Li R, Liu KB, Li X *et al.* Assessing diagnosis approaches for wireless sensor networks: Concepts and analysis. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 29(5): 887–900 Sept. 2014. DOI 10.1007/s11390-014-1476-z

# Assessing Diagnosis Approaches for Wireless Sensor Networks: Concepts and Analysis

Rui Li[1,2] (李 瑞), Ke-Bin Liu[3,4] (刘克彬), *Member*, *ACM*, *IEEE*
Xiangyang Li[2,5] (李向阳), *Senior Member*, *IEEE*, *Member*, *ACM*, Yuan He[3,4] (何 源), *Member*, *ACM*, *IEEE*
Wei Xi[2,*] (惠 维), *Member*, *CCF*, *ACM*, *IEEE*, Zhi Wang[2] (王 志)
Ji-Zhong Zhao[2] (赵季中), *Member*, *CCF*, *ACM*, *IEEE*, and Meng Wan[6] (万 猛), *Member*, *ACM*, *IEEE*

[1] *Institute of Software Engineering, Xidian University, Xi'an 710071, China*

[2] *School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China*

[3] *School of Software, Tsinghua University, Beijing 100084, China*

[4] *Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China*

[5] *Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, U.S.A.*

[6] *Center for Science and Technology Development, Ministry of Education, Beijing 100080, China*

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

Received February 27, 2014; revised June 27, 2014.

**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 in measuring the effectiveness of their diagnosis 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 applications. 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 the proliferation of the sensor network applications in the wild[1-4], this trend has accelerated as the diagnosis is even harder caused by complex topography and dynamic environmental factors[5-7]. 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. These approaches work well in a reliable network. They, however, may experience significant performance degradation when more failures occur. For example, some diagnostic tools deliver diagnosis information

Regular Paper

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61190110, 61325013, 61103187, 61170213, 61228202, 61170216, and 61422207, the National Basic Research 973 Program of China under Grant No. 2014CB347800, the Natural Science Foundation of USA under Grant Nos. CNS-0832120, CNS-1035894, ECCS-1247944, and 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 of China under Project No. 20130201120016.

*Corresponding Author

using initial network protocols, thus network failures lead to incomplete information for the diagnosis engine and thus inaccurate judgments. 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 instance, 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 the 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 corresponding 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 it is the first step towards better understanding of 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 studies are illustrated in Section 2 and motivations of this work in Section 3. In Section 4, we define the 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 the five factors. Section 6 contains the skyline query algorithm 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 work.

## 2 Related Work

Most debugging tools[13-14] target finding software bugs in sensor nodes. They are considered as pre-deployment diagnosis. Cao *et al.*[8] reported a debugging system, which could automatically watch program states to detect bugs. Clairvoyant[9] enables the code-level debugging for WSNs which allow users to remotely execute debugging commands such as step and breakpoint. Debugging tools are effective at finding network failures. However, they can bring relatively high overhead in term of massive control messages.

Operating period diagnosis attracts many efforts. MintRoute[15] visualizes the network topology by collecting neighbor tables from sensor nodes. SNTS[16] constructs network infrastructure for logging and retrieving state information at runtime and EmStar[17] supports the 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 combines the view of the node itself with the diagnosis process.

Post-deployment diagnosis is usually log-based analysis. For example, 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 does not need network statistical data. And the diagnostic tool is loosely coupled with the network.

Skyline query is considered as a promising technique in multi-criteria optimization process in database community. The intuitive method of processing skyline computation, such as BNL[20], 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 it is efficient to implement.

## 3 Motivation

To reveal the factors that influence the diagnosis performance, we introduce some basic observations in the 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, 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.

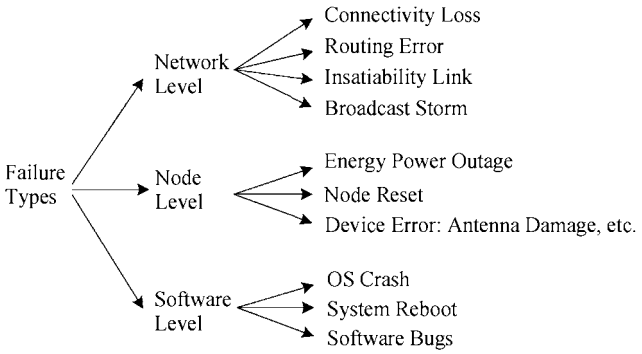



Fig.1. Three main types of failures that appear in the GreenOrbs system.

*Observation* 1. We can distinguish node failure and network failure by data value changes and data quantity changes. Node failure can only influence the data values, but network failure can result in the change of the packet's quantity. As shown in Fig.2, the number of transmissions varies from each other on different nodes, which may indicate the appearance of the node failures in the network. However, Fig.3 illustrates the packets received by sink during 48 sample points, where the severe network failures occur at 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 leading to three types of failures. However, the

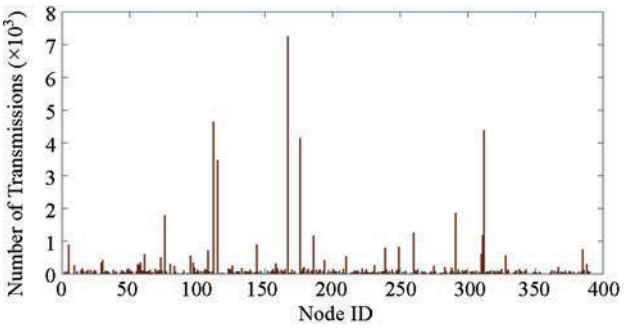



Fig.2. Number of transmissions on different sensor nodes in the GreenOrbs system.

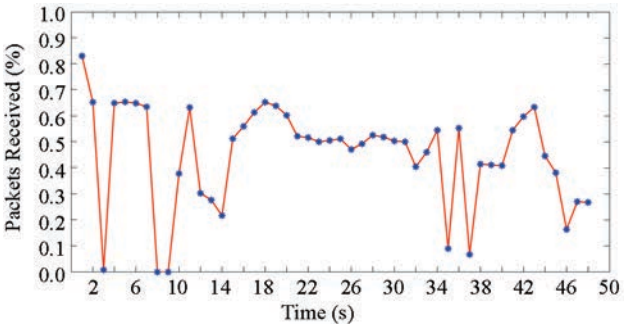



Fig.3. Packets received by 48 sample points during five days.

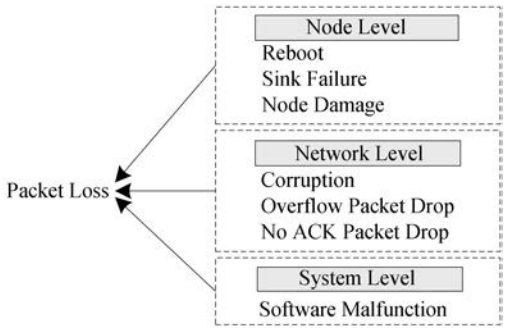



Fig.4. Causes of packet loss.

combination of several necessary conditions may form the sufficient condition of a certain failure.

According to the above observations, we discuss the main factors that influence the diagnosis results in WSNs that come from the intrinsic nature of different diagnosis approaches. We classify the root causes that can influence the diagnosis results into five main factors. The five factors are inner-correlated with each other as shown in Fig.5. And the figure is established by cause-and-effect diagram[25]. In the figure, smaller arrows connect the sub-causes to major causes, and reflect the inner-correlations among five factors.

The network reliability and the diagnostic tool reliability influence 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. Consequently, we need an

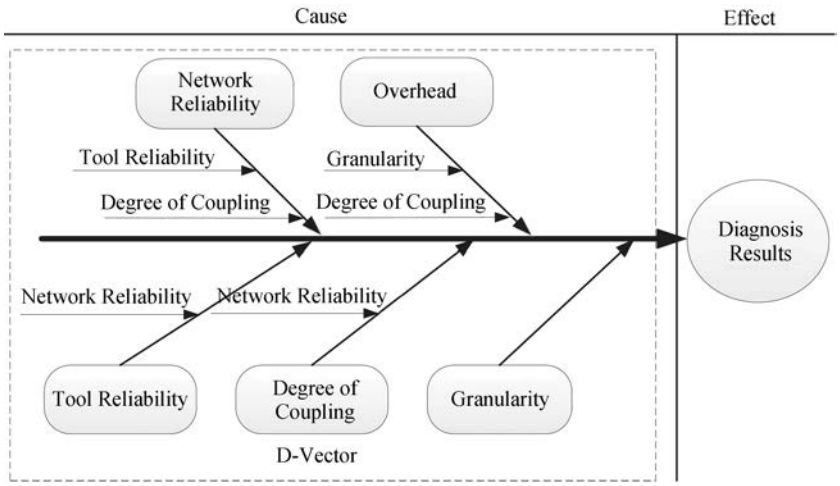



Fig.5. Cause-and-effect diagram of diagnosis results.

independent diagnostic tool and the diagnosis approach should not collect more information from the network itself. Fig.5 elaborates that the tool reliability and the degree of coupling both influence the network reliability. However, the tool reliability is only affected by the network reliability.

Since the effective diagnosis approaches are restricted by the collected information types, the effective diagnosis information overhead is affected by granularity and degree of coupling.

The degree of coupling is affected by the network reliability, and 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. Therefore, 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 the other factors. Since the granularity determines the capability of diagnosis approaches, with more fine-grained granularity of diagnostic information, the diagnosis results can be more accurate.

## 4 Quantification of D-Vector

The five main factors may affect the diagnosis result of a diagnosis approach, and they are correlated with each other. In this section, we aim to quantify the main factors so as to indicate various diagnosis approaches effectively. Some symbols used later are defined in Table 1.

### 4.1 Definition of 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 every D-vector corresponds to an existing diagnosis approach.

**Table 1.** Useful Notations

| Symbol | Definition |
|---|---|
| Degree of coupling | Amount of effective information a diagnostic tool needs to collect from the sensor network |
| $R_{\text{sensor}}$ | Network reliability that can be measured by network yield |
| $R_{\text{diag}}$ | Diagnostic tool reliability which depends on the true positive and true negative diagnosis results |
| Granularity | 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 |
| $N_d = \{N_{d1}, N_{d2}, \ldots, N_{dk}\}$ | Destination of all information 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, $R_{\text{sensor}}$, $R_{\text{diag}}$, granularity, overhead) quintuple to indicate a specific 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 the 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 DC 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 can a diagnostic tool get from the sensor network? And how much overhead does a diagnostic tool take? To share a unified criterion, we need to find a quantitative metric for all the tools and calculate the effective information that a diagnostic tool needs.

We introduce mutual information to measure how much information a diagnostic tool collects, even though different diagnostic tools collect different types of information and 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 noises, the information can be calculated as the entropy of the source information, which is defined by:

$$H(X) = -\sum\nolimits_i P(x_i)\log P(x_i). \tag{1}$$

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

$$I(X;Y) = H(X) - H(X|Y),$$

where $H(X|Y)$ is conditional entropy, which can be illustrated as:

$$H(X|Y) = -\sum\nolimits_i \sum\nolimits_j P(x_i, y_j)\log P(x_i|y_j).$$

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

• Non-negative property, i.e., $I(X;Y) \geqslant 0$, where the equality holds if and only if the sending information and the receiving information are independent;

• Mutual information is no larger than the source information entropy, i.e., $I(X;Y) \leqslant H(X)$. When the noises do not exist in the channel, $I(X;Y)$ equals $H(X)$ in numerical value;

• Symmetry property, which means the mutual information is equal to the source information and the destination information.

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

$$H(X) = -\int P(x)\log P(x)\mathrm{d}x,$$

$$I(X;Y) = \iint P(x,y)\log\frac{P(x,y)}{P(x)P(y)}\mathrm{d}x\mathrm{d}y.$$

**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. All the destination information 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 (1), we can describe the feature entropy of the above three statistical information sets as follows:

$$H(N_s) = -\sum_{i=1}^{m}\Big(\frac{N_{si}}{S_1}\Big)\log\Big(\frac{N_{si}}{S_1}\Big), \tag{2}$$

$$H(N_d) = -\sum_{i=1}^{k}\Big(\frac{N_{di}}{S_2}\Big)\log\Big(\frac{N_{di}}{S_2}\Big), \tag{3}$$

$$H(N_p) = -\sum_{i=1}^{l}\Big(\frac{N_{pi}}{S_3}\Big)\log\Big(\frac{N_{pi}}{S_3}\Big), \tag{4}$$

where $S_1 = \sum_{i=1}^{m} n_{si}$, $S_2 = \sum_{i=1}^{k} n_{di}$, and $S_3 = \sum_{i=1}^{l} n_{pi}$ illustrate the collected information traffic during the observation period. (2)∼(4) 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 of $[0, \log N]$. When the value of $n_i$ is approximate to 0, different $n_i$ may experience the same event. When the distribution of the feature is maximized, the value is approximate to $\log N$, i.e., $n_1 = n_2 = \cdots = n_i$.

Therefore, we can calculate the distribution of entropy for each diagnostic tool in order to get DC. We use the feature entropy of each diagnostic tool to quantify DC, when the feature entropy is large. That means the diagnostic tool needs more effective information from sensor networks. Then DC can be calculated as follows:

$$\mathcal{D} = \frac{H(N_p) - H(N_p|N_d)}{H(N_s) - H(N_s|N_d)},$$

where $\mathcal{D}$ is the ratio of the effective input information for the diagnostic tool to all information collected from the sensor network.

### 4.3 Network Reliability

The network reliability is one of the major factors that influence the selection of diagnosis approaches. 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 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:

$$Yield_i = \frac{\text{Number of packets received by sink from } i \text{ during } p}{\text{Number of packets sent by } i \text{ during } p},$$

where $i$ is the node ID and $p$ is the observation period. The network yield is calculated by:

$$Yield = \frac{\text{Number of packets received by sink during } p}{\text{Number of packets sent by all nodes during } p}.$$

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 $R_{\text{sensor}}$, is influenced by the network yield and the node yield. The threshold $R_{\text{sensor0}}$ is specified in different sensor network applications as tailored to diverse purposes.

In most sensor networks, the value of $R_{\text{sensor}}$ 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 where 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 $R_{\text{sensor}}$. $R_{\text{sensor}}$ can be calculated by:

$$R_{\text{sensor}} = \frac{1}{n} \sum_{i=1}^{n} Yield_i,$$

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 the information from sniffer nodes that can overhear the network status. Whatever methods that diagnosis approaches adopt, they all need information as input to unveil the faults occurred in sensor networks.

There are three main approaches in diagnostic tools. The first approach is the 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 find at where the faults locate. The third and the most frequent way, to deal with performance degradation in sensor networks, is that the administrators monitor the network and use domain knowledge to judge if there were faults happened in sensor networks. Unveiling fault manually 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 adopts in diagnosis, which is shown in Table 2.

**Table 2.** Diagnostic Tools and the Types of Faults They Can Detect

| Diagnostic Tool | Fault Type | Information Type | Recovery Method |
|---|---|---|---|
| PAD, Powertracing | Node, link | Out-network | Node reboot, node replacement |
| Sympathy, TinyD2 | State | In-network | Network or node reconfiguration |
| Declarative Tracepoints, Clairvoyant | Source code | Global | Code correction or rewrite |

Nevertheless, we need to illustrate the tool reliability. For instance, Sympathy and Declarative Trace-

points are detecting different faults. Sympathy may be of a high reliability using the information collected from the sensor networks. However, Declarative 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. We just simply choose the reliability of each tool under the same condition of different granularities and the reliability is calculated by:

$$R_{\text{diag}} = \frac{t\_pos + t\_neg}{pos + neg}, \tag{5}$$

where $t\_pos$ is the number of true positives in diagnosis results, and $t\_neg$ is the number of true negatives. $pos$ is the number of positive results of the diagnostic tools and $neg$ is the number of negative results of the diagnostic tools. (5) 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 and 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 granularities they have, we obtain the results in Table 3.

**Table 3.** Granularity of Each Diagnostic Tool Determined by the Collected Information Types

| Diagnostic Tool | Granularity | DC |
|---|---|---|
| PAD, Powertracing① | Out-network (3) | Relative high |
| Sympathy, TinyD2 | In-network (2) | Relative fair |
| Declarative Tracepoints, Clairvoyant② | Global (1) | Relative low |

The overhead is defined as the amount of diagnostic packets. Nevertheless, the cost of diagnosis approach is considered as the traffic overhead plus the hardware cost. As most diagnosis approaches are tightly coupled with sensor network and do not need any extra hardware, we define the overhead only contains traffic overhead. If a diagnostic tool contains extra hardware equipment, we ignore the traffic overhead and simply set the traffic overhead to 1.

In this study, we define the overhead as the result calculated by the diagnostic information traffic flow over the whole traffic flow 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 subsection, 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 and cause relatively high overhead. On the contrary, if the overhead is relatively high, we cannot derive that 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 the 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 lower DC than other diagnostic tools. That means PAD slightly relies

①Powertracing holds a DC of 0, since the diagnostic tool is an independent system and it does not need statistical data from sensor nodes.

②Some 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.

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 characterizing the properties of diagnosis approaches, and of real usability for the 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 literatures focus on designing and developing diagnostic tools, guiding design of diagnostic tools is of importance to all applications.

### 5.2 Constraints of DC and Granularity

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

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

Constraint 1 is easy to validate. If diagnostic tools rely more on network information, the diagnosis results could become more uncertain. Since the diagnostic information is not always available due to the 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 becomes 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.*

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 aims at unique failure due to information type constraints. Secondly, the granularity can help the 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, and 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 the design and selection of diagnostic tools.

## 6 Design Guidance Using Skyline Query

Designing a tailored diagnostic tool for network diagnosis is urgent, as faults may lead to severe performance degradation of the entire system. The D-vector is introduced to characterize the properties of diagnosis approaches, so as to guide the design of diagnosis approaches. A straightforward question is how to guide design or selection of diagnostic tools by given different D-vector points. Since the design of diagnostic tools is deemed as a multi-criteria decision process, we need to consider trade-offs among different factors to make the decision.

Through the analysis in Sections 4 and 5, we derive five principal dimensions that may affect the decision. In this section, we are motivated by the skyline query, to select the most effective diagnostic tools considering trade-offs 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 designing diagnostic tools.

### 6.1 Skyline Query Principles

The skyline query deals with a multi-criteria optimization problem. Given a set of objects $p_1, p_2, \ldots, p_N$, 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. 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.

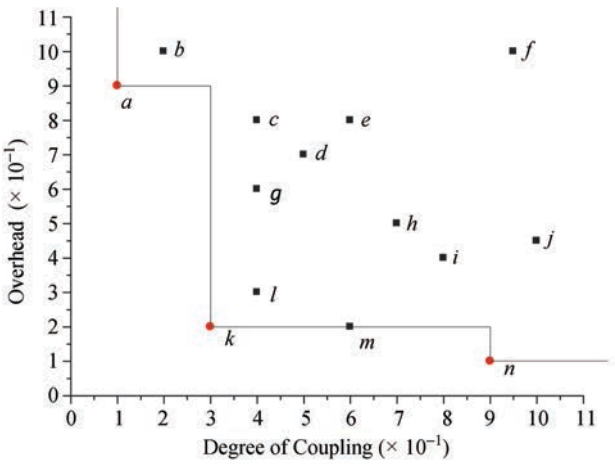



Fig.6. Example of skyline query using selection of diagnostic tools on two dimensions.

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 assume 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 $D_i$ $(1 \leqslant i \leqslant n)$, $u.D_i \leqslant v.D_i$, and there exists a dimension $D_{i_0}$ $(1 \leqslant i_0 \leqslant n)$ 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 exist 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 properties 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? Do 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 combining and selecting 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 by using state-of-the-art techniques. With the 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 is described in Subsection 6.2.

### 6.2 Skyline Query Algorithm

There exist many skyline query algorithms, such as Block Nested Loop (BNL), Divide and Conquer (D & C), Bitmap, Index. And we take the 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.

Considering the example of choosing diagnostic tools, the first chosen point is $k$, where point $k$ $(k_x, k_y)$ is the nearest neighbor of point (0, 0). According to Algorithm 1, $k$ is a skyline point. 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.7(a) that the first partition contains

**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 $S$

```
1 foreach point p_i in S do
2     Calculate p_i according to L1-norm;
3     /* min Dist = min ‖·‖_{L1} */
4     Partition space into d subdivisions according to the
      coordinate of p_i;
5     Prune the dominance region of p_i;
6     Add d subdivisions to to-do list;
7     Obtain p_i;
8 end
```

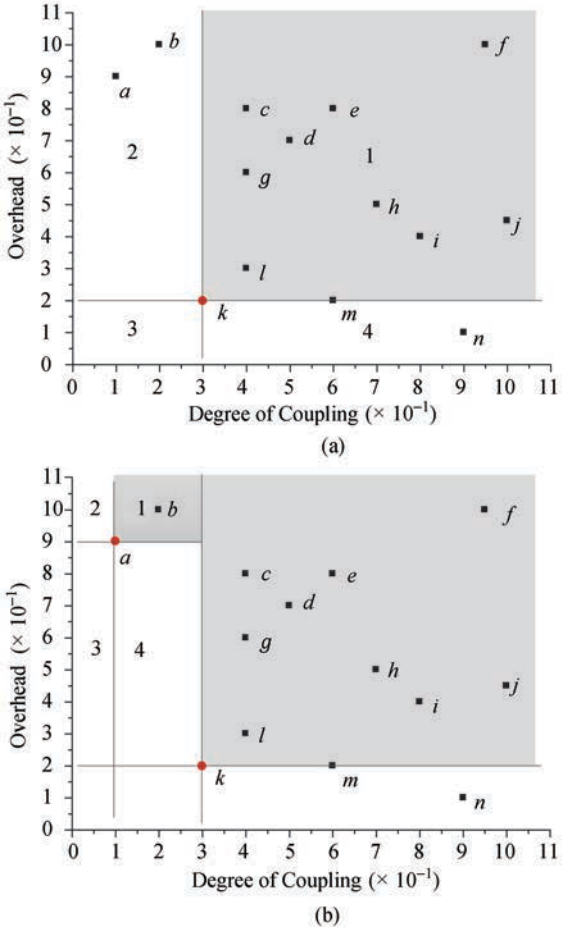



Fig.7. Example of NNS using two impact factors of diagnostic tools. (a) obtains diagnostic tool $k$ as part of the skyline query, while (b) gets the query result of point $a$.

regions 2 and 3 and the second partition contains regions 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 does 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.7(b)). 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 an $N$-dimensional space, finding each SP costs $N$ times recursive executions of NNS. The detailed NNS is described in Algorithm 2.

**Algorithm 2.** NNS

**Input:** nearest neighbors in each partition $p_i$

**Output:** SPs in $S$, which is denoted by $Q$ $(q_1, q_2, \ldots, q_M)$

```
1  foreach subdivision p_i do
2      if the partition is not empty in to-do list then
3          do Nearest Neighbor Query;
4          Add points to Q;
5          else
6              The to-do list is empty;
7          end
8      end
9      return stop;
10 end
```

The D-vector contains five dimensions, such that we need to do NNS five times to obtain an SP. NNS stops until all of the partitions become empty.

## 7 Evaluations

### 7.1 Experimental Setup

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

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. Some of the approaches cannot be implemented in our system, such as 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-vector points 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.

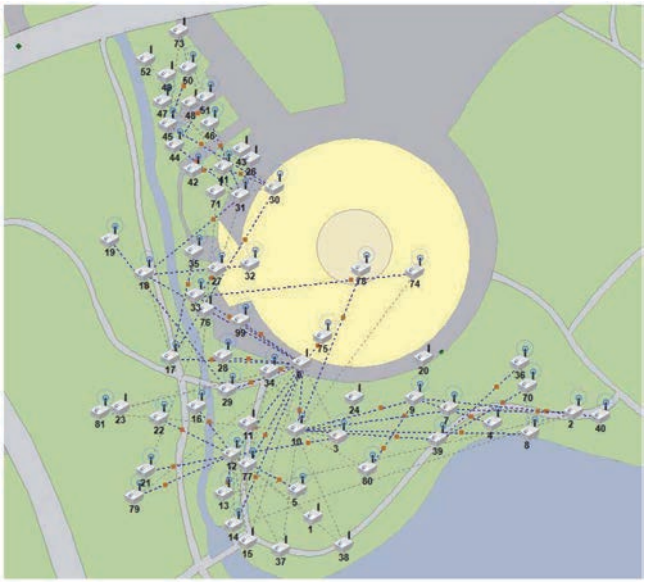


Fig.8. Physical topology of our prototype. Our prototype contains 61 TelosB motes and the physical topology shows the data retrieval process. The blue links represent the ongoing data collection period, and the gray links show once two nodes have been communicated.

**Table 4.** D-Vector Points in Represent of Typical Diagnosis Approaches

| Diagnostic Tool | D-Vector (DC, Granularity, Overhead, Network Reliability, Tool Reliability) |
|---|---|
| PAD | (0.312 2, 3, 0.05, 0.82, 0.89) |
| Powertracing | (0, 3, 1, 0.82, –) |
| Sympathy | (0.693 8, 2, 0.12, 0.82, 0.85) |
| TinyD2 | (0.892 0, 2, 0.03, 0.82, 0.94) |
| Clairvoyant | (– , 1, 0.42, 0.82, –) |

Note: Due to the implementation constraints, some values are missing.

### 7.2 Methodology

We implement diagnostic tools in two different granularity levels. One is out-network information level, and we compare two typical diagnostic tools, called PAD[11] and Powertracing[19]. The other is in-network information level, and we compare Sympathy[10] and TinyD2[18] in this level. Since the diagnostic tools of global information level usually contain code level debugging tools, we cannot employ this change after the network is established. Thus 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 the 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 com-

**Table 5.** Metrics Taken to Evaluate Different Diagnostic Tools

| Diagnostic Tool | 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 time |
| PAD | Network topology | |
| Powertracing | Current pattern (measured by power meter) | |

Note: Powertracing is a power meter aided method. We cannot acquire the power meter supply to implement the evaluation, therefore we get the results from [19].

pare the D-vector points and the detected fault types. For the implementation of skyline query, we generate potential D-vector points based on the combination 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, DC also stays high according to Table 4. That is to say, even though TinyD2 can detect more failures, the tool still meets a high risk of unreliable diagnosis. Since it heavily relies on the statistical data collected from the sensor nodes.

**Table 6.** Fault Types Detected by Various Diagnostic Tools

| 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 |

Note: TinyD2 can detect more failures than other diagnostic tools (i.e., Sympathy, PAD, Powertracing) in our experiment.

Fig.9 shows the composition pattern of detected fault types with different diagnostic tools. The capability of each tool is constrained by the granularity level. We can see that Sympathy and TinyD2 receive a similar compositional pattern of the detected fault types, since they are on the same granularity level. However, PAD is a little different from Sympathy. And we cannot derive the results of Powertracing neither in our prototype nor in [19]. The results of the fault compositional pattern illustrate that the diagnostic tools which hold similar capability are coming from the same granularity level.

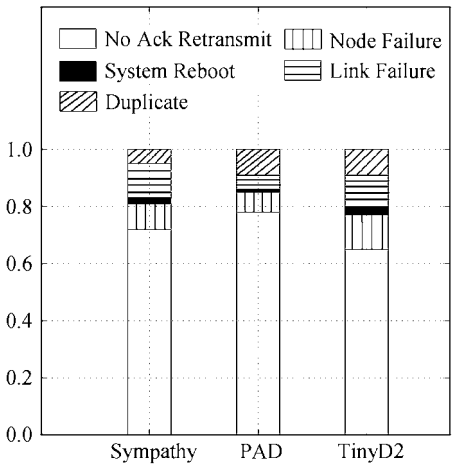



Fig.9. Fault compositional patterns of typical diagnostic tools.

SPs are the diagnosis approaches which contain more delighted features that can be found through the combination of existing diagnosis approaches. Table 7 shows the search results of all SPs in the space of potential D-vector points. Powertracing is easy to validate, since it is an independent diagnosis approach. It 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

**Table 7.** Evaluation of Typical Diagnostic Tools in Our Prototype

| Diagnostic Tool | DC | Granularity | Avg. Overhead | SPs |
|---|---|---|---|---|
| Sympathy | 0.693 8 | In-network | 0.12 | No |
| TinyD2 | 0.892 0 | In-network | 0.03 | Yes |
| PAD | 0.312 2 | Out-network | 0.05 | No |
| Powertracing | 0.000 0 | Out-network | 1.00 | Yes |

the nodes and then transmits them to sink if the diagnosis reports are required. On the in-network level, there exist no other points than TinyD2, which can dominate its overhead.

### 7.4 Impact of Overhead

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

### 7.5 Design Guidance for GreenOrbs

GreenOrbs is a large-scale working system which contains up to 500 TelosB motes. In the operational period, we can get three types of packets to judge if there are faults happened during the deployment. The procedure of how to use our proposed method to choose diagnostic tool is shown in Fig.10.

Firstly, we classify the granularity level of candidate diagnosis approaches into in-network level, and then consider a D-vector is ($\mathcal{D}$, 2, overhead, $R_{\text{sensor}}$, $R_{\text{diag}}$). Secondly, we find that the SP of the granularity is 2, and TinyD2 is selected as the best approach that can be achieved. Table 8 describes the property of TinyD2. Since the granularity and the 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 ($\mathcal{D}$, 2, 0.03, 0.82, $R_{\text{diag}}$), with $\mathcal{D} \leqslant DC_{\text{TinyD2}}$ and $R_{\text{diag}} \geqslant R_{\text{diag0}}$.

**Table 8.** Chosen Approach for GreenOrbs Meeting Requirements of Most Effective Tools under the Granularity of 2

| | |
|---|---|
| Diagnostic tool | TinyD2 |
| DC | 0.581 9 |
| Granularity | 2 |
| Metrics | Neighbor list, routing list, packet send/receive time, number of packets transmitted/retransmitted, radio on time, number of duplicate packets, number of parent changes |
| Fault types detected | Node failure, link failure, routing failure, ingress drop |

## 8 Conclusions

This paper presents the D-vector to characterize various diagnostic tools. Working with a skyline query algorithm, named NNS, we derived a set of diagnosis approaches with dominated features in every dimension, which can be used as a guidance for future designs. As far as we are concerned, it is the first effort on analyzing and quantifying the correlation among properties of different diagnostic tools under the adoption of real data trace.

With adoption of the D-vector, we can efficiently evaluate diagnosis approaches under the same system settings, and it can also help users select a proper diagnostic tool with low overhead. Furthermore, we also took steps in understanding diagnosis in a different view through extensive experiments on the design guidance of diagnosis approaches. For future work, we may focus on more practical ways of efficiently designing diagnosis approaches in real applications.

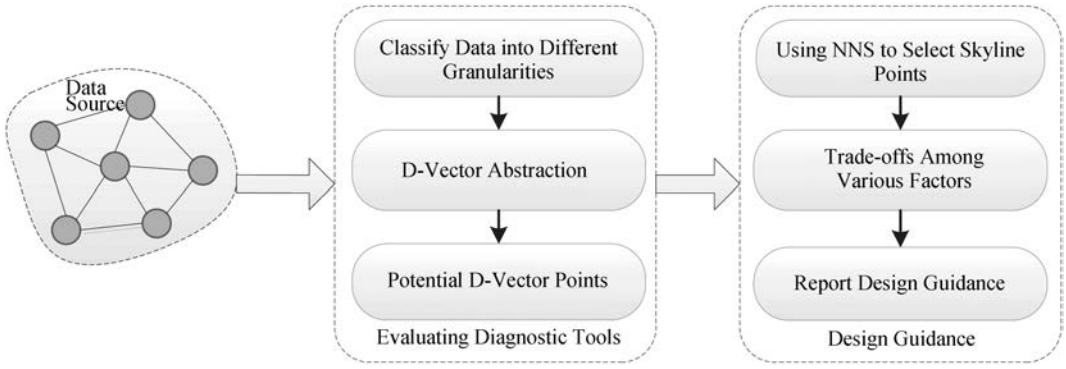



Fig.10. Workflow for design guidance of diagnostic tools using analysis of skyline points from all potential D-vectors.

## References


[1] Mainwaring A, Culler D, Polastre J *et al.* Wireless sensor networks for habitat monitoring. In *Proc. the 1st ACM WSNA*, Sept. 2002, pp.88-97.

[2] Tolle G, Polastre J, Szewczyk R *et al.* A macroscope in the redwoods. In *Proc. the 3rd ACM SenSys*, Nov. 2005, pp.51-63.

[3] Mao X, Miao X, He Y *et al.* Citysee: Urban $CO_2$ monitoring with sensors. In *Proc. the 31st IEEE INFOCOM*, March 2012, pp.1611-1619.

[4] Xia M, Dong Y, Xu W *et al.* $MC^2$: Multi-mode user-centric design of wireless sensor networks for long-term monitoring. *ACM Transactions on Sensor Networks*, 2014, 10(3): Article No.52.

[5] Dong W, Liu Y, Chen C *et al.* 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 *et al.* 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 *et al.* 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 *et al.* Declarative tracepoints: A programmable and application independent debugging system for wireless sensor networks. In *Proc. the 6th ACM SenSys*, November 2008, pp.85-98.

[9] Yang J, Soffa M L, Selavo L *et al.* Clairvoyant: A comprehensive source-level debugger for wireless sensor networks. In *Proc. the 5th ACM SenSys*, November 2007, pp.189-203.

[10] Ramanathan N, Chang K, Kapur R *et al.* Sympathy for the sensor network debugger. In *Proc. the 3rd ACM SenSys*, November 2005, pp.255-267.

[11] Liu Y, Liu K, Li M. Passive diagnosis for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 2010, 18(4): 1132-1144.

[12] Khan M M H, Le H K, Ahmadi H *et al.* Dustminer: Troubleshooting interactive complexity bugs in sensor networks. In *Proc. the 6th ACM SenSys*, November 2008, pp.99-112.

[13] Li P, Regehr J. T-check: Bug finding for sensor networks. In *Proc. the 9th ACM/IEEE IPSN*, April 2010, pp.174-185.

[14] Sookoor T, Hnat T, Hooimeijer P *et al.* Macrodebugging: Global views of distributed program execution. In *Proc. the 7th ACM SenSys*, November 2009, pp.141-154.

[15] Woo A, Tong T, Culler D. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proc. the 1st ACM SenSys*, November 2003, pp.14-27.

[16] Khan M M H, Luo L, Huang C, Abdelzaher T. SNTS: Sensor network troubleshooting suite. In *Proc. the 3rd IEEE DCOSS*, June 2007, pp.142-157.

[17] Girod L, Elson J, Cerpa A *et al.* EmStar: A software environment for developing and deploying wireless sensor networks. In *Proc. the USENIX Annual Technical Conference*, June 27-July 2, 2004, pp.283-296.

[18] Liu K, Ma Q, Zhao X, Liu Y. Self-diagnosis for large scale wireless sensor networks. In *Proc. the 30th IEEE INFOCOM*, April 2011, pp.1539-1547.

[19] Khan M M H, Le H K, LeMay M *et al.* Diagnostic powertracing for sensor node failure analysis. In *Proc. the 9th ACM/IEEE IPSN*, April 2010, pp.117-128.

[20] Börzsöny S, Kossmann D, Stocker K. The skyline operator. In *Proc. the 17th IEEE ICDE*, April 2001, pp.421-430.

[21] Tan K L, Eng P K, Ooi B C. Efficient progressive skyline computation. In *Proc. the 27th VLDB*, Sept. 2001, pp.301-310.

[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. In *Proc. the 2003 ACM SIGMOD*, June 2003, pp.467-478.

[24] Mo L, He Y, Liu Y *et al.* Canopy closure estimates with GreenOrbs: Sustainable sensing in the forest. In *Proc. the 7th ACM SenSys*, November 2009, pp.99-112.

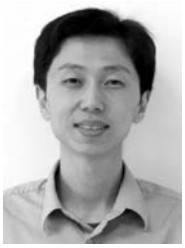
[25] Stamatis D. Failure Mode and Effect Analysis: FMEA from Theory to Execution. WI, USA: ASQ Quality Press, 2003.

[26] Werner-Allen G, Lorincz K, Johnson J, Lees J, Welsh M. Fidelity and yield in a volcano monitoring sensor network. In *Proc. the 7th USENIX OSDI*, November 2006, pp.381-396.


**Rui Li** received his B.S. degree in mathematics from the Department of Applied Mathematics of Xidian University in 2006, and the Ph.D. degree in computer science from the Department of Computer Science and Technology of Xi'an Jiaotong University in 2014. He is currently a lecturer in Xidian University. His main research interests include wireless ad hoc and sensor networks, and pervasive computing.

**Ke-Bin Liu** received his B.S. degree from the Department of Computer Science of Tongji University, and his M.S. and Ph.D. degrees in computer science from the Department of Computer Science and Technology of Shanghai Jiao Tong University. 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 ACM and IEEE.

**Xiangyang 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, University of Science and Technology of China, and Tongji University. Dr. Li received his M.S. and Ph.D. degrees in computer science from University of Illinois at Urbana-Champaign in 2000 and 2001 respectively. His research interests include mobile computing, cyber physical systems, wireless networks, security and privacy, and algorithms. He is a senior member of IEEE and a member of ACM.



**Yuan He** received his B.E. degree from the Department of Computer Science and Technology of University of Science and Technology of China (USTC), his M.E. degree in computer science from Institute of Software, Chinese Academy of Sciences, and his Ph.D. degree in computer science from 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 ACM and IEEE.

**Wei Xi** is a postdoctoral research fellow at Xi'an Jiaotong University. He received his Ph.D. degree in computer science from Xi'an Jiaotong University in 2014. His main research interests include wireless networks, smart sensing, and mobile computing. He is a member of CCF, ACM, and IEEE.

**Zhi Wang** is a Ph.D. candidate at Xi'an Jiaotong University. His research interests include smart sensing, pervasive computing, localization, and wireless security.

**Ji-Zhong Zhao** is a professor at the School of Electronic and Information Engineering of Xi'an Jiaotong University. He got his Ph.D. degree in computer science from Xi'an Jiaotong University in 2001. His research interests include computer software, pervasive computing, distributed systems, and network security. He is a member of CCF, ACM, and IEEE.

**Meng Wan** is an associate professor at the Center for Science and Technology Development, Ministry of Education, Beijing. He received his Ph.D. degree in business administration from Wuhan University in 2008, and his M.S. degree in business administration 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 the 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 ACM and IEEE.