Link Scanner: Faulty Link Detection for Wireless Sensor Networks

Qiang Ma^{1,2}, Kebin Liu², Xiangrong Xiao³, Zhichao Cao^{1,2}, Yunhao Liu^{1,2}

¹ Department of Computer Science and Engineering, Hong Kong University of Science and Technology

² School of Software and TNLIS, Tsinghua University, Beijing, China

³ School of Information Engineering, Zhejiang Agriculture&Forest University

{maq, kebin, xiaoxr, caozc, yunhao}@greenorbs.org

Abstract—In large-scale wireless sensor networks, it proves very difficult to dynamically monitor system degradation and detect bad links. Faulty link detection plays a critical role in network diagnosis. Indeed, a destructive node impacts its links' performances including transmitting and receiving. Similarly, other potential network bottlenecks such as network partition and routing errors can be detected by link scan. Since sequentially checking all potential links incurs high transmission and storage cost, existing approaches often focus on links currently in use, while overlook those unused yet ones, thus fail to offer more insights to guide following operations. We propose a novel scheme Link Scanner (LS) for monitoring wireless links at real time. LS issues one probe message in the network and collects hop counts of the received probe messages at sensor nodes. Based on the observation that faulty links can result in mismatch between the received hop counts and the network topology, we are able to deduce all links' status with a probabilistic model. We evaluate our scheme by carrying out experiments on a testbed with 60 TelosB nodes and conducting extensive simulation tests. A real outdoor system is also deployed to verify that LS can be reliably applied to surveillance networks.

I. INTRODUCTION

Wireless sensor networks (WSNs) have been widely used in many application areas such as infrastructure protection, environment monitoring and habitat tracing. The reliability of individual links' transmissions is crucial in these applications, e.g., in a surveillance network [14], the transmissions must be reliable to avoid false alarms and missed detections. Compared to the wired networks, it seems much more essential to detect link faults rather than node faults in WSNs. A wireless link itself virtually exists, which means we can't directly observe and assess whether it performs well or not [17], [21]. What is more, It proves difficult to localize the faulty links under a dynamic mal-condition in the wild, for the link quality will significantly be impacted by the nature environment like trees in the forest and flow in the ocean [3], [14].

Multi-hop networks always suffer more harm than single-hop networks due to link failures. For example, a critical link may cause a large area of partition, or significantly interfere with routing protocol among the nodes, producing some problems such as routing cycle and even network partition. Compared to single-hop networks, faulty link detection is more difficult to proceed in the multi-hop networks due to their topology features. A packet has to traverse multiple links to the sink, it is for this reason that exactly localizing a faulty link becomes really hard if only on the basis of whether

the packet arrives at the sink or not. Therefore, faulty link detection becomes one of the most critical issues in multi-hop network diagnosis. Indeed, according to the status of a link, we are able to explain many failures like packet loss, routing failure, partition and so on. Notably, link performance actually reflects a network's reliability and bottleneck if exist.

Although single link failures are more common, multiple link failures occur due to shared risks such as failure of a link while another link is under maintenance, or natural disasters that cause links traversing a region to fail. In [1], the authors use monitoring paths and cycles to localize single link and Shared Risk Link Group (SRLG) failures. They also prove that $(k+2)$ -edge connectivity¹ was necessary and sufficient to uniquely localize all SRLG failures involving up to k links with one monitor. In practice, however, not all sensor networks can satisfy this strict condition, especially in the cases we spread the sensor nodes randomly in the area of interest. In addition, in most cases we are not allowed to set any more monitors after the deployment. What we expect is to utilize the rule-free probes (i.e., without computing the exact probing paths) to achieve our detection.

One of the most peculiar routing characteristics of WSN is routing dynamics. It is not surprising that a sensor node frequently changes its parent to forward packets. Unfortunately, many existing approaches just aim to detect the faulty links which had been behaving badly, but fail to offer an inspection report about other unused yet ones, thus never further suggest the nodes how to reroute when the current routing path is less than satisfactory. To solve the above problems, in this work we propose Link Scanner (LS), a novel probe-based and rule-free detection approach for discovering faulty links in sensor networks. The object of LS is to provide a blacklist containing all possible faulty links. With such a blacklist, further analysis and recovery processes become possible, including (i) exploring the root causes of observed symptoms in the network, (ii) adjusting routing strategy for the related nodes, (iii) offering the spare list of links for every node. As a result, we not only achieve the goal of diagnosis, but also take a big picture of current network, which guides the following applications. In LS, we first flood a probe message in the network. Then each node is expected to receive multiple

¹A network is said to be k -edge-connected if the removal of any $k-1$ links will retain network connectivity.

Fig. 1. LS Design Structure. The module of Probe flooding presents the front-end action, including flooding protocol, data reduction and collection. Code Generator is used to initiate back-end processing program. Diagnosis module depicts the detailed process. Data preprocessing converts the raw data to DLV language, including the fundamental facts, rules and constraints. Fault report finally generates a feedback of parameters for DLV to modify the weak constraints.

probe messages through different paths. Each probe message contains a hop counter that records the number of hops from sink to current node. Since faulty links may cause probes dropped, there must be mismatches between the received hop counts in sensor nodes and our expectations according to the topology. We then analyze the mismatch information from sensor nodes with a probabilistic and heuristics based inference model thus deduce the faulty links. Specifically, the main contribution of this paper can be summarized as follows:

- To the best of our knowledge, we are the first to investigate a probe-based and rule-free method of detecting the faulty links including those potential but not used yet ones in sensor networks.
- According to the unique features of sensor networks, we design an efficient probe marking scheme that reveals the inner dependencies of sensor networks.
- LS proposes hierarchical inference models to infer the faulty links which capture the multi-level dependencies among the network elements and achieve high accuracy. We further introduce a learning-based inference scheme which increases the inspection accuracy and is thus scalable for large scale networks.
- We evaluate the performance of LS on a real testbed and extensive simulation, the results show that LS indeed detects most faulty links accurately, and helps in exploring the root causes of observed symptoms. A field study on a real outdoor deployment is also presented to verify that LS is practical to surveillance networks.

The rest of the paper is organized as follows. Section II presents our design and provides additional techniques to deal with several practical issues on implementation. Section III and IV show the performance evaluation results from both real indoor testbed experiments and simulations. An outdoor field study is presented in section V. Section VI summarizes the related work. Section VII concludes the paper.

II. MAIN DESIGN

We consider this problem in a large-scale wireless sensor network, and the network topology is known [6], [23], [4], [19]. A link exists if it's length is within the communication range of radio. In our previous work [10], [9], we find that most of links, which either have a good PRR (Packet Reception Ratio) more than 90%, or have a bad PRR less than 10%. Therefore, in this work, the basic idea is to use one flooding probe to check all the links. If a link is good, this probe should traverse on it, otherwise not.

In this design, we consider both efficiency and diagnosis accuracy. First, in order to reduce the transmission overhead for each node while still obtaining sufficient information for inferring the link failures, we need to compress the local information in a certain way, then proceed the parse work at the base station. Second, our LS should be able to work even if no sensor node knows its location, which also means that no computational operation at the sensor side. Most importantly, we desire to explore the failures on currently used links, as well as those unused links which potentially may fail.

A. Overview

In traditional ways of using active probes [25], [1] to detect faulty links, the monitor first maps out the paths for the active probes, then diagnoses the network according to the symptoms or other information obtained through probes. In our LS design (Figure 1), we simply issue a broadcast message in the network, and each node is expected to receive multiple probe messages through different paths. Each probe message contains a hop counter that records the number of hops from sink to current node. As faulty links may impact this process, there will be mismatches between the received hop counters in sensor nodes and our expectations based on the topology. We then analyze such mismatch with a probabilistic inference model and deduce the faulty links. What we gain from the detection is a global diagnosis for the network, including how to explain the system performance as well as the prediction about the potential system bottleneck like contention and routing failures.

B. Probe Flooding

Recording all node IDs along the collection path of each data packet or calculating PRR of different paths are frequently used for estimating link qualities. Clearly, these two ways consume much overhead in information preservation and propagation. Also, since links actually exist in WSN in a virtual manner, it proves difficult to observe and assess all links' performance with these fixed-path probes. They may fail to perceive those links which are not used yet but useful when the nodes reroute in the future.

For a certain node, its flooding probe can be overheard by all its neighbors, i.e., the other end of each possibly existing link can respond to the broadcasting. For example, as shown in Fig. 2, the sink actively broadcasts a probe. Node A and B hear the probe through one hop, and continuously broadcast it, so node C fetches this probe through two hops whatever

Fig. 2. Probes in Flooding. When sink and node B,C respectively broadcasts the probe, node A should overhear all of them. Notably, the probe from sink is only 1-hop as well as the other two are 2-hop and 3-hop (i.e., $sink \rightarrow B \rightarrow A$ and $sink \rightarrow B(A) \rightarrow C \rightarrow A$). But we will fail to distinguish the two 2-hop probes at C: $sink \rightarrow A \rightarrow C$ and $sink \rightarrow B \rightarrow C$.

from A or B. Since node C isn't aware of its location, it will broadcast the probe to ensure the coverage of flooding. Eventually, node A receives three probes from sink, node B and node C respectively, i.e., the number of received probes should be equal to that of in-edges. To cover all the possible links in the network, besides the number of probes, the record also needs to distinguish the received probes. Now if A only gets two probes, for example, from sink and C, it means that either B fails to get the probe from the sink, i.e., link $sink \rightarrow B$ is bad; or the link $B \rightarrow A$ is broken. In addition, if B records that it actually gets the probe, we obviously can infer that link $B \rightarrow A$ is broken.

A straightforward way to distinguish the probes is adding the node ID into the probes, so that each node records the parent where the probe comes from. Such an approach, being beneficial and accurate in a small network, or some of specific topology, may not work well for large-scale sensor networks, or dense networks where the number of neighbors may be more than 30: where even we use 2 bytes to identify nodes means each node should consume 60 bytes overhead. Clearly, we need to set up a good tradeoff between the transmission overhead and information usefulness. Based on our recently deployed sensor network system, GreenOrbs [14], we observe that hop count can distinguish different probes to some degree. For example, if node A records the corresponding hop count for every probe, we can expect that there is one for each hop count number, e.g., one-hop probe comes directly from sink, 2-hop probe is from node B and the 3-hop one is from node C. To record the hop count number consumes much less resource than to record the node ID. In our design, we use the data structure in the form $(MinHop, n_0, n_1, \dots, n_{k-1})$, where $MinHop$ represents the minimum hop count in the received probes and n_0 represents the number of probes with $MinHop$ -hop. Similarly, n_t ($0 \leq t \leq k-1$) is the number of probes with $(MinHop + t)$ -hop, and $MinHop + k - 1$ is the maximum hop count in the received probes.

In many cases, each node only needs to maintain a record of 4 bytes, i.e., $(MinHop, n_0, n_1, n_2)$. For each node A, all A's probes potentially come from A's neighbors. Assume that A's $MinHop$ is C, then every A's neighbor (e.g., B) possibly will receive A's $(C+1)$ -hop probe such that the hop count of B's probe is at most $C+2$. So A only needs to record the hop count

Fig. 3. Probe Lost. Node N gets a 1-hop probe from sink while node N_k fails to get the probe from N since link $N \rightarrow N_k$ failed. Along another path $sink \rightarrow N_1 \rightarrow N_2 \dots \rightarrow N_k$, N_k gets a k-hop probe, which is the only probe received by N_k . So N_k broadcasts a $(k+1)$ -hop probe. If N receives this probe, it maintains a record like $(1, 1, 0, \dots, 1)$, where there are $k-1$ '0's.

between C and $C+2$ (including C and $C+2$). However, some corner cases happen due to asymmetric links as illustrated in Fig. 3. Node N_k ($k > 2$) loses node N's 2-hop probe but gets a k-hop probe along the path $N_1 \rightarrow N_2 \dots \rightarrow N_k$ which is its only probe. So N_k should broadcast a $(k+1)$ -hop probe to its neighbors. If N actually receives this probe, it finally reports like $(1, 1, 0, \dots, 1)$, where there are $k-1$ '0's. This kind of asymmetric information extremely improves our latter inference but costs much transmission overhead. To reduce the transmission overhead, we design each byte in the record as follows:

- If the first bit is '0'. That means it indicates a number of hop count.
- Else the first bit is '1'. Then the remaining 7 bits indicate the number of successive '0' in the record.

With our design, in Fig. 3, the $k-1$ successive '0's can be represented by one byte " $1x_6x_5x_4x_3x_2x_1x_0$ ", where " $0x_6x_5x_4x_3x_2x_1x_0$ " equals to $k-1$ after binary-decimal conversion. This design is feasible as we assume that there are no more than "01111111", i.e., 127 probes traverse the same hop count, and no more than 127 '0's in the final report.

In practice, however, much information vagueness still exists when we are inferring faulty links with hop count. For example, in Fig. 2, assume that both A and B perform as our expectation (i.e., $(1, 1, 1, 1)$), and C only gets one 2-hop probe (i.e., $(2, 1)$). In this case, what we can accurately infer is that C misses a probe from A or B, that is, one of link $B \rightarrow C$ and link $A \rightarrow C$ is broken. To solve this problem, the most straightforward metric is link length, i.e., without any information provided by other nodes, we can simply deduce that the longer link may result in probe loss. In a large-scale network with a more intricate performance, simply relying on the relationship between link length and PRR is proved to be inaccurate, thus we need to consider more link features based on real system observations.

C. DLP

DLP (Disjunctive Logic Programming) is a formalism representing indefinite information. Similar to Prolog, language statements consist of facts, inference rules, strong constraints and weak constraints. In DLP, we initiate some facts based on our observation, which are definitely true and used to explore

FACT	MEANING
$node(N, X, Y)$	a node with ID as N and location (X, Y)
$e(A, B, HopCount)$	a directional link from A to B , in which the probe is $HopCount$ -hop.
$lost(A, B)$	$e(A, B, HopCount)$ exists but B did not receive A 's probe.
$g(A, HopCount)$	node A 's neighbor group with hop count as $HopCount$.
$inGroup(B, g(A, HopCount))$	node B is in $g(A, HopCount)$.
$noProbe(A)$	node A has not received any probe yet.
$distance(A, B, Length)$	the physical distance between A and B is $Length$
$lostCount(A, HopCount)$	the number of A 's lost probes with hop count as $HopCount$.
$faultyLink(A, B)$	Link $A \rightarrow B$ is detected as a faulty link.

TABLE I
DLP FACTS

potential possibilities, or models in a disjunctive logic program. In addition, we need to enumerate both the strong constraints and weak constraints. Each strong constraint only presents a conjunction of facts, while a weak constraint additionally assigns a numeric cost to the conjunction. The program will prune the models which violate any strong constraints, then rank the remaining models according to the weak constraints they violate, finally outputs the lowest cost model of inferred facts generated from the observation facts and inference rules.

The specific DLP implementation we use is DLV [8]. In DLV, we set the disjunctive inference rules as follows:

$$fact_1 \vee fact_2 \vee \dots \vee fact_k : - fact.$$

In this rule $fact$ should be pre-defined as a possible truth either input as an observation, or inferred by others. $fact_i (1 \leq i \leq k)$ are disjunctive, which means $fact$ must infer one and only one of $\{fact_1, fact_2, \dots, fact_k\}$. Besides, we can define a strong constraint as:

$$: - fact_1, fact_2, \dots, fact_k.$$

and a weak constraint as:

$$:\sim fact_1, fact_2, \dots, fact_k. [Weight : Level].$$

As mentioned above, $fact_1, fact_2, \dots, fact_k$ is a conjunction of facts. If a model includes a conjunction in a strong constraint (i.e., violates a strong constraint), it will be removed from the set of solutions. Weak constraints can be weighted according to their importance (i.e., the higher the *weight*, the more important the constraint). In the presence of weights, best models minimize the sum of *Weight* of violated weak constraints. *Level* can be used to prioritize weak constraints but omitted in our program.

D. Probabilistic Reasoning

In order to detect all the potential links even if they haven't been used yet, we consider every link, provided the physical distance of the corresponding two nodes is within effective communication range. In our expectation, every link should generate two probe records (i.e., two-way broadcast). If the collection record mismatches with our expectation, we can judge that some links must fail to deliver the probes. Due to the incompleteness and vagueness of information, we should find some metrics to infer the potential solutions.

1) *Data Pre-processing*: Raw record data must be converted into observation facts for DLP, including topology features and probe records. Topology features mainly depict the node location and link length, thus initialize our expectation collection during flooding. Probe records describe the observation facts, i.e., for node S and hop count C , how many probes with hop count C does S receive in the flooding? With topology knowledge, we can further group the neighbors of S by hop counts. For example in Fig. 2, we divide the neighbors of A into three groups. If we find that the probe with hop count 2 is lost, we can say that B fails to send the probe to A due to some certain reasons. Normally, one group contains more than one node. For node C , its neighbors A and B are in one group since the probes from both of them are 2-hop. When C 's records show that only one 2-hop probe is received, we can only say that 2-hop group loses one probe but not attribute to one specific node's fault.

In the program of DLP, we can refine the facts like what described in Table I. Based on these facts, we can set the inference rules to tell some relationships between them:

$$g(A, HopCount) : - e(B, A, HopCount).$$

$$inGroup(B, g(A, HopCount)) : - e(B, A, HopCount).$$

The first rule tells us that, if there exist two nodes, A and B , the distance between which is within the communication range, and the corresponding probe is $HopCount$ -hop. We should expect that A 's neighbor group with hop count as $HopCount$, i.e., $g(A, HopCount)$ must not be empty. What not described in this rule is that the number of probes should be equal to the size of this group. The second rule divides neighbors into groups, narrowing our inference choices of nodes when some probes are lost. Actually our program contains many other rules. As we can see, most of facts should be input by the records, like $e(A, B)$ and $distance(A, B, Length)$ completely depend on prior topology knowledge. The $noProbe(A)$ is determined by the collection records, for checking that whether A 's record exists at the sink or be implied by others' records.

What we want to know is the potential facts, inferred by the observation facts. To be precise, we roughly know how many probes are lost during the flooding process from the records. Furthermore, we can even know the losing probe's hop count number (i.e., which group of neighbors). What follows is to infer that which specific neighbor in the group

fails to broadcast the probe, so as to delve deeply into the exact faulty links. The fundamental inference rule is:

$$noProbe(A) \vee faultyLink(A, B) : - lost(A, B).$$

It means that when we believe that A didn't send its probe to B , there are two disjunctive possibilities. One is A has not received any probe from others; the other one is that link $A \rightarrow B$ performs poor. These two reasons may also be both true. In DLP, however, one and only one of them will be regarded as truth, and the other is ignored. Actually, we can easily judge whether A has probe or not in the collected reports.

2) *Cost Function*: Strong constraints are inviolable truths. Whatever models generated by DLP should not contain any conjunction of facts in a strong constraint. In LS there are two strong constraints:

$$: - \#count(A : lost(A, B), e(A, B, HopCount)) != lostCount(B, HopCount).$$

$$: - noProbe(A), not lost(A, B), edge(A, B, HopCount).$$

The first constraint guarantees that every lost probe of node B should be traced to its specific neighbor. In DLP, this constraint helps us to traverse every combination of neighbors in each group. The second one presents a simple logical contradiction, that is, if node A has no probes, every one of its neighbors must not receive any probe from A .

Weak constraints are set based on observed correlations between the links corresponding to engineering practices. Each practice holds as a general rule of thumb, but may be violated by an individual model. Thus the model that violates the fewest practices seems to be the best approximation of reality. Here we list the weak constraints in order of importance.

1. One node's fault may cause its neighbors simultaneously fail to receive the probe from it. That is, a faulty node makes all its out-going links look poor simply because there is no probe being sent out.
2. Similar with the first one, one node may simultaneously lose its neighbors' probes. That is, a faulty node make all its in-going links look poor.
3. Regional correlation. Many studies show that environment factor can significantly impact link performance. At the same time, regional channel collision sometimes bring a critical degradation of link qualities.
4. Symmetric links. Although there are a few asymmetric links in the network, most pairs of links perform consistently, especially for small packets.
5. The longer the distance between two nodes, the less the PRR, especially when the PRR is less enough, the tendency becomes more obvious.

The cost for a model is assigned based on the number of practices violated, weighted by the above importance of the practice. It also proves possible for DLP to output multiple equal-cost models. To give another group of possible solutions, after each round DLP generates a set of models, we then add one strong constraint in which the fact conjunction is set as same as the output into the program, hence the program will first eliminate the previous solutions.

Fig. 4. Indoor Testbed.

III. TESTBED EVALUATION

We first evaluate LS through a real indoor testbed consisting of 60 TelosB motes (Figure 4), where the network diameter is 7 hops. Two metrics are used for evaluating LS: false negative rate (i.e., miss detection rate) and false positive rate (i.e., false alarm rate). False negative rate is defined as the proportion of faulty nodes which are detected as normal, while false positive rate is defined as the proportion of normal nodes which are detected as faulty. Because our inferring process is based on record collection, fault reports may be different if each collection differs with the other. To examine the reliability of LS, we conduct 100 experiments for each group of comparison. We also take an external record as ground truth for comparison through USB serial channel, so as not to impact the regular probe flooding and collection.

In the first experiment, we examine how network scale affects the fault report, since intuitively more nodes may indicate more links. In the second experiment we change the network density, thus change the potential possibilities of probe transmissions. From the angle of DLV, this may bring more solution combinations, further disturb regular inference. What is more, a dense network may significantly degrade the probe flooding, hence impact our record for the real topology. Unlike the previous two, our third experiment utilizes the former fault reports to assist the current inference model to eliminate vagueness and correct the weak constraints about the link correlation.

A. Network Scale

We run a series of experiments with 20, 40, 60 motes respectively, with the same density and no feedback from the fault report (i.e., 100 experiments are totally independent). As described in Fig. 5, the more nodes in the network, the more difficulties existing in our inference. False negative rate stays about 7.3% when only 20 motes in the network, and it increases to 11.4% and 13.8% when network scale enlarges to 40 and 60 motes. False positive rate follows this trend as well as false negative rate. As we can see, when our network includes 60 nodes, one out of every 7 links which are reported as normal is actually bad, while one out of even

Fig. 5. Impact of network scale

Fig. 6. Impact of network density

Fig. 7. Impact of algorithm feedback

every 9 links which are reported as faulty is normal in fact. For each node, once it lost one probe, the more neighbors, the more combinations of solutions in DLV inference model. Furthermore, a large-scale network may have other problems like channel collision and routing error. Finally, from this set of results, LS is also proved to perform reliably under these three network scales.

B. Network Density

Network density can also significantly change the network topology. What is more, a dense network should suffer more channel collision and packet lost due to hidden terminal, thus may impact the probe flooding process and cover the real link performance. Here we define a network density in terms of average neighbor number. As presented in Fig. 6, we show three densities in a 60-node network. Clearly, when the network is sparse (i.e., average 8 neighbors), LS can achieve a false negative rate about 5%, which means there is only one out of 20 normal links reported by LS is faulty in fact, while the false positive rate is only 4.8%. Following the network density increases, each node is expected to receive more probes with the same hop count number, since its neighbors are more centralized around itself. In DLV, its corresponding group has a larger size, hence produces more possibilities once the number of probes mismatches the group size.

C. DLV Feedback

From the previous two groups of experiments, we can see that a multiple-node, dense network may bring much vagueness into our DLV inference model, leading to a biased fault report. Because DLV generates the optimal inference result just based on its facts and rules. The facts include node location, link distance and probe records while the rules show the inner relationship between the links (detailed in section II-D). Unlike the previous experiments, we run 100 experiments which are not independent. That is, every fault report can generate a feedback for the next DLV programs. Our experiments try three schemes for feedback: weak constraints, facts, and both. In our implementation, there are four weak constraints. The feedback of weak constraints optimizes the cost functions, like importance order and proportions of different links in the current network. The feedback of facts are more straightforward and emphasized, including which part of links or which specific links are bad, thus guide the program totally

avoid to obey the rules, and eliminate the wrong inferences by the rules. Figure 7 clearly shows that whatever feedback of constraints are provided by the fault report, it only produces little or even no benefits for the following work. By contrast, feedback of facts concretely increases the diagnosis accuracy.

IV. LARGE-SCALE SIMULATION STUDY

Since the testbeds can only investigate a limited design space in terms of the network scale and corner cases, we further conduct a large scale simulation study. In this evaluation, we pay more attention on the importance order of weak constraints, as well as their respective inner parameters.

A. Simulation Setup

In our simulation, 2000 sensor nodes are randomly deployed on a 2500m $\times$ 2500m map, and the communication range is 100m, thus the network density is average 10 neighbors for each node when the nodes are uniformly distributed. We use the logarithmic distance path loss model [16] to simulate received signal strength, and the received signal strength S_i can be formulated as:

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

where β is the signal fading factor and is set to 4 as in [26]. d_i is the link length while d_0 is the reference distance which is set to 1m. We also add a random noise factor X_i into our simulation, which follows a 0-mean normal distribution with variance σ^2 where σ is set to 4. A link's quality is decided by its received signal strength, an extra random factor and other manual settings. However, we will emphasize the inference procedure but not to simulate a complicate scene for data collection. Compared to real testbed system evaluation, the results of simulation seems more reasonable on the basis of network topology. Hence we primarily discuss about the inference model itself. We will study the impact of importance order of weak constraints and some individual specific weak constraints in DLV. For each setup we take 1000 runs.

B. Impact of Importance Order

In our expectation, different orders enable DLV to generate different solutions. For example, if we pay great attention to the correlation between link length and link quality, the inference model will significantly depend on network topology. By

(a) False negative rate in different importance orders of weak constraints. (b) False positive rate in different importance orders of weak constraints. (c) False negative rate in different settings of asymmetric links. (d) False positive rate in different settings of asymmetric links.

Fig. 8. Simulation Evaluation.

contrast, if we believe that some bad nodes probably exist in the networks and their self-contained hardware faults definitely cause their links failures, we may rank this fact the highest. Otherwise, it does not make sense that we are discussing about link features and correlations while there are even no links in the networks.

Figure 8(a), 8(b) describe the LS's fault report under four importance orders. We use "12345" to represent the order we list in section II-D, then vary the order to "45123", "54123", and "34512" in our simulation, in order to verify our expectation. As we can see, LS achieves a false negative rate around 10.2% as well as a false positive rate around 9.5%. Under the order "45123" and "34512", LS fails to accurately explore some links' real performance. They both put constraint 4 before constraint 5, 1, and 2. So the inference model mainly considers about asymmetric links in the network and takes less concern on some existing bad nodes. What is more, the order "54123" regards the link length as the most critical factor to judge a link's performance, which strongly differs from our real system observation.

In addition, we try to observe the variations by removing some weak constraints from the inference model at the beginning. First we remove the weak constraints which emphasize that most links have a poor performance just because they are all related to one bad node. Then we remove the weak constraint which is defined on the basis of observation about asymmetric links in the network, which implies that a link probably performs well if its reverse link is good. We find that the impact of constraint 1 and 2 is so large (nearly 17% difference). As mentioned above, a sensor node's own hardware failure or program error may cause all of its links even its neighbors' links destructive. Without these two constraints in the interference model, DLV fails to utilize the inner relationship between links and nodes. For example, it should be seriously challenged if the final fault report says that only one or two neighbors have received node A's probe but node A actually has 20 neighbors around. Constraint 4 also partially impacts the interference in our simulation where we set about 20% asymmetric links. After all, we expect to leverage each reasonable observation in link features to construct the interference model.

C. Impact of Asymmetric Links

In this section, we delve deeply into the parameter settings in the weak constraint about asymmetric links. In the sim-

Fig. 9. Field Topology. We use grey level to depict the number of neighbors for each node. Darker the node, more neighbors around the node.

ulation setup, we set 20% asymmetric links in the network. We respectively initialize DLV's prior knowledge about asymmetric links number to 10%, 20%, 30% and 40%, and DLV potentially regards a link as asymmetric link with a possibility of this manual prior knowledge. As described in Fig. 8(c) and 8(d), LS's accuracy fluctuates when the prior knowledge mismatches with the ground truth, but keeps relative accurate average false negative rate and false positive rate. We believe that this constraint's low importance more or less eliminates the impacts because the first three important constraints have covered its potential conclusive decision, which can be also explained by the system evaluation on DLV feedback.

V. FIELD STUDY

In section III we evaluate LS under an indoor environment, discussing how different network topologies impact our inference results. Then we conduct a large-scale simulation study to examine LS by adjusting the parameters in the program. In this section, we deploy a real outdoor system to verify that LS can be reliably applied to surveillance networks. We deploy 80 sensor nodes in a 75m $\times$ 20m forest. As illustrated in Fig. 9, we put the nodes in a 5 $\times$ 16 manner, which enables us to easily locate the nodes and compute the links' length. Nodes' transmission power is set as 15 which guarantees each node has 10 neighbors on average.

A. Implementation

LS incurs memory overhead on RAM and ROM respectively for data and program storage. (i)As mentioned in section II, every report is usually generated in the form

(a) Correlation between the number of neighbors and that of lost packets. (b) Correlation between the number of neighbors and that of asymmetric links. (c) Evaluation in terms of false negative rate, for LS and learned LS. (d) Evaluation in terms of false positive rate, for LS and learned LS.

Fig. 10. Field Study.

($MinHop, n_0, n_1, n_2$), i.e., only 4 bytes storage consumption, which is remarkably few. (ii) To evaluate LS' ROM overhead, we first implement a simple collection application using CTP. Then we compare this benchmark and the same one with LS module. We have found that the original benchmark consumes 18270 bytes ROM while the LS-embedded version consumes 22440 bytes, which indicates the LS module consumes approximately 4.1KB ROM, which is acceptable compared to 48KB ROM in telosB. Besides, no computation overhead is needed as probe path is not needed to compute.

B. Basic Observations

First we observe the distribution of faulty links. Due to hidden terminal and receiving queue overflow, a node with more neighbors is more likely to lose packets, thus degrades its links' performance. In this field study, the nodes are divided into 4 groups according to the number of their neighbors, and we record each probe's sender ID to identify the sources of received probes for each node. Figure 10(a) shows the ratio for each group. For those nodes which have less than 9 neighbors, each of them loses 1.5 probes on average, while the nodes with more than 12 neighbors averagely lose 4 probes.

In addition, we compare every pair of nodes to find out the asymmetric pairs, i.e., only one of two nodes has received the other's probe. Similarly, we divide the nodes into 4 categories according to the number of their neighbors. Figure 10(b) compares the ground truth and our inference results. For those nodes which have less than 9 neighbors, each of them has 1.2 asymmetric links on average. For those nodes which have more than 12 neighbors, however, they averagely have 3.2 asymmetric links. In LS's fault report, we set the prior knowledge about asymmetric link ratio is 20%, which means the inference program expects that around 20% links in the network are asymmetric. As we can see, our program has a better accuracy for the nodes with less neighbors, and it deduces that there exist average 1.3 and 3.7 asymmetric links respectively for the nodes with less than 9 neighbors and those with more than 12 neighbors.

C. Performance Evaluation

We also evaluate LS in terms of false negative rate and false positive rate. We try to add the basic observations into our inference model, to help the program specify different ratios of asymmetric links for different groups of nodes. Figure 10(c) compares the false negative rate before and after the learning.

As we can see, for those nodes with less than 9 neighbors, the false negative rates are both 8.9%. In contrast, LS improves its accuracy for those nodes with more than 12 neighbors, i.e., detect the links of those nodes more accurately. Before learning, LS has a false negative rate as 17%, while it achieves to 15.4% after we modifying the program according to Fig. 10(b), i.e., adjust the prior knowledge about the number of asymmetric links for different group of nodes.

We observe similar improvement of false positive rate in Fig. 10(d). The evaluations before and after learning stay the same for the nodes with less than 9 neighbors, i.e., both are 7.4%. In contrast, the links of nodes which have more than 12 neighbors are more accurately detected. Learning decreases the false positive rate from 15.5% to 13.2%. To explain the benefit generated by learning, we compare the faulty links in the fault reports of two inference models (i.e., LS before and after learning). We find that the parts related to the links of nodes with less than 9 nodes are totally the same. Figure 10(b) shows that the average number of asymmetric links of nodes which have less than 9 neighbors in ground truth is 1.2, while 1.3 in LS' report, which means that tiny difference of knowledge about the number of asymmetric links could't change the detection significantly. In contrast, for those nodes with more than 12 neighbors, the numbers are respectively 3.2 and 3.7, thus the learning process improves the evaluation by adjusting the corresponding knowledge in the inference model.

VI. RELATED WORK

Network diagnosis has been extensively studied in recent years. Existing approaches can be broadly divided into two categories: debugging tools and inference schemes. This work belongs to the later category. Claivoyant [22] is a notable tool which focuses on debugging sensor nodes at the source-level, and enables developers to wirelessly connect to a remote sensor and execute debugging commands. Declarative Tracepoints [2] allows the developers to insert a group of action-associated checkpoints at runtime, which are programmed in an SQL-like declarative language. Existing inference-based diagnosis schemes for WSNs like Sympathy [15] or Emstar [5] rely heavily on an add-in protocol that periodically reports a large amount of network information from individual sensor nodes to the sink, introducing huge overhead to the resource constrained and traffic sensitive sensor network. In order to minimize the overhead, some researchers propose to establish inference models by marking the data packets [11], [13],

and then parse the results at the sink to infer the network status, or conduct the diagnosis process in local areas [12]. Steinder and Sethi [18] apply Belief Network with the bipartite graph to represent dependencies among links and end to end connections, then the root causes can be deduced by conducting inference on the Belief Network.

Besides, most approaches actively design their probes to fetch desired information for faulty link detection [1], [20], especially in the managed enterprise WLANs and wireless mesh networks, where the monitors are easy to deploy. In [24], A cycle cover was leveraged to monitor network links. For each cycle, a node is required to monitor the cycle's performance. [7] develops a non-adaptive fault diagnosis through a set of probes where all the probes are employed in advance. The authors in [6] propose a failure detection scheme, in which monitors are assigned to each optical multiplexing and transmission section. These approaches usually compute the probe paths according to different network symptoms, so as to combine the network topology to infer the link status. For a large scale sensor network, however, deploying monitors in the wild not only increases the cost, but also needs to guarantee sustainable management. Sniffers can be used to collect the information. Indeed, to use sniffer also needs to take into account the cost of maintenance and other deployment details like coverage and timeline accordance.

VII. CONCLUSION

A wireless network often contains a large number of links which virtually exist in the air, but we can never directly observe whether they perform well or not. We propose a novel and low-cost link scanning scheme LS for faulty link detection. LS infers all links statuses on the basis of data collection from a prior probe flooding process, in which we leverage hop count to reflect node in/out-going link performances. In the inference model, we use DLP to describe the inner relationship among the links, and finally output the optimal fault report with some constraints, which reversely generates a feedback for DLP's following computation. We evaluate our algorithm through a testbed consisting of 60 TelosB sensor motes and an extensive simulation study, while a real outdoor system is deployed to verify that LS can be reliably applied to surveillance networks.

ACKNOWLEDGMENT

This work is supported in part by the NSFC Distinguished Young Scholars Program under Grant No. 61125202, National Basic Research Program of China (973) under Grants No. 2011CB302705 and 2012CB316200, NSFC under Grants No. 61103187 and 61100236, National High-Tech R&D Program of China (863) under Grant No. 2011AA010100, and China Postdoctoral Science Foundation under Grant No. 2011M500330.

REFERENCES

- [1] S.S. Ahuja, S. Ramasubramanian, and M.M. Krunz. Single-link failure detection in all-optical networks using monitoring cycles and paths. *IEEE/ACM Transactions on Networking*, 17(4):1080–1093, 2009.
- [2] Q. Cao, T. Abdelzaher, J. Stankovic, K. Whitehouse, and L. Luo. Declarative tracepoints: a programmable and application independent debugging system for wireless sensor networks. In *Proceedings of ACM SenSys, Raleigh, NC, USA*, 2008.
- [3] A. Cerpa, J.L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. In *Proceedings of IEEE IPSN, UCLA, Los Angeles, California, USA*, 2005.
- [4] H. Chang et al. Spinning beacons for precise indoor localization. In *Proceedings of ACM SenSys, Raleigh, NC, USA*, 2008.
- [5] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: a software environment for developing and deploying wireless sensor networks. In *Proceedings of the USENIX Annual Technical Conference, Boston, MA, USA*, 2004.
- [6] Y. Hamazumi, M. Koga, K. Kawai, H. Ichino, and K. Sato. Optical path fault management in layered networks. In *Proceedings of IEEE GlobeCom, Sydney, Australia*, 1998.
- [7] N.J.A. Harvey, M. Patrascu, Y. Wen, S. Yekhanin, and V.W.S. Chan. Non-adaptive fault diagnosis for all-optical networks via combinatorial group testing on graphs. In *Proceedings of IEEE INFOCOM, Anchorage, Alaska, USA*, 2007.
- [8] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3):499–562, 2006.
- [9] Z. Li, Y. Liu, M. Li, J. Wang, and Z. Cao. Exploiting ubiquitous data collection for mobile users in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 24(2):312–326, 2013.
- [10] Y. Liu, Y. He, M. Li, J. Wang, K. Liu, L. Mo, W. Dong, Z. Yang, M. Xi, J. Zhao, et al. Does wireless sensor network scale? a measurement study on greenorbs. In *Proceedings of IEEE INFOCOM, Shanghai, China*, 2011.
- [11] Y. Liu, K. Liu, and M. Li. Passive diagnosis for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 18(4):1132–1144, 2010.
- [12] Q. Ma, K. Liu, X. Miao, and Y. Liu. Sherlock is around: Detecting network failures with local evidence fusion. In *Proceedings of IEEE INFOCOM, Orlando, FL, USA*, 2012.
- [13] E. Magistretti, O. Gurewitz, and E. Knightly. Inferring and mitigating a link's hindering transmissions in managed 802.11 wireless networks. In *Proceedings of ACM MobiCom, Chicago, Illinois, USA*, 2010.
- [14] L. Mo, Y. He, Y. Liu, J. Zhao, S.J. Tang, X.Y. Li, and G. Dai. Canopy closure estimates with greenorbs: Sustainable sensing in the forest. In *Proceedings of ACM SenSys, Berkeley, California, USA*, 2009.
- [15] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the sensor network debugger. In *Proceedings of ACM SenSys, San Diego, USA*, 2005.
- [16] T.S. Rappaport et al. *Wireless communications: principles and practice*, volume 207. Prentice Hall PTR New Jersey, 1996.
- [17] D. Son, B. Krishnamachari, and J. Heidemann. Experimental analysis of concurrent packet transmissions in low-power wireless networks. In *Proceedings of ACM SenSys, San Diego, USA*, 2005.
- [18] M. Steinder and A.S. Sethi. Probabilistic fault localization in communication systems using belief networks. *IEEE/ACM Transactions on Networking*, 12(5):809–822, 2004.
- [19] I. Stojmenovic and X. Lin. Loop-free hybrid single-path flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1023–1032, 2001.
- [20] Y. Wen, V.W.S. Chan, and L. Zheng. Efficient fault-diagnosis algorithms for all-optical WDM networks with probabilistic link failures. *Journal of Lightwave Technology*, 23(10):3358, 2005.
- [21] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of ACM SenSys, Los Angeles, CA, USA*, 2003.
- [22] J. Yang, M.L. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant: a comprehensive source-level debugger for wireless sensor networks. In *Proceedings of ACM SenSys, Sydney, Australia*, 2007.
- [23] K. Yedavalli and B. Krishnamachari. Sequence-based localization in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 7(1):1–14, 2008.
- [24] H. Zeng, C. Huang, and A. Vukovic. Monitoring cycles for fault detection in meshed all-optical networks. In *International Conference on Parallel Processing Workshop, Montreal, Quebec, Canada*, 2004.
- [25] Q. Zheng and G. Cao. Minimizing probing cost and achieving identifiability in probe based network link monitoring. *IEEE Transactions on Computers*, 2011.
- [26] Z. Zhong, T. Zhu, D. Wang, and T. He. Tracking with unreliable node sequences. In *Proceedings of IEEE INFOCOM, Rio de Janeiro, Brazil*, 2009.