Directional Diagnosis for Wireless Sensor Networks

Wei Gong, *Member, IEEE*, Kebin Liu, *Member, IEEE*, and Yunhao Liu, *Senior Member, IEEE*

Abstract—Network diagnosis is crucial in managing a wireless sensor network (WSN) since many network-related faults, such as node and link failures, can easily happen. Diagnosis tools usually consist of two key components, information collection and root-cause deduction, while in most cases information collection process is independent with root-cause deduction. This results in either redundant information which might pose high communication burden on WSNs, or incomplete information for root-cause inference that leads false judgments. To address the issue, we propose DID, a directional diagnosis approach, in which the diagnosis information acquirement is guided by the fault inference process. Through several rounds of incremental information probing and fault reasoning, root causes of the network abnormalities with high credibility are deduced. We employ a node tracing scheme to reconstruct the topical topology of faulty regions and build the inference model accordingly. We implement the DID approach in our forest monitoring sensor network system, GreenOrbs. Experimental results validate the scalability and effectiveness of this design.

Index Terms—Diagnosis, directional, sensor networks

1 Introduction

An increasing large number of wireless sensor networks (WSNs) have been applied in many different applications such as habitat surveillance, infrastructure protection and clinical monitoring [1], [2], [3], [4], [5]. Typically, sensor nodes are usually provisioned with low-capacity batteries and work in an ad-hoc and energy conservative manner. Although the connectivity and functionalities of WSNs have been greatly improved by many efforts [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], sensor networks still suffer many network-related faults, e.g., failure nodes or lossy links [16], [17] due to the error-prone nodes and other uncertainties, like environmental interference. The faults in deployed WSNs are usually difficult to detect and localize due to its improvisational nature and invisibility of internal running status. Therefore, the design of effective online WSN diagnosis tools, which can help network administrators monitor the network operational status and maintain a sensor network system, has drawn significant attentions.

Existing works on fault diagnosis in WSNs typically consist of two independent parts: collecting running information from nodes and deducing root causes of network exceptions. For instance, Sympathy [17] periodically collects runtime status from all sensor nodes and leverages a decision-tree based scheme to find the most possible causes of observed network exceptions. Khan et al. [18] propose to use external power measurements to accumulate all sensors' power information. A classifier is then applied to determine the internal health condition of an unresponsive host and the possible cause of its failure. Such methods, however, share a common drawback: their diagnosis information gathering process is static and preplanned without the aid of useful intermediate reports. Clearly, the predetermined scheme may either result in redundant information, posing unnecessary communication burden, or incompleteness of information for root-cause inference so that it leads to a large number of false results.

To address the above issues, we propose an online diagnosis approach, DID, in which information acquirement is directed by the intermediate results of a probabilistic inference model. Combining an incremental probing scheme and a dynamic probabilistic inference model, DID effectively localizes the root causes of various network abnormalities. Differing prior methods, this approach dynamically recognizes the most useful information for further diagnosis according to current results of the fault inference engine. Additionally, the diagnosis is confined to topical area covering the problematic network elements in high potential. With a low overhead, a high-accurate real-time network diagnosis service is thus provided.

Indeed, probing is a widely used diagnosis technique for obtaining internal network status in Internet and enterprise networks [19], [20]. Several types of probing schemes have been proposed for different applications, while most of them rely on expert knowledge or prior information like network topology. Besides, the network models are often assumed remaining unchanged during the running phase. Compared to the Internet and enterprise networks, however, sensor networks have the following unique features: 1) Sensor nodes usually have limited computational resources and energy supply; 2) due to the environment interference and uncertainty of the wireless medium, the network topology is highly dynamic. What is more, prior information are difficult to be acquired since WSNs are self-organized. As shown in Fig. 1, the up-left corner is the physical deployment of our GreenOrbs project. The other three topology pictures are snapshots at different time that shows that link statuses between nodes change from time to time;

• The authors are with the School of Software and TNLIST, Tsinghua University, Beijing 100086, China.
E-mail: {gongwei, kebin, yunhao}@greenorbs.com.

Manuscript received 2 Sept. 2013; revised 14 Dec. 2013; accepted 26 Jan. 2014. Date of publication 20 Apr. 2014; date of current version 8 Apr. 2015.
Recommended for acceptance by J. Chen.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2014.2308173

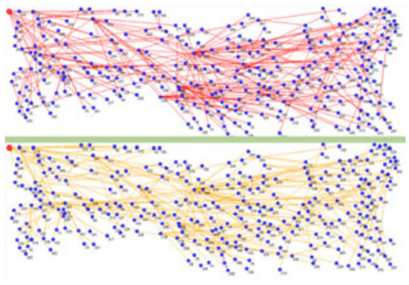


Fig. 1. GreenOrbs deployment and dynamic topologies.

3) as the sensor nodes are error-prone, the number of simultaneously failures is hard to predict in advance, while in enterprise networks it is often assumed that this number is no more than a predefined constant *k*. Thus, existing approaches cannot be directly applied to WSNs because no information about inner dependencies among network elements is really available, and frequently changed topologies further make the schemes infeasible for WSNs.

In this design, first, we introduce a node tracing scheme, in which each node caches its communication traces. When the sink detects that some nodes violate operation rules, it sends a probe request to collect information from sensor nodes that are related to the abnormalities, or we say the nodes in the problematic region. Consequently, sink is able to construct the topical topology of the problematic region. An inference model can thus be dynamically built from this topical topology. Such a process is on-demand and only involves nodes in the problematic region, so it minimizes the traffic overhead and can adapt to frequent topology changes. Second, we employ an incremental probing scheme which is combined with a probabilistic inference model. We input the probing results to the inference model and the inference results will guide the selection of next probe, aiming at gathering the most useful information to deduce the root-causes. The advantage of this scheme is that it always selects and sends probes as needed in response to problems that actually occur in the network. Third, we design a real-time inference engine which is capable of conducting incremental reasoning to localize the faulty network elements. The diagnosis reports output by this inference engine will gradually achieve the ground truth. Eventually, the diagnose process terminates after only a few probes when current information is sufficient to diagnose the network faults. To the best of our knowledge, we are the first to investigate a directional diagnosis for sensor networks.

The major contributions of this work are as follows:

1) According to the feature of dynamic topology in WSNs, we propose a node tracing scheme to recover topical topology which only includes nodes involved in the faulty region.
2) We introduce an incremental probing scheme in which the selection of next probe is based on the results of current fault inference.
3) We implement our diagnosis approach, DID, and verify its effectiveness in our forest monitoring project, GreenOrbs. The results of our field test show that the root causes are effectively and accurately localized.
4) We conduct extensive simulations under varied conditions to evaluate the scalability and effectiveness of DID design.

The rest of this paper is organized as follows. Section 2 introduces related works. Section 3 describes the architecture of our design. The node tracing scheme is detailed in Section 4. We discuss the incremental probing scheme together with the inference model in Section 5. In Section 6, we present our simulation and implementation results. We conclude this work in Section 7.

2 Related Work

Most of existing approaches for sensor network diagnosis follow similar patterns: they gather information from sensor nodes in predetermined manner and then use various inference algorithms which take collected information as inputs to deduce root-causes of network exceptions. Some researchers propose to scan the residual energy and other metrics of each sensor node in the running time [21], so that the sink can analyze the gathered information to obtain network operational conditions. Sympathy [17] periodically collects pre-defined in-network information from each sensor node such as traffic flow and neighbor list and uses an empirical decision-tree based scheme to root-cause the network failures. It selects an optimal set of information metrics to minimize the communication cost. A passive diagnosis approach, PAD, which passively observes the network symptoms from the network, is proposed for sensor network by Liu et al. [22]. Using the marks collected at the sink, a probabilistic inference model can be dynamically constructed and maintained which takes both positive and negative symptoms as inputs and reports the inferred posterior probability of possible root-causes. Khan et al. [18] propose to use an independent power-metering subsystem which can collect power consumption traces of high-end sensor nodes. The authors also introduce various algorithms to determine possible causes of the network failures and infer application states by analyzing the power traces. In Gong et al. [36] proposed multi-center diagnosis solution to improve the quality of diagnosis quality. Most of existing works, if not all, however, fail to link the fault inference process and the diagnosis information gathering process.

Detecting and localizing network anomalies is an important issue in Internet and enterprise networks, and many efforts have been devoted to this field. Bejerano and Ragosti propose to actively monitor all the links of a network in an efficient way [23]. They form the problem of selecting a minimal set of paths which covers all links in network as an instance of the well-known NP-hard problem, minimal set cover, and employ a heuristic algorithm to solve it. Duffield proposes a tomographic method in [24], which can infer the location of problematic links assuming binary network performance characteristics. Dhamdhere et al. [25] extend the work of Duffield by using additional network measurements. Score [26] introduces a shared risk model in which a two-level graph is simplified for inference model and also formulates the problem of localizing fault as a NP-hard problem. By exploring the bipartite graph inference model, Kandula et al. [27] reduces the complexity of the inference process. In [28], Sherlock proposes a multi-state

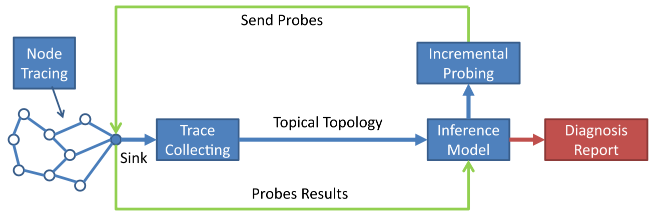


Fig. 2. DID architecture overview.

and multi-level inference graph for the network diagnosis and a scoring function to determine root causes for observed exceptions. The selection of optimal probes has been addressed in [29], where Song et al. apply Bayesian experimental design to select active probes for maximizing the amount of information about path properties. The above schemes either require prior information of the network dependencies such as topology or take simplified methods to model the network dependencies. However, highly dynamic and self-organizing properties of WSN make it impractical to assume that network topology can be acquired in advance and remain relatively unchanged during running time. Also, the assumption that there are no more than a small constant number of failures in Internet and enterprise networks which can avoid NP-hard computation complexity is also invalid for WSNs.

3 System Architecture

We consider sensor networks where sensing data are periodically sent from source nodes to a sink through multi-hop communication. There could be both full-function nodes which involve in both communication and reporting sensing data and single-function nodes which only relay packets for others in the sensor networks.

We design a directional diagnosis approach, DID, for such sensor networks. The goal of DID is to aid network administrator in discovering the root causes of the network abnormal phenomenon. DID implants a tiny tracing sniffer into each node in WSNs which records packets relay information so that sink can collect these information from nodes to reassemble topical topology for further diagnosis when there are network exceptions happened. Usually the network exceptions are predefined and used as a symbol of fault detected which triggers diagnosis process. The definition of exception can be in various criteria according to specific application requirement, such as a long time delay of data arrival or insufficient amount of data. However, only tracing information from sniffers does not suffice to precisely infer and localize the root cause of network faults. Thus by employing an incremental probing scheme, we continuously refine the inference model to derive network element statuses more exactly.

As illustrated in Fig. 2, DID is mainly composed of four components: a node tracing module, a trace collecting module, a probabilistic inference model and an incremental probing module. The node tracing module resides in each node and counts the source of all packets passing by. At the sink, the tracing collection module gathers above tracings from nodes which participate in

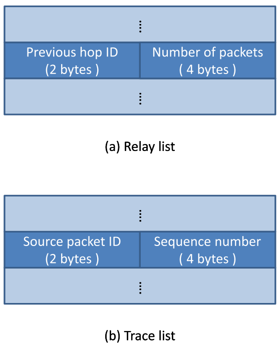

Previous hop ID (2 bytes)
Number of packets (4 bytes)
(a) Relay list
Source packet ID (2 bytes)
Sequence number (4 bytes)
(b) Trace list

Fig. 3. Data structures in node tracing.

communication with suspiciously faulty nodes when any network exception is discovered. Then the coarse inference model can be built on the topical topology reconstructed. The inference model guides the selection of next probe sent to sensor networks. At the same time with the help of probing results, the inference model is continuously updated. When incremental probing stops, the diagnosis report which contains the posterior probability of network elements being faulty is yielded.

4 Node Tracing

The network structure of WSN frequently change due to the innate self-organization; therefore prior information about topology is hard to be obtained for constructing the inference model which can be easily built in Internet or enterprise networks. The topology of WSN is highly dynamic during the runtime; so the network statuses also need to be acquired continually to reflect network changes in real time. To address the above requirements, we propose a node tracing algorithm which dynamically reconstructs topical topology and derive the inner dependencies among network elements.

4.1 Data Structures in Node Tracing

The node tracing algorithm lets each sensor node maintain two lists: one is a relay list which records the previous one-hop IDs of packet and the number of packets relayed for, the other is a trace list in which each entry contains source ID of packets and its newest sequence number, as illustrated in Fig. 3. This tracing scheme only adds 6 bytes in each row of relay list for each one hop neighbor and 6 bytes for each entry of trace list. During the packet delivery, each packet is counted by the nodes on the path from source to sink. Supposed that N is the number of the nodes in the WSN, thus the size of relay list on each node is no more than 6N bytes, so as the size of relay list. The total memory consumption of each node is no more than 12N bytes. Assume that in a WSN of 1,024 nodes, the maximum memory consumption on each node is no more than 12 KB. And also we know that the number of previous-hop ID and source packet ID is usually far from N in practice. Thus we can conclude that this memory consumption is acceptable.

At the sink side, the trace collecting module gathers the information from nodes which included in potential problematic area when network exceptions found. Through assembling relay lists of these nodes, topical topology can easily be reconstructed. The collecting process is on-demand when network fault events are detected, so network conditions variations are included in the topical topology in time. There are two obvious advantages of this design: one is that diagnosis area is confined to topical nodes, the other is that no traffic overheads would be incurred if no network exceptions are found.

4.2 Tracing Scheme on Sensor Nodes

As shown in Algorithm 1, upon receiving a packet, the intermediate node first checks relay list. If there is no entry for the previous hop node ID, it creates a new entry for it and sets the number of relayed packet to 1. Otherwise, it just increases the packets relay number by 1 in the corresponding entry. Then it checks the trace list. A new entry for this source node is created in its cache and the sequence number for the packet is recorded if there is no entry for source ID of the packet. If there exists an entry in the trace list for the source node and the sequence number in the packet is larger than trace list entry, the intermediate node updates the trace list entry with the new sequence number. The sink also participates in the tracing process. However, the only difference is that the sequence number of entry in the trace list is the oldest at the sink while other nodes record the newest. When one-time topical diagnosis completed, we should reset sequence number of the faulty nodes diagnosed which indicates a new start time point.

Algorithm 1 Node Tracing (Packet p)

```
1:  check relay list for previous hop node ID of packet $p$.
2:  **if** no entry for packet $p$ **then**
3:      create entry with previous hop node ID and set number
        of relayed packet to 1.
4:  **else**
5:      increase the number of relayed packet of corresponding
        row by 1.
6:  **end if**
7:  check trace list for source node ID of packet $p$.
8:  **if** no entry for packet $p$ **then**
9:      create entry with source node ID and sequence number
        of $p$.
10: **else**
11:     **if** sequence number in the list is larger than $p$'s **then**
12:         **return**
13:     **else**
14:         update sequence number in corresponding entry.
15:     **end if**
16: **end if**
17: **return**
```

4.3 Collecting Trace from Sensor Nodes

At the sink, trace collecting process is trigged by some network exceptions happened. One of the widely used criteria is that insufficient amount of sensing data from nodes during some time interval indicates possibly some network faults detected. Then the sink extracts the rows of these suspicious nodes from the trace list and then packages this information as a collecting probe which is broadcasted to networks. For example, suppose we have seven nodes, as A (sink), B, C, D, E, F, and G. If A finds that the amount of sensing data from both B and G is below a predefined threshold in the time interval. Then it extracts corresponding rows in trace list, e.g., (B, 1,010) and (G, 1,308). This flooding scheme is different from normal broadcasting because only if the relay list of nodes contains any suspicious node, the probe is rebroadcasted to neighbor nodes. Otherwise the probing is stopped on that node. By using this controlled flooding, additional traffic overhead incurred by probe broadcasting is effectively restricted.

As shown in Algorithm 2, on receiving a probe, each node needs to check whether it has ever relayed packets for any suspicious node. It first examines whether there is suspicious node ID in its trace list or not. If so, it then compares the sequence numbers of row in the probe and trace list. If the sequence number in the probe is larger, then the node should hand in the trace information which contains relay list and one-hop neighbor list which is often easily obtained from down-level network protocols such as routing protocol, and then broadcast this probe to neighbor nodes. Otherwise, larger sequence number in probe p means that this node had relayed packets for suspicious nodes before the problem time interval in which network exceptions happened, so it is not involved in the diagnosis process. The collected relay list is used to facilitate the construction of the inference model as it tells the strength of the dependency between the parent and its successive nodes.

Algorithm 2 Reply Probe(Probe p)

```
1: for each row in $p$
2: **if** row.ID is in the trace list **then**
3:     **if** row.sequencenumber is larger than its corresponding
       entry in the trace list **then**
4:         send thisNode.relaylist and thisNode.neighborlist to
           sink.
5:         broadcast $p$ to its neighbor.
6:         **return**
7:     **end if**
8: **end if**
9: **return**
```

After gathering demanded information, the sink starts to reconstruct the topical topology for diagnosing. The neighbor lists and suspicious nodes set are used to recover topology. The reconstruction starts from the sink, then we connect all neighbor links between each other in the replied neighbor lists. Obviously, there might be some irrelevant edges and nodes included. Therefore, we need to remove those irrelevant network elements. As requirements of topical diagnosis, we can conclude that leaf nodes of topical topology can only possibly be the sink or suspicious nodes. So we iteratively cut off those nodes whose degree is one and their connected edges until all unqualified leaf nodes are removed.

5 Incremental Probing

As we all know that network elements are correlated, the crash of a parent node could cause that its children need to find a new route to sink and disconnected to the sink if there is no new route found. Another instance is that, simultaneous high ratio loss of multiple paths might indicate a shared link is abnormal. Thus, we build a probabilistic inference model which encodes the dependencies among different network elements, such as node status, link status and sensing function, based on the reconstructed topology from node tracing module. Exterior symptoms like loss of data samples or link status are considered as inputs. When specific symptoms are observed by our probing algorithm, we can reason the posterior probability of the failures of each network element and find the most probable explanation for observed symptoms.

Most existing diagnosis schemes for sensor networks have no feedbacks on diagnosis information collecting. In our approach, inference engine starts when network abnormality is detected. However, as the initial observed symptoms is not sufficient to root-cause the real faults, we use incremental probing scheme to discover the most critical symptoms that are important to explain the problem. The selection of probe which is used to collect more information about network elements is under the guide of the newest updated inference model. Our approach can effectively localize multiple faults of WSNs in a timely manner.

5.1 Background of Belief Network

As no prior information about topology can be acquired in advance and topologies are also highly dynamic in running time, most existing probabilistic inference schemes for Internet and static enterprise networks which use the bipartite graph or tree-based inference models are not applicable in WSNs. Here, we employ a multi-level inference model which is similar to [22] to encode the inner dependencies in sensor networks.

A belief network is a directed acyclic graph (DAG) that consists of a number of vertices which denote random variables and edges which denote causal relations between variables. A directional arc from vertex A to vertex B means B is the outcome of cause A. Each vertex in belief networks is assigned with its probability distribution. The difference is that the vertex which has no parent is assigned prior probability, we call it CAUSE for short, and the other intermediate vertex is assigned conditional probability distribution (CPD), we call it SYMPTOM. Given certain evidence which might contain several SYMPTOMS, the belief network can deduce posterior probability of all CAUSES.

5.2 Constructing the Belief Network

In our approach, the belief network structure is constructed from the topical network topology obtained from the node tracing module. An example topology is depicted in Fig. 4a which is composed of a sink and two sensor nodes. The directional edge between two nodes denotes a wireless data transmitting link. There are six types of vertices in our belief network, each of which has two statuses: UP and DOWN. We explain each type as follows. The radio vertex indicates

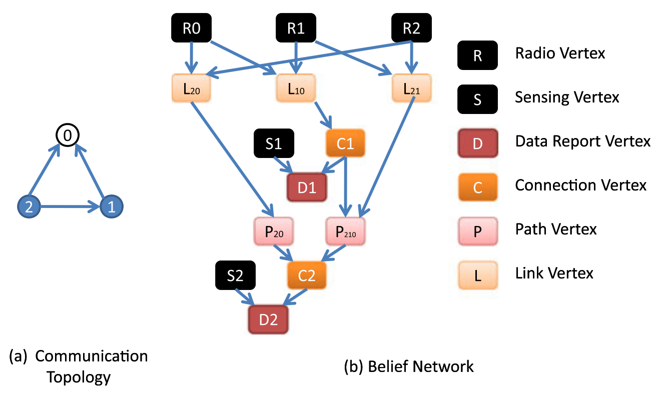


Fig. 4. Belief network. (a) Communication topology. (b) Belief network built on communication topology.

the radio communication function of the correspondent sensor node. It impacts related link status. The sensing vertex indicates the sensing function of the node. The connection vertex describes network connectivity from sensor node to the sink. The path vertex stands for network condition of specific course. The link vertex represents the communication conditions between two nodes. The data report vertex denotes the status of the data reception of the source node at the sink. For example, if the sink observes insufficient sensing data in a period of time or a long time delay from a source, this vertex is observed as DOWN status which may trigger a diagnosis process. Note that, the relay node which has no sensing function should not contain this vertex. The status of the data report vertex depends on two parents, the sensing vertex and the connection vertex; while the link vertex status is decided by the status of two neighbor nodes. It is obvious that the dependency among connection, path and link vertex is deduced from communication topology. According to above rules, we can obtain a multi-level belief network is composed of these six types of variables, as shown in Fig. 4b. Among the six types of variables, the statuses of the radio and sensing vertex are the CAUSES that we need to infer, as they cannot be observed from outside. The link, path and connection vertices are SYMPTOMS which we can obtain through probe technique. The data report vertex statuses is directly observed at the sink. The belief network structure can be easily updated from node tracing module as network topology changes over time.

Complexity analysis. Here we give both the time and spatial analysis of our belief network construction in worst case. Supposed there are V nodes and E edges/links in the original topology. In worst case, the tropical topology is the same as the original topology, i.e., we are diagnosing the whole network. Our belief network construction mainly consists of three steps: First, each node is connected to a sensing vertex and a connectivity vertex. The complexity of this step is $\mathcal{O}(V)$ and the spatial complexity is $3V$ vertices; second, finding all possible paths from each node to the sink by breadth-first-search (bfs) can be finished in $\mathcal{O}(V(V+E))$ [30] and the spatial complexity is at most $V * V^2 = V^3$ vertices; the third step is to connect each links to the radio vertices. Thus the time complexity of this step is $\mathcal{O}(E)$ and the spatial complexity is V. Therefore, in total the time complexity of constructing our belief network is $\mathcal{O}(V(V+E))$ and the number of total vertices in belief network should be less than $(4V + V^3)$. Although this time and spatial cost is relatively high for normal PC, fortunately this belief network construction is always performed on servers which are the high-performance clustered-computers or the clouds [31].

5.3 Inferring in Belief Network

When the belief network construction completes, the next important issue is to assign CPD to each vertex. The prior fault probability distribution of the radio communication and sensing function are assigned according to experience data. The CPDs of other SYMPTOM vertices are encoded as noisy-OR gate [32] and Select gate [28]. In a noisy-OR gate, any one of the parent vertices in DOWN status results in the DOWN status of the child variable. In our model, the connection and sensing function vertices affecting the data report vertex, the link and connection vertices affecting the path vertex, two radio vertices affecting the link vertex, and the link and connection vertices affecting path vertex are represents as noisy-OR gate. The relation between multiple paths and a connection is represented as the Select gate, because if any one path is in UP status; the connection is in UP status. The contribution of each path to connection vertex can be obtained from the relay lists collected in trace collecting module. The outputs of the inference process are the posterior probabilities about the radio and sensing vertices which provides important guidance for probe selection and root-cause judgment.

5.4 Probing the Problematic Network

Once the probabilistic inference model is constructed, we can start our incremental probing process according to information provided by belief network. The main idea of our approach is that if the observed SYMPTOMS are not sufficient to explain the problem, it selects additional optimal probe to explore the most critical SYMPTOMS that are important to explain the problem. There are three major modules: Fault deducing, fault evaluation and probe selection. Fault deducing module puts observed SYMPTOMS into inference model and generates a candidate fault set according to updated posterior probabilities of CAUSES. Then this set is sent to faults evaluation module to compute its belief value and check if it is satisfactory. If the correlated SYMPTOMS are sufficient to explain the fault candidates then probing process terminates. Otherwise, the unobserved symptoms that contribute to the fault candidate set are sent to the probe selection module. An optimal probe is selected to discover statuses of those unobserved symptoms. Afterward the probe result is sent to fault deducing module for another round of reasoning until a high belief value of the candidate set found or all SYMPTOMS observed. The pseudo code of this probing scheme is described as Algorithm 3.

5.5 Deducing Faults Candidates

The task of fault deducing is to search for root causes of the observed SYMPTOMS. The observed SYMPTOMS sent to fault deducing contain data report statuses from the sink and statuses of network elements examined by dynamic probing. When new observed SYMPTOMS arrive, the fault deducing module sets those discovered evidences to inference model and update probability of each CAUSE. Then we use a fault probability threshold as a criterion to find which CAUSE is fault. The CAUSE whose newest updated probability of DOWN statuses exceeds the threshold is included into candidate fault set. This is a simplified method to find faults which have the most informative contribution to observed SYMPTOMS. Note that, if the threshold is set too high, at initial stage maybe no CAUSE is qualified as a fault. So we can simply choose the CAUSE which has maximum probability of status DOWN as a fault for a start point. It is also noticed that there is no direct relation between two successive deduced candidate fault set because new SYMPTOMS observed may increase or decrease their fault probability. As networks statuses are progressively probed, more SYMPTOMS are observed which indicate that faults deduced is more closed to ground truth.

Algorithm 3 IncrementalProbing()

```
1:  Construct Belief network and initial OS(observed symptoms).
2:  while OS is less than all SYMPTOMS do
3:      generate FSet (candidate fault set).
4:      Compute $BV$ for FSet.
5:      if the $BV$ of FSet is less than threshold then
6:          select optimal path to probe.
7:          OS += probe result.
8:      else
9:          return
10:     end if
11: end while
12: return FSet
```

5.6 Evaluating Faults

The candidate fault set generated by the fault deducing module maybe inaccurate because the information about the SYMPTOMS statuses is insufficient. The fault evaluation module measures the credibility of a candidate fault set given correlated SYMPTOMS and observed SYMPTOMS.

We design a belief function $BV(C)$ to compute the belief value of given candidate fault set. We assume a candidate fault set consists of a number of faults, denoted as $C_f = \{f_1, f_2, \ldots\}$. And the SYMPTOMS related to f_i are included in set $S_{f_i} = \{SYM_1, SYM_2, \ldots\}$. So the correlated SYMPTOMS set of a candidate fault set is denoted as $S_c = \bigcup_{f_i \in C_f} S_{f_i}$. We use S_o to denote observed SYMPTOMS set. Thus, the belief value of a candidate fault set is computed as follows:

$$BV(C) = \frac{\prod_{SYM_i \in S_r}(1 - \prod_{f_i \in C_f}(1 - p(SYM_i = D|f_i)))}{\prod_{SYM_i \in S_o}(1 - \prod_{f_i \in C_f}(1 - p(SYM_i = D|f_i)))}, \quad (1)$$

in which, $SYM_i = D$ means that there is a $DOWN$ event with SYMPTOM *i*.

In our probabilistic inference model, even though related SYMPTOMS are all observed, the belief value of a candidate set can still be less than 1 because some

SYMPTOMS may not happen even when the faults deduced are the ground truth. Thus, the administrator can configure a belief value threshold to decide whether a candidate fault set is acceptable. The configuration of this threshold can be based on long-term observation and previous experience. If the threshold is set too high, correct fault set will be unqualified; but if the threshold is too low, false positive faults might be included. Belief function is used to evaluate each candidate fault set and if the result value exceeds the pre-defined threshold, the fault localization process terminates and the candidate fault set is sent to diagnosis report.

Complexity analysis. Here we analyze the complexity of equation (1). Supposed that there are c faults in C_f. The main source of complexity comes from the number SYMPTOMS in S_o and S_c. According to analysis in Section 5.2, we know that there are at most $(V^3 + V)$ SYMPTOMS in our belief network. Thus, the complexity of equation (1) is $\mathcal{O}(cV^3)$ in worst case.

5.7 Selecting Probe

If the belief value of a candidate fault set is below the threshold, we should select an optimal probe to gather information about the statuses of SYMPTOMS which we most want to know. As stated in [23], [26], [20], the problem of probe selection whose goal is to minimize probe cost is equal to a minimal set cover problem which is NP-complete. Therefore, we propose a heuristic greedy algorithm to solve this problem. The main idea is that we always select the probe which covers the maximum number of unobserved SYMPTOMS that related to candidate fault set. If there are several probes which cover the same number of unobserved SYMPTOMS, the probe which explores least number of SYMPTOMS is selected. Our algorithm intuitively appreciates probes that discover more symptom correlation.

5.8 Fault Sources

In this section, we aim to choose the simplest fault source that explains the failure. Through analysis of the fault reports generated by DID, we can mainly group fault sources as follows:

1) *Node failure.* Node failure means that the fault source is localized to the node itself. This type of failure includes two main sub-sources: hardware failure and software failure. For example, battery component falls off and sensing unit down are typical hardware failures. Deadlock in critical section assignment and network sending queue overflow are typical operating system failures. Remedial actions involve interacting with the node itself, such as changing its batteries, rebooting operating system, replacing hardware components.
2) *Network failure.* Due to instability and uncertainty of wireless transmission, network failure mainly relates to traffic flow in networks. The typical network failures are network congestions which lead to a low packet delivery ratio excessive collisions along the routing path. Remedial actions involve moving or change the positions of specific nodes and add nodes in some region to balance traffic load.
3) *Temporary failure.* Another important type of failure is temporary failure which may happen due to complex interior or exterior interferences and quick self-recovery. The DTN-Style errors, which are characterized by a lack of instantaneous end-to-end paths, have much in common as temporary failure in wireless sensor networks. Thus, we treat DTN-Style error as a particular type of temporary failure. In DID, we use a common technique, replicated copies, to increase the probability of probing message being successfully transferred [33]. Through replicating several copies of probing message, we hope that at least one will succeed in reaching the destination. Noted that this technique can only partially solve the temporary failure problem. The complete solution for this is the main interest of our future research.
4) *Multiple failures.* Maybe the most difficult situation is multiple failures, e.g., network partitions. Normally, as stated in Section 5, multiple failures can be successfully localized by our belief network inference framework, since those failures may be correlated through our intermediate SYMPTOMS, e.g., path vertex or link vertex. But if those multiple errors mask each other, i.e., the status of one SYMPTOM is totally uncovered the other SYMPTOM, the masked errors are unable to be detected in all online diagnosis models [17], [22], including our DID. In this situation, one possible solution is to use the mobile diagnosis center to dynamically gather the information from nodes in masked/partitioned area and then introduce proper inference model to deduce root-causes.

6 Evaluation

We evaluate DID by extensive simulations and implementation. For the inference model, we use the BNJ implementation of the belief network. We implement the node tracing scheme on the TinyOS platform [34] with nesC language and incremental probing scheme with java. The simulation platform is TOSSIM. The sensor nodes in our field experiments are TelosB motes.

6.1 Simulations

In our simulation, a sensor network is composed of a sink which is located at the center and several nodes which scattered in two-dimensional space. Sensing data is periodically sent to sink through multihop communications. Tree-based routing scheme and DAG-based routing scheme are both evaluated. In tree-based routing scheme, the topology is a spanning tree rooted at the sink and each node only has one route to the sink (one parent). In DAG-based routing, or called multipath routing, since each node connects multiple parents, it can transmit data to any one of its parents based on the connectivity metrics, e.g., loss rate. A number of different faults are injected into the network under various simulation settings. We use 0.5 as the fault posterior probabilities threshold. We also apply the detection ratio and false positive ratio to analyze the effectiveness of our incremental probing scheme and inference model.

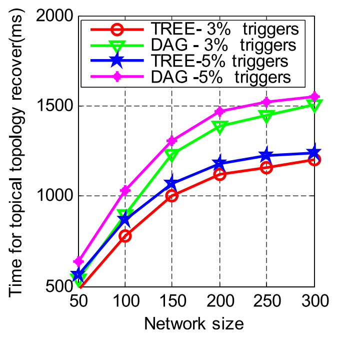


Fig. 5. Topical topology recovery time on varying network sizes.

6.1.1 Efficiency of the Node Tracing Scheme

We evaluate topical topology recovery time under different network settings. In tests, we simulate a data acquisition sensor network of both TREE routing and DAG routing. Each sensor node generates sensing data and delivers to sink every 1 s. Besides the routing schemes, the number of triggers which initiate the trace collecting process also impacts the topology reconstruction. Thus, we evaluate the performance of DID with different number of triggers. Here, 5 percent triggers means that there are 5 percent number of nodes which fails to deliver sufficient sensing data to the sink. In tests, we measure the reconstruction time of topical topology. According to the results in Fig. 5, it is very fast that the topical topology recovered by 5 percent triggers under tree-based routing can be within 1,200 ms with up to 300 nodes. As nodes in DAG always have multiple parents, the topology of DAG routing is more complicated than it of TREE, so more time is needed to recover it. As the network size increases, the growing speed of topical topology recovery time is less than the linearity which indicates a proper scalability of our approach.

6.1.2 The Performance of Incremental Probing

We then evaluate the performance of incremental probing with four different groups of simulations. We inject artificial errors into the network and use incremental probing scheme to refine the inference model which generates diagnosis reports according to the posterior probability estimations.

First, only sensing failures are inject into sensor nodes. We randomly set the sensing function disabled in 5 percent of the nodes. The network size varies from 50 to 100 nodes. As the detection ratios shown in Fig. 6, our approach achieves detection ratios higher than 90 percent in both TREE and DAG settings. Fig. 7 describes the false positive ratio. We see that the false positive ratio decreases as the network size increases. Through analyzing the intermediate and final diagnosis reports and, this can be explained by that the bigger network size, the more link-related SYMPTOMS are easily observed, and so the higher accuracy of reasoning sensing failure. Otherwise, the posterior probability of sensing failure is heavily relied on the prior fault probability and improper faults threshold might result in negative judgment.

We then randomly inject sensing failure and radio communication failure into sensor nodes. The results are shown in Figs. 8 and 9. Both decrease rate of the detection ratio and increase ratio of the false positive ratio is relatively high when the number of nodes is below 60. After the network size grows beyond 60, the detection ratio and false positive ratio are both relatively stable. This is mainly because the small size of belief network always performs better than the large size in probabilistic inference.

Another group of simulation evaluates different belief value threshold of candidate fault set. The experiment is conducted under TREE routing setting. The other settings are the same as previous group simulation. As the experiment results shown in Figs. 10 and 11, both thresholds achieve detection ratios higher than 85 percent. We also can find that the higher of threshold value, the lower of detection ratio and false positive ratio. This phenomenon is consistent with the intention of designing belief value function to evaluate the quality of candidate faults.

In the last group of simulations, we fix the network size at 30 and examine the effect of simultaneous multiple faults in our model which vary from one to five. As shown in Figs. 12 and 13, as more faults occur, the detection ratio decreases and the false positive ratio increases. This is the result of that mutual interference of multiple faults in the inference model deteriorate inference performance.

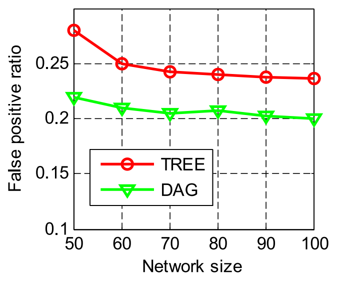


Fig. 7. Sensing failure false positive ratio.

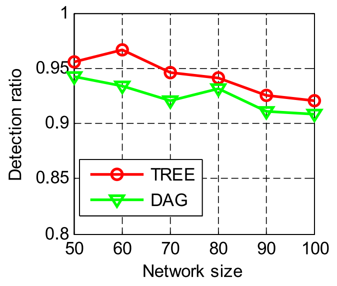


Fig. 6. Sensing failure detection ratio.

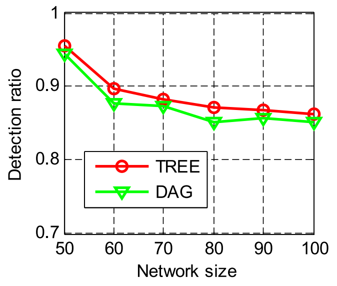


Fig. 8. Sensing and radio failure detection ratio.

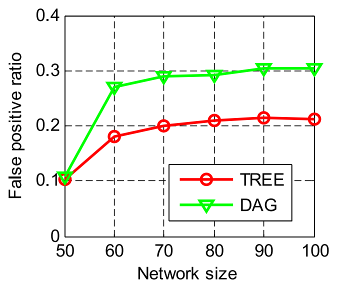


Fig. 9. Sensing and radio failure false positive ratio.

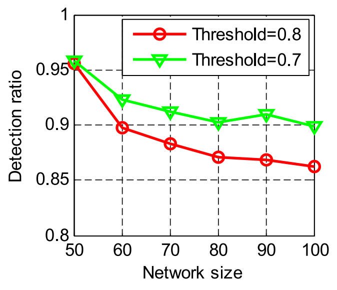


Fig. 10. Detection ratio for varied thresholds.

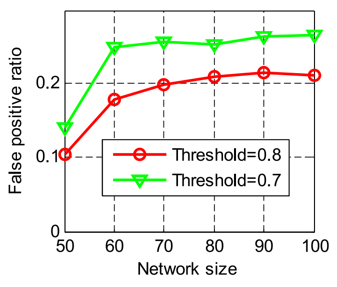


Fig. 11. False positive ratio for varied thresholds.

6.2 Trace-Driven Experiments

In trace-driven experiments, we collect data traces from both in-network data, which is transmitted through the same route as sensing data, and offline data, which is stored in the flash memory of each node. With the help of these data traces, we can exactly know the statues of each node in every timestamp and thus rebuild the virtual sensor networks to perform diagnosis. The data traces are from a period of more than three months in our forest monitoring sensor networks, GreenOrbs [35]. After extensively examining traces of a few months, we find that the occurrences of failures in network is not evenly distributed. In order to demonstrate the effectiveness of our scheme, thus, we select the time interval in which the occurrences of failure is relatively concentrated. Finally, we perform trace-driven study on a six-day data trace that consists of a segment of 42,228 received packets from 309 sensor nodes.

6.2.1 *The Trace Diagnosis*

Our trace study mainly consists of three parts. First, we count the sensing data receptions in TIMEPERIOD1, if some nodes did not sufficiently hand in their sensing data, we call them suspicious nodes. Then we start the probing diagnosis in TIMEPERIOD2. The topical topology is constructed based on the neighbor list of nodes

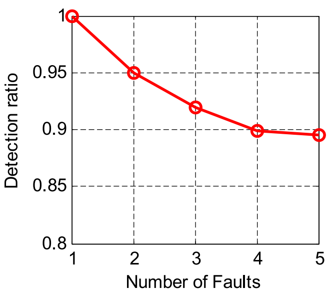


Fig. 12. Detection ratio for multiple faults.

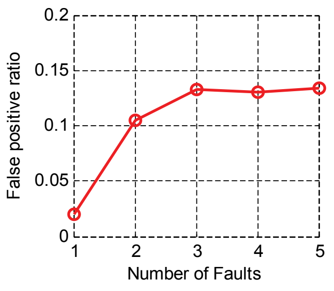


Fig. 13. False positive ratio for multiple faults.

which are involved in the relaying paths of suspicious nodes during TIMEPERIOD1. The probing process is equal to searching the corresponding paths in the TIMEPERIOD2. According to the incremental probing results, the inference model finally deduces the sensing failures and node failures. In Table 1, we list the diagnosis reports of two time periods in our six-day experiment. In diagnosis report I, through 21 incremental probes, we localize 12 sensing failures and seven radio failures of nodes while we localize 15 sensing failures and nine radio failures of nodes using 25 probes in diagnosis report II. Moreover, time cost of diagnosis in diagnosis reports I and II are 204 and 274 seconds, respectively, showing the time efficiency of DID.

6.2.2 *Diagnosis Traffic Overhead*

We compare the extra diagnosis traffic overhead introduced by DID, PAD and Sympathy. PAD acquired running information of networks through added marks on each regular traffic packets and inferred the possible failure, while Sympathy aimed to collect all necessary diagnosis information and deduced the root-causes. Both of them achieved nice performance in their settings, however, they also produced extra diagnosis traffic even when there is no exception in the running networks. Here we select empirical cumulative distribution functions (ECDFs) to compare the overhead. In Fig. 14, the x-axis denotes the ratio of the diagnosis traffic to the total network traffic, and the y-axis is the ECDF. The results in Fig. 14 are combined from diagnosis reports I and II. As an illustration, a point with value (7, 0.7) means that the diagnosis traffic overhead dominates less than 7 percent of the total traffic in 70 percent of the time. As in Fig. 14, we compare the DID with PAD and different epoch size of Sympathy. DID significantly outperform Sympathy all

TABLE 1
Trace Driven Diagnosis Reports

	Diagnosis report I	Diagnosis report II
Sensing failures	12	15
Radio communication failures	7	9
Number of probes	21	25
Diagnosis time (in second)	204	274

the time. Compared to PAD, DID achieves better performance at lower overhead since that almost no diagnosis overhead is generated if there is no exception found in the networks, while in the trace collecting process, DID may incur more traffic overhead than PAD by probing network status.

6.3 Test-Bed Diagnosis

In this testbed experiment, we artificially inject one sensing failure node *NodeA* and one radio failure node *NodeB* into the network with 30 sensors. As shown in Fig. 15, the variations of inferred posterior fault probability of both faulty nodes are given. As probes incrementing, more SYMPTOMS are effectively observed. After each probe, the fault probabilities are gradually close to the ground truth which indicates that *NodeA* and *NodeB* are both faulty.

7 Conclusion

We propose DID, a directional diagnosis approach which can be easily implemented for providing online diagnosis of working WSNs. The incremental probing scheme enables the diagnosis information collecting guided by the updated inference model which can quickly localize faults as well as minimize communication cost incurred by collecting useless information. The proposed node tracing scheme can successfully recover topical topology for diagnosis with relatively low overhead. We implement our approach and examine its effectiveness in our forest monitoring project. The diagnosis result is accurate in real time. We now plan to introduce a decentralized mechanism for diagnosis information collecting, so that the sink does not have to maintain the most updated state of all nodes in the network. In addition, more network exceptions and unexpected faults such as environmental interference are considered to be included into inference model for further reasoning.

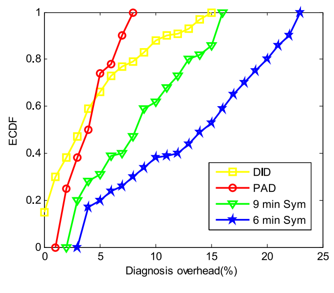


Fig. 14. Diagnosis traffic overhead (DID, PAD, Sympathy).

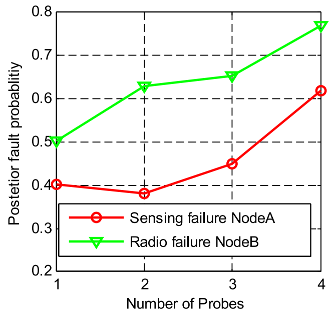


Fig. 15. The posterior probability variations.

Acknowledgments

This work was supported in part by NSF China Distinguished Young Scholars Program No. 61125202, the NSFC Young Scholar No.61303196 and 61103187, National Basic Research Program of China (973) Grant No. 2012CB316200, and China Postdoctoral Science Foundation No. 2013M540950.

References

[1] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, “An Analysis of a Large Scale Habitat Monitoring Application,” *Proc. Second Int’l Conf. Embedded Networked Sensor Systems (SenSys)*, 2004.

[2] N. Xu, S. Rangwala, K.K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, “A Wireless Sensor Network for Structural Monitoring,” *Proc. Second Int’l Conf. Embedded Networked Sensor Systems (SenSys)*, 2004.

[3] T. He, S. Krishnamurthy, J.A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, “Energy-Efficient Surveillance System Using Wireless Sensor Networks,” *Proc. Second Int’l Conf. Mobile Systems, Applications, and Services (MobiSys)*, 2004.

[4] O. Chipara, C. Lu, T.C. Bailey, and G.-C. Roman, “Reliable Clinical Monitoring Using Wireless Sensor Networks: Experiences in a Step-Down Hospital Unit,” *Proc. Eighth ACM Conf. Embedded Networked Sensor Systems (SenSys)*, 2010.

[5] J. Lu, T. Sookoor, V.Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse, “The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes,” *Proc. Eighth ACM Conf. Embedded Networked Sensor Systems (SenSys)*, 2010.

[6] T. Zhu, Z. Cao, W. Gong, Y. He, and Y. Liu, “Illuminations and the Revelations: Lessons Learned from GreenOrbs Project Development,” *ACM Mobile Computing and Comm. Rev.*, vol. 17, pp. 42-46, 2013.

[7] Y. Liu, X. Mao, Y. He, K. Liu, W. Gong, and J. Wang, “CitySee: Not Only a Wireless Sensor Network,” *IEEE Network Magazine Special Issue on Cloud-Assisted Mobile Computing and Pervasive Services*, vol. 27, no. 5, pp. 42-47, Sept./Oct. 2013.

[8] Z. Yang, C. Wu, T. Chen, Y. Zhao, W. Gong, and Y. Liu, “Detecting Outlier Measurements Based on Graph Rigidity for Wireless Sensor Network Localization,” *IEEE Trans. Vehicular Technology*, vol. 62, no. 1, pp. 374-383, Jan. 2013.

[9] Y. Liu, Q. Zhang, and L. Ni, “Opportunity-Based Topology Control in Wireless Sensor Networks,” *IEEE Trans. Parallel and Distributed Systems*, vol. 21, no. 3, pp. 405-416, Mar. 2010.

[10] Y. Cai, W. Lou, M. Li, and X.-Y. Li, “Energy Efficient Target-Oriented Scheduling in Directional Sensor Networks,” *IEEE Trans. Computers*, vol. 58, no. 9, pp. 1259-1274, Sept. 2009.

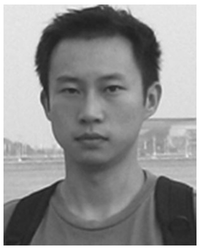
[11] D. Dong, X. Liao, Y. Liu, C. Shen, and X. Wang, “Edge Self-Monitoring for Wireless Sensor Networks,” *IEEE Trans. Parallel and Distributed Systems*, vol. 22, no. 3, pp. 514-527, Mar. 2011.

[12] C. Zhang, X. Bai, J. Teng, D. Xuan, and W. Jia, “Constructing Low-Connectivity and Full-Coverage Three Dimensional Sensor Networks,” *IEEE J. Selected Areas in Comm.*, vol. 28, no. 7, pp. 984-993, Sept. 2010.

[13] Y. Ling, C.-M. Chen, and S. Chen, “Analysis of Power-Aware Buffering Schemes in Wireless Sensor Networks,” *ACM Trans. Sensor Networks*, vol. 7, no. 3, article 26, Oct. 2010.

[14] S. Zahedi, M. Srivastava, C. Bisdikian, and L. Kaplan, "Quality Tradeoffs in Object Tracking with Duty-Cycled Sensor Networks," *Proc. IEEE 31st Real-Time Systems Symp. (RTSS)*, 2010.
[15] X. Mao, S. Tang, J. Wang, and X.Y. Li, "iLight: Device-Free Passive Tracking Using Wireless Sensor Networks," *IEEE Sensors J.*, vol. 13, no. 10, pp. 3785-3792, Oct. 2013.
[16] G. Tolle and D. Culler, "Design of an Application-Cooperative Management System for Wireless Sensor Networks," *Proc. Second European Workshop Wireless Sensor Networks (EWSN)*, 2005.
[17] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the Sensor Network Debugger," *Proc. Third Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2005.
[18] M.M.H. Khan, H.K. Le, M. LeMay, P. Moinzadeh, L. Wang, Y. Yang, D.K. Noh, T. Abdelzaher, C.A. Gunter, J. Han, and X. Jin, "Diagnostic Powertracing for Sensor Node Failure Analysis," *Proc. Ninth ACM/IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN)*, 2010.
[19] L. Cheng, X. Qiu, L. Meng, Y. Qiao, and R. Boutaba, "Efficient Active Probing for Fault Diagnosis in Large Scale and Noisy Networks," *Proc. IEEE INFOCOM*, 2010.
[20] I. Rish, M. Brodie, N. Odintsova, S. Ma, and G. Grabarnik, "Real-Time Problem Determination in Distributed Systems Using Active Probing," *Proc. IEEE/IFIP Network Operations and Management Symp. (NOMS)*, 2004.
[21] Y. Zhao, R. Govindan, and D. Estrin, "Residual Energy Scan for Monitoring Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC)*, 2002.
[22] Y. Liu, K. Liu, and M. Li, "Passive Diagnosis for Wireless Sensor Networks," *IEEE/ACM Trans. Networking*, vol. 18, no. 4, pp. 1132-1144, Aug. 2010.
[23] Y. Bejerano and R. Rastogi, "Robust Monitoring of Link Delays and Faults in IP Networks," *IEEE/ACM Trans. Networking*, vol. 14, no. 5, pp. 1092-1103, Oct. 2006.
[24] N. Duffield, "Network Tomography of Binary Network Performance Characteristics," *IEEE Trans. Information Theory*, vol. 52, no. 12, pp. 5373-5388, Dec. 2006.
[25] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "NetDiagnoser: Troubleshooting Network Unreachabilities Using End-to-End Probes and Routing Data," *Proc. Third Int'l Conf. Emerging Networking Experiments and Technologies (CoNEXT)*, 2007.
[26] R.R. Kompella, J. Yates, A. Greenberg, and A.C. Snoeren, "IP Fault Localization via Risk Modeling," *Proc. Second Conf. Symp. Networked Systems Design and Implementation (NSDI)*, 2005.
[27] S. Kandula, D. Katabi, and J.-P. Vasseur, "Shrink: A Tool for Failure Diagnosis in IP Networks," *Proc. ACM SIGCOMM Workshop Mining Network Data (MineNet)*, 2005.
[28] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D.A. Maltz, and M. Zhang, "Towards Highly Reliable Enterprise Network Services via Inference of Multi-Level Dependencies," *ACM SIGCOMM Computer Comm. Rev.*, vol. 37, no. 4, pp. 13-24, Aug. 2007.
[29] H.H. Song, L. Qiu, and Y. Zhang, "NetQuest: A Flexible Framework for Large-Scale Network Measurement," *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 34, no. 1, pp. 121-132, June 2006.
[30] C.E. Leiserson, R.L. Rivest, C. Stein, and T.H. Cormen, *Introduction to Algorithms*. MIT Press, 2001.
[31] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, and X. Li, "Does Wireless Sensor Network Scale? A Measurement Study on GreenOrbs," *IEEE Trans. Parallel and Distributed Systems*, vol. 24, no. 10, pp. 1983-1993, Oct. 2013.
[32] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
[33] A. Vahdat et al., "Epidemic Routing for Partially Connected Ad Hoc Networks," Technical Report CS-200006, Duke Univ., 2000.
[34] Tinyos, http://www.tinyos.net/, 2014.
[35] Greenorbs, http://www.greenorbs.org/, 2014.
[36] W. Gong, K. Liu, X. Li, X. Miao, and T. Zhu, "Quality of Interaction for Sensor Network Energy-Efficient Management," *The Computer J.*, vol. 56, no. 8, pp. 926-937, 2013.

Wei Gong received the BS degree from the Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2003, and the MS and PhD degrees from the School of Software and the Department of Computer Science and Technology, Tsinghua University, in 2007 and 2012, respectively. His research interests include wireless sensor networks, RFID applications, and mobile computing. He is a member of the IEEE.



Kebin Liu received the BS degree from the Department of Computer Science, Tongji University, and the MS degree from Shanghai Jiaotong University, China. He is a joint PhD student in the Department of Computer Science and Engineering at Shanghai Jiaotong University and the Department of Computer Science and Engineering in the Hong Kong University of Science and Technology, under the supervision of Dr. Yunhao Liu. His research interests include sensor networks and distributed systems. He is a member of the IEEE.

Yunhao Liu received the BS degree in automation from Tsinghua University, Beijing, China, in 1995, and the MS and PhD degrees in computer science and engineering from Michigan State University, East Lansing, in 2003 and 2004, respectively. He is a professor at the School of Software, Tsinghua National Lab for Information Science and Technology, and director of the MOE Key Lab for Information Security, Tsinghua University. He is a senior member of the IEEE.