Fault Localization Using Passive End-to-End Measurements and Sequential Testing for Wireless Sensor Networks

Bing Wang, *Member*, *IEEE*, Wei Wei, *Member*, *IEEE*, Hieu Dinh, Wei Zeng, *Student Member*, *IEEE*, and Krishna R. Pattipati, *Fellow*, *IEEE*

Abstract—Faulty components in a network need to be localized and repaired to sustain the health of the network. In this paper, we propose a novel approach that carefully combines active and passive measurements to localize faults in wireless sensor networks. More specifically, we formulate a problem of *optimal sequential testing guided by end-to-end data*. This problem determines an optimal testing sequence of network components based on end-to-end data in sensor networks to minimize expected testing cost. We prove that this problem is NP-hard, and propose a recursive approach to solve it. This approach leads to a polynomial-time optimal algorithm for line topologies while requiring exponential running time for general topologies. We further develop two polynomial-time heuristic schemes that are applicable to general topologies. Extensive simulation shows that our heuristic schemes only require testing a very small set of network components to localize and repair *all* faults in the network. Our approach is superior to using active and passive measurements in isolation. It also outperforms the state-of-the-art approaches that localize and repair all faults in a network.

Index Terms—Wireless sensor networks, fault localization, sequential testing.

1 Introduction

Wireless sensor networks have been deployed in a wide range of applications. A deployed sensor network may suffer from many network-related faults, e.g., malfunctioning or lossy nodes or links [23], [19]. These faults affect the normal operation of the network, and hence should be detected, localized, and corrected/repaired. Existing studies on sensor network fault localization use active or passive measurements (see Section 2). Active measurement incurs additional monitoring traffic (a node needs to monitor itself or its neighbors, and transmit the monitoring results locally or to a centralized server), which consumes precious resources of sensor nodes, and may reduce the lifetime of the network. On the other hand, it has the advantage that it can exactly pinpoint the faults. Passive measurement uses existing end-to-end data inside the network: if end-to-end data indicate faulty end-to-end behaviors, then some components in the network must be faulty. It introduces no additional traffic into the network, and hence is an attractive approach for energy-stringent sensor networks. On the other hand, it poses the challenge of fault *inference*—accurate inference from end-to-end data (i.e., locating all faults with low false positives) is not always possible because end-to-end measurements can have inherent ambiguity (see Section 3).

Motivated by the complementary strengths of active and passive measurements, we propose a novel approach that uses active measurements to resolve ambiguity in passive measurements, and uses passive measurements to guide active measurements to reduce expected testing cost (i.e., cost incurred from active measurements, see Section 3). More specifically, we formulate a problem of *optimal sequential testing guided by end-to-end data*. This problem determines an optimal testing sequence of network components that minimizes the total testing cost: it picks the first component to be tested, based on the test result (i.e., it is faulty or not faulty) and the end-to-end data (passive measurements, which indicate potential faults); it determines the next component to be tested. This sequential testing continues until the identified faulty components have explained all end-to-end faulty behaviors. Since these identified faults may not have included all faults in the network, we identify all faults by solving the optimal sequential testing problem in iterations. In an iteration, based on end-to-end data in this iteration, we solve the optimal sequential testing problem to identify a set of faulty components. We then repair all the identified faulty components and start the next iteration. The iteration repeats until all end-to-end behaviors are normal. At this time, *all* faulty components have been identified and repaired.

We prove that the problem of optimal sequential testing is NP-hard, and propose a recursive approach to solve it. This recursive approach leads to a polynomial-time optimal algorithm for line topologies, while requiring exponential running time for general topologies. Therefore, we further

- *B. Wang, H. Dinh, and W. Zeng are with the Computer Science and Engineering Department, University of Connecticut, 371 Fairfield Way, Unit 2155, Storrs, CT 06269-2155. E-mail: {bing, hdinh, wei.zeng}@engr.uconn.edu.*
- *W. Wei is with the Computer Science Department, University of Massachusetts Amherst, 140 Governors Drive, Amherst, MA 01003-9264. E-mail: weiwei@cs.umass.edu.*
- *K.R. Pattipati is with the Electrical and Computer Engineering Department, University of Connecticut, 371 Fairfield Way, Unit 2157, Storrs, CT 06269-2157. E-mail: krishna@engr.uconn.edu.*

Manuscript received 20 Apr. 2010; revised 25 Oct. 2010; accepted 9 Feb. 2011; published online 29 Apr. 2011.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2010-04-0182. Digital Object Identifier no. 10.1109/TMC.2011.98.

develop two polynomial-time heuristic schemes for general topologies. We evaluate the performance of these two heuristic schemes through extensive simulation in sensor networks of static and dynamic topologies. Our simulation results show that both schemes only need a few iterations and testing a very small subset of components to localize all faults in a network. These results demonstrate the benefits of our approach: it is superior to using active measurements alone since it selectively tests a very small subset of components; it is superior to fault inference using passive measurements alone since it localizes *all* faulty components while fault inference may suffer from a large number of false positives and false negatives [9], [14], [15].

Our approach also outperforms two state-of-the-art approaches [5], [13] that identify and repair all faults in a network. These two approaches also run in iterations. The exhaustive inspection approach [5] differs from our approach in that at each iteration, it infers *in parallel* (rather than in sequence) a set of potential faulty components from end-to-end measurements, tests each identified component, and repairs the faulty ones at the end of the iteration. It has to test all identified components because some of them may be false positives. Hence, the number of tests is at least equal to the total number of faulty components, while our approach requires much less number of tests. The approach in [13] infers the best component to be tested in each iteration, and repairs the component, if necessary. Since it only tests a single component in an iteration, it may lead to a large number of iterations to localize and correct all faults. To reduce the number of iterations, the authors also consider identifying multiple faults *in parallel* in an iteration, which is similar in spirit to exhaustive inspection [5] and suffers from the same drawbacks.

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 presents the problem setting. Section 4 describes a recursive approach to solve the optimal sequential testing problem, and an optimal solution for line topologies. Section 5 describes the two heuristic schemes. Section 6 presents evaluation results. Finally, Section 7 concludes the paper and presents future work.

2 Related Work

We consider the problem of localizing and correcting network-related faults in a deployed sensor network. Most existing studies use either active or passive measurements for this purpose.

Active measurements provide accurate view of the network at the price of introducing additional monitoring traffic into the network. Zhao et al. design a residual energy scan for a sensor network that depicts the remaining energy inside the network [29]. From the scan, a network operator can discover areas with low residual energy and take corrective actions. The same authors also propose an architecture that computes aggregates for sensor network monitoring [30]. This approach continuously collects aggregates (sum, average, count) of network properties (e.g., loss rates, energy level), triggers scan of the network when observing sudden changes in the aggregates, and further debugs the problem through detailed dumps of node states. To limit the scope of the monitoring traffic to a local area, Hsin and Liu propose a distributed monitoring architecture where each node monitors its neighbors by periodically sending them probes [10]. More recently, Tolle and Culler design a sensor network management system (SNMS) that allows a network operator to query the network for health information [23]. Whitehouse et al. propose Marionette, which extends SNMS by providing users the ability to call functions, and read or write variables in embedded applications [27]. Ramanathan et al. propose Sympathy, a tool for detecting and debugging failures in sensor networks [19]. Sympathy carefully selects metrics that enable efficient failure detection and includes an algorithm that analyzes root causes. Rost and Balakrishnan design a health monitoring system, Memento, that delivers state summaries and detects node failures [21]. Last, Gruenwald et al. propose a remote management system for a wide area sensor network that contains multiple and heterogenous networks [8].

Active measurements consume precious resources of the sensor network. Furthermore, malicious behaviors can mislead fault localization that solely relies on active measurement, e.g., a node may not report its status or its neighbors' status honestly, or an intermediate node may manipulate the forwarded messages or aggregates. Passive measurement using existing end-to-end data in sensor networks does not suffer from the above drawbacks. Hartl and Li use traditional network tomography techniques to infer node loss rates [9], and Mao et al. use a factor graph decoding method to infer link loss rates [14]. Both techniques, however, heavily rely on a data aggregation procedure (that is used to guarantee correlation among packets). Furthermore, their assumption of a fixed tree limits their applicability. Nguyen and Thiran propose lossy link inference schemes that use uncorrelated packets and take account of dynamic network topologies [15]. Their schemes, however, may lead to a large number of false positives and false negatives in certain circumstances. Broadly speaking, fault inference from end-to-end data falls into network tomography, i.e., inferring internal properties through end-to-end measurement. A rich collection of network tomography techniques has been developed in the past (see [1] for a review). Most of these techniques are developed for wired networks and cannot be applied directly to wireless sensor networks. This is because most of them rely on correlated packets (through multicast or striped unicast packets) and require static topology. In wireless sensor networks, however, end-to-end data are not correlated and topology may change over time.

Our approach differs from existing studies on sensor network fault localization in that it carefully combines active measurements and end-to-end data. The study in [19] also uses end-to-end data together with active measurements: it uses end-to-end data to detect faults, which in turn trigger active measurements and root-cause analysis. It, however, does not utilize end-to-end data to guide selective active measurement (to reduce expected testing cost) as in our study. Two existing studies [5], [13] combine end-to-end measurements and active testing to identify and correct all faults in a network. Our approach outperforms these two approaches (see Sections 1 and 6).

Finally, sequential testing (or online decision) has been studied in the fields of machine troubleshooting, medical diagnosis, and computer decision making (e.g., [3], [2], [7], [17], [18]). Our sequential testing problem differs from that

in other fields in two important aspects: 1) our sequential tests are on individual components (instead of multiple components) with the guidance of end-to-end data that provide insights into the status of multiple components; and 2) we diagnose multiple faults instead of a single fault that is often assumed in other fields.

3 Problem Setting

3.1 Assumptions

Consider a sensor network where sensed data are sent (periodically) from sources to a sink. As in [19], we assume the amount of end-to-end data can be used to detect faults in the network: insufficient amount of data indicates faults, while sufficient amount of data indicates that the network is operating normally. The status of a component (i.e., whether faulty or not) can be tested through active measurements, e.g., by monitoring the component locally, or looking into the internal states of relevant components (e.g., using [27]). This test incurs a *testing cost*, which accounts for personnel wages when human is involved, or the resources used at a sensor node to monitor itself and neighboring nodes/links, or the energy and network bandwidths used to transfer the monitoring results to the sink.

A fault in a sensor network can be of various forms. For ease of exposition, we only consider faults in the form of lossy links (we briefly discuss lossy nodes in Section 5.5 and present results when nodes can be lossy in Section 6.2). In particular, our goal is to locate *persistently* lossy links that are used in routing. These persistent faults can be due to physical obstacles and/or faults at the sensor nodes (e.g., low battery, hardware faults, or program bugs). They can be corrected by removing the root causes. We do not consider transient lossy links (e.g., due to interference or background noise) since they are not caused by persistent faults that need to be localized and corrected.

We determine whether a link is lossy or not based on its average loss rate or reception rate (defined as one minus the average loss rate). Since existing studies (e.g., [4], [11], [24], [12]) show that packets of different sizes may experience different loss rates at a link, we assume the data packets that are transmitted from the sources to the sink are of the same size, and use the loss rate (or reception rate) experienced by these data packets at a link as the loss rate of that link. We say link l is *lossy* or *bad* if its average reception rate lies below a threshold, t_l. Otherwise, we say l is *not lossy* or *good*. We assume the threshold, t_l, can clearly separate good and bad links, i.e., good links have reception rates much larger than t_l while bad links have reception rates much lower than t_l. This is reasonable since measurement studies have shown that link loss rates in sensor networks are either large or small, but rarely in between [20], [15].[1] We assume that the losses at different links are independent of each other (as shown in [15]). Furthermore, if a link is good (or bad) on one path, then it is good (or bad) on all paths that use the link.

1. Several other studies reveal links that have intermediate loss rates [28], [6], [32]. Since these types of links can have significant negative impact on the performance of upper layer protocols [6], [31], we assume the routing protocol used in the sensor network avoids using such links, and hence most links used in the routes have either large or small loss rates.

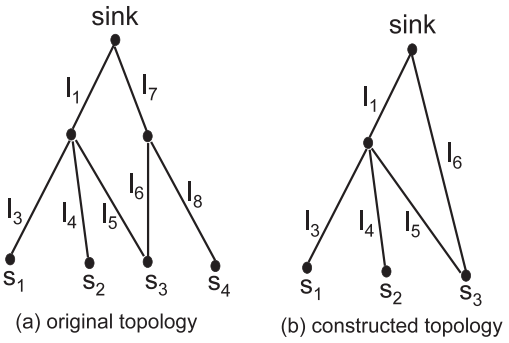


Fig. 1. (a) An example network topology, where four sources send data to a sink. (b) The constructed topology by contracting links l_7 and l_8 in (a). (we assume end-to-end data indicate that the path from source s_4 to the sink is good, and hence links l_7 and l_8 are good.)

The routing path from a source to the sink can be static or dynamic (e.g., due to a dynamic routing technique [28]). Fig. 1 shows an example topology, where sources s_1, s_2, and s_4 use a single path; source s_3 uses two paths dynamically. We consider two settings of the problem: 1) we know *complete* path information, i.e., we know the path used by a source at any point of time; and 2) we only know *probabilistic* path information, i.e., we only know the set of paths that are used by a source and the probability using each path. The first setting applies to static topologies, and dynamic topologies where up-to-date path information is available, e.g., obtained through a path reporting service which reports the parent of a node periodically as in [22], or from information embedded in data packets [28]. The second setting applies to dynamic topologies where it is too costly to obtain complete path information (e.g., it might consume too much energy), and hence a path reporting service only runs at coarse time scales, and the probability to use a path is estimated from the frequency that this path is reported to be used.

When knowing complete path information, we define *path reception rate* as the probability that a packet traverses a path successfully. We assume that data are not aggregated while being transmitted inside the network, and path reception rate can be estimated from end-to-end data: when n data packets are transmitted along a path and m packets are received successfully, the path reception rate is estimated as m/n. As mentioned earlier, we assume that the amount of end-to-end data indicates whether a fault exists along a path. In particular, we assume, for path P, there exists a threshold, t_P, so that it contains at least one bad link *if and only if* its reception rate is below t_P. We next illustrate when this assumption holds. Consider a path of h links. Assume good links have reception rate at least α, while bad links have reception rate no more than β, $0 \leq \beta < \alpha \leq 1$. When the path contains no bad links, the path reception rate is at least α^h (since each good link has reception rate at least α and the losses at the various links are independent, a path of h good links has reception rate at least α^h); otherwise, it is no more than β. Therefore, when $\alpha^h > \beta$, there exists a threshold (any number in (β, α^h), e.g., $(\beta + \alpha^h)/2$) so that the path contains at least one bad link if and only if its reception rate is below this threshold. The condition $\alpha^h > \beta$ holds when good and bad links have

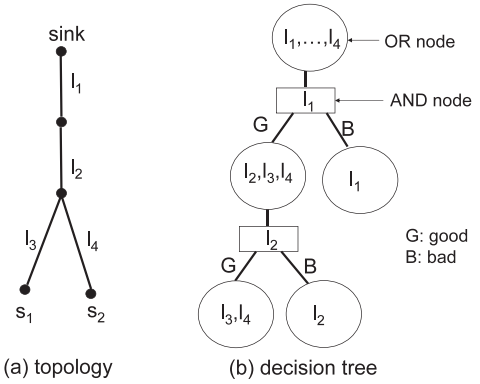


Fig. 2. An example illustrating sequential testing.

significantly different reception rates, which is true in wireless sensor networks as mentioned earlier. For convenience, we say path P is *lossy* or *bad* if its reception rate is below the threshold, t_P (in other words, it contains at least one lossy link); otherwise, it is *not lossy* or *good*.

When only knowing probabilistic path information, we define *source-sink reception rate* as the probability that a packet is sent from a source to the sink successfully. It can be estimated from end-to-end data: when n packets are sent from a source to the sink, and m packets arrive successfully, it is estimated as m/n. We again assume that there exists a threshold so that at least one link used by a source-sink pair is lossy *if and only if* the source-sink reception rate is below this threshold. Again, we say a source-sink pair (or simply a pair) is *lossy* or *bad* if its reception rate is below the threshold; otherwise, it is *not lossy* or *good*.

The above assumptions imply that all the links on a good path/pair are good, and a bad path/pair contains at least one bad link. Therefore, the potential bad links are the ones that are used by bad paths/pairs, excluding those used by good paths/pairs.

3.2 Sequential Testing Guided by End-to-End Data

Using end-to-end data (i.e., passive measurements), we have narrowed down the potential lossy links to the set of links that are used by bad paths/pairs, excluding those used by good paths/pairs (since all the links on a good path/pair are good). Testing individual links (i.e., active measurements) can pinpoint which potential lossy links are indeed lossy. We now motivate the benefits of sequential testing using an example in Fig. 2a. This example shows two paths, $P_1 = (l_3, l_2, l_1)$ and $P_2 = (l_4, l_2, l_1)$. Suppose end-to-end data indicate that both paths are bad. When determining which links are lossy, we face the following ambiguities. First, since links l_1 and l_2 are used by both paths, they cannot be differentiated solely from end-to-end data. Second, since both paths are lossy, the lossy links can be the common links (i.e., l_1 and/or l_2), both leaf links (i.e., l_3 and l_4), or a combination of the above two scenarios. The above ambiguity can be resolved by testing individual links. An advantage of sequential testing is that as we reveal the status of one link, this knowledge may provide information on other links, and hence inform later decisions.

We next formulate the problem of sequential testing guided by end-to-end data. For simplicity, the formulation below assumes complete path information (the scenario where we know probabilistic path information only differs in that we use source-sink pairs instead of paths). A sequential testing problem takes as input a topology constructed from the original one through edge contraction [26].[2] More specifically, the links (i.e., edges) that are known to be good are contracted. Therefore, the resultant topology only contains potential bad links that are used by bad paths but not used by good paths. Let $\mathcal{L} = \{l_1, \ldots, l_M\}$ denote the set of potential bad links. Let $\mathcal{P} = \{P_1, \ldots, P_N\}$ denote the set of bad paths that are identified from end-to-end data. Furthermore, let $\mathcal{P}_i \subseteq \mathcal{P}$ represent the set of bad paths that use link l_i. Fig. 1b shows a topology constructed from Fig. 1a by contracting links l_7 and l_8 that are known to be good. Henceforth, unless otherwise specified, topology refers to the topology constructed from the original one through edge contraction. Each potential lossy link, l_i, is associated with a testing cost $c_i > 0$, and a probability p_i that denotes the prior probability that link l_i is lossy.

A solution to the sequential testing problem is to determine the next link to test depending on the previous link that is tested, its corresponding test result (i.e., whether it is lossy or not), and end-to-end data, so that all bad paths are explained (i.e., each bad path contains at least one link that has been found to be lossy). More conveniently, we can describe a solution to the sequential testing problem using a binary AND/OR decision tree. In the tree, an OR node represents the set of potential lossy links, and an AND node represents testing a link. A leaf node is reached when all lossy paths have been explained, and the leaf node is marked with the set of bad links that has been identified to explain the lossy paths. Each AND node has two branches, leading to two OR nodes based on whether the link tested is lossy or not. If the link is good, the AND node branches left and the arc is marked with "G;" otherwise, the AND node branches right and the arc is marked with "B."

The *expected total testing cost* of a solution to the sequential testing problem is the sum of the expected testing costs over all links (in the input of the problem), where the expected testing cost of a link is the testing cost of this link times the probability that this link is tested. More conveniently, the expected total testing cost can be calculated from the binary decision tree: it is the sum of the expected testing costs over all AND nodes, where the expected testing cost of an AND node is the testing cost for the link tested at this AND node times the probability that this AND node is reached in the tree. An *optimal* solution to the sequential testing problem is one that leads to the minimum expected total testing cost.

We next use an example to illustrate sequential testing. Fig. 2b shows a binary decision tree (not necessarily the optimal one) for the example in Fig. 2a. The root of the tree contains all potential lossy links. The first link tested is l_1. If l_1 is bad, sequential testing stops (since both lossy paths

2. In graph theory, an edge contraction is an operation that removes an edge from a graph while simultaneously merging together the two vertices that it previously connected. More formally, contraction of an edge $e = (u, v)$ replaces the two end points u and v with a single vertex w such that edges incident to w each correspond to an edge incident to u or v. The resulting graph has one less edge and one less node than the original one, and may contain loops and multiedges.

have been explained) and identifies one lossy link, l_1. Otherwise, the set of potential bad links is reduced to l_2, l_3, l_4, and the next link tested is l_2. If l_2 is bad, sequential testing stops and identifies one lossy link, l_2. Otherwise, sequential testing stops and concludes that links l_3 and l_4 are lossy. The expected total testing cost of this decision tree is $c_1 + (1 - p_1)c_2$ since there are two AND nodes in the decision tree; the expected testing cost of the AND node that tests l_1 is c_1, while the expected testing cost of the AND node that tests l_2 is $(1 - p_1)c_2$ since l_2 is only tested when l_1 is good, which happens with probability $(1 - p_1)$.

Note that we may not have identified all lossy links when the above sequential testing stops. For instance, in Fig. 2a, when both l_1 and l_3 are lossy, the decision tree in Fig. 2b stops after finding link l_1 to be lossy. To identify all lossy links, we run sequential testing in iterations; at the end of an iteration, we repair all lossy links (by finding out the root causes and removing the root causes) that have been found in the iteration. The iteration continues until all end-to-end behaviors are good. The reason why we do not intend to find all lossy links in one iteration is that end-to-end data in a later iteration may reveal link status at no additional testing cost. For instance, in the above example, if l_1 is lossy and the next iteration shows no lossy links, then we know that the rest of the links are good without any additional test.

3.3 Discussion

Our sequential testing problem differs from existing studies [5], [15], [16] in important ways. The goal of existing studies is to find the most likely set of lossy links that explains the faulty end-to-end behaviors, while our goal is to minimize testing cost. Therefore, we may purposely test a link that is likely to be good as long as it can reduce testing cost. Our approach and the ones in existing studies [5], [15], [16] can run in iterations until all lossy links are located and repaired. As we shall see (Section 6), our approach requires a similar or lower number of iterations and a much lower testing cost.

The goal of our sequential testing problem is to minimize expected testing cost. In practice, minimizing the total number of iterations to locate and repair all faulty components can also be an important goal. Our formulation can be extended to incorporate this goal as follows: we can use a cost to represent the number of iterations required to locate and repair all faulty components, and minimize a weighted sum of this cost and the testing cost, or we can formulate a constrained optimization problem that minimizes total testing cost under a constraint on the number of iterations, or minimizes the required number of iterations under a constraint on total testing cost. Further exploration of these problems is left as future work.

4 Optimal Sequential Testing

The sequential testing problem formulated in Section 3 is NP-hard; the proof is found in the Appendix I, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2011.98. In the following, we first present a recursive approach to solve it under general topologies, and then consider two special types of topologies, namely lines and trees.

4.1 General Topology

For a given instance of the optimal sequential testing problem, I, let $J^*(I)$ denote the minimum expected testing cost from optimal sequential testing. It is clear that $J^*(I) = 0$ when no potential lossy link is in I. In general, we can determine $J^*(I)$ using the following recursive equation:

$$J^*(I) = \min_{l_k \in \mathcal{L}} \{c_k + p_k J^*(I_{kb}) + (1 - p_k) J^*(I_{kg})\}, \qquad (1)$$

where c_k is the testing cost of link l_k, I_{kb} is the resultant instance when l_k is found to be lossy, and I_{kg} is the resultant instance when l_k is found to be good. The expression, $c_k + p_k J^*(I_{kb}) + (1 - p_k) J^*(I_{kg})$, is the optimal expected testing cost when link l_k is chosen to be tested.

We next describe how to obtain instances I_{kb} and I_{kg}. Before that, we first define two types of links. We say a link is **responsible** if there exists at least one bad path that can only be explained by this link being lossy; we say a link is **irrelevant** if it is used by none of the bad paths to be explained. Algorithm 1 describes, after determining that l_k is lossy, how to obtain I_{kb} from I. More specifically, it contracts link l_k (i.e., removes l_k from $\mathcal{L}$ while merging the two end nodes of l_k) and removes $\mathcal{P}_k$ from $\mathcal{P}$ (since all the bad paths in $\mathcal{P}_k$ have been explained). After that, it finds irrelevant links, and contracts these links. Algorithm 2 describes, after determining that l_k is good, how to obtain I_{kg} from I. It first contracts link l_k, which may lead to responsible links. It then contracts these responsible links, and remove all the bad paths that use these links from $\mathcal{P}$, which may further lead to irrelevant links to be contracted. It is easy to see that the running time of Algorithm 1 is $O(\max_k |\mathcal{P}_k| + |\mathcal{L}|)$, while the running time of Algorithm 2 is $O(|\mathcal{L}||\mathcal{P}| + |\mathcal{L}|)$.

Algorithm 1. Obtain instance I_{kb}

```
contract link l_k
remove all the paths in P_k from P
for all l_i ∈ L do
    if l_i is irrelevant then
        contract link l_i
    end if
end for
```

Algorithm 2. Obtain instance I_{kg}

```
contract link l_k
for all l_i ∈ L do
    if l_i is responsible then
        contract link l_i
        remove all the paths in P_i from P
    end if
end for
for all l_i ∈ L do
    if l_i is irrelevant then
        contract link l_i
    end if
end for
```

Following the recursive equation (1), a solution to the optimal sequential testing problem can be obtained in a top-down manner by constructing the full decision tree. Fig. 3 shows an example. This topology has three potential lossy

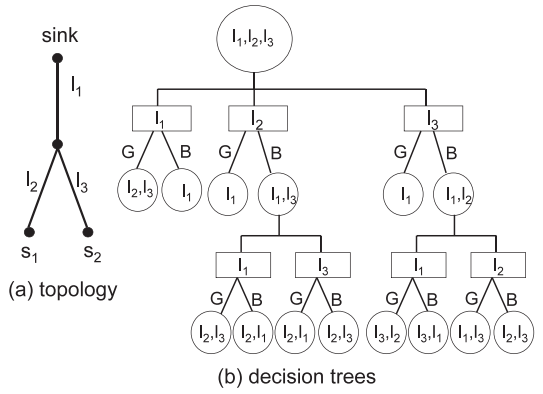


Fig. 3. An example illustrating the recursive approach for the sequential testing problem.

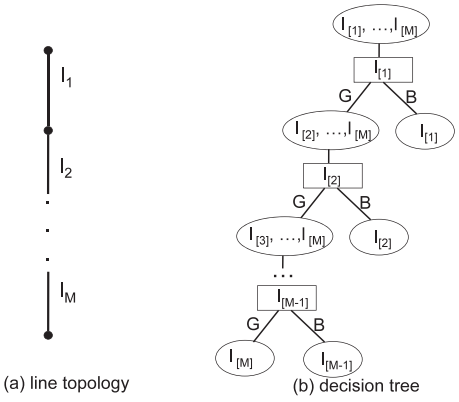


Fig. 4. Optimal sequential testing in a line topology.

links, l_1, l_2, and l_3, and two bad paths, $P_1 = (l_1, l_2)$ and $P_2 = (l_1, l_3)$. As shown in Fig. 3b, at the top level, we may choose to test link l_1, l_2, or l_3. Suppose we test l_1. If it is good, then following Algorithm 2, we contract l_1, and discover that l_2 and l_3 are responsible links since they are the only potential lossy links on paths P_1 and P_2, respectively. After that, we contract l_2 and l_3, and remove the paths that use l_2 and l_3, leading to a leaf state (i.e., an empty topology) that concludes l_2 and l_3 as lossy. If l_1 is bad, then following Algorithm 1, we contract l_1 and remove the two paths that use l_1, which leads to a leaf state that concludes l_1 as lossy. Since testing costs of leaf nodes in the decision tree are zero, the expected testing cost when testing l_1 at the top level is c_1. Similarly, we can construct the decision trees when testing l_2 or l_3 at the top level, and obtain the corresponding expected testing costs. The optimal testing cost is the minimum of the three expected testing costs (starting with l_1, l_2, or l_3 at the top level, respectively).

The above procedure generally requires exponential running time. We next discuss optimal sequential testing in two special types of topologies: lines and trees.

4.2 Line Topology

We next derive an optimal solution for a line topology. Consider a line topology with M links, $\mathcal{L} = \{l_1, \ldots, l_M\}$, as shown in Fig. 4a. Since the topology contains a single path, we can stop after finding one lossy link (since the single bad path has been explained by this lossy link); otherwise, we proceed to test the next link. We need to test at most $(M-1)$ links, since we can stop when the $(M-1)$th link is either bad or good—when it is good, then the Mth link (i.e., the link that has not been tested) must be bad. Suppose the testing sequence is $l_{[1]}, \ldots, l_{[M-1]}$. Then, the decision tree is as shown in Fig. 4b. Let $c_{[i]}$ and $p_{[i]}$ denote, respectively, the testing cost and the prior probability of being lossy for link $l_{[i]}$. The expected test cost for this testing sequence is

$$J = c_{[1]} + (1-p_{[1]})c_{[2]} + (1-p_{[1]})(1-p_{[2]})c_{[3]} + (1-p_{[1]})\cdots(1-p_{[M-2]})c_{[M-1]}. \quad (2)$$

From (2), it is easy to see that when the testing costs of all the links are the same, i.e., $c_i = c$, $\forall i$, the optimal testing sequence is one of sorting the links in decreasing order according to their prior probabilities of being lossy, testing the first $M-2$ links in order, and then testing one of the two remaining links (we can choose either one since they have the same cost and the cost J is independent of $p_{[M-1]}$, i.e., the prior probability of the $(M-1)$th link that is tested). Similarly, when the prior probabilities are the same, i.e., $p_i = p$, $\forall i$, the optimal testing sequence is sorting the links in increasing order according to their testing costs, and testing the first $M-1$ links in order. For general settings, the following theorem describes a property that an optimal solution should satisfy.

Theorem 1. *For a line topology with potential lossy links $l_1, \ldots, l_M$, the first $M-2$ tests in an optimal testing sequence $l_{[1]}, \ldots, l_{[M-1]}$ must satisfy the following property:*

$$\frac{p_{[1]}}{c_{[1]}} \geq \frac{p_{[2]}}{c_{[2]}} \geq \cdots \geq \frac{p_{[M-2]}}{c_{[M-2]}}. \quad (3)$$

Proof. We prove this theorem by contradiction. Suppose the optimal testing sequence $l_{[1]}, \ldots, l_{[M-2]}, l_{[M-1]}$ does not satisfy the stated property. That is, we can find two links, $l_{[i]}$ and $l_{[i+1]}$, $i < M-2$, such that

$$\frac{p_{[i]}}{c_{[i]}} < \frac{p_{[i+1]}}{c_{[i+1]}}.$$

Let J denote the testing cost of this testing sequence. Let J' denote the testing cost of a testing sequence that switches the order of $l_{[i]}$ and $l_{[i+1]}$. Let $q_{[i]} = 1 - p_{[i]}$. Then,

$$\begin{aligned} J &= c_{[1]} + \cdots + q_{[1]}q_{[2]}\cdots q_{[i-1]}c_{[i]} \\ &\quad + q_{[1]}q_{[2]}\cdots q_{[i]}c_{[i+1]} + \cdots + q_{[1]}\cdots q_{[M-2]}c_{[M-1]} \\ J' &= c_{[1]} + \cdots + q_{[1]}q_{[2]}\cdots q_{[i-1]}c_{[i+1]} \\ &\quad + q_{[1]}q_{[2]}\cdots q_{[i-1]}q_{[i+1]}c_{[i]} + \cdots + q_{[1]}\cdots q_{[M-2]}c_{[M-1]}. \end{aligned}$$

Hence,

$$\begin{aligned} J - J' &= c_{[i]}(1 - q_{[i+1]}) + c_{[i+1]}(q_{[i]} - 1) \\ &= c_{[i]}p_{[i+1]} - c_{[i+1]}p_{[i]} > 0. \end{aligned}$$

This contradicts the assumption that $l_{[1]}, \ldots, l_{[M-2]}$ is the optimal testing sequence. Therefore, the first $M-2$ links in an optimal testing sequence must satisfy

$$\frac{p_{[1]}}{c_{[1]}} \geq \frac{p_{[2]}}{c_{[2]}} \geq \cdots \geq \frac{p_{[M-2]}}{c_{[M-2]}}.$$

□

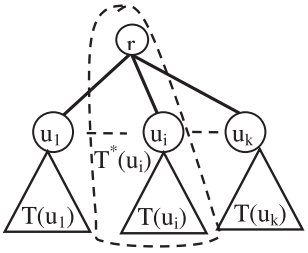


Fig. 5. An illustration of tree topology in Proposition 4.1.

Based on Theorem 1, we derive an optimal sequential testing algorithm for line topologies as follows: by randomly choosing $M-2$ links out of $l_1, \ldots, l_M$, and testing them following the optimal property (3), we construct $M(M-1)$ testing sequences that satisfy the optimal property (since there $M(M-1)$ ways of choosing $M-2$ links out of M links). For each such sequence, according to expression (2), the $(M-1)$th link to be tested should be the one with the lower cost of the two remaining links that have not been tested. Therefore, by appending the $(M-1)$th link thus selected, we construct $M(M-1)$ testing sequences, each with $M-1$ links, and the optimal testing sequence is the one with the lowest expected testing cost among all these sequences.

The complexity of the above algorithm is $O(M^2)$. When $M=2$, it reduces to testing the link with the lower cost. For $M>2$, it tends to test a link with high prior probability of being faulty and a low testing cost first.

We next use an example to illustrate the optimal solution for line topology. Consider a line topology with four links. The testing costs of these four links are 7, 6, 4, and 4, respectively. Their prior probabilities of being faulty are 0.23, 0.18, 0.10, and 0.09, respectively. Hence, $p_1/c_1 = 0.033$, $p_2/c_2 = 0.030$, $p_3/c_3 = 0.025$, and $p_4/c_4 = 0.023$. According to Theorem 1, the first two links tested in an optimal solution should satisfy condition (3). In this example, since $p_1/c_1 > p_2/c_2 > p_3/c_3 > p_4/c_4$, the first two links tested, l_i and l_j, should satisfy $i<j$. Following the optimal algorithm described above, we obtain the optimal testing sequence: test l_2 first; if l_2 is good, then test l_3; if l_3 is good, then test l_4. The expected testing cost of this optimal solution is 12.23. Note that, maybe counterintuitively, the link with the highest ratio of p_i/c_i, l_1, is not included in the optimal solution.

4.3 Tree Topology

We have the following proposition for a tree topology:

Proposition 4.1. *Consider a tree topology, T. The root of the tree is r, which has k children, $u_1, \ldots, u_k$, as illustrated in Fig. 5. Let $T(u_i)$ denote the subtree rooted at u_i, and $T'(u_i)$ denote the tree consisting of tree $T(u_i)$ and the link (u_i, r), $i = 1, \ldots, k$. Then, $J^*(T) = \sum_{i=1}^{k} J^*(T'(u_i))$.*

Proof. Since T is a tree, any decision made in $T'(u_i)$ does not affect that in $T'(u_j)$, $i \neq j$ (this is clear from Algorithms 1 and 2). Therefore, testing $T'(u_i)$ and $T'(u_j)$ is independent of each other, and the optimal testing cost for T is the sum of the optimal testing costs of $T'(u_i)$, $i = 1, \ldots, k$. □

Based on Proposition 4.1, it is sufficient to consider trees in which the root has a single child. We next show that, even for the simplest trees of such form, the optimal solution depends on many factors. Consider a tree with three links and two branches as shown in Fig. 3a. For link l_i, its testing cost is c_i and its prior probability of being lossy is p_i, $i = 1, 2, 3$. Following the decision tree shown in Fig. 3b, let J_i denote the optimal expected testing cost when starting with link l_i. Then,

$$J_1 = c_1$$
$$J_2 = c_2 + p_2 \min(c_1, c_3)$$
$$J_3 = c_3 + p_3 \min(c_1, c_2).$$

We next derive the optimal solution by considering the following cases:

- Case 1 ($c_1 \leq c_2$ and $c_1 \leq c_3$). In this case, $J_1 < J_2$ and $J_1 < J_3$, and hence the optimal solution is to test l_1, following the decision tree rooted as l_1 in Fig. 3b.
- Case 2 ($c_1 > c_2$ and $c_1 \leq c_3$). In this case, $J_1 < J_3$, and $J_2 = c_2 + p_2 c_1$. If $c_2 < c_1 \leq c_2/(1-p_2)$, then $J_1 \leq J_2$, and the optimal solution is to test l_1. Otherwise, the optimal solution is to test l_2 first; if l_2 is good, then test l_1.
- Case 3 ($c_1 \leq c_2$ and $c_1 > c_3$). Similar as above, if $c_3 < c_1 \leq c_3/(1-p_3)$, the optimal solution is to test l_1. Otherwise, the optimal solution is to test l_3 first; if l_3 is good, then test l_1.
- Case 4 ($c_1 > c_2$ and $c_1 > c_3$). In this case, $J_1 = c_1$, $J_2 = c_2 + p_2 c_3$, and $J_3 = c_3 + p_3 c_2$. The optimal testing sequence depends on the relative ordering of J_1, J_2, and J_3, which can be any one of the six possible orderings.

The above example shows that, even for the simplest tree topology, determining the optimal solution is nontrivial. Section 5 proposes two heuristic schemes for general topologies.

5 Heuristic Sequential Testing Schemes

In this section, we develop two heuristic algorithms to solve the sequential testing problem. Both algorithms choose a sequence of links to test. After testing a link, based on whether the link is good or bad, it obtains the resultant topology following Algorithm 1 or 2, and then chooses another link to test. We next describe these two algorithms in detail.

5.1 Ordering Algorithm

In each step, this algorithm picks the link with the highest $n_k p_k/c_k$ (breaking ties arbitrarily), where n_k is the number of paths that use link l_k, i.e., $n_k = |\mathcal{P}_k|$. Intuitively, it favors links with high values of p_k/c_k. Furthermore, it favors links that are used by more paths. This is because, intuitively, knowing the status of such links may provide more information. For instance, for the tree in Fig. 5, after finding link (r, u_i) to be lossy, all the paths using (r, u_i) are removed, and hence none of the links in the subtree rooted at u_i needs to be tested. The complexity of this scheme is

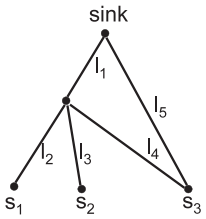


Fig. 6. An example to illustrate the heuristic sequential testing schemes, $c_i = 1$, $p_i = 0.2$, $i = 1, \ldots, 5$. The ground truth is that links l_1 and l_5 are bad.

$O(|\mathcal{L}|^2|\mathcal{P}|)$ since each step has running time of $O(|\mathcal{L}||\mathcal{P}|)$, and there are at most $O(|\mathcal{L}|)$ steps.

We next use an example to illustrate this scheme. The topology is shown in Fig. 6, where three sources send data to a sink. It has five links, $l_1, \ldots, l_5$. The ground truth is that links l_1 and l_5 are bad. The cost for testing a link is 1 unit. The probability that a link is lossy is $p = 0.2$. We first assume that we know complete path information. In this case, end-to-end data indicate that all paths are lossy. We denote the bad paths as $P_1 = (l_2, l_1)$, $P_2 = (l_3, l_1)$, $P_3 = (l_4, l_1)$, and $P_4 = (l_5)$. Without any testing, we identify l_5 as a responsible link since it is only used by P_4, and P_4 is lossy. We, therefore, contract link l_5 and remove P_4. Afterward, we obtain $\mathcal{P}_i$, the set of paths that use link l_i as $\mathcal{P}_1 = \{P_1, P_2, P_3\}$, $\mathcal{P}_2 = \{P_1\}$, $\mathcal{P}_3 = \{P_2\}$, $\mathcal{P}_4 = \{P_3\}$. Since all the links have the same testing cost and prior probability of being lossy, and l_1 is used by more paths than other links, we test link l_1 first. Since l_1 is lossy, all the lossy paths have been explained, and the algorithm terminates. To summarize, for this example, the ordering algorithm uses one iteration and one test to identify all lossy links.

Let us look at the above example again assuming that we only know probabilistic path information. In this case, we consider three source-sink pairs, all identified as bad from end-to-end data. Denote these bad pairs as $P_1 = (l_2, l_1)$, $P_2 = (l_3, l_1)$, and $P_3 = (l_4, l_1, l_5)$. Since l_1 is used by more source-sink pairs than others, we will test link l_1 first and find it is lossy. After l_1 is repaired, in the next iteration, we find that only P_3 is lossy. Since P_3 uses three links, l_1, l_4, and l_5, and l_1 is used by good pairs, we are left with two potential lossy links, l_4 and l_5. Since both of them are used by one path, we can choose to test either of them. In either case, the algorithm locates the lossy link l_5 and terminates. To summarize, for this example, the ordering algorithm uses two iterations and two tests to identify all lossy links.

5.2 Greedy Algorithm

In each step, this algorithm picks the link that provides the highest *gain* (breaking tie arbitrarily). The gain from knowing the status of a link is defined as the cost savings (from the links that do not need to be tested due to this knowledge) subtracted by the testing cost of this link. More specifically, for link $l_k \in \mathcal{L}$, let θ_k denote the expected gain from knowing the status of link l_k, let θ_{kb} denote the cost savings when knowing l_k is bad, and let θ_{kg} denote the cost savings when knowing l_k is good. From Algorithm 1, θ_{kb} is the sum of the testing costs of all the irrelevant links identified after knowing that l_k is bad. Similarly, from Algorithm 2, θ_{kg} is the sum of the testing costs of all the responsible and irrelevant links identified after knowing that l_k is good. Then, $\theta_k = p_k\theta_{kb} + (1 - p_k)\theta_{kg} - c_k$, where p_k is the probability that l_k is bad, and c_k is the cost of testing l_k.

The complexity of this algorithm is $O(|\mathcal{L}|^3|\mathcal{P}|)$. This is because in each step, it needs to run Algorithms 1 and 2 to obtain the expected gain for each link that remains to be tested, leading to a complexity of $O(|\mathcal{L}||\mathcal{P}|)$ for each link (see Section 4 on the complexities of Algorithms 1 and 2) and a complexity of $O(|\mathcal{L}|^2|\mathcal{P}|)$ for all the links. Since there are at most $O(|\mathcal{L}|)$ steps, the complexity of this scheme is $O(|\mathcal{L}|^3|\mathcal{P}|)$.

We now use the example in Fig. 6 to illustrate this scheme. Let us first assume that we know complete path information. We again first contract l_5 and remove path P_4, and are left with three paths $P_1 = (l_2, l_1)$, $P_2 = (l_3, l_1)$, and $P_3 = (l_4, l_1)$. Then, we calculate the gains from testing the various links as $\theta_1 = 3p + 3(1 - p) - 1 = 2$, $\theta_2 = \theta_3 = \theta_4 = 2(1 - p) - 1 = 0.8$. The expected gain θ_1 is calculated as above because if l_1 is bad, then P_1, P_2, and P_3 are explained and hence we do not need to test links l_2, l_3, and l_4, leading to a saving of 3; if l_1 is good, then l_2, l_3, and l_4 must be bad, leading to a saving of 3. The expected gain θ_2 is $2(1 - p) - 1$ since if l_2 is bad, then it leads to no savings; if l_2 is good, then l_1 must be bad, which makes testing l_3 and l_4 unnecessary (since P_2 and P_3 have been explained by l_1), leading to a saving of 2. The gains θ_3 and θ_4 are obtained in a similar manner as θ_2. Since testing l_1 provides the maximum gain, we test l_1 first. Since l_1 is lossy, all the lossy paths have been explained, and the algorithm terminates. To summarize, for this example, the greedy algorithm uses one iteration and one test to identify all lossy links.

When we only know probabilistic path information, the three lossy source-sink pairs are $P_1 = (l_2, l_1)$, $P_2 = (l_3, l_1)$, and $P_3 = (l_4, l_1, l_5)$. We calculate the gains from testing the various links as $\theta_1 = 4p + 2(1 - p) - 1 = 2p + 1 = 1.4$, $\theta_2 = \theta_3 = 3(1 - p) - 1 = 2 - 3p = 1.4$, $\theta_4 = \theta_5 = -1$. The expected gain θ_1 is calculated as above because if l_1 is bad, then P_1, P_2, and P_3 are explained and hence we do not need to test links $l_2, \ldots, l_5$, leading to a saving of 4; if l_1 is good, then l_2 and l_3 must be bad, leading to a saving of 2. The expected gain θ_2 is $3(1 - p) - 1$ since if l_2 is bad, then it leads to no savings; if l_2 is good, then l_1 must be bad, which makes testing l_3, l_4, and l_5 unnecessary, leading to a saving of 3. The gain θ_3 is obtained in a similar manner. Knowing the status of l_4 or l_5 does not lead to savings, and hence the gain is -1. Since testing links l_1, l_2, and l_3 provide the same highest gain, we break the tie arbitrarily. Suppose we choose to test l_2. We find that l_2 is good, and hence l_1 must be bad, which explains all source-sink pairs. After l_1 is repaired, the second iteration is similar to that under the ordering algorithm. To summarize, for this example, the greedy algorithm uses two iterations and two tests to identify all lossy links.

5.3 Performance under Special Topologies

For a general topology, it is difficult to obtain the approximation ratios of the two heuristic schemes compared to the optimal solution. In the following, we briefly discuss the performance of these two heuristic schemes under line topologies and the simplest three-link tree

topology (optimal solutions for these two types of topologies are described in Sections 4.2 and 4.3, respectively) to provide some insights; their performance under general topologies is studied through simulation in Section 6.

For line topology, $n_i = 1, \forall i$, and hence the ordering scheme picks the link with the highest p_i/c_i to test first. When $c_i = c, \forall i$, it tests the links in decreasing order of their prior probabilities of being lossy, which is an optimal solution. When $p_i = p$, $\forall i$, it tests the links in increasing order of their costs, which is again an optimal solution. Under the greedy scheme, for link l_k, $\theta_{kg} = 0$ and $\theta_{kb} = \sum_{l_j \in \mathcal{L}} c_j - c_k$, and hence $\theta_k = p_k \sum_{l_i \in \mathcal{L}} c_i - p_k c_k - c_k$. It is clear that this scheme also leads to an optimal solution when $c_i = c$, $\forall i$ or $p_i = p$, $\forall i$. In general, however, these two heuristic schemes may not provide optimal solutions. For instance, for the four-link line example in Section 4.2, neither scheme leads to optimal solution. In particular, the ordering scheme tests link l_1 first; if l_1 is good, it tests l_2; if l_2 is good, it tests l_3, leading to an expected testing cost of 14.15. The greedy scheme tests l_3 first; if l_3 is good, it tests l_4; if l_4 is good, it tests l_2, leading to an expected testing cost of 12.51. Therefore, the testing costs of both schemes are larger than the optimal cost of 12.23.

For the three-link tree topology, we describe the performance of the two heuristic schemes under the following special cases:

- Case 1 ($c_i = c$ and $p_i = p$, $\forall i$). Both the ordering and greedy schemes obtain the optimal solution (i.e., testing l_1, the top link in the tree).
- Case 2 ($c_i = c$, $\forall i$, $p_i \neq p$). The optimal solution is testing l_1 (see Section 4.3). The ordering scheme may not choose to test l_1, since $n_1 p_1/c_1 = 2p_1/c$, $n_2 p_2/c_2 = p_2/c$, and $n_3 p_3/c_3 = p_3/c$, which may not satisfy $n_1 p_1/c_1 < n_2 p_2/c_2$ and $n_1 p_1/c_1 < n_3 p_3/c_3$. It is easy to see that the approximate ratio of the ordering scheme for this case is 2 since in the worst case it leads to an expected test cost of $\max(c + p_2 c, c + p_3 c) \leq 2c$. Under the greedy scheme, we have

$$\begin{aligned}\theta_1 &= c_2 + c_3 - c_1 = c, \theta_2 = (1 - p_2)(c_1 + c_3) - c_2 \\ &= c - 2p_2 c < \theta_1, \theta_3 = (1 - p_3)(c_1 + c_2) - c_3 \\ &= c - 2p_3 c < \theta_1.\end{aligned}$$

  Hence, testing l_1 leads to the highest gain, and the greedy scheme tests l_1 first, which is the optimal solution.
- Case 3 ($p_i = p$, $\forall i$, $c_i \neq c$). In this case, neither the ordering scheme nor the greedy scheme is guaranteed to provide an optimal solution. An example is $c_1 = 0.09, c_2 = 0.84, c_3 = 0.07$, and $p_i = 0.2, \forall i$. The optimal solution is testing link l_3 first; if it is lossy, testing link l_1, leading to an optimal testing cost of 0.088. The ordering scheme and greedy schemes both test link l_1, leading to a test cost of 0.09.
- Case 4 ($c_1 = c, c_2 = c_3 = 2c$). When $p_i = p$, both the ordering and greedy schemes obtain the optimal solution. When $p_i \neq p$, the ordering scheme may not lead to an optimal solution. Its approximate ratio is 4 since in the worst case, the expected testing cost is $\max(c_2 + p_2 c_3, c_3 + p_3 c_2) \leq 4c$ while the optimal testing cost is c (i.e., testing l_1). The greedy scheme obtains the optimal solution since

$$\begin{aligned}\theta_1 &= c_2 + c_3 - c_1 = 3c, \theta_2 = (1 - p_2)(c_1 + c_3) - c_2 \\ &= 3(1 - p_2)c - 2c < \theta_1, \theta_3 = 3(1 - p_3)c - 2c < \theta_1,\end{aligned}$$

  and hence the greedy scheme tests l_1 since it leads to the highest gain, which is the optimal solution.

In general, neither scheme is guaranteed to provide an optimal solution in the three-link tree topology. In fact, we can construct scenarios where these two heuristic schemes can lead to much larger testing costs than the optimal solutions.

5.4 Trade-Offs between the Ordering and Greedy Schemes

The greedy scheme has a higher complexity than the ordering scheme. On the other hand, in some of the special settings discussed above, the greedy scheme leads to optimal solutions while the ordering scheme may not lead to optimal solutions. In Section 6, we compare the performance of these two schemes in more general settings where we, however, do not know the prior probability that a link is lossy, and assume that the prior probabilities of all the links are the same. As we shall see, their performances are similar, while the ordering scheme can slightly outperform the greedy scheme.

5.5 Discussion

So far, for ease of exposition, we only consider lossy links. The above two heuristic schemes are also applicable to the more general scenarios where both nodes and links can be lossy. A lossy node can cause its adjacent links to be lossy (e.g., when a node's battery level is low, its incoming and outgoing links can lose packets). Taking account of this, in both ordering and greedy heuristics, when we have a tie between a node and link, we select to test the node first since if it is indeed lossy, once it is repaired, the correlated faulty links caused by it are also repaired.

6 Performance Evaluation

We evaluate the performance of our heuristic algorithms through extensive simulation (using a simulator that we developed) in a sensor network. This network is deployed in a $10 \text{ unit} \times 10 \text{ unit}$ square. A single sink is deployed at the center, and 500 other nodes (sources and/or relays) are uniformly randomly deployed in the square. The transmission range of each node is 3 units. At a given point of time, the paths from the sources to the sink form a reversed tree. In the tree, node u has a directed link to v if v is in the transmission range of u, and forward data for u.

We consider both static and dynamic routings. Under static routing, the paths from the sources to the sink are fixed, and we know the complete path information. More specifically, the paths from the sources to the sink form a spanning tree rooted at the sink. The leaves of the tree are the sources. The number of branches of an intermediate node in the tree is uniformly distributed in $[1, b]$, $b = 10$ or 5. We refer to b as *branch ratio*. A tree generated using $b = 5$ is "taller" and "thinner" than one generated using $b = 10$. Under dynamic routing, the routing tree from the

sources to the sink is chosen randomly from multiple trees at every time unit, and we assume a path report service (as in [22]) that reports path information periodically to the sink. When this service runs at every time unit, it provides complete path information; when it runs at coarser time scales (every 20 time units in our simulation), it provides probabilistic path information and the probability to use a route is estimated from the frequency that this route is used. Note that when we only have probabilistic path information, the topology is a general graph since for each source we consider all the paths from the source to the sink together during fault localization.

For testing costs, we consider two models, referred to as *homogeneous* and *heterogeneous* cost models, respectively. In the homogeneous cost model, all the nodes and links have the same testing costs of 1 unit. This applies to the scenarios where the monitoring overheads at the sensor nodes lead to the most dominant testing cost, and the monitoring overheads at all the sensor nodes are very close. In the heterogeneous cost model, the testing cost of a node or link is proportional to its distance to the sink. More specifically, a node that is k hops away from the sink has a testing cost of k units; a link that is adjacent to the root has a testing cost of 1 unit, and k hops away from the sink has a testing cost of k units. This models the transmission costs of sending monitoring data to the sink: data from a node k hops away from the sink use k links to transmit data back to the sink. Our simulation shows that the results under these two cost models have similar trends. We, therefore, focus on the homogeneous cost model, and only briefly describe the results under the heterogeneous cost model in Section 6.1.3.

The prior probability that a node or link is lossy can be estimated from historical data if a network has been operating for a long time. It can also be estimated online from end-to-end data using the technique in [16]. This technique is, however, computational intensive, and we were not able to obtain results in a reasonable amount of time for our simulation scenarios. In the following, for simplicity, we assume that the prior probabilities are unknown, and all nodes and links have the same probability of being lossy (as in [5]). Under this assumption, the ordering sequential testing scheme does not depend on the exact value of p. For the greedy sequential testing scheme, we set p to 0.2, 0.4, 0.6, or 0.8, and our simulation results show that its performance is not sensitive to the choice of p. The results below are for $p = 0.2$.

The performance metrics we use are the *number of iterations* required to identify all lossy components, and the *normalized testing cost*, which is the total testing cost normalized by the sum of the testing costs of the faulty components. When the testing cost of each component is 1 unit, normalized testing cost is the number of components that are tested divided by the number of lossy components in the network.

We compare our sequential testing schemes with two existing studies [5], [13] that also localize and repair all faults in a network (Section 1 describes them briefly). One approach [13] tests a single link in each iteration, which requires a large number of iterations to localize and repair all lossy links (the number of iterations increases linearly with the number of bad paths, and it can require 30 iterations for only 30 bad paths). To reduce the number of iterations, Lee et al. [13] propose another approach that tests multiple links in parallel. This approach is similar in spirit to the exhaustive inspection approach [5], and under our setting (i.e., all links have the same probability of being lossy and the same testing cost), the best heuristic using this approach is essentially the same as exhaustive inspection. In the following, we only present comparison results with exhaustive inspection.[3] In our preliminary version [25], we also evaluate a baseline sequential testing scheme that randomly selects a component (from the remaining potential lossy components) to test, and show that its performance is much inferior to the greedy scheme. In this paper, in the interest of space, we do not present results under this randomized scheme.

Our simulation results are under three scenarios: where only nodes are lossy, when only links are lossy, and when both nodes and links can be lossy. In the following, we only present the results under the second and third scenarios; the results under the first scenario are similar to those under the second one.

6.1 Lossy Links

We say a link is lossy if its transmission rate lies below 0.8 (i.e., its loss rate is above 0.2); otherwise, it is good. We use two loss models. In the first model, a good link has transmission rate of 0.99 and a bad link has transmission rate of 0.75 (this model is also used in [14], [15]). In the second loss model, a good link has transmission rate uniformly distributed in $[\alpha, 1]$ and a bad link has transmission rate uniformly distributed in $[0, \beta]$, $\alpha > 0.8$ and $\beta < 0.8$. We only describe the results under the second loss model; the results under the first one are similar.

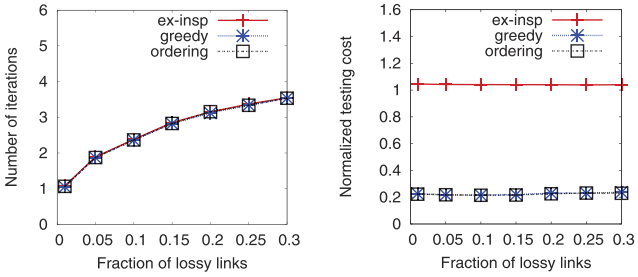
We assume the losses at a link follow a Bernoulli or Gilbert process. Under Bernoulli process, a packet traversing a link is dropped with a probability that is equal to the link loss rate. Under Gilbert process, a link is in a good or bad state. When in a good state, the link does not drop any packet; while in a bad state, the link drops all packets. The transition probabilities between good and bad states are chosen so that the average loss rate is what assigned to the link. We only show the results under Bernoulli loss process; the results under Gilbert loss process are similar (since our algorithms use average loss rates and hence are not sensitive to the loss process).

For each topology, we vary the percentage of lossy links from 1 to 30 percent, and randomly choose lossy links. For simplicity, we assume that once a lossy link is repaired, it remains good (although our schemes can handle the case where a repaired link becomes lossy again in a later iteration).

6.1.1 Static Routing

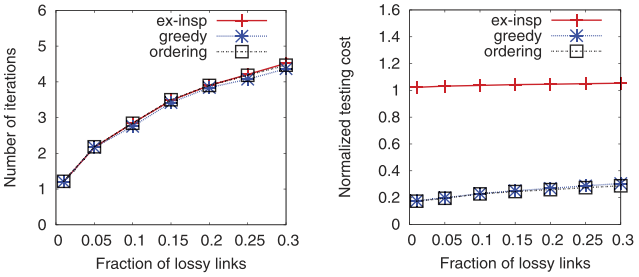
In this setting, a good link has reception rate uniformly distributed in $[0.95, 1]$; a bad link has reception rate uniformly distributed in $[0, 0.60]$. We can find a threshold

3. When implementing exhaustive inspection, we use the inference algorithm in [16], which is applicable to general topologies and is the same as that in [5] for a tree topology (the algorithm in [5] only considers trees).



(a) Number of iterations. (b) Normalized testing cost.

Fig. 7. Lossy link scenario, simulation results under static routing, $b = 10$.



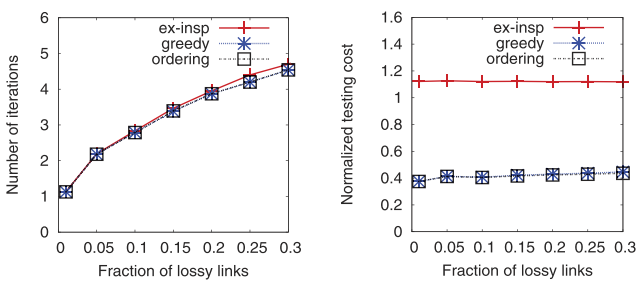
(a) Number of iterations. (b) Normalized testing cost.

Fig. 9. Lossy link scenario, simulation results under dynamic routing when knowing complete path information, $b = 5$.

to determine whether a path is good or bad. This is because for a path of h links, its reception rate is at least 0.95^h if it is good and is at most 0.60 otherwise. We have $h \leq 5$ when the branch ratio is 10 (i.e., $b = 10$) and $h \leq 7$ when the branch ratio is 5 (i.e., $b = 5$). In both cases, $0.95^h > 0.60$, and hence we can find a threshold to determine whether this path is good or bad. In particular, we use $(0.95^h + 0.60)/2$ as the threshold. In each iteration, a source sends 400 packets to the sink; the reception rate of a path is estimated from these packets (when using 400 packets, the decision on whether a path is good or bad is correct with probability close to one in our setting). Each confidence interval below is obtained from 150 simulation runs, using five randomly generated routing trees and 30 simulation runs in each tree. We do not plot them in the figures since they are very tight.

Fig. 7 plots the results when $b = 10$. The results of exhaustive inspection, and greedy and ordering sequential testing schemes are plotted in the figure. We observe that all three schemes require only a few iterations to localize all lossy links. Under exhaustive inspection, the normalized testing cost is at least 1 (it is higher than 1 since some identified links are false positives). The two sequential testing schemes have similar normalized testing costs (the two curves overlap in the figure), which are around 0.2 for all the settings, indicating that only a small fraction of links (0.002 to 0.06 when the fraction of lossy link changes from 0.01 to 0.30) needs to be tested to localize all lossy links.

Fig. 8 plots the results when $b = 5$. As expected, the number of iterations and the normalized testing cost are larger than those when $b = 10$ for all the three schemes, since end-to-end data in a "taller" and "thinner" tree posses a larger amount of ambiguity in determining the status of individual links. In this case, all three schemes still need only a few iterations to localize all lossy links;



(a) Number of iterations. (b) Normalized testing cost.

Fig. 8. Lossy link scenario, simulation results under static routing, $b = 5$.

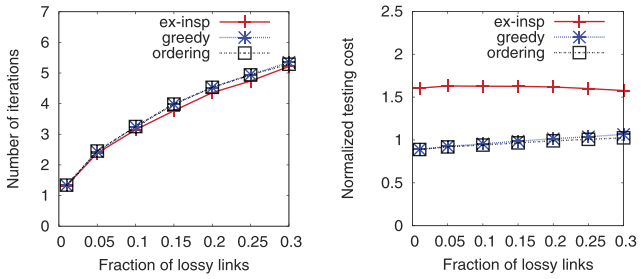
greedy and ordering sequential testing schemes still maintain low normalized testing costs (around 0.4 for all the settings), while exhaustive inspection has normalized testing costs above 1.

6.1.2 Dynamic Routing

Under dynamic routing, we assume that the routing tree from the sources to the sink is chosen randomly with equal probability from two trees at every time unit. This is motivated from the measurement results that for each source, a small number of paths carry most traffic in a sensor network under dynamic routing [15]. The two routing trees have the same set of leaves (which are the sources), and differ in the structure of intermediate nodes (we form the second tree by randomly changing the parents of the nodes in the first tree). Again, the number of branches of an intermediate node is uniformly distributed in $[1, b]$, $b = 10$ or 5. In all the topologies we generated, above 94 and 91 percent of the sources use two paths when $b = 10$ and 5, respectively, the rest of the sources use one path. For a source that uses two paths, the path lengths are the same. Again, each confidence interval below is obtained from 150 simulation runs, over five randomly generated topologies and 30 simulation runs in each topology; we omit them in the figures since they are very tight.

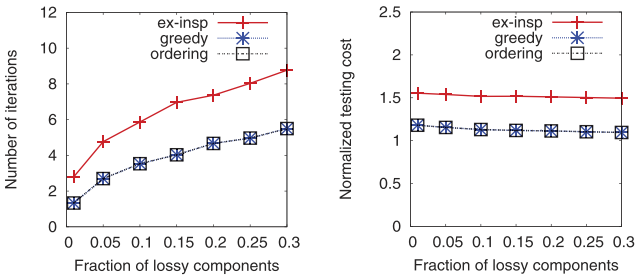
When knowing complete path information, the settings for good and bad links are the same as those in the static routing. In each iteration, a source sends 800 packets to the sink (if a source uses two paths, approximately 400 packets are sent on each path). Fig. 9 plots the results when $b = 5$ (the results when $b = 10$ are slightly better, figures omitted). Again, all three schemes require a few iterations to localize all lossy links. The normalized testing costs under all three schemes are lower than those under static routing (see Fig. 8). This is because the path diversity when using two routing trees leads to less ambiguity than that when using a single routing tree. Last, the normalized testing costs under greedy and ordering sequential testing schemes are below 0.3 for all the settings, much lower than those under exhaustive inspection.

When knowing probabilistic path information, a good link has reception rate uniformly distributed in $[0.99, 1]$; a bad link has reception rate uniformly distributed in $[0, 0.60]$. In this setting, we can find a threshold to determine whether a source-sink pair is good or bad. This is because, for a source using two paths with equal probability, each path of h links, the source-sink reception rate is at least 0.99^h



(a) Number of iterations. (b) Normalized testing cost.

Fig. 10. Lossy link scenario, simulation results under dynamic routing when knowing probabilistic path information, $b = 5$.



(a) Number of iterations. (b) Normalized testing cost.

Fig. 11. Simulation results under static routing when both nodes and links can be lossy, $b = 10$.

when the source-sink pair is good, and is at most $(1 + 0.6)/2$ otherwise. Since $0.99^h > (1 + 0.6)/2$ ($h \leq 5$ when $b = 10$ and $h \leq 7$ when $b = 5$), we can find a threshold to determine whether the source-sink pair is good or bad. In particular, we choose the threshold to be in the middle of $(1 + 0.6)/2$ and 0.99^h. In each iteration, a source sends 800 packets to the sink, and the path report service sends 40 path reports (it runs every 20 time units) to the sink for each source.

Fig. 10 plots the results under probabilistic path information when $b = 5$ (the results when $b = 10$ are slightly better, figures omitted). Again, all three schemes require a few iterations, and the two sequential testing schemes incur much lower testing costs than exhaustive inspection. Observe that the number of iterations under the two sequential testing schemes is slightly larger than that under exhaustive inspection, which is opposite to the trend when knowing complete path information (Figs. 8a and 9a). This difference is due to the different amount of ambiguity in these two settings: the setting with only probabilistic path information has much more ambiguity than the setting with complete path information (since the end-to-end information in the former is for each source-sink pair, containing all the links that are used by a source-sink pair over all paths, while the end-to-end information in the latter is for each individual path). Also, observe that the testing costs under all three schemes are much larger than those when knowing complete path information. This is again due to the much larger amount of ambiguity when not knowing the exact path information. Under the sequential testing schemes, the number of links tested is close to the number of lossy links (the normalized testing cost is close to 1). Furthermore, for large fraction of lossy links, the ordering heuristic slightly outperforms the greedy heuristic. This is because, when all the links have the same prior probability of being lossy, the ordering scheme does not need to know the exact value, while the greedy scheme uses the value to calculate the gain from testing a link, and hence can be affected by inaccurate estimates of this value (as in our case).

6.1.3 Heterogenous Cost Model

Results under the heterogeneous cost model have similar trends as those under the homogeneous cost model. However, we observe that the testing costs of the two heuristic sequential testing schemes have slightly more dramatic difference under the heterogeneous cost model. For instance, under static routing and $b = 10$, the normalized testing costs using the greedy scheme are 4 to 7 percent higher than those using the ordering scheme (figure omitted), while in the homogeneous model, the performance of these two schemes are very close (see Fig. 7). This might be because the performance of the greedy scheme is impacted more significantly by the inaccurate estimates of the prior probabilities that links are lossy under the heterogeneous cost model.

6.2 Lossy Nodes and Links

When both nodes and links can be lossy, we again say a node or link is bad if its transmission rate lies below 0.8; otherwise, it is good. A good node or link has transmission rate uniformly distributed in $[\alpha, 1]$ and a bad link has transmission rate uniformly distributed in $[0, \beta]$, $\alpha > 0.8$ and $\beta < 0.8$. Furthermore, we assume that when a node is lossy, it causes all incoming and outgoing links to be lossy. When choosing faulty components, we first decide whether to choose a node or link (with equal probability), and then randomly choose a node or link to be faulty. The percentage of faulty components is from 1 to 30 percent. Since a lossy node causes its adjacent links to be lossy, the number of lossy components is larger than the specified value.

We again investigate static and dynamic routings, and find that in all the cases the two sequential test schemes have similar performances, and both require significantly less iterations and lower testing costs than exhaustive inspection. Fig. 11 shows one result under static routing, $b = 10$. We observe that the number of iterations required by sequential test schemes is 37 to 52 percent lower than that by exhaustive inspection, and the normalized inspection cost is around 26 percent lower than that by exhaustive inspection. The more dramatic difference in the number of iterations compared to those when only links are lossy is because sequential testing schemes are more effective in detecting and repairing lossy nodes in earlier iterations. When a lossy node is repaired, all the links that are faulty due to this node become good, which leads to less ambiguity in end-to-end results, and hence reduces the number of iterations.

7 Conclusions and Future Work

In this paper, we formulated an optimal sequential testing problem that carefully combines active and passive measurements for fault localization in wireless sensor networks. This problem determines an optimal testing sequence of

network components based on end-to-end data to minimize testing cost. We proposed a recursive approach and two heuristic algorithms to solve it. Extensive simulation demonstrated that our heuristic algorithms only require a few iterations and testing a small subset of components to identify all lossy components in a network.

As future work, we are pursuing in the following two directions: 1) evaluating the performance of our approach under other scenarios, for instance, when the location of faulty components follows a more clustered distribution instead of uniform random distribution and 2) developing more scalable approaches to reduce the overhead in obtaining complete path information under dynamic topologies.

ACKNOWLEDGMENTS

A preliminary version of this paper appeared in [25]. This work was partially supported by US National Science Foundation CAREER award 0746841 and Qualtech Systems Incorporated. The authors would like to thank Yoo-Ah Kim, Jim Kurose, Prashant Shenoy, and Patrick Thiran for their helpful comments and suggestions. They also thank the anonymous reviewers for their insightful comments and associate editor Saurabh Bagchi for handling the paper.

REFERENCES

[1] A. Adams, T. Bu, R. Caceres, N. Duffield, T. Friedman, J. Horowitz, F.L. Presti, S. Moon, V. Paxson, and D. Towsley, "The Use of End-to-End Multicast Measurements for Characterizing Internal Network Behavior," *IEEE Comm. Magazine,* vol. 38, no. 5, pp. 152-159, May 2000.

[2] M. Adler and B. Heeringa, "Approximating Optimal Binary Decision Trees," *Proc. Int'l Workshop Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques (APPROX/RANDOM '08),* Aug. 2008.

[3] V.T. Chakaravarthy, V. Pandit, S. Roy, P. Awasthi, and M. Mohania, "Decision Trees for Entity Identification: Approximation Algorithms and Hardness Results," *Proc. 26th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems,* 2007.

[4] W. Dong, X. Liu, C. Chen, Y. He, G. Chen, Y. Liu, and J. Bu, "DPLC: Dynamic Packet Length Control in Wireless Sensor Networks," *Proc. IEEE INFOCOM,* Mar. 2010.

[5] N. Duffield, "Network Tomography of Binary Network Performance Characteristics," *IEEE Trans. Information Theory,* vol. 52, no. 12, pp. 5373-5388, Dec. 2006.

[6] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks," Technical Report UCLA/CSD-TR 02-0013, Feb. 2002.

[7] M.R. Garey, "Optimal Binary Identification Procedures," *SIAM J. Applied Math.,* vol. 23, no. 2, pp. 173-186, Sept. 1972.

[8] C. Gruenwald, A. Hustvedt, A. Beach, and R. Han, "SWARMS: A Sensornet Wide Area Remote Management System," *Proc. Third Int'l Conf. Testbeds and Research Infrastructure for the Development of Networks and Communities (TridentCom),* May 2007.

[9] G. Hartl and B. Li, "Loss Inference in Wireless Sensor Networks Based on Data Aggregation," *Proc. Third Int'l Symp. Information Processing in Sensor Networks (IPSN),* Apr. 2004.

[10] C.-F. Hsin and M. Liu, "A Distributed Monitoring Mechanism for Wireless Sensor Networks," *Proc. First ACM Workshop Wireless Security (WiSe),* Sept. 2002.

[11] P. Jelenkovic and J. Tan, "Dynamic Packet Fragmentation for Wireless Channels with Failures," *Proc. ACM MobiHoc,* May 2008.

[12] J. Korhonen and Y. Wang, "Effect of Packet Size on Loss Rate and Delay in Wireless Links," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC),* Mar. 2005.

[13] P.P. Lee, V. Misra, and D. Rubenstein, "Toward Optimal Network Fault Correction in Externally Managed Overlay Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 21, no. 3, pp. 354-366, Mar. 2010.

[14] Y. Mao, F.R. Kschischang, B. Li, and S. Pasupathy, "A Factor Graph Approach to Link Loss Monitoring in Wireless Sensor Networks," *IEEE J. Selected Areas in Comm.,* vol. 23, no. 4, pp. 820-829, Apr. 2005.

[15] H.X. Nguyen and P. Thiran, "Using End-to-End Data to Infer Lossy Links in Sensor Networks," *Proc. IEEE INFOCOM,* Apr. 2006.

[16] H.X. Nguyen and P. Thiran, "The Boolean Solution to the Congested IP Link Location Problem: Theory and Practice," *Proc. IEEE INFOCOM,* May 2007.

[17] K.R. Pattipati and M.G. Alexandridis, "Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis," *IEEE Trans. Systems, Man and Cybernetics,* vol. 20, no. 4, pp. 872-887, July/Aug. 1990.

[18] V. Raghavan, M. Shakeri, and K.R. Pattipati, "Optimal and Near-Optimal Test Sequencing Algorithms with Realistic Test Models," *IEEE Trans. Systems, Man and Cybernetics,* vol. 29, no. 1, pp. 11-26, Jan. 1999.

[19] 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),* Nov. 2005.

[20] N. Reijers, G. Halkes, and K. Langendoen, "Link Layer Measurements in Sensor Networks," *Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS),* Oct. 2004.

[21] S. Rost and H. Balakrishnan, "Memento: A Health Monitoring System for Wireless Sensor Networks," *Proc. Third Ann. IEEE Comm. Soc. on Sensor and Ad Hoc Comm. and Networks (SECON),* Sept. 2006.

[22] T. Schmid, H. Dubois-Ferriére, and M. Vetterli, "SensorScope: Experiences with a Wireless Building Monitoring," *Proc. Workshop Real-World Wireless Sensor Networks,* June 2005.

[23] G. Tolle and D. Culler, "Design of an Application-Cooperative Management System for Wireless Sensor Networks," *Proc. Second European Workshop Wireless Sensor Networks (EWSN),* Jan. 2005.

[24] M. Vuran and I. Akyildiz, "Cross-Layer Packet Size Optimization for Wireless Terrestrial, Underwater, and Underground Sensor Networks," *Proc. IEEE INFOCOM,* Apr. 2008.

[25] B. Wang, W. Wei, W. Zeng, and K.R. Pattipati, "Fault Localization Using Passive End-to-End Measurement and Sequential Testing for Wireless Sensor Networks," *Proc. Ann. IEEE Comm. Soc. Conf. Sensor, Mesh and Ad Hoc Comm. and Networks (SECON),* June 2009.

[26] D.B. West, *Introduction to Graph Theory,* second ed. Prentice Hall, Sept. 2000.

[27] K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, and D. Culler, "Marionette: Providing an Interactive Environment for Wireless Debugging and Development," *Proc. Fifth Int'l Conf. Information Processing in Sensor Networks (IPSN),* Apr. 2006.

[28] A. Woo, T. Tong, and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," *Proc. First Int'l Conf. Embedded Networked Sensor Systems (SenSys),* Nov. 2003.

[29] J. Zhao, R. Govindan, and D. Estrin, "Residual Energy Scans for Monitoring Wireless Sensor Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC),* Mar. 2002.

[30] J. Zhao, R. Govindan, and D. Estrin, "Computing Aggregates for Monitoring Wireless Sensor Networks," *Proc. IEEE Int'l Workshop Sensor Network Protocols and Applications (SNPA),* May 2003.

[31] G. Zhou, T. He, S. Krishnamurthy, and J.A. Stankovic, "Models and Solutions for Radio Irregularity in Wireless Sensor Networks," *ACM Trans. Sensor Networks,* vol. 2, no. 2, pp. 221-262, May 2006.

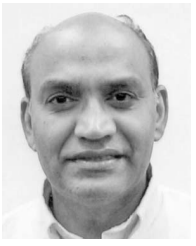
[32] M. Zuniga and B. Krishnamachari, "An Analysis of Unreliability and Asymmetry in Low-Power Wireless Links," *ACM Trans. Sensor Networks,* vol. 3, no. 2, June 2007.

Bing Wang received the BS degree in computer science from Nanjing University of Science & Technology, China, in 1994, and the MS degree in computer engineering from Institute of Computing Technology, Chinese Academy of Sciences, in 1997. She then received two MS degrees in computer science and applied mathematics and the PhD degree in computer science from the University of Massachusetts, Amherst, in 2000, 2004, and 2005, respectively. Afterward, she joined the Computer Science and Engineering Department at the University of Connecticut as an assistant professor. Her research interests are in computer networks, multimedia, and distributed systems. She received the US National Science Foundation CAREER award in February 2008. She is a member of the IEEE and the IEEE Computer Society.

Wei Wei received the BS degree in applied mathematics from Beijing University, China, in 1992, and the MS degree in statistics from Texas A&M University in 2000. He then received two MS degrees in computer science and applied mathematics and the PhD degree in computer science from the University of Massachusetts, Amherst, in 2004, 2004, and 2006, respectively. He is currently a senior postdoctoral researcher in the Department of Computer Science at the University of Massachusetts, Amherst. His research interests are in the areas of computer networks, distributed embedded systems, and performance modeling. He is a member of the IEEE and the IEEE Computer Society.

Hieu Dinh received the BS degree in computer science from the Vietnam National University at Hanoi in 2005 and the MS degree in computer science and engineering from the University of Connecticut in 2008. He is currently working toward the PhD degree in the Computer Science and Engineering Department at the University of Connecticut, working on combinatorial optimization and approximation algorithms with applications in networks and bioinformatics.

Wei Zeng received the BS and MS degrees in computer science and engineering from the South China University of Technology, Guangzhou, China, in 2000 and 2003, respectively. Currently, she is working toward the PhD degree in the Computer Science and Engineering Department at the University of Connecticut. Her research topics are in network diagnosis, network measurement, and network management for wireless sensor networks. She is a student member of the IEEE.



Krishna R. Pattipati is a professor of electrical and computer engineering at the University of Connecticut, Storrs. He has published more than 330 articles, primarily in the application of systems theory and optimization techniques to large-scale systems. He received the Centennial Key to the Future Award in 1984 from the IEEE Systems, Man and Cybernetics (SMC) Society and was elected a fellow of the IEEE in 1995 for his contributions to discrete-optimization algorithms for large-scale systems and team decision making. He received the Andrew P. Sage Award for the best *IEEE Transactions on Systems, Man, and Cybernetics* paper for 1999, the Barry Carlton award for the best *IEEE Transactions on Aerospace and Electronic Systems* paper for 2000, the 2002 NASA Space Act Award, the 2003 AAUP Research Excellence Award, and the 2005 School of Engineering Teaching Excellence Award at the University of Connecticut. He also won best technical paper awards at the 1985, 1990, 1994, 2002, 2004, and 2005 IEEE AUTOTEST Conferences and at the 1997 and 2004 Command and Control Conferences. He served as editor-in-chief of the *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* during 1998-2001.