm, named *NNS*, we derive a set of diagnosis approaches with dominated features in every dimensions, which can be used as a guidance for future designs. As far as we concern, it is the first work on analyze and quantify the correlation among properties of different diagnostic tools under the employment of real data traces.

With adoption of the D-vector, we can efficiently evaluate diagnosis approaches under the same system settings, and it also brings efficiently design or select diagnostic tools with relatively low overhead. Furthermore, we also take steps on understanding diagnosis in an different view through extensive experiments on design guidance of diagnosis approaches. For future works, we may focus on more practical ways of efficiently design diagnosis approaches in real applications.

References

[1] Mainwaring A, Culler D, Polastre J, Szewczyk R, Anderson J. Wireless sensor networks for habitat monitoring. Proc. of ACM WSNA, 2002.

[2] Tolle G, Polastre J, Szewczyk R, Culler D, Turner N, Tu K, Burgess S, Dawson T, Buonadonna P, Gay D, et al. A macroscope in the redwoods. Proc. of ACM SenSys, 2005.

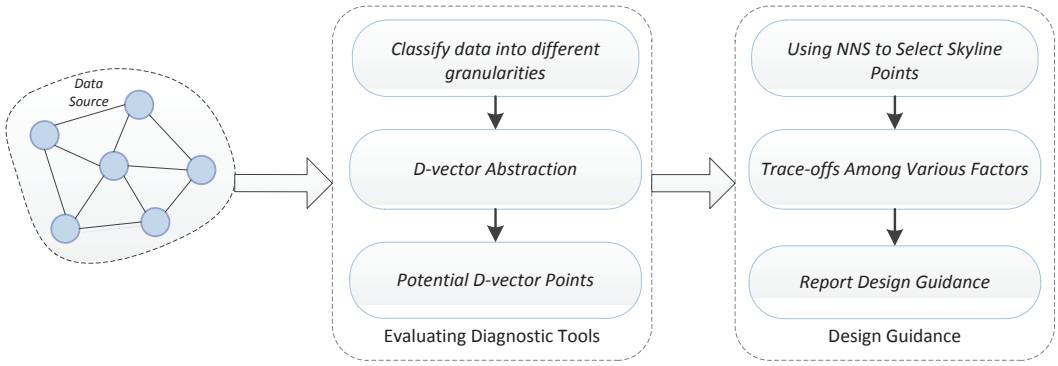


Fig. 10: **Workflow for design guidance.** Design guidance of diagnostic tools using analysis of skyline points from all potential D-vectors.

[3] Mao X, Miao X, He Y, Li X Y, Liu Y. Citysee: urban co_2 monitoring with sensors. Proc. of IEEE INFOCOM, 2012.

[4] Xia M, Dong Y, Xu W, Li X, Lu D. Mc2: Multimode user-centric design of wireless sensor networks for long-term monitoring. *ACM Transactions on Sensor Networks (TOSN)*, 2014, 10(3):52.

[5] Dong W, Liu Y, Chen C, Bu J, Huang C, Zhao Z. R2: Incremental reprogramming using relocatable code in networked embedded systems. *IEEE Transactions on Computers*, 2013, 62(9):1837–1849.

[6] Liu Y, He Y, Li M, Wang J, Liu K, Li X. Does wireless sensor network scale? a measurement study on greenorbs. *IEEE Transactions on Parallel and Distributed Systems*, 2013, 24(10):1983–1993.

[7] Zhang H, Ma H, Li X Y, Tang S. In-network estimation with delay constraints in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 2013, 24(2):368–380.

[8] Cao Q, Abdelzaher T, Stankovic J, Whitehouse K, Luo L. Declarative tracepoints: A programmable and application independent debugging system for wireless sensor networks. Proc. of ACM SenSys, 2008.

[9] Yang J, Soffa M L, Selavo L, Whitehouse K. Clairvoyant: A comprehensive source-level debugger for wireless sensor networks. Proc. of ACM SenSys, 2007.

[10] Ramanathan N, Chang K, Girod L, Kapur R, Kohler E, Estrin D. Sympathy for the sensor network debugger. Proc. of ACM SenSys, 2005.

[11] Liu K, Li M, Liu Y, Li M, Guo Z, Hong F. Passive diagnosis for wireless sensor networks. Proc. of ACM SenSys, 2008.

[12] Khan M M H, Le H K, Ahmadi H, Abdelzaher T F, Han J. Dustminer: Troubleshooting interactive complexity bugs in sensor networks. Proc. of ACM SenSys, 2008.

[13] Li P, Regehr J. T-check: Bug finding for sensor networks. Proc. of ACM/IEEE IPSN, 2010.

[14] Sookoor T, Hnat T, Hooimeijer P, Weimer W, Whitehouse K. Macrodebugging: Global views of distributed program execution. Proc. of ACM SenSys, 2009.

[15] Woo A, Tong T, Culler D. Taming the underlying challenges of reliable multihop routing in sensor networks. Proc. of ACM SenSys, 2003.

[16] Khan M M H, Luo L, Huang C, Abdelzaher T. Snts: Sensor network troubleshooting suite. Proc. of IEEE DCOSS, 2007.

[17] Girod L, Elson J, Cerpa A, Stathopoulos T, Ramanathan N, Estrin D. Emstar: A software environment for developing and deploying wireless sensor networks. Proc. of USENIX Annual Technical Conference, 2004.

[18] Liu K, Ma Q, Zhao X, Liu Y. Self-diagnosis for large scale wireless sensor networks. Proc. of IEEE INFOCOM, 2011.

[19] Khan M, Le H, LeMay M, Moinzadeh P, Wang L, Yang Y, Noh D, Abdelzaher T, Gunter C, Han J, et al. Diagnostic powertracing for sensor node failure analysis. Proc. of ACM/IEEE IPSN, 2010.

[20] Borzsony S, Kossmann D, Stocker K. The skyline operator. Proc. of IEEE ICDE, 2001.

[21] Tan K, Eng P, Ooi B. Efficient progressive skyline computation. Proc. of VLDB, 2001.

[22] Hjaltason G, Samet H. Distance browsing in spatial databases. *ACM Transactions on Database Systems*, 1999, 24(2):265–318.

[23] Papadias D, Tao Y, Fu G, Seeger B. An optimal and progressive algorithm for skyline queries. Proc. of ACM SIGMOD, 2003.

[24] Mo L, He Y, Liu Y, Zhao J, Tang S, Li X Y, Dai G. Canopy closure estimates with greenorbs: Sustainable sensing in the forest. Proc. of ACM SenSys, 2009.

[25] Stamatis D. Failure mode and effect analysis: Fmea from theory to execution. *Productivity Press*, 2003.

[26] Werner-Allen G, Lorincz K, Johnson J, Lees J, Welsh M. Fidelity and yield in a volcano monitoring sensor network. Proc. of USENIX OSDI, 2006.

Rui Li received his BS degree in Department of Applied Mathematics from Xidian University in 2006. He is currently a master-doctoral program graduate student at Xi' an Jiaotong University, and has been qualified as a PhD candidate in Department of Computer Science and Technology. His main research interests include wireless ad hoc and sensor networks, and pervasive computing. He is a student member of ACM and CCF.

Dr. Kebin Liu received his BS degree in Department of Computer Science from Tongji University, and an MS and PhD degree in Shanghai Jiaotong University, China. He is currently an assistant research fellow in Tsinghua University. His research interests include Wireless Sensor Networks, Pervasive Computing, and Network Diagnosis. He is a member of IEEE and a member of ACM.

Dr. Xiang-Yang Li is a professor at the Illinois Institute of Technology. He holds EMC-Endowed Visiting Chair Professorship at Tsinghua University. He currently is distinguished visiting professor at Xi' an Jiaotong University and

University of Science and Technology of China. He is a recipient of China NSF Outstanding Overseas Young Researcher (B). Dr. Li received MS (2000) and PhD (2001) degree at Department of Computer Science from University of Illinois at Urbana-Champaign, a Bachelor degree at Department of Computer Science and a Bachelor degree at Department of Business Management from Tsinghua University, P.R. China, both in 1995. His research interests include mobile computing, cyber physical systems, wireless networks, security and privacy, and algorithms. Dr. Li is an editor of several journals, including IEEE Transaction on Parallel and Distributed Systems, IEEE Transaction on Mobile Computing. He has served many international conferences in various capacities. He is a senior member of IEEE and a member of ACM.

Dr. Yuan He received his BE degree in Department of Computer Science and Technology from University of Science and Technology of China (USTC) , his ME degree in Institute of Software, Chinese Academy of Sciences, and his PhD degree in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He is currently a lecturer in Tsinghua University. His research interests include sensor networks, peer-to-peer computing, and pervasive computing. He is a member of the IEEE and ACM.

Dr. Wei Xi received the B.S. degree from Xidian University, Xi' an, China, in 2006, and the Ph.D. degree from Xi'an Jiaotong University, Xi'an, China, in 2014, both in computer science. He has been a postdoctoral research fellow at Xi'an Jiaotong University. His main research interests include wireless networks, smart sensing, and mobile computing. Dr. Xi is a member ACM, IEEE, and CCF.

Zhi Wang is currently a doctoral program graduate student at Xi' an Jiaotong University, and has been qualified as a PhD candidate in Department of Computer Science and Technology. His main research interests include wireless ad hoc and sensor networks, smart space and pervasive computing.

Dr. Jizhong Zhao received the B.S. and M.S. degrees in mathematics and Ph.D. degree in computer science with a focus on distributed systems from Xi' an Jiaotong University, Xi' an, China, in 1992, 1995, and 2001, respectively. He is a Professor with the Computer Science and Technology Department, Xi' an Jiaotong University. His research interests include computer software, pervasive computing, distributed systems, network security. Dr. Zhao is a member of ACM, IEEE and CCF.

Dr. Meng Wan is an associate professor at Center for Science and Technology Development, Ministry of Education, Beijing. He received his Ph.D. degree from Wuhan University

in 2008, and his M.S. degree from Central University of Finance and Economics in 2000. His research interests include computer network architecture, network and systems management, science and technology management, system engineering, etc. He is currently serving as the division director of Department of Network and Information, Center for Science and Technology Development，Ministry of Education. He is the associate editor of Sciencepaper Online. He is a member of the ACM and IEEE.