Coverage-Based Lossy Node Localization in Wireless Sensor Networks Using Chi-Square Test

Forrest Sheng Bao
Dept. of Electrical and Computer Engineering
University of Akron
Akron, OH 44325
forrest.bao [AT] gmail.com

Wu-Jün Zhou
California State University
Fullerton, CA 92831
wujun_zhou [AT] csu.fullerton.edu

Wu Jiang
R&D Division
Truveris Inc.
475 Park Ave. S.
New York, NY 10016
wu [AT] morediff.info

Chen Qian
Dept. of Computer Science
University of Kentucky
Lexington, KY 40506
qian [AT] cs.uky.edu

***Abstract*—Locating lossy nodes in wireless sensor networks (WSNs) is difficult due to the large amount of sensor nodes, and their limited resources. The state-of-the-art work frames lossy node localization in WSNs as an optimal sequential testing problem guided by end-to-end data. It combines both active and passive measurements to minimize testing cost and number of iterations. However, this hybrid approach has many limitations. Inspired by the success of statistic methods in coverage-based software testing, and the similarity between software testing and lossy node localization, we develop an improved approach by employing Chi-square test in WSN lossy node localization. Supported by well-established statistic theories, our elegant approach delivers great performance. Experiments on randomly generated networks and deployed networks show significant performance improvement using the proposed algorithm. We expect to use this approach for other diagnostic problems in WSNs.**

I. Introduction

In Wireless Sensor Networks (WSNs), data are often sampled by nodes outside the transmission range of sink nodes. They are relayed to sinks by intermediate nodes. If intermediate nodes become lossy, i.e., dropping packets, sensing data will be lost in transmission. In order to maintain network performance, it is important to locate[1] lossy nodes.

Existing approaches can be categorized into two groups: active and passive. Active approaches [1]–[3] generate query traffic which in turn consumes precious resources of sensor nodes. Passive approaches [4]–[6], using existing end-to-end data, are more suitable for resource constraints of WSNs but they could also lead to false detection of lossy nodes. A better strategy is to combine the two approaches.

Combining active and passive approaches, the state of the art is a framework developed by Wang et al. that formulates lossy node localization problem as an iterative sequential testing problem [7]. In this framework, suspect lossy nodes are first inferred and ranked according to end-to-end data. They are then sequentially tested based on their ranks. The first step is passive while the second is active. Carefully combining active and passive (end-to-end data) measurements, this hybrid approach provides a good balance between accuracy and cost. Compared with previous work, it reduces testing cost and number of iterations.

[1]In this paper, we refer to topological location instead of geographical location.

However, several assumptions in this approach greatly limit its performance and scalability. First of all, it assumes that good links have reception rate (i.e., package delivery rate) at least α while lossy links have reception rate below β. For example, $\alpha = 0.95$ and $\beta = 0.6$ in [7]. Setting the two thresholds is empirical. It does not automatically adapt for different networks or when the performance of the same network changes. This reduces its robustness. Second, the maximum length of paths is assumed to be $h = \log_\alpha \beta$. This places a limitation on network size. We observe from experimental study on deployed networks that this assumption does not hold for large-scale WSNs. Path length usually go beyond h. The last assumption that limits the scalability of the framework is that only terminal nodes of paths sample data. In practice, intermediate nodes can also generate data.

Besides limitations, there is room to further reduce number of iterations and testing cost, the two performance metrics on which Wang et al. claim to outperform other approaches. When links and paths are categorized as lossy or good by thresholding, information about how lossy or good they are is lost. If the degrees of lossy can be used, it is possible to generate more optimal testing sequences.

To remove limitations and also to improve the performance of the innovative framework introduced by [7], in this paper, we propose a new approach to ranking suspect lossy nodes. Bug localization based on coverage testing results and Chi-square test has been proven effective in software engineering [8]. Observing the similarity between lossy node localization and software bug localization, here we apply coverage-based Chi-square test for solving lossy link localization. In our approach, two reception rate thresholds α and β used in [7] are no longer needed. Instead, we use the result of every end-to-end transmission, no matter whether it is successful. Dropping the two thresholds also allows the algorithm to work on much larger networks, without path length limitation.

In summary, our main contribution is a new suspect lossy node identification and ranking algorithm, which can be used to replace its counterpart in original framework. This change allows the framework to work with much less restrictions and prompts the performance greatly. Experimental evaluation shows that our approach can reduce the number of iterations by over 50%, on both randomly generated networks and real deployed networks. Even better, the improvement on number of iterations does not come at a price of increasing testing cost. In some cases, our approach can even further reduce the testing cost. Because of the similarity between lossy node localization and other troubleshooting problems in WSNs, we expect that this approach can be extended to other diagnostic problems in WSNs in the future.

The rest of the paper is organized as follows. In Section II, we formulate the node ranking problem. Our methodology and algorithm are given in Section III. Section IV reports our comprehensive empirical comparison between our method and the state of the art. After discussions in Section IV, Section V concludes the paper.

II. Problem Description

A. Lossy node localization in WSNs

The framework proposed by [7] uses a nested iterative process, as shown in Algorithm 1. This is the first approach combining active and passive measurements in lossy node localization for WSNs. It outperforms previous work and we intent to further improve it in this paper.

In each inner iteration of Algorithm 1, nodes are ranked based on end-to-end data. Then a subset of nodes are checked sequentially based on their ranks and replaced if found lossy. The inner iteration stops when at least one node is checked for every lossy path. Since paths consist of links between nodes, a path is lossy if it contains at least one lossy link [2]. The algorithm continues until there is no packet dropped in the network.

Algorithm 1: Overall framework introduced by [7]

```
1 repeat
2     repeat
3         1. Collect passive data
4         2. Rank nodes based on testing results
5         3. Check a subset of nodes from top of the list
             and replace them if needed
6     until at least one node is checked on a lossy link;
7 until no packets dropped in the network;
```

In this paper, our contribution goes to Step 2 of the inner loop where nodes are ranked. Wang et al. has their own ranking algorithm which inferences on contracted network [7]. They also propose several heuristics to speed up the inference for certain network topologies. We will use the same framework but with our ranking algorithm to be introduced in Section III.

[2]Our definition to "lossy link" is slightly different from that in [7]. Here, a link is lossy if at least one end of it is a lossy node. In [7], a lossy link is a link whose reception rate is below the threshold β. In other words, in this paper a link is lossy if at least one packet is dropped instead of dropping enough fraction of packets. Hence, our definition to "lossy link" is stronger than that in [7].

B. Node ranking problem

Before we introduce our node ranking algorithm, we first formulate the problem.

Definition 1 (Node ranking problem)**.** *Given a WSN consisting of nodes $\mathcal{N} = \{n_1, \ldots, n_M\}$ and transmission results $\mathcal{T} = \{t_1, \ldots, t_T\}$, where each t_i $(1 \leq i \leq T)$ is a tuple $\langle c, r\rangle$ such that $c \subseteq \mathcal{N}$ and $r \in \{failed, passed\}$, rank all nodes in $\mathcal{N}$.*

Each transmission result t_i has two parts, the nodes on the path of an end-to-end transmission and the status (i.e., failed or passed). We say a transmission $t_i = \langle c, r\rangle$ **covers** a node n_j if $n_j \in c$. For example, a transmission result $\langle\{n_1, n_2, n_5\}, failed\rangle$ means that the transmission via nodes n_1, n_2 and n_5 failed, and the transmission t_i covers the nodes n_1, n_2 and n_5.

Based on transmission results, ranks of nodes can be calculated. The higher a node is ranked, the more likely the node is lossy and thus should be checked first.

III. Coverage-Based Node Ranking Using Chi-Square Test

In this section we propose our solution to the node ranking problem. Our approach is inspired by the work of [8] which has been proven effective in identifying bug-causing lines of code in software engineering. We call our method "*coverage-based*" because we assume that lossy nodes are likely covered by failed transmissions, whereas good nodes are likely covered by passed transmissions.

A. Methodology

The foundation of our method is Chi-square test [9], a statistic measure to test hypotheses. By computing Chi-square statistic, we can know how strongly a node is associated with transmission results (both passed and failed). The next step is to tell whether the node is associated with passed transmissions or failed transmissions. Only the association with failed transmissions is our interest, because lossy nodes cause failed transmissions. We use a method proposed in [8] to determine whether the association is with failed transmissions.

For a given node n_i, we propose a null hypothesis H_0:

Transmission results are independent from the node n_i.

To test this hypothesis, Chi-square statistic is estimated [8]:

$$\chi^2(n_i) = \frac{(C + F + P + N) \cdot (CF \cdot NP - CP \cdot NF)^2}{C \cdot F \cdot P \cdot N} \tag{1}$$

where

- CF: number of transmissions **C**overing n_i and **F**ailed
- CP: number of transmissions **C**overing n_i and **P**assed
- NF: number of transmissions **N**ot covering n_i and **F**ailed
- NP: number of transmissions **N**ot covering n_i and **P**assed

and

- $C = CF + CP$
- $F = CF + NF$

- $P = CP + NP$
- $N = NF + NP$

The relationship between the terms in Eq. (1) can be represented using a contingency table (Table I).

TABLE I: The contingency table for Chi-square test

	node is covered	node is not covered	$\sum$
transmissions failed	CF	NF	F
transmissions passed	CP	NP	P
$\sum$	C	N	

Chi-square statistic $\chi^2(n_i)$ indicates the degree of association/dependency between transmission results and the status (i.e., lossy or not) of the node n_i. Hence, we can use it to rank nodes and help identify suspicious lossy node.

However, a large value of $\chi^2(n_i)$ implies both the association between a lossy node and failed transmissions, and the association between a good node and passed transmissions. To locate lossy nodes, we only care about the former one. In order to distinguish the two kinds of association, and to tell whether node n_i is lossy or good, we use another measure introduced in [8]:

$$\varphi(n_i) = \frac{CF/F}{CP/P}.$$

A large value of $\chi^2(n_i)$ and $\varphi(n_i) > 1$ indicate that node n_i is likely responsible for failed transmissions. There is no need to check nodes whose $\varphi(n_i) <= 1$.

The larger $\varphi(n_i)$ is, the more likely node n_i is associated with failed transmissions. Hence, we can use $\varphi(n_i)$ to scale up $\chi^2(n_i)$ for a better estimation on the likelihood that node n_i is lossy:

$$score(n_i) = \begin{cases} \chi^2(n_i) \cdot \varphi(n_i) & \text{if } \varphi(n_i) > 1 \\ 0 & \text{if } \varphi(n_i) <= 1 \end{cases} \quad (2)$$

In the end, nodes are ranked according to scores obtained from Eq. (2) in descending order. High ranking of a node implies that the node is very likely to be lossy and it will be checked before nodes ranked below it.

B. Algorithm

Our method has two steps: scoring nodes by Algorithm 2 and then ranking (sorting) nodes based on their scores. Since the second step, sorting, is a well-studied topic, here we focus on the first step.

For every node n_i ($i \in [1..M]$), Algorithm 2 first scans over all transmission results to count four numbers $CF[i]$, $NF[i]$, $CP[i]$, and $NP[i]$. Then it computes Chi-square statistic $\chi^2(n_i)$ and $\varphi(n_i)$. In the end, it estimates the score.

Since our scoring algorithm scans through end-to-end transmissions only once, it is linear with respect to the problem size, the sum of the number of transmissions T, and the number of nodes M. In implementation, we add 0.1 to each term in denominators in Algorithm 2 to avoid division by zero.

Algorithm 2: Scoring nodes

```
input: transmission results $\mathcal{T}$, number of nodes $M$
Initialize 4 empty 1-by-$M$ arrays: $CF$, $NF$, $CP$, $NP$.
foreach $t_j = \langle c, r \rangle \in \mathcal{T}$ do
    if $r == failed$ then
        if $t_j$ covers $n_i$ then $CF[i]++$
        else $NF[i]++$
    else
        if $t_j$ covers $n_i$ then $CP[i]++$
        else $NP[i]++$
Initialize one empty 1-by-M array, denoted as $score$.
for $i = 1$ to $M$ do
    $C = CF[i] + CP[i]$
    $F = CF[i] + NF[i]$
    $P = CP[i] + NP[i]$
    $N = NP[i] + NF[i]$
    $\varphi = \frac{CF[i]/F[i]}{CP[i]/P[i]}$
    if $\varphi > 1$ then
        $\chi^2 = \frac{(C+F+P+N)\cdot(CF[i]\cdot NP[i]-CP[i]\cdot NF[i])^2}{C\cdot F\cdot P\cdot N}$
        score[i] = $\chi^2 \cdot \varphi$
    else
        score[i]=0
return score
```

C. An illustrative example

Now we use an example to illustrate how the proposed algorithm can score and rank nodes based on transmission results.

As shown in Fig. 1, the example WSN consists of 8 nodes, including a source node n_1 and a sink node n_8. When n_1 tries to send data to n_8, the rest nodes (n_2 to n_7) perform as intermediate nodes in charge of forwarding and receiving data. Let n_5 be the lossy node, which drops packets randomly.

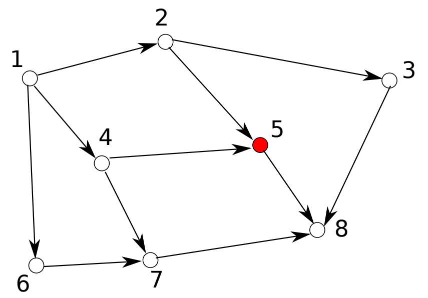


Fig. 1: The example WSN

Six transmissions are carried out. Their coverage and status are shown in Table II, where a 1 (or 0) in row tx_j column n_i means that the j-th transmission covers (or does not cover) node n_i, and the last column indicates whether the transmission fails or passes.

Then we count the numbers needed to compute scores for all nodes, as shown in Table III.

TABLE II: Transmission coverage and status

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	**status**
tx_1	1	1	1	0	0	0	0	1	passed
tx_2	1	1	0	0	1	0	0	1	failed
tx_3	1	0	0	1	1	0	0	1	passed
tx_4	1	0	0	1	0	0	1	1	passed
tx_5	1	0	0	0	0	1	1	1	passed
tx_6	1	0	0	1	1	0	0	1	failed

TABLE III: Numbers to computer score

node	**CF**	**CP**	**NF**	**NP**	**C**	**F**	**P**	**N**
n_1	2	4	0	0	6	2	4	0
n_2	1	1	1	3	2	2	4	4
n_3	0	1	2	3	1	2	4	5
n_4	1	2	1	2	3	2	4	3
n_5	2	1	0	3	3	2	4	3
n_6	0	1	2	3	1	2	4	5
n_7	0	2	2	2	2	2	4	4
n_8	2	4	0	0	6	2	4	0

In the end we compute scores for all nodes and rank them, as shown in Table IV. According to Table IV, node n_5, the lossy node indeed, is ranked the highest and should be examined first.

TABLE IV: Scores and ranks

node	χ^2	φ	**score**	**rank**
n_1	0	1	0	3
n_2	0.75	2	1.5	**2**
n_3	1.2	0	0	3
n_4	0	1	0	3
n_5	6	4	24	**1**
n_6	1.2	0	0	3
n_7	3	0	0	3
n_8	0	1	0	3

Note that nodes of the same rank will be checked in a random order.

IV. Experimental Evaluation

In order to compare our algorithm with the state-of-the-art approach, we carry out a series of experiments on networks of different parameters, including both randomly generated networks and real deployed networks.

A. Testing topologies

Two types of topologies are used to compare the performance of algorithms: randomly generated ones and the ones derived from a well-known deployed WSN system GreenOrbs [10].

To better model real applications, in our simulation, all nodes except the sinks are sources. On the contrary, in experiments of [7], only leaf nodes of the reverse routing tree are sources.

1) Random topologies: On randomly generated topologies, in order to provide a fair comparison, we keep the parameters and settings as close as possible to those used in [7]. Sensor nodes are distributed in a 10 unit by 10 unit square. A single sink is placed in the center. In order to test the effect that network density may cause, we choose two network densities: 200 nodes and 500 nodes. The transmission range between nodes is 1 unit. At a given point of time, a static reversed tree is formed from sources to the sink [7] and provides routing information. All packets follow the reverse tree to sinks. The number of branches of intermediate nodes on the tree distributes uniformly in the interval $[1, b]$ when generating the tree. b is called the branching ratio. We choose $b = 5$ and $b = 10$ which are used in [7]. Parameters used only by the algorithm in [7] are left intact.

TABLE V: Parameters for random networks

parameter	**values**
branching ratio (b)	5 or 10
number of nodes	200 or 500

Table V lists parameters used to generate different random networks in simulation. Therefore, we have 4 types of random networks.

2) Real deployed topologies: In addition to random networks, we compare algorithms using the trace data of GreenOrbs [10], a large-scale WSN project deployed in Tianmu Mountain, China. We focus on two different network deployments named “Network1” and “Network2”, with 179 and 199 TelosB motes respectively. When two nodes can receive packets from each other and the RSSI is higher than a threshold (-80 dBm), we consider that a communication link exists between them. We only consider the connectivity information and ignore others such as ETX, latency, and duty-cycles. Since trace data includes how packets are relayed, we do not form routing trees.

Due to the limitations mentioned earlier, the original algorithm in [7] cannot directly run on real deployed networks which are of large size. The following modifications are made to accommodate deployed networks:

1) Lengths of paths are used as h instead of heights of trees when computing threshold for separating lossy paths and good paths.
2) Responsible link inference is disabled if the length of path excesses maximum allowance.

B. Performance measuring

Two performance metrics, number of iterations and testing cost, are used. Both of them are the lower the better. Our definition to both metrics are the same as in [7]. To be specific, “number of iterations” means the number of times that the outer loop of Algorithm 1 is executed, and each node is assigned with a testing cost, uniformly distributed between 0 and 1.

We call every complete execution of Algorithm 1 as one “run.” Both our approach and the state-of-the-art approach are simulated 100 runs, and averages of the 100 runs are used to compare performances.

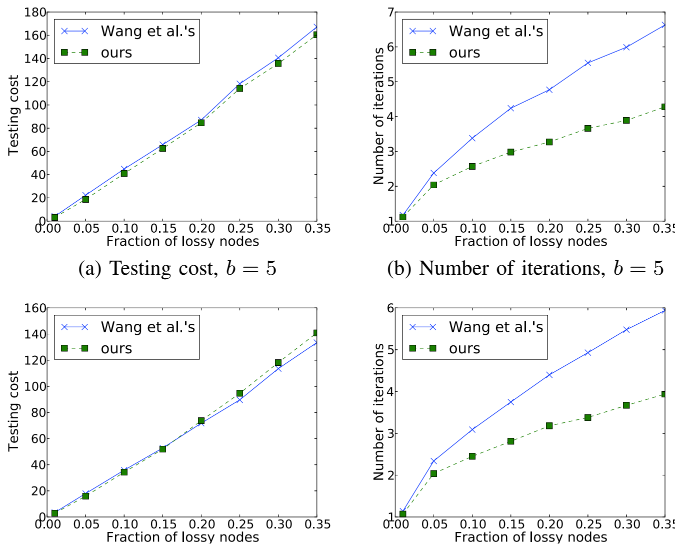
In each run, a portion of nodes are randomly chosen as lossy nodes. Lossy nodes randomly drop packets at a uniformly distributed rate between 0 and 1. The term “fraction of lossy node" refers to the ratio of lossy nodes to total number of nodes in the network [7]. We collect values of the two performance metrics while varying fraction of lossy nodes from 0.05 to 0.35 at the step of 0.05. We also estimate performance metrics when fraction of lossy nodes is 0.01.

For randomly generated networks, every source sends 400 packets to the sink in each iteration (i.e., the outer loop of Algorithm 1). For deployed networks, we use real packets from trace data. For either Network1 or Network2, trace data on two days, denoted as Trace1 and Trace2, are used to provide packets. In other words, packets are actual activities in deployed sensor networks.

C. Results

Consistent results are observed among different topologies that our algorithm can greatly reduce the number of iterations in all cases. Our advantage on number of iterations enlarges to over 50%, when fraction of lossy nodes increases to 0.35. In terms of testing cost, our approach can reduce the testing cost on real deployed networks and gives almost identical results on randomly generated networks.

From the simulation results, we can conclude that our algorithm outperforms the state of the art, especially on real deployed networks.

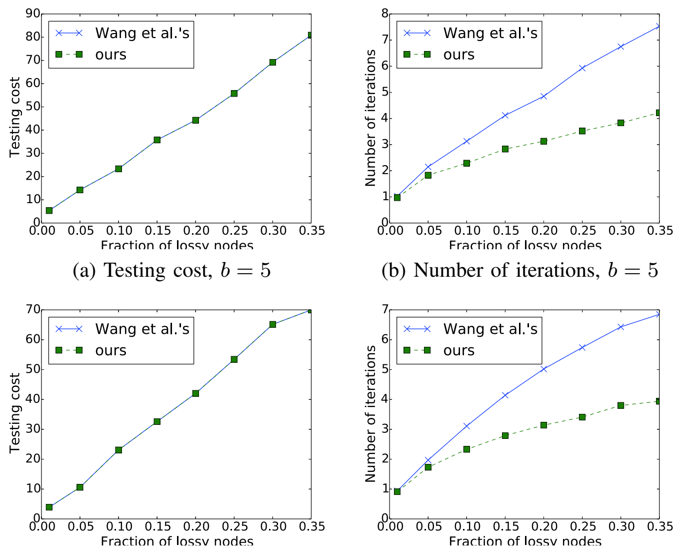


(a) Testing cost, $b = 5$ (b) Number of iterations, $b = 5$

(c) Testing cost, $b = 10$ (d) Number of iterations, $b = 10$

Fig. 2: Simulation results on random networks, 500 nodes

1) On random topologies: Figs. 2 and 3 show the simulation results on random networks of 500 nodes and 200 nodes, respectively. Our algorithm greatly reduces the number of iterations using testing cost similar to that of the state-of-the-art approach. This observation is consistent amount all 4 configurations (2 branching ratios and 2 network densities). When fraction of lossy nodes reaches 0.35, meaning that 35% of the nodes in the network are lossy, the number of iterations can be reduced by more than 50% (Figs. 2b and 2d, 3b and 3d). The testing cost for both algorithms are almost the same while sometimes our approach uses less cost on dense networks (Figs. 2a and 2c). On sparse networks (200 nodes), the curves of testing cost from both algorithms overlap (Figs. 3a and 3c).



(a) Testing cost, $b = 5$ (b) Number of iterations, $b = 5$

(c) Testing cost, $b = 10$ (d) Number of iterations, $b = 10$

Fig. 3: Simulation results on random networks, 200 nodes

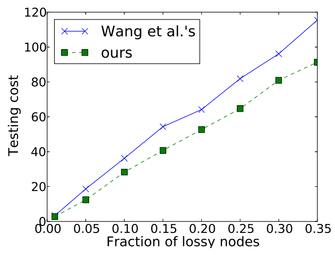
2) On deployed topologies: The results on two real deployed topologies Network1 and Network2 are shown in Fig. 4 and Fig. 5, respectively. Consistent results are obtained across 4 instances (2 networks and 2 traces for each network). Overall trends are the same as those on randomly generated networks. On both Network1 (Figs. 4b and 4d) and Network2 (Figs. 5b and 5d), the number of iterations can be greatly reduced by more than 50%.

Unlike the cases for randomly generated networks, our approach can also reduce testing cost by approximately 10% on both Network1 (Figs. 4a and 4c) and Network2 (Figs. 5a and 5c). This means that on real deployed networks, our approach can improve both metrics.

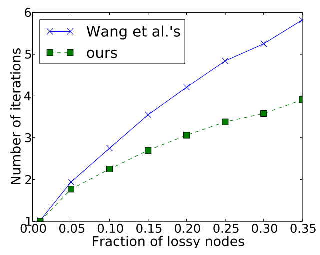
V. Discussions

We would like to address why our algorithm does not have large advantage in terms of testing cost. There are two reasons for this. First of all, in Wang et al.'s algorithm, many nodes are avoided from ranking by path contraction. They will never be ranked and therefore no testing cost will be wasted on them. In contrast, our algorithm does not have the step of path contraction. We plan to add this step into our future work in order to reduce both testing cost and number of iterations. Second, the price of Wang et al.'s low testing cost is the high number of iterations. The high number of iterations of Wang et al.'s algorithm actually helps reduce the testing cost.

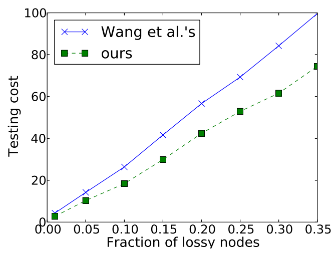
From experiments, we also find that Wang et al.'s algorithm has better performance when the reversed routing tree is near-balanced and has a high branching ratio near the root of



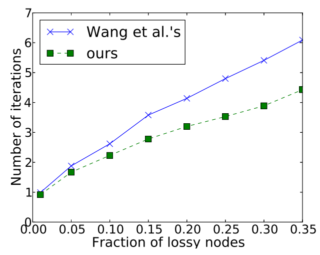
(a) Testing cost, Trace1



(b) Number of iterations, Trace1

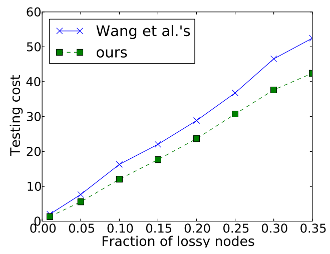


(c) Testing cost, Trace2

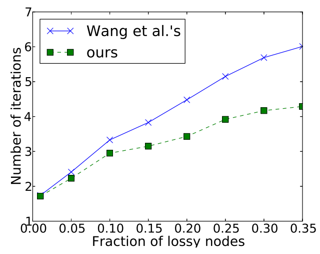


(d) Number of iterations, Trace2

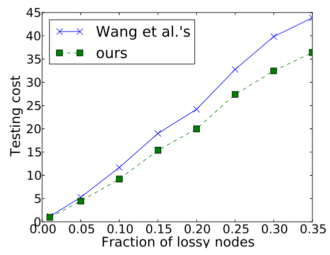
Fig. 4: Simulation results on deployed network Network1



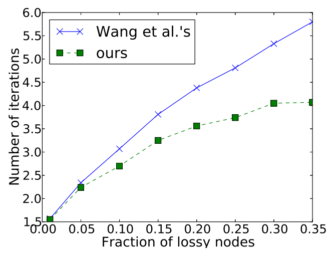
(a) Testing cost, Trace1



(b) Number of iterations, Trace1



(c) Testing cost, Trace2



(d) Number of iterations, Trace2

Fig. 5: Simulation results on deployed network, Network2

the tree. This is because in this situation, lossy nodes are distributed evenly in each branch. Hence, more lossy nodes can be inferred out and the testing on unnecessary nodes can be avoided.

Chi-square test only works well when there are enough data samples, e.g., abundant transmission results. For instance, if we do not have tx_6 in the example in Section III-C, nodes n_2 and n_5 will tie as top 1 whereas testing node n_2 is a waste of nodes' resources. When there are insufficient amount of data, Fisher's test should be used instead. Using the variables in contingency table (Table I), Fisher's statistic is:

$$\mathcal{F} = \frac{C! \cdot F! \cdot P! \cdot N!}{(C+F+P+N)! \cdot CF! \cdot NP! \cdot CP! \cdot NF!}$$

However we still choose Chi-square test for two reasons. First, WSNs are designed to collect a large amount of data and thus massive transmissions could occur easily. Second, Chi-square statistic has a lower time complexity which could better accommodate limited resources in WSNs.

VI. Conclusion

In this paper, we propose an improvement to the state-of-the-art method for solving lossy node localization problem in WSNs. Inspired by coverage-based software testing, we employ Chi-square test, a well-established statistic tool, for lossy node localization in WSNs. Our method eliminates several limitations in the state of the art, and boosts performance in terms of the number of iterations and testing cost. In particular, the improvement to the number of iterations is significant. Experiments show the merits our approach consistently across random topologies and real deployed topologies. In the future, we will further improve our algorithm to lower the testing cost. This approach can also be extended to other types of diagnostic problems in WSNs.

Acknowledgements

The authors would like to thank GreenOrbs Team [10] for sharing the network trace data.

References

[1] J. Zhao, R. Govindan, and D. Estrin, "Residual energy scans for monitoring wireless sensor networks," in *Proceedings of IEEE Wireless Communication and Networking Conference (WCNC)*, 2002.

[2] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Proceedings of Third International Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[3] S. Rost and H. Balakrishnan, "Memento: A health monitoring system for wireless sensor networks," in *Proceedings of Third Annual IEEE Communication Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2006.

[4] N. Duffield, "Network tomography of binary network performance characteristics," *IEEE Transactions on Information Theory*, pp. 1–32, 2006.

[5] P. P. C. Lee, V. Misra, and D. Rubenstein, "Toward Optimal Network Fault Correction in Externally Managed Overlay Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 3, pp. 354–366, 2010.

[6] H. Nguyen and P. Thiran, "Using end-to-end data to infer lossy links in sensor networks," in *Proceedings of IEEE International Conference on Computer Communications (InfoCom)*, vol. 00, no. c, 2006.

[7] B. Wang, W. Wei, H. Dinh, W. Zeng, and K. R. Pattipati, "Fault Localization Using Passive End-to-End Measurements and Sequential Testing for Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 3, pp. 439–452, 2012.

[8] W. E. Wong, T. Wei, Q. Yu, and L. Zhao, "A crosstab-based statistical method for effective fault localization," in *International Conference on Software Testing, Verification, and Validation*, 2008, pp. 42–51.

[9] B. S. Everitt, *The analysis of contingency tables*. Chapman and Hall, 1977.

[10] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, J. Zhao, and X.-Y. Li, "Does wireless sensor network scale? A measurement study on GreenOrbs," in *Proceedings of IEEE International Conference on Computer Communications (InfoCom)*, 2011, pp. 873–881.