Probabilistic Fault Localization in Communication Systems Using Belief Networks

Małgorzata Steinder, *Member, IEEE,* and Adarshpal S. Sethi, *Member, IEEE*

***Abstract*—We apply Bayesian reasoning techniques to perform fault localization in complex communication systems while using dynamic, ambiguous, uncertain, or incorrect information about the system structure and state. We introduce adaptations of two Bayesian reasoning techniques for polytrees, iterative belief updating, and iterative most probable explanation. We show that these approximate schemes can be applied to belief networks of arbitrary shape and overcome the inherent exponential complexity associated with exact Bayesian reasoning. We show through simulation that our approximate schemes are almost optimally accurate, can identify multiple simultaneous faults in an event driven manner, and incorporate both positive and negative information into the reasoning process. We show that fault localization through iterative belief updating is resilient to noise in the observed symptoms and prove that Bayesian reasoning can now be used in practice to provide effective fault localization.**

***Index Terms*—Fault localization, probabilistic inference, root cause diagnosis.**

I. Introduction

TO IMPROVE the reliability and performance of a communication system it is important to quickly and accurately detect and diagnose its faults. Fault localization (also problem determination or root cause diagnosis) [24], [28], [54] isolates the most probable set of faults based on their external manifestations called symptoms or alarms. Fault localization techniques available today concentrate on detecting and isolating faults related to network connectivity [15], [28], [51], [54]. The diagnosis focuses on lower layers of the protocol stack (typically the physical and data-link layers) [36], [54], and its major goal is to isolate faults related to the availability of network resources, such as a broken cable, an inactive interface, etc. These techniques are usually deterministic, i.e., they assume that the dependencies among the system components are known with certainty [24], [54] or that the information about the current system state is always accurate and complete [19], [28]. Most fault localization techniques reported in the literature are window based [28], [29], [51], i.e., they analyze groups of symptoms that are collected over a period of time, called time window. Such techniques are inadequate for use in the management of modern enterprise services, which impose new challenges on the fault localization task. In addition to resource availability problems in lower layers of the protocol stack, high-level and performance-related problems have to be diagnosed. The root causes of higher layer problems may be located in a different protocol layer or on a different network host, which is frequently not visible to the higher layers. Moreover, information about the system's internal structure may be incomplete or out of date, and it changes dynamically. It is also likely for the observation of the system state to be ambiguous or inaccurate. In complex, multilayer communication systems it is difficult to determine symptom latencies in order to determine the length of an appropriate correlation time window and it may be necessary to intertwine fault localization with testing. Thus, window-based fault localization is no longer sufficient.

Considering these difficulties, we argue that in complex multilayer communication systems, fault localization techniques should aim at the following objectives. They should:

1) be *able to isolate multiple simultaneous faults* even if their symptoms overlap.
2) be *accurate* and *efficient*.
3) be *resilient to lost or spurious alarms*.
4) be *event-driven* rather than window based. Event-driven fault localization techniques maintain the system state that is updated after every symptom observation. They allow a symptom to be analyzed independently of other symptoms, which may happen as soon as the symptom is received by the fault localization process. Therefore, they utilize time more efficiently. In addition, they allow fault localization to be intertwined with testing procedures, which, with event-driven techniques, may be designed dynamically based on the analysis of the previously observed symptoms.
5) *facilitate other fault management tasks* such as on-demand testing and impact analysis.

A. Contributions of This Paper

This paper investigates an application of Bayesian reasoning using belief networks [37] to nondeterministic fault diagnosis in complex communication systems. The paper makes the following contributions to the field of fault management.

- It advances the state of the art in fault management by introducing the requirements of: 1) applicability to various fault-localization problems, 2) event-driven diagnosis, and 3) facilitating fault-management tasks other than fault localization. The paper also emphasizes and justifies the importance of the technique's ability: 1) to isolate multiple simultaneous faults, 2) to deal with positive, lost, and spurious symptoms, and 3) to be accurate even if the system model is not.

Manuscript received August 23, 2002; revised June 2, 2003; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor G. Pacifici. This work was supported in part by the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technologyp Alliance Program under Cooperative Agreement DAAD19-01-2-0011.

M. Steinder was with the Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716 USA. She is now with the IBM T. J. Watson Research Center, Hawthorne, NY 10532 USA (e-mail: steinder@us.ibm.com).

A. S. Sethi is with the Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716 USA (e-mail: sethi@cis.udel.edu).

Digital Object Identifier 10.1109/TNET.2004.836121

- It applies two Bayesian inference algorithms that calculate belief-updating and most-probable-explanation queries in singly connected belief networks [37] to perform fault localization in belief networks with loops. It proposes a heuristic that applies the belief-updating algorithm [37] to perform event-driven diagnosis in nonpolytree belief networks representing noisy-OR probability distribution and, based on the results of belief updating, calculates the explanation hypothesis. To apply the most-probable-explanation algorithm [37] the paper uses an approximation that reduces the algorithm's complexity to polynomial.
- It extends the system model to incorporate uncertainty involved in observations of system state resulting from lost and spurious symptoms. It shows that the proposed algorithms may be effectively used to perform fault localization in the presence of observation noise.
- It evaluates and compares the proposed algorithms using the problem of end-to-end service failure diagnosis as a case study. It also analyzes the impact of including lost and spurious symptoms into the analysis.
- It proposes an application of other canonical belief network models to solve fault management problems for which the noisy-OR model is inadequate.

B. Paper Organization

In Section II, we define belief network concepts used in this paper. We also motivate the application of belief networks to modeling fault propagation patterns in communication systems.

In Section III, we discuss the application of known algorithms for performing Bayesian reasoning in belief networks to fault localization. They include the bucket elimination framework [12] and two algorithms for calculating queries in singly connected belief networks (polytrees): 1) belief updating and 2) most-probable explanation [37]. Then, we design and present the approximate polynomial fault localization techniques which utilize these algorithms.

Section IV extends the fault propagation model to incorporate uncertainty involved in observations of system state resulting from lost and spurious symptoms.

In Section V, the algorithms proposed in Section III are evaluated through simulation using the problem of end-to-end service failure diagnosis as a case study. We compare the performance and accuracy of algorithms proposed in Section III and investigate whether through increasing the system instrumentation and taking observation noise into account the robustness of nondeterministic techniques may indeed be improved.

In Section VI, we discuss the examples of fault management problems for which the noisy-OR model is inadequate and propose the application of other canonical belief network models.

Section VII compares the approach proposed in this paper against the related work.

II. Belief Network as Fault Propagation Model

Most fault localization techniques [7], [17], [19], [27], [28], [41], [54], including the technique introduced in this paper, rely on a fault propagation model, which represents information about the impact of system entity failures on other system entities. Fault propagation models expose information about the system structure that is useful to fault localization while hiding information that is irrelevant to this process. By providing this abstract view of the system, fault propagation models facilitate the development of fault localization techniques that are applicable to a wide range of fault localization problems, i.e., related to failures occurring in various protocol layers, both in hardware and software, and on various levels of abstraction.

Fault management systems model fault propagation by representing either causal relationships among events [7], [19], [54] or dependencies among (both abstract and physical) communication system entities [17], [27], [28], [41]. A dependency-graph representation is more natural as it encodes perceivable relationships among system components. This representation is also easier to obtain as the relationships among system entities are frequently explicitly decided at the system design stage. However, the representation of causal relationships among events presents fault localization task with more precise information on fault propagation patterns. It is also easier to use because it allows the fault localization technique to deal with a simple notion of an event rather than with possibly multitype and multistate system entities.

The fault-propagation modeling approach adopted in our research uses a layered nondeterministic dependency graph as a system model, which is mapped into a probabilistic causality graph [49], [50]. The dependency graph is a layered template that associates multiple failure modes with every physical or abstract system component which is conceptualized as a *function* or a *service*. The template may be refined by developing micromodels of functions and services to provide a higher degree of detail. Our previous work on this subject discusses the dependency-graph model in detail and shows a general technique of mapping this model into a causality graph [49], [50].

It should be pointed out that obtaining dependency-graph representations of complex multilayer systems is a challenging problem as many dependencies are hidden or dynamic. From the research performed so far, it is fairly obvious that the precise methodology of obtaining dependency information depends on the type of system and its various characteristics. Thus, the discussion of techniques that could be used to build a fault propagation model is beyond the scope of this paper. However, there are a number of publications that focus on this issue while investigating its various aspects [4], [26], [35], [39]. In Section V-A, we cite some of the techniques that can be used to build a model for end-to-end service failure diagnosis, which is the subject of our experimental study.

Building upon our previous work [49], which maps a dependency graph into a causality graph, this paper uses a causality graph as a fault propagation model. A causality graph may be easily interpreted as a belief network with binary-valued nodes. Belief networks are attractive to use as a fault propagation model because they provide a powerful yet intuitive representation of causality—a concept at the core of fault propagation. As a fault propagation model they are general enough to represent various types of relationships that may exist between both abstract and physical components of a communication system and provide the right degree of abstraction by exposing only the information that is critical to the fault localization process while hiding the specifics of the managed resources. In addition, they are well understood as a formalism and are accompanied by extensively investigated algorithmic techniques.

A *belief network* [12], [37] is a directed acyclic graph (DAG) in which each node represents a random variable

over a multivalued domain. We will use terms "node" and "random variable" interchangeably and denote them by V_i. The set of all nodes is denoted by V. The domain of random variable V_i will be represented by symbol D_i. The set of directed edges E denotes an existence of causal relationships between the variables and the strengths of these influences are specified by conditional probabilities. Formally, a belief network is a pair (G, P), where G is a DAG, $P = \{P_i\}$, and P_i is the conditional probability matrix associated with a random variable V_i. Let $Par(V_i) = \{V_{i_1}, \ldots, V_{i_n}\}$ be the set of all parents of V_i. P_i is a $(|Par(V_i)| + 1)$-dimensional matrix of size $|D_i| \times |D_{i_1}| \times \cdots \times |D_{i_n}|$, where $P_i(v_i, v_{i_1}, \ldots, v_{i_n}) = P(V_i = v_i | V_{i_1} = v_{i_1}, \ldots, V_{i_n} = v_{i_n})$. We will denote by $\mathcal{A} = \{V_1 = v_1, \ldots, V_n = v_n\}$ an assignment of values to variables in set V where each $v_j \in D_j$. We will use $v_j^{\mathcal{A}}$ to denote the value of variable $V_j \in V$ in assignment $\mathcal{A}$. Given a subset of random variables $U_k = \{V_{k_1}, \ldots, V_{k_m}\} \subseteq V$, we will denote by $U_k^{\mathcal{A}} = \{V_{k_1} = v_{k_1}^{\mathcal{A}}, \ldots, V_{k_m} = v_{k_m}^{\mathcal{A}}\}$ an assignment of values to variables in set U_k that is consistent with assignment $\mathcal{A}$. An evidence set e is an assignment $U_o^{\mathcal{A}}$, where $U_o \subseteq V$ is a set of variables whose values are known, and for each $V_{o_j} \in U_o$, $v_{o_j}^{\mathcal{A}}$ is its known value.

Belief networks are used to make four basic queries given evidence set e: belief assessment, most probable explanation, maximum *a posteriori* hypothesis, and maximum expected utility [12]. The first two queries are of particular interest in the presented research. The *belief assessment* task is to compute $bel(V_i = v_i) = P(V_i = v_i | e)$ for one or more variables V_i. The *most probable explanation* (MPE) task is to find an assignment $\mathcal{A}_{\max}$ that best explains the observed evidence e, i.e., $P(\mathcal{A}_{\max}) = \max_{\mathcal{A}} \Pi_{i=1}^{n} P\left(V_i = v_i^{\mathcal{A}} | Par(V_i)^{\mathcal{A}}\right)$ [12]. It is known that these tasks are NP-hard in general belief networks [8]. A belief updating algorithm, polynomial with respect to $|V|$, is available for *polytrees*, i.e., directed graphs without undirected cycles [37]. However, in unconstrained polytrees, the propagation algorithm still has an exponential bound with respect to the number of a node's neighbors.

Since exact inference in belief networks is NP-hard, approximation techniques have been investigated [13], [37], [40]. To the best of our knowledge, no approximation has been proposed that works well for all types of networks. Moreover, some approximation schemes have been proven to be NP-hard [10].

In this paper, we focus on a class of belief networks representing a simplified model of conditional probabilities called *noisy-OR gates* [37]. The simplified model contains binary-valued random variables. It associates an inhibitory factor with every cause of a single effect. The effect is absent only if all inhibitors corresponding to the present causes are activated. Thus, instead of conditional probability matrices associated with belief network nodes, the noisy-OR belief network assigns conditional probability values to the belief network edges. The model assumes that all inhibitory mechanisms are independent [20], [37]. This assumption of independence is ubiquitous in probabilistic fault localization approaches reported in the literature [28], [29]. It indicates that all alternative causes of the same effect are independent, which is consistent with system model refinements discussed in [49]. These refinements help avoid exponential time and memory otherwise needed to process and store conditional probability matrices associated with random variables in a belief network. Furthermore, belief assessment in polytrees with the noisy-OR model has polynomial complexity, which makes it attractive as an approximation scheme to solve the fault localization problem.

This paper uses a belief network whose nodes are $\{0, 1\}$-valued random variables. A variable represents a failure of a particular system entity. An assignment of one or zero indicates that the system entity experiences or does not experience the represented failure, respectively. Several distinct variables may be associated with the same entity to represent its various failures [49]. The fact that a failure of one entity may cause a failure of another entity is represented by a causal edge between the corresponding belief network nodes, which is weighted with the probability of the causal implication.

A symptom is defined as an observation that an entity experiences a particular failure (*negative* symptom) or that it does not experience this failure (*positive* symptom). We will denote by $\mathcal{S}$ the set of all possible symptoms. If V_i is a belief network node corresponding to a failure of a system entity, then the negative symptom is interpreted as an instantiation of V_i with value one, and the positive symptom is interpreted as an instantiation of V_i with value zero. The set of all observed negative symptoms and the set of all observed positive symptoms will be denoted by $\mathcal{S}_N$ and $\mathcal{S}_P$, respectively. The set of all observed symptoms, which will be denoted by $\mathcal{S}_O = \mathcal{S}_N \cup \mathcal{S}_P \subseteq \mathcal{S}$, becomes the evidence set e. Note that in general, $\mathcal{S}_N \cup \mathcal{S}_P \neq \mathcal{S}$, as some symptoms may be unobservable. If a symptom is unobservable, e.g., as a result of the current management system configuration, the lack of a negative observation may not be interpreted as a positive observation. The ratio $|\mathcal{S}_O|/|\mathcal{S}|$ will be called an *observability ratio* (OR). The belief network in which $\mathcal{S}_O$ represents the set of all observable symptoms will be denoted by $\mathcal{BN}(\mathcal{S}_O)$.

A fault is a failure of a system entity that may not be further explained given the fault propagation model. It is represented by the assignment of one to the corresponding belief network node. The set of all possible faults is denoted by $\mathcal{F}$. The fault localization problem is to find the set of faults $\mathcal{F}_c \subseteq \mathcal{F}$ that best explains the set of observed symptoms $\mathcal{S}_O$, which may be solved by computing the MPE query in belief network $\mathcal{BN}(\mathcal{S}_O)$ based on the evidence set e.

III. Fault Localization Techniques

In this section, we address the problem of finding the set of root problems that best explains the set of observed symptoms using a belief network as a fault propagation model. In general, the problem is known to be NP-hard [8]; the exact calculation of the best explanatory hypothesis requires a number of steps that is exponential with respect to the number of graph nodes. One of the most popular exact algorithms is bucket elimination [12]. We use this algorithm as a reference algorithm against which algorithms introduced in this paper are compared. In Sections III-B and III-C, we present two algorithms derived from Pearl's iterative propagation in polytrees [37], which we adapt to solving the fault localization problem in arbitrary belief networks. These algorithms were first introduced in [50]. The first algorithm, which is presented in Section III-B, utilizes iterative belief updating to calculate the marginal posterior probability distribution given the observed evidence. From this probability distribution, we derive an approximation of the most probable explanation of the evidence. The second algorithm (Section III-C) applies iterative calculation of the MPE query for polytrees [37]

to arbitrary networks. Since the original algorithm [37] has exponential complexity with respect to the maximum node degree, we introduce an approximation that allows the calculations to be performed in polynomial time [49].

A. Bucket Elimination Algorithm

Bucket elimination proposed in [12, Algorithm 1] is one of the most popular algorithmic frameworks for computing queries in belief networks. The *bucket elimination algorithm* for computing MPE is exact and always outputs a solution. We consider it the optimal algorithm for computing the explanation of the observed symptoms. The computational complexity of the algorithm is bound by $\mathcal{O}(|V|\exp(w^*(o)))$, where $w^*(o)$ is the width of the graph induced by ordering o [12].

B. Iterative Belief Updating

Recall from Section II that in singly connected networks (polytrees) representing the noisy-OR-gate model of conditional probability distribution, Bayesian inference (belief updating) may be computed in polynomial time using the algorithm presented in [37]. Belief networks used as fault propagation models typically are not polytrees because they contain undirected loops [49].

Networks with loops violate certain independence assumptions based on which the local computation equations were derived for polytrees. Nevertheless, successful applications of the iterative algorithm have been reported. The most famous of them are turbo-codes [2] that offer near Shannon limit-correcting coding and decoding. The turbo-codes decoding algorithm was shown to be an instance of iterative belief propagation in polytrees applied to loopy networks [34]. Previous applications of a deterministic decoding schema to deterministic fault localization [54] inspire the application of probabilistic decoding to fault localization with nondeterministic fault models.

Recall from Section II that the problem of fault localization may be translated into the most probable explanation (MPE) query in belief networks. The iterative algorithms for polytrees proposed in [37] include the algorithm for calculating MPE. Nevertheless, we start presenting iterative algorithms from the description of belief updating, which is conceptually simpler. We also present an adaptation of belief updating to estimating the MPE.

1) Iterative Belief Propagation Concepts: Iterative belief propagation utilizes a message passing schema in which the belief network nodes exchange λ and π messages (Fig. 1). Message $\lambda_X(v_j)$ that node X sends to its parent V_j for every valid V_j's value v_j denotes *a posterior* probability of the entire body of evidence in the subgraph obtained by removing link $V_j \rightarrow X$ that contains X, given that $V_j = v_j$. Message $\pi_{U_i}(x)$ that node X sends to its child U_i for every valid value of X, x denotes a probability that $X = x$ given the entire body of evidence in the subgraph containing X created by removing edge $X \rightarrow U_i$. In this section, we present a summary of the iterative algorithm for polytrees. The complete description of the iterative algorithm for polytrees along with some illustrative examples may be found in [37].

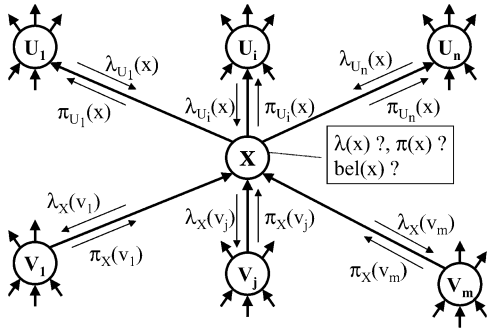

Fig. 1. Message passing in Pearl's belief propagation.

Based on the messages received from its parents and children, node X computes $\lambda(x)$, $\pi(x)$, and $bel(x)$ as follows [37]:

$$\lambda(x) = \prod_{i=1}^{n} \lambda_{U_i}(x) \tag{1}$$

$$\pi(x) = \begin{cases} \alpha \prod_{j=1}^{m}(1 - c_{V_jX}\pi_{jX}), & \text{if } x = 0 \\ \alpha\left(1 - \prod_{j=1}^{m}\left(1 - c_{V_jX}\pi_{jX}\right)\right), & \text{if } x = 1 \end{cases} \tag{2}$$

$$bel(x) = \alpha\lambda(x)\pi(x). \tag{3}$$

In the above equations, $\pi_{jX} = \pi_X(v_j)$ for $v_j = 1$, α is a normalizing constant, and β is any constant. In a noisy-OR polytree, let us denote by q_{XU_i} the probability of activating the inhibitor controlling link $X \rightarrow U_i$. The probability that U_i occurs given X occurs is $c_{XU_i} = 1 - q_{XU_i}$. The messages $\lambda_X(v_j)$ and $\pi_{U_i}(x)$ are computed using the following [37]:

$$\lambda_X(v_j) = \beta\Bigg(\lambda(1) - q_{V_jX}^{v_j}(\lambda(1) - \lambda(0)) \times \prod_{k \neq j}(1 - c_{V_kX}\pi_{kX})\Bigg) \tag{4}$$

$$\pi_{U_i}(x) = \alpha \prod_{k \neq i} \lambda_{U_k}(x)\pi(x). \tag{5}$$

In the initialization phase, for all observed nodes X, $\lambda(x)$ is set to one if x is the observed value of X. For other values of x, $\lambda(x)$ is set to zero. For all unobserved nodes $\lambda(x)$ is set to one for all values of x. Parentless nodes have their $\pi(x)$ set to the prior probabilities. The belief propagation algorithm in polytrees starts from the evidence node and propagates the changed belief along the graph edges by computing $bel(x)$, $\lambda_X(v_i)$, and $\pi_X(u_i)$ in every visited node. In loopy graphs, several iterations are performed in which the entire graph is searched according to some pre-defined ordering.

2) Application of Belief Propagation to Fault Localization: For the purpose of event-driven fault localization this paper adapts the iterative belief updating as follows. The fault localization algorithm starts with a belief network all of which have evidence nodes corresponding to observable symptoms assigned to zero, and all other nodes are unassigned, i.e., their $\lambda(0) = \lambda(1) = 1$. Then, the algorithm proceeds in an event-driven manner, after every symptom observation applying one iteration of belief updating traversing the graph according to some order. For every symptom, we define a different ordering that is equivalent to the breadth-first order started from the node representing the observed symptom. The

initialization phase considers all observable symptoms positive and calculates fault probability distribution in the presence of no negative observations. When a negative symptom is observed, the propagation of evidence reverses the results of the corresponding positive symptom analysis performed in the initialization phase.

Results of negative symptom analysis may also be reversed when the symptom is canceled or a corresponding positive symptom is observed. The set of observable alarms may be easily modified during the fault localization process by redefining λ values of nodes whose observability status has changed.

It may be noticed that in the unobserved symptom nodes, because their $\lambda(x) = 1$ for any value of x, $\lambda_X(v_j) = 1$ regardless of the values of other messages used in the expression. Since symptom nodes have no children, there is no need to compute π messages. Because of these properties, calculations performed in an unobserved symptom node do not change any of the messages published by the node. As a result, an unobserved symptom node acts as a barrier through which belief does not propagate. Therefore, the fault localization algorithm does not visit unobserved symptom nodes, nor does it explore subgraphs accessible only through unobserved symptom nodes.

For a positive symptom node X, and for any parent node of X, V_j, observe that

$$\lambda_X(v_j) = \beta q_{V_jX}^{v_j} \prod_{k \neq j} (1 - c_{V_kX}\pi_{kX}).$$

Since β is any constant, assignment $\beta = \alpha = 1/(\lambda_X(V_j = 1) + \lambda_X(V_j = 0))$ leads to the following:

$$\lambda_X(v_j) = \begin{cases} \frac{1}{(1+q_{V_jX})}, & \text{if } v_j = 0 \\ \frac{q_{V_jX}}{(1+q_{V_jX})}, & \text{if } v_j = 1. \end{cases}$$

Since $\lambda_X(v_j)$ does not depend on the received values of functions $\pi_X(v_k)$, node X does not propagate evidence among its parents. As a result, the iterative belief updating need not continue past a positive symptom node.

The iterative belief propagation described above allows us to obtain the marginal posterior distribution resulting from the observation of the evidence (symptoms). We use this distribution to estimate the set of faults that are the most probable causes of the observed symptoms. For this purpose, we introduce the following heuristic: 1) choose a fault node with the highest posterior probability, 2) place the corresponding fault in the MPE hypothesis, 3) mark the node as observed with value one, and 4) perform one iteration of belief propagation starting from the chosen node. Steps 1)– 4) are repeated until: 1) the posterior distribution contains fault nodes whose probability is greater than 0.5 and 2) unexplained negative symptoms remain.

Observe that an inherent property of Algorithm 2 is the capability to isolate multiple simultaneous faults even if their symptoms overlap. Observe also that the algorithm does not necessarily provide an explanation to all observed symptoms. When none of the available alternative hypotheses is associated with sufficient belief (i.e., 0.5), the risk of giving an incorrect answer by proposing one of the hypotheses is high. We favor not explaining some symptoms over proposing a hypothesis that is likely to contain nonexistent faults. Formally, the fault localization algorithm is defined as follows.

```
Algorithm 2: Fault localization using
iterative belief updating
Inference iteration starting from node Y_i:
 let o be the breadth-first order starting
 from Y_i
 for all nodes X along ordering o do
  if X is not an unobserved or positive
 symptom node then
   compute λ_X(v_j) for all X's parents , V_j,
    and for all v_j ∈ {0,1}
   compute π_{U_i}(x) for all X's children, U_i,
    and for all x ∈ {0,1}
Initialization phase:
 for every symptom S_i ∈ S_O do
  mark S_i as observed to have value of zero
  run inference iteration starting from S_i
Symptom analysis phase:
 for every symptom S_i ∈ S_N do
  mark S_i as observed to have value of one
  run inference iteration starting from S_i
 compute bel(v_i) for every node V_i, v_i ∈ {0,1}
Fault selection phase:
 while ∃ fault node V_j for which bel(1) > 0.5 and
 S_N ≠ ∅ do
  take V_j with the greatest bel(1) and
   place it in the set of detected faults F_D
  mark V_j as observed to have value of one
  remove all symptoms explained by V_j from S_N
  run inference iteration starting from V_j
  compute bel(v_i) for every node V_i, v_i ∈ {0,1}.
```

Note that, contrary to other approaches to fault localization [28], [29] which delay symptom analysis until all symptoms are collected, Algorithm 2 does not require all symptoms to be observed before their analysis may be started. On the contrary, it analyzes a symptom independently of other symptom observations. The knowledge resulting from analyzing a symptom is stored for the next iterations in the belief network nodes in the form of λ and π messages, allowing Algorithm 2 to utilize time more efficiently. Moreover, for every fault the algorithm continuously provides the probability of its occurrence given the symptoms observed so far.

Local computations in nodes require $\mathcal{O}(d)$ operations, where d is the maximum node degree. A single iteration visits every node at most once; therefore, its computational complexity is $\mathcal{O}(|V|(d)) = \mathcal{O}(|E|)$. During the entire computation, at most $|\mathcal{S}_O| + |\mathcal{F}|$ iterations are performed, where usually $|\mathcal{F}| \ll |\mathcal{S}_O|$. Thus, the complexity of the entire algorithm is $\mathcal{O}(|\mathcal{S}_O||E|)$.

C. Iterative Most Probable Explanation

In this section, we present an application of the iterative MPE algorithm for polytrees [37] to networks with undirected loops. The MPE algorithm in every iteration produces the most probable value assignment to the belief network nodes. Thus, in every iteration the algorithm produces a complete explanation of the observed symptoms and no selection phase is necessary.

1) Iterative MPE Concepts: Similar to belief updating, the MPE computation algorithm proceeds from the evidence nodes by passing messages λ^* and π^* along the belief network edges. Message $\lambda_X^*(v_j)$ sent by node X to its parent V_j represents the

conditional probability of the most probable prognosis for the values of nodes located in the subgraph containing X resulting from the removal of the link $V_i \rightarrow X$, given the proposition $V_i = v_j$. Message $\pi^*_{U_i}(x)$ sent by node X to its child U_i represents the probability of the most probable values of nodes in the subgraph resulting from the removal of link $X \rightarrow U_i$, which include the proposition $X = x$. Belief metric $bel^*(x)$ stands for the probability of the most probable explanation of evidence e that is consistent with the proposition $X = x$. Messages $\lambda^*_X(v_j)$ and $\pi^*_{U_i}(x)$ and belief metric $bel^*(x)$ are computed using the following equations [37]:

$$\lambda^*_X(v_j) = \max_{x,\{v_k \neq v_j\}} \left(\prod_i \lambda^*_{U_i}(x) P(x|v_1,\ldots,v_m) \times \prod_{k\neq j} \pi^*_X(v_k) \right) \quad (6)$$

$$\pi^*_{U_i}(x) = \prod_{k\neq i} \lambda^*_{U_k}(x) \times \max_{\{v_k\}} \left(P(x|v_1,\ldots,v_m) \prod_k \pi^*_X(v_k) \right) \quad (7)$$

$$bel^*(x) = \beta \prod_k \lambda^*_{U_k}(x) \times \max_{\{v_k\}} \left(P(x|v_1,\ldots,v_m) \prod_k \pi^*_X(v_k) \right). \quad (8)$$

Using notation from Section III-B, in noisy-OR belief networks, the maximization may be expressed as in (9), shown at the bottom of the page.

$$\max_{\{v_k\}} \left(P(x|v_1,\ldots,v_m) \prod_k \pi^*_X(v_k) \right) = \begin{cases} \max_{\{v_k\}} \left(\left(\prod_{V_k|v_k=1} q_{V_kX}\right) \prod_k \pi^*_X(v_k) \right), & \text{if } x=0 \\ \max_{\{v_k\}} \left(\left(1 - \prod_{V_k|v_k=1} q_{V_kX}\right) \prod_k \pi^*_X(v_k) \right), & \text{if } x=1 \end{cases} \quad (9)$$

2) Application of Iterative MPE to Fault Localization in Belief Networks With Loops: The calculation of λ^*, π^*, and bel^* is the primary difficulty in using iterative MPE in practical applications. While for $x = 0$ expression $\max_{\{v_k\}} (P(x|v_1,\ldots,v_m) \cdot \prod_k \pi^*_X(v_k))$ may be simplified to $\prod_k \max(\pi^*_X(v_k = 0),\ q_{V_kX}\pi^*_X(v_k = 1))$, the exact computation of the maximization for $x = 1$ requires enumerating all possible combinations of value assignments to the parents of X and choosing a combination that maximizes the value of the expression. Clearly, listing all combinations is computationally infeasible. In this paper, we use an approximation that allows us to compute the maximization expression in polynomial time.

The algorithm for computing MPE calculates λ^* and π^* for every network node traversing the graph starting from the observed symptom in a breadth-first order. A single traversal is repeated for every observed symptom. At the end, bel^* values are computed for all network nodes. The MPE contains all fault nodes with $bel^*(x = 1) > bel^*(x = 0)$.

```
Algorithm 3: Fault localization through
iterative MPE
Inference iteration starting from node Y_i:
 let o be the breadth-first ordering starting
 from Y_i
 for all nodes X along ordering o do
  compute λ*_X(v_j) for all X's parents, V_j,
   and for all v_j ∈ {0,1}
  compute π*_{U_i}(x) for all X's children, U_i,
   and for all x ∈ {0,1}
Initialization phase:
 for every symptom S_i ∈ S_O do
  mark S_i as observed to have value of zero
  run inference iteration starting from S_i
Symptom analysis phase:
 for every symptom S_i ∈ S_N do
  mark S_i as observed to have value of one
  run inference iteration starting from S_i
 compute bel*(v_i) for every node V_i, v_i ∈ {0,1}
Fault selection phase:
 choose all fault nodes with bel*(X = 1) >
 bel*(X = 0)
  and place them in F_D.
```

Local computations in nodes require $\mathcal{O}(d^2)$ operations, where d is the maximum node degree. There are $|\mathcal{S}_O|$ iterations visiting every belief network node at most once. This leads to the computational complexity of the entire algorithm of $\mathcal{O}(|\mathcal{S}_O||V|d^2)$.

IV. Dealing With Noisy Observations

In real-life communication systems, an observation of network state is frequently disturbed by the presence of lost and/or spurious symptoms (usually referred to as observation noise).

In a management system, alarms may be lost as a result of using an unreliable communication mechanism to transfer alarms from their origin to the management node. For example, since the SNMP protocol [6] exploits an unreliable transport layer protocol (UDP), SNMP traps [6] issued by an SNMP agent are not guaranteed to be delivered to the destination. Too liberal threshold values may also prevent an existing problem from being reported, thereby causing alarm loss. When the fault localization algorithm relies on positive information to create the most likely fault hypothesis, alarm loss, if ignored by the algorithm, could lead to an incorrect solution.

Another frequent disturbance in an observation of network state is due to spurious alarms, which are caused by intermittent network faults or by overly restrictive threshold values. The method proposed in this section assumes that spurious symptoms occur independently of one another. This assumption is justified since interdependencies among spurious symptoms, even if they exist, usually are very difficult to learn.

We address the problem of lost and spurious alarms by augmenting the belief network model presented in Section II using the technique we introduced in [48]. Let $V_S = \{V_{S_1},\ldots,V_{S_m}\} \subset V$, where $m = |\mathcal{S}|$, be the set of all belief network nodes which correspond to symptoms $\{S_1,\ldots,S_m\} = \mathcal{S}$. We introduce a set of belief network nodes $\{V'_{S_1},\ldots,V'_{S_m}\}$ which represent unobservable failures. Then, for every node $V_{S_i} \in V_S$ and for every $V_j \in V - V_S$ such that $(V_j, V_{S_i}) \in E$ we: 1) remove (V_j, V_{S_i}) from E and 2) add (V_j, V'_{S_i}) to E. Then, we add directed edges $V'_{S_i} \rightarrow V_{S_i}$,

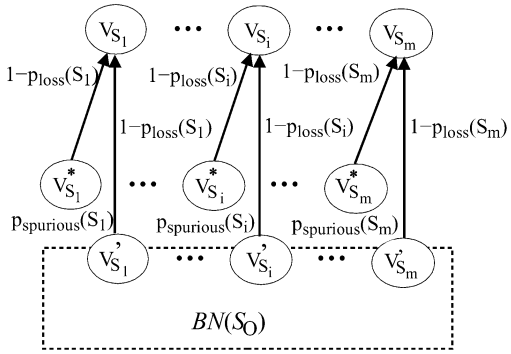

Fig. 2. Belief network modeling lost and spurious symptoms ($\mathcal{BN}(\mathcal{S}_O, p_{\text{loss}}, p_{\text{spurious}})$).

$i = 1 \ldots m$. With every directed edge $V'_{S_i} \rightarrow V_{S_i}$ we associate the probability of causal relationship between V'_{S_i} and V_{S_i} equal to $1 - p_{\text{loss}}(S_i)$, where $p_{\text{loss}}(S_i)$ is the probability that alarm S_i is lost. The values of $p_{\text{loss}}(S_i)$ may be obtained, for example, by analyzing packet loss rate in the network used to transport symptom S_i.

To model spurious symptoms, we introduce nodes $V^*_{S_i}$, $i = 1, \ldots, m$. Then, we add directed edges $V^*_{S_i} \rightarrow V_{S_i}$. With every $V^*_{S_i}$ we associate prior belief $p_{\text{spurious}}(S_i)$ that represents the cumulative probability of events (other than persistent faults) that trigger alarm S_i. The value of $p_{\text{spurious}}(S_i)$ may be learned by analyzing historical alarm log files. Every directed edge $V^*_{S_i} \rightarrow V_{S_i}$ is labeled with $1 - p_{\text{loss}}(S_i)$.

The resultant belief network, $\mathcal{BN}(\mathrm{S}_O, p_{\text{loss}}, p_{\text{spurious}})$ is presented in Fig. 2. When $p_{\text{loss}}(S_i) = 0$, edges $V'_{S_i} \rightarrow V_{S_i}$ (for $i = 1, \ldots, N_s$) are redundant, and nodes V'_{S_i} and V_{S_i} may be considered identical ($V'_{S_i} \equiv V_{S_i}$). Also, when $p_{\text{spurious}}(S_j) = 0$, nodes $V^*_{S_j}$ are redundant and may be reduced ($V^*_{S_j} \equiv V_{S_j}$). Thus, $\mathcal{BN}(\mathcal{S}_O, 0, 0)$ is equivalent to $\mathcal{BN}(\mathcal{S}_O)$. Belief network $\mathcal{BN}(\mathrm{S}_N, 0, 0)$ is equivalent to $\mathcal{BN}(\mathcal{S}_N)$ which is the fault propagation model that does not take either positive, lost, or spurious symptoms into account. When the existence of either lost or spurious (but not both) alarms may be neglected, one should use $\mathcal{BN}(\mathcal{S}_O, 0, p_{\text{spurious}})$ or $\mathcal{BN}(\mathcal{S}_O, p_{\text{loss}}, 0)$, respectively.

To perform fault localization that includes lost and spurious symptoms in the analysis using $\mathcal{BN}(\mathcal{S}_O, p_{\text{loss}}, p_{\text{spurious}})$ as a fault propagation model, algorithms proposed in Section III may be applied with no modification.

V. End-to-End Service Failure Diagnosis—A Simulation Study

In this section, we describe the application of fault localization techniques proposed in Sections III-C and IV to the diagnosis of end-to-end service failures. Network connectivity in a given protocol layer is frequently achieved through a sequence of intermediate nodes invisible to the layers above. A failure of an intermediate node may cause availability or performance problems on one or more end-to-end paths established using the malfunctioning node. End-to-end service failure diagnosis deals with isolating causes of failures associated with end-to-end paths. Diagnosing end-to-end service failures, both availability- and performance=related ones, is a crucial step toward multilayer fault localization. In this paper, end-to-end service failure diagnosis is used as a case study through which the performance and accuracy of algorithms introduced in Section III are evaluated.

A. End-to-End Service Model

When connectivity between nodes **a** and **b** in network layer L is achieved through a sequence of intermediate nodes, we say that the end-to-end service offered by layer L between hosts **a** and **b** is implemented in terms of multiple host-to-host services offered by layer L between subsequent hops on the path of the layer L packet from node **a** to node **b**. A failure of a host-to-host service such as a broken link, a buffer overflow, or a transmission link noise may cause a failure of a dependent end-to-end service such as a loss of connectivity, excessive delay, or excessive packet loss rate. How a specific failure of a host-to-host service affects a dependent end-to-end service depends on a communication protocol used in the given layer.

A fault propagation model for end-to-end service failure diagnosis is a bipartite belief network in which parentless nodes (called *link* nodes) represent host-to-host service failures and childless nodes (called *path* nodes) represent end-to-end service failures. To build this model, a knowledge of the logical network topology is required. When the logical topology is isomorphic to the physical one, the relationships between end-to-end and host-to-host services may be obtained by analyzing the physical network connectivity. An automatic means of discovering this information is built in many commercially available network management systems. Also, IETF proposes a standardized means of representing the physical network topology, the physical topology MIB, [3], which may be used to obtain connectivity information, if it is implemented in the managed domain. When the logical topology is not isomorphic to the physical one, other techniques have to be used to obtain the fault propagation model, which differ depending on the system layer and routing protocol. Very useful information is available through network management protocols such as SNMP [5]. For example, the logical topology established in the data-link layer by the spanning tree protocol [38] may be collected using the `dot1dBase` Group of bridge MIB [14]. Run-time updates of the spanning tree may be triggered by newRoot and topology change traps [14]. In the network layer of the Internet, current routes may be calculated from ipRoutingTable of *TCP/IP MIB-II* [33]. Besides, a plethora of other, more advanced techniques has been proposed by the research community for different types of networks, e.g., [4], [18], [32], [35], and [43].

When the fault management system does not distinguish among multiple failure types, each system component may be in one of two possible states: malfunctioning or working properly. In this case, the fault propagation model is completely determined by the logical network topology and consists of one path or link node for every end-to-end path or host-to-host link, respectively. An example of such fault propagation models is presented in Fig. 3, which shows a bipartite belief network for end-to-end service failure diagnosis in the data-link layer of a simple network composed of four learning bridges [38].

When the fault management system distinguishes among multiple failure types, each path or link may experience multiple types of problems (e.g., delay, excessive packet loss, erroneous transmission, total loss of connectivity, etc.). This situation is represented in the fault propagation model by creating multiple path or link nodes for each end-to-end path

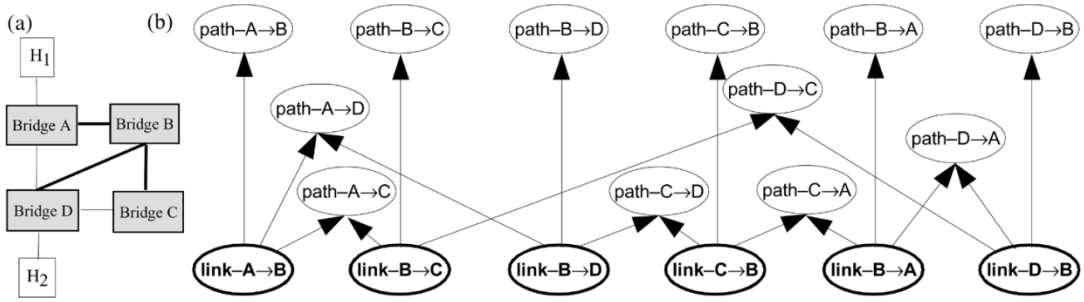

Fig. 3. (a) Example bridge topology with the current spanning tree marked in bold. (b) Belief network built based on the spanning tree in (a).

or host-to-host link, each corresponding to a different failure type. A complete fault propagation model contains multiple, possibly overlapping, graphs as the one presented in Fig. 3, each focusing on a different failure type. The connectivity between the graphs is determined by the communication protocol used in a given layer. For example, if the protocol implements an error-detection mechanism, a belief network node representing erroneous transmission in one of the host-to-host links is connected to belief network nodes representing excessive data loss in dependent end-to-end paths. Otherwise, it is connected to belief network nodes representing erroneous transmission in the dependent end-to-end paths.

End-to-end service diagnosis aims at correlating symptoms related to path failures in order to isolate one or more responsible link failures. Thus, an observed path failure is mapped into the corresponding belief network node. Then, the node is assigned to 1 and an event-driven iteration of an algorithm (either Algorithm 2 or Algorithm 3) is executed starting from this node. After symptoms are processed in this manner, the algorithm produces the most probable value assignment to link nodes. While using Algorithm 1, the algorithm (which is not event driven) is executed once, when all symptoms have been observed and their corresponding nodes appropriately assigned. At the end, failure conditions corresponding to link nodes that are assigned to one in the most probable assignment are reported as the root causes of the observed disorder.

B. Application of Belief Networks

The application of algorithms introduced in Section III for diagnosing end-to-end services using a bipartite model is straightforward. In this section, we analyze the algorithms' complexity in this fault localization problem.

1) *Bucket elimination*

Recall from Section III-A that the computational complexity of bucket elimination in arbitrary belief networks is $\mathcal{O}(|V|\exp(w^*(o)))$, where $w^*(o)$ is the width of the graph induced by ordering o, defined in Section II. One can show that the minimum width of the bipartite graph for end-to-end service failure diagnosis is bound by the maximum path length in the original network graph [i.e., spanning tree in Fig. 3(a)]. Therefore, the complexity of bucket elimination in application to end-to-end service diagnosis is $\mathcal{O}(n^2\exp(n))$.

2) *Iterative belief propagation*

In Section III-B, we determined the complexity of fault localization iterative belief propagation in arbitrary belief networks to be $\mathcal{O}(|\mathcal{S}_O||E|)$. In a bipartite belief network representing end-to-end service model of an n-node network, there are at most n^2 paths (i.e., possible symptoms) and every path may be composed of at most n links. Thus, the graph contains at most n^3 edges leading to the complexity of the entire algorithm of $\mathcal{O}(|\mathcal{S}_O|n^3) = \mathcal{O}(n^5)$.

3) *Iterative MPE*

In Section III-C, we wrote that local computations in nodes require $\mathcal{O}(d^2)$ operations resulting in the complexity of the entire algorithm of $\mathcal{O}(|\mathcal{S}_O||V|d^2)$. Indeed, the calculation of λ messages, which is performed only by path nodes, requires $\mathcal{O}(d^2) = \mathcal{O}(n^2)$ steps. However, messages π, which are calculated only by fault nodes, may be obtained in $\mathcal{O}(d) = \mathcal{O}(n^2)$ steps. Thus, the calculation of λ or π messages in a single iteration requires $\mathcal{O}(n^4)$ or $\mathcal{O}(n^3)$ steps, respectively. As a result, the complexity of the entire algorithm in the application to end-to-end service diagnosis is $\mathcal{O}(|\mathcal{S}_O|n^4) = \mathcal{O}(n^6)$.

C. Simulation Model

The simulation study presented in this paper uses tree-shaped network topologies, which result, for example, from the usage of the spanning tree protocol [38]. The usage of tree-shaped topologies greatly simplifies their random generation, while not having any significant impact on the accuracy of the results presented in this section.

We evaluated the algorithms presented in Section III using the following simulation model. Let OR represent the alarm observability ratio, i.e., $\mathrm{OR} = |\mathcal{S}_O|/|\mathcal{S}|$. Let LR represent alarm loss rate, i.e., the ratio of the number of generated alarms that were lost to the number of all generated alarms. Let SSR represent the spurious alarm rate, i.e., the probability that an alarm is generated spontaneously. The three ratios OR, LR, and SSR are parameters of the simulation model.

Given the simulation model with parameters OR, LR, and SSR, for a given network topology size n, where n represents the number of intermediate network nodes, such as bridges or switches, we design K_n simulation cases as follows.

- We create a random tree-shaped n-node network $\mathcal{N}_i (1 \leq i \leq K_n)$. We denote the set of faults, the set of unobservable end-to-end failures, and the set of symptoms for network $\mathcal{N}_i$ as $\mathcal{F}_i$, $\mathcal{E}_i$, and $\mathcal{S}_i$, respectively. For every link in $\mathcal{N}_i$, we create one $f_j \in \mathcal{F}_i$. Note that in an n-node tree-shaped network there are $n-1$ links, i.e., $|\mathcal{F}_i| = n-1$. For every end-to-end path in $\mathcal{N}_i$, we create one $e_l \in \mathcal{E}_i$ and one $s_l \in \mathcal{S}_i$. (Recall that $|\mathcal{E}_i| = |\mathcal{S}_i| \in \mathcal{O}(n^2)$.)
- We randomly generate prior fault probability distribution $p_f: \mathcal{F}_i \rightarrow [0.001, 0.01]$; $p_f(f_j)$s are uniformly distributed over the range [0.001, 0.01]. We randomly generate conditional probability distribution $p_{fe}: (\mathcal{F}_i \times \mathcal{E}_i) \rightarrow [0, 1)$, defined as the probability that e_l occurs given f_j occurs. For

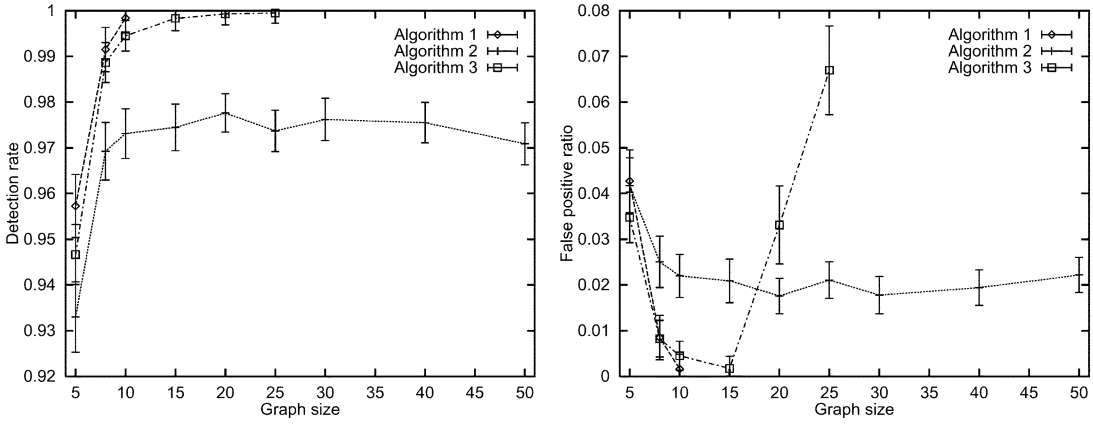

Fig. 4. Comparison of accuracies achievable with Algorithms 1–3 for different network sizes.

all $f_j \in \mathcal{F}_i$ and $e_l \in \mathcal{E}_i$ such that the path corresponding to e_l includes the link corresponding to f_j, $p_{fe}(f_j, e_l)$s are uniformly distributed over the range $(0, 1)$; otherwise, $p_{fe}(f_j, e_l) = 0$.

- We randomly generate the set of observable alarms $\mathcal{S}_{iO} \subseteq \mathcal{S}_i$ such that on average $(|\mathcal{S}_{iO}|/|\mathcal{S}_i|) = \mathrm{OR}$.
- Given network $\mathcal{N}_i$, we build its belief network model $\mathcal{BN}_i(\mathcal{S}_{iO}, p_{\mathrm{loss}}, p_{\mathrm{spurious}})$. We choose $p_{\mathrm{loss}} = LR(p_{\mathrm{spurious}} = \mathrm{SSR})$ to perform fault localization that includes alarm loss (spurious alarms) in the analysis. We use $p_{\mathrm{loss}} = 0 (p_{\mathrm{spurious}} = 0)$ to perform fault localization that disregards lost (spurious) symptoms.

For the ith simulation case $(1 \leq i \leq K_n)$, we create M_S simulation scenarios as follows.

1) Using p_f, we randomly generate the set of faulty links in network $\mathcal{N}_i$, $\mathcal{F}_{iC}^k (1 \leq k \leq M_S)$ and create probability distribution p_e^k: $\mathcal{E}_i \rightarrow [0, 1]$, where $p_e^k(e_j) = P\{e_j \text{ occurs}|\text{all faults in } \mathcal{F}_{iC}^k \text{ occur}\}$.
2) Using p_e^k, we randomly generate the set of events $\mathcal{E}_{iC}^k$ resulting from faults in $\mathcal{F}_{iC}^k$ and build $\mathcal{S}_{iC}^k \subseteq \mathcal{S}_i$, the set of alarms corresponding to events in $\mathcal{E}_{iC}^k$. We set $\mathcal{S}_{iG}^k = \mathcal{S}_{iC}^k \cap \mathcal{S}_{iO}$.
3) Using SSR, we randomly generate the set of spurious alarms $\mathcal{S}_{iS}^k \subseteq \mathcal{S}_{iO}$. We set $\mathcal{S}_{iG+S}^k = \mathcal{S}_{iG}^k \cup \mathcal{S}_{iS}$.
4) We create $\mathcal{S}_{iG-L}^k$, the set of generated alarms that are not lost by the communication system, by randomly removing alarms from $\mathcal{S}_{iG+S}^k$ so that on average $(|\mathcal{S}_{iG-L}^k|/|\mathcal{S}_{iG+S}^k|) = 1 - LR$.
5) We set $\mathcal{S}_{iN}^k = \mathcal{S}_{iG-L}^k$; $\mathcal{S}_{iN}^k$ constitutes the set of negative symptoms observed in the simulation scenario k.
6) Using one of the fault localization algorithms, we compute $\mathcal{F}_{iD}^k \subseteq \mathcal{F}_i$, the most likely explanation of symptoms in $\mathcal{S}_{iN}^k$. We calculate *detection rate* $\left(\mathrm{DR}_i^k\right)$ and *false positive rate* $\left(\mathrm{FPR}_i^k\right)$ defined using

$$\mathrm{DR}_i^k = \frac{\left|\mathcal{F}_{iD}^k \cap \mathcal{F}_{iC}^k\right|}{\left|\mathcal{F}_{iC}^k\right|} \quad \mathrm{FPR}_i^k = \frac{\left|\mathcal{F}_{iD}^k \setminus \mathcal{F}_{iC}^k\right|}{\left|\mathcal{F}_{iD}^k\right|}.$$

For the ith simulation case, we calculate the mean detection rate $\mathrm{DR}_i = (1/M_S)\sum_{k=1}^{M_S} \mathrm{DR}_i^k$ and mean false positive rate $\mathrm{FPR}_i = (1/M_S)\sum_{k=1}^{M_S} \mathrm{FPR}_i^k$. Then, we calculate the expected values of detection rate and false positive rate denoted by $\mathrm{DR}(n)$ and $\mathrm{FPR}(n)$, respectively. In our study, we used $K_n = 100$ and $M_S = 100$ or 200, depending on the variability of DR_i^k. We varied n from 5 to 50.

In Sections V-D and V-E, we apply this simulation model to evaluate the performance and accuracy of fault localization algorithms described in Section III. We used JavaBayes [1] implementation of bucket elimination (Algorithm 1). We implemented Algorithms 2 and 3 in Java.

D. Comparison of Algorithms

The first simulation study was conducted to compare the performance and accuracy of the fault localization algorithms described in Section III. We intentionally ignore positive, lost, and spurious symptoms. Formally, we set $\mathrm{OR} = |\mathcal{S}_N|/|\mathcal{S}|$, $\mathrm{LR} = 0$, and $\mathrm{SSR} = 0$ and use belief network $\mathcal{BN}(\mathcal{S}_N)$ as a fault propagation model. The link failure probabilities are uniformly distributed random values of the order of 10^{-6}, and the conditional probabilities on causal links are uniformly distributed random values in the range $[0.5, 1)$. Because of excessive simulation time, we had to limit the tested network size range for Algorithms 1 and 3 to 10 and 25, respectively.

Fig. 4(a) presents the relationship between detection rate and network size. The detection rate is shown within statistically computed confidence intervals. We observe that Algorithms 1 and 3 outperform Algorithm 2 by 1%–3%. The shape of the graphs in Fig. 4(a) indicates a strong dependency of the detection rate on the network size. This relationship may be explained as follows. For small (five-node) networks, the number of symptoms observed is typically small (less than ten), which in some cases is not sufficient to precisely pinpoint the actual fault. Since in small networks the size of $\mathcal{F}_c$ is also small, any mistake in fault detection significantly reduces the detection rate. The fault-localization accuracy in networks of small size can be improved by increasing the system instrumentation degree (i.e., increasing the observability ratio OR) or by using positive symptoms (i.e., the lack of negative observations) in addition to the negative ones. Our previous work on this subject has shown that the accuracy fault localization in poorly instrumented networks can be substantially improved by including positive symptoms in the reasoning process [48].

When the network gets larger, the number of observed symptoms increases, thereby increasing the ability to precisely detect the faults. On the other hand, as the network size grows, the

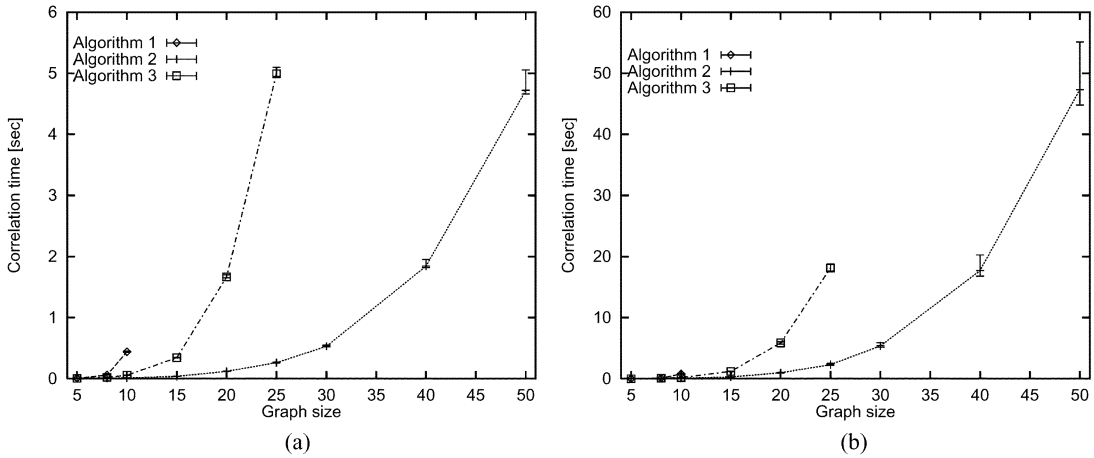

Fig. 5. Comparison of fault localization time for Algorithms 1– 3 versus network size.

TABLE I
COMPARISON OF ALGORITHMS 1–3

Algorithm	Bucket Elimination (Alg. 1)	Iterative Belief Updating (Alg. 2)	Iterative MPE (Alg. 3)
Theoretical bound	$n^2exp(n)$	n^5	n^6
Detection rate	96-100%	93-98%	75-100%
False positive rate	0-4%	2-5%	0-20%
Max. network size with localization time <10s	10	50	25
Multi-fault scenarios	yes	yes	yes
Lost and spurious symptoms	yes	yes	yes
Is algorithm event-driven?	no	yes	yes
Prediction capabilities	yes/no answer	confidence assessment	yes/no answer

multifault scenarios become more and more frequent. In a multifault experiment, it is rather difficult to detect all actual faults, which leads to partially correct solutions and, in consequence, the lower accuracy of Algorithm 2.

As depicted in Fig. 4(b), the false positive rate of Algorithm 1 is 1%–2% lower than that of Algorithm 2. For small networks, Algorithm 3 has false positive rate comparable to Algorithm 1. However, as networks get bigger, the false positive rate of Algorithm 3 grows sharply, suggesting that Algorithm 3 has a tendency to propose too big a set of faults as a final hypothesis than is actually needed to explain all symptoms. A number of additional tests performed on Algorithm 3 for bigger network sizes [which are not shown in Fig. 4(b)] revealed that its false positive rate further increases as the network size grows, but it saturates with the value of approximately 20% for networks composed of 50 nodes. These experiments also revealed that, when the network size increases, the detection rate of Algorithm 3 decreases faster than that of Algorithm 2. As a result, we conclude that Algorithm 2 is more reliable in offering high accuracy than Algorithm 3.

Fig. 5(a) and (b) presents the dependency of the execution time on the network size in the presence of one and four network faults, respectively. Although in the tested network size range, Algorithm 1 exhibited the best accuracy, the difference between its accuracy and that of other algorithms is too small to justify the substantially worsened performance. Algorithm 2 proved to be the most efficient one while preserving very good accuracy. The execution of the order of several seconds, even for large networks and multifault scenarios, is encouraging.

Table I summarizes the results of the described simulation study. To make the comparison of the algorithms complete, we also evaluate them according to the criteria introduced in Section I. Clearly, all three algorithms are able to detect multiple simultaneous faults and are robust against the information noise, given they are provided with an appropriate fault propagation model. Algorithms 2 and 3 are event driven; Algorithm 1 is not, because running the inference after every symptom observation would repeat calculations performed in the previous iterations. In addition to fault localization, Algorithms 1 and 3 make a binary prediction of network service failures; Algorithm 2 offers an estimation of confidence that a service is affected.

The results presented in this section show that Algorithm 2 is the most promising one in the application to fault localization. In [48], we show that its accuracy may be further improved by including positive symptoms in the analysis and that it is insensitive to the inaccuracies of the probabilistic model, offering almost the same accuracy when a small number of confidence levels (e.g., 3), rather than their real values, is used to specify conditional probabilities. This implies that instead of exact probability values, meaningful probability assignments can be used, e.g., *unlikely*, *likely*, and *very likely*, with almost no effect on the effectiveness of the algorithm. This result confirms the practicality of Algorithm 2 in real-life applications, because it shows that the algorithm does not require a precise knowledge of probability distributions, which are usually difficult to characterize. In Section V-E, we continue evaluating Algorithm 2 looking into its robustness against observation noise.

E. Impact of Lost and Spurious Symptoms

To isolate the impact of symptom loss on the accuracy of the fault localization process, we set $\mathrm{SSR} = 0$ in the simulation model. Loss rate is either 0.05 or 0.1. We compare the accuracy

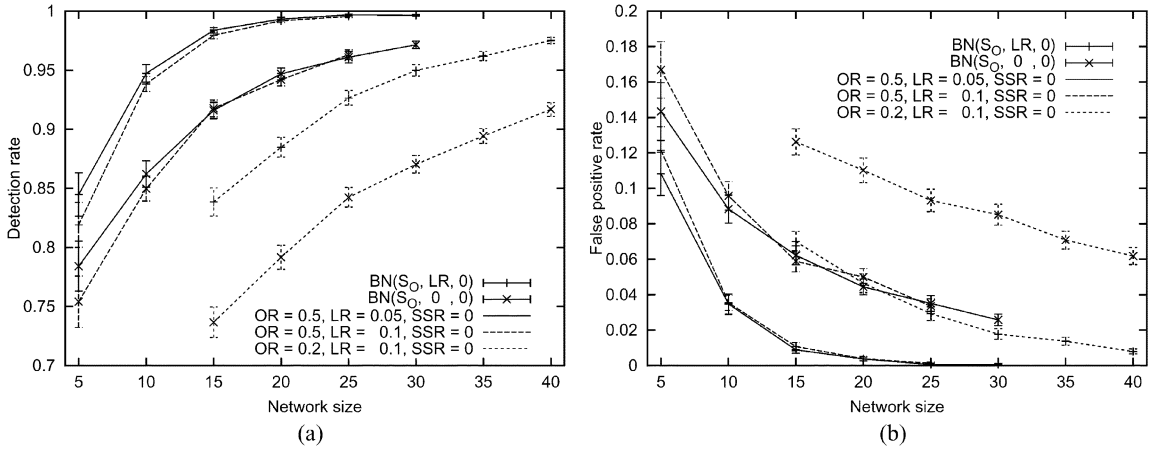

Fig. 6. Comparison of accuracies obtained with Algorithm 2 using fault propagation models $\mathcal{BN}(\mathcal{S}_O, LR, 0)$ and $\mathcal{BN}(\mathcal{S}_O, 0, 0)$.

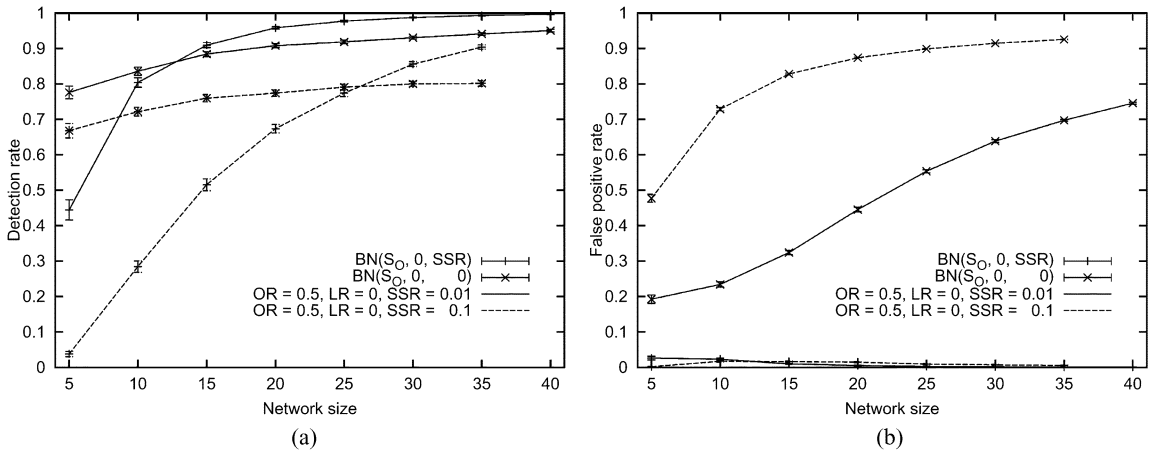

Fig. 7. Comparison of accuracies obtained with Algorithm 2 using fault propagation models $\mathcal{BN}(\mathcal{S}_O, 0, SSR)$ and $\mathcal{BN}(\mathcal{S}_O, 0, 0)$.

of Algorithm 2 using belief networks $\mathcal{BN}(\mathcal{S}_O, LR, 0)$ (taking symptom loss into account) and $\mathcal{BN}(\mathcal{S}_O, 0, 0)$ (disregarding the possibility of symptom loss), varying OR between 0.2 and 0.5. Fig. 6(a) and (b) shows that by including loss rate in the analysis, the detection (false positive) rate may be increased (decreased) by up to 10%. Moreover, given constant OR, this gain is insensitive to the value of LR.

The impact of including spurious symptoms in the fault localization process is evaluated by applying Algorithm 2 to fault propagation models $\mathcal{BN}(\mathcal{S}_O, 0, SSR)$ and $\mathcal{BN}(\mathcal{S}_O, 0, 0)$ using $\mathrm{LR} = 0$ and $\mathrm{OR} = 0.5$. We vary SSR between 0.01 and 0.1. As shown in Fig. 7(a), the inclusion of spurious symptoms in the fault localization process in small networks decreases detection rate. This is explained by the fact that in small networks (in particular, with small observability ratios), only a few symptoms are available to the fault localization process. When the possibility of spurious symptoms is taken into account, and the number of symptoms is very small, the algorithm concludes that there is no sufficient evidential support for the existence of faults and considers most of these symptoms spurious. When $\mathrm{SSR} = 0.1$, for small networks, the probability that all observed symptoms are spurious is frequently higher than the probability of fault occurrence. Therefore, the algorithm refuses to identify a fault thereby achieving very low detection rate. One can conclude that small networks need to be better instrumented (i.e., have higher OR) to allow fault localization to benefit from the analysis of spurious symptoms. In larger networks, the inclusion of spurious symptoms does not cause a decrease in the detection rate; in fact, as shown in Fig. 7(a), it allows the detection rate to be improved. Fig. 7(b) presents the impact of including spurious symptoms on the false positive rate. It shows that regardless of the network size, by taking spurious symptoms into account, the false positive rate of the fault localization process may be significantly decreased.

VI. Other Canonical Models

Thus far in this paper, we have focused on fault propagation models represented by noisy-OR-gate belief networks, which allow fault localization to be performed in polynomial time using Algorithm 2. In noisy-OR-gate networks, a variable value is obtained by combining its predecessors' values using logical OR. This is, clearly, the most useful and wide-spread representation in the area of fault management [28], [29].

Unfortunately, for some fault management problems, the noisy-OR-gate model is inadequate. In this section, we explore the application of other canonical belief network models to solving these problems.

A. *AND Model*

Let us consider a popular high-availability scenario in which two alternative physical network connections are provided be-

tween two neighboring hosts. To model this situation using a belief network, we create node X to represent connectivity failure between the two hosts and nodes Y_1 and Y_2 to represent failures of the two physical connections, respectively, where X is caused by Y_1 and Y_2. When one of the physical connections fails, i.e., Y_1 or Y_2 occurs, the entire traffic between the two hosts is transferred to the second, still operating connection. Thus, the connectivity failure between the hosts may be observed only if both physical connections fail. Clearly, Y_1 and Y_2 do not independently contribute to X, and therefore this scenario may not be represented using the noisy-OR model.

A possible solution to the problem is to detect hard failures of physical connections using configuration-change monitoring tools available in many management systems and, accordingly, modify the model. However, in many situations, a preferable approach would rather embed the information about the high-availability configuration into the fault propagation model so that the monitoring of the configuration changes may be avoided. The relationship between X and Y_1 and Y_2 should be modeled by combining X's predecessors' values using logical AND. To perform belief updating using Algorithm 2, the (2) and (4) in Section III-B used to calculate λ and π in an AND-node X should be modified as follows:

$$\pi(x) = \begin{cases} \alpha \prod_{j=1}^{m} c_{V_jX}\pi_{jX}, & \text{if } x = 1 \\ \alpha\left(1 - \prod_{j=1}^{m} c_{V_jX}\pi_{jX}\right), & \text{if } x = 0 \end{cases} \quad (10)$$

$$\lambda_X(v_j) = \beta\left(\lambda(0) + v_j c_{V_jX}^{v_j}(\lambda(1) - \lambda(0)) \times \prod_{k \neq j} c_{V_kX}\pi_{kX}\right). \quad (11)$$

B. NOT Model

In the NOT model, a variable value is calculated as a logical negation of its single predecessor's value. Intuitively, for a variable to be true, its predecessor must not be activated, or if its predecessor is activated, the inhibitor on the corresponding causal link is activated as well. The replacement for (10) and (11) in the NOT model is obtained in a straightforward manner by setting $m = 1$ and exchanging $\pi(x = 1)$ with $\pi(x = 0)$ and $\lambda_X(v_j = 1)$ with $\lambda_X(v_j = 0)$, respectively.

C. Hybrid Model

In real-life scenarios, a hybrid model is useful, in which a node may apply different logical operators to different subsets of its predecessors. In the extreme case, a hybrid model associated with variable V_i could involve an arbitrary logical expression over V_i's predecessors. In this case, calculating π and λ would correspond to performing inference in a belief network identical with the syntax-decomposition tree of the logical expression, which is rooted at V_i. In practice, much simpler hybrid models are needed, in which π and λ may be calculated using compact and easy-to-obtain formulas.

VII. Related Work

In the past, various event correlation techniques were proposed including rule-based systems [31], [53], model-based reasoning systems [24], [36], model traversing techniques [25], [27], case-based systems [30], graph-theoretic approaches [19], [28], and the code-book approach [54]. Most of these approaches are deterministic, while this paper focuses on nondeterministic fault diagnosis techniques.

So far, the most comprehensive approach to nondeterministic fault localization has been proposed by Katzela *et al.* [28]. The fault propagation algorithm in [28] uses a dependency graph as a fault model and exploits the maximum mutual dependency heuristics to isolate multiple simultaneous faults. This heuristic relies on the assumption that of two sets of faults which explain all the observed symptoms, the better solution is the one that exhibits the higher mutual dependency among its members. We believe that in communication systems in which most faults are independent of one another, this assumption does not hold. In addition to this problem, the approach presented in [28] does not allow lost or spurious symptoms. Other limitations are that correlation is window based, and only one failure mode per object is allowed.

Kliger *et al.* [29] propose a probabilistic model to be used with the codebook approach [54], which creates a symptom-fault dictionary (called a codebook) by reducing the fault propagation model to a bipartite graph using serial and parallel reduction operators. In a nondeterministic case, this reduction is an NP-hard problem on its own. In addition, the reduction to a bipartite graph loses some dependency information that may be useful in fault localization. More importantly, no nondeterministic decoding schema is proposed in [29]. The algorithms proposed in this paper (Algorithms 2 and 3) can be used for this purpose. These algorithms do not require graph reduction and therefore take advantage of all the available dependency information. In addition, multiple simultaneous faults may be detected in an event-driven manner.

The literature on event correlation contains reports of applying belief networks to fault diagnosis. However, the approaches are limited to rather specific fault diagnosis problems, which use simplified belief network models. In fault diagnosis in linear light-wave networks [15] and in diagnosing connectivity problems in communication systems [52] conditional probabilities are 0-, 1-valued. Bayesian reasoning used by Smyth [45] to monitor and diagnose the state of an antenna is applicable to the management of a system with a small number of possible states. As a result, it cannot be used in a system with a big number of (possibly multistate) components. To trouble shoot printing services a tree-shaped belief network is used [20]. Hood *et al.* [23] reported an application of Bayesian network theory to proactive fault detection. The belief network used there is also tree-shaped based on the structure of SNMP [6] MIBs [33]. A bipartite belief network is used in [7] to pinpoint LAN segments suspected of having a particular fault. The reasoning mechanism used in [7] is not able to precisely identify the fault (e.g., the malfunctioning node). Moreover, it is based on observations that are particular to the LAN environment and are not applicable in general. The techniques proposed in [7], [15], [20], and [52] derive heuristics that approximate Bayesian reasoning in the particular fault localization problems they attempt to solve. The solutions proposed in [7], [15], [20], and [52] do not allow lost or spurious symptoms, and some of them are window based [7], [15], [52]. In contrast, this paper adapts techniques of Bayesian reasoning in belief networks of

arbitrary shape to provide a single event-driven solution to a wide range of problems.

Other approaches to dealing with uncertainty in network fault diagnosis include an application of Dempster–Shafer theory to detect break faults in communications networks [11]. Similarly to [52], this technique is tailored specifically to diagnosing connectivity problems in networks with known and static topologies. This solution would not be sufficient for the purpose of diagnosing performance problems or when the knowledge of the network topology is uncertain or incomplete.

Statistical data analysis methods were used for nondeterministic fault diagnosis in bipartite-graphs in [16]. The solution is proposed to detect link failures in wireless and/or battlefield networks based on the observed set of broken end-to-end connections focusing on unknown and constantly changing network topologies. This technique [16] is not able to deal with lost or spurious symptoms and it is window based.

The hierarchical modeling of uncertainty with belief networks presented in this paper is consistent with other work on constructing diagnostic models [46]. Contrary to the approach in [46], in our model, each component may have more than one output, which allows representation of different types of influences caused by one component on its dependent components.

VIII. Conclusion and Future Work

This paper investigates an application of Bayesian reasoning using belief networks [37] to nondeterministic fault diagnosis in complex communication systems. We propose a belief network as a representation of causal relationships among system events and show that the fault localization problem may be solved by calculating the MPE query in belief networks. Then, we investigate an application of known algorithms for performing Bayesian reasoning in belief networks to fault localization. We directly apply the bucket elimination framework [12] and adapt two algorithms for calculating queries in singly connected belief networks (polytrees): 1) belief updating and 2) MPE [37].

We conclude that, due to its exponential computational complexity, the exact Bayesian inference is not feasible in the application to fault localization. However, as revealed by the simulation study on end-to-end service failure diagnosis, the approximate techniques offer much better (polynomial) performance while retaining almost optimal accuracy. The approximate techniques proposed in this paper meet all or most of the objectives of our research. We, therefore, conclude that belief networks are a promising model for nondeterministic fault localization.

The performance of fault localization using belief networks could be improved by applying hierarchical reasoning, in which the fault localization is performed first using a high-level model to pinpoint the location of the problem, and then a more detailed model would be used to identify the precise location of the fault within the previously pinpointed location. The theoretical foundation for such an approach has been laid in [46]. Since the high-level model of the entire system and a detailed model of the pinpointed location are both smaller in size than a single detailed system model, the hierarchical reasoning reduces the complexity of the fault localization task. This paper implicitly advocates the usage of hierarchical reasoning by presenting a case study on the analysis of end-to-end service failures, which is an example of a high-level fault localization problem. The services identified as faulty during end-to-end service diagnosis need to be analyzed on the detailed level to precisely determine the root causes.

In the area of end-to-end fault diagnosis, the future research will investigate distributed fault localization techniques in which multiple fault localization applications cooperate to isolate the causes of end-to-end service failures. Such a distributed solution should explore domain semantics of typical communication networks [47]. We will also address end-to-end service failure diagnosis in situations where the diagnostic model is difficult to build and investigate fault localization with incomplete and ambiguous dependency information.[1]

References

[1] JavaBayes Version 0.346 [Online]. Available: http://www-2.cs.cmu.edu/~javabayes/Home/

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. Int. Commun. Conf.*, 1993, pp. 1064–1070.

[3] A. Bierman and K. Jones, "Physical topology MIB," IETF Network Working Group, Paper RFC 2922, 2000.

[4] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz, "Topology discovery in heterogeneous IP networks," in *Proc. IEEE INFOCOM*, 2000, pp. 265–274.

[5] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protovol (SNMP)," IETF Network Working Group, Paper RFC 1157, 1990.

[6] J. D. Case, K. McCloghrie, M. T. Rose, and S. Waldbusser, "Protocol operations for Version 2 of the simple network management protocol (SNMPv2)," IETF Network Working Group, Paper RFC 1905, 1996.

[7] C. S. Chao, D. L. Yang, and A. C. Liu, "An automated fault diagnosis system using hierarchical reasoning and alarm correlation," *J. Network Syst. Manage.*, vol. 9, no. 2, pp. 183–202, 2001.

[8] G. F. Cooper, "Probabilistic inference using belief networks is NP-hard," Stanford Univ., Stanford, CA, Tech. Rep. KSL-87-27, 1988.

[9] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter, "Probabilistic networks and expert systems," in *Statistics for Engineering and Information Science*. New York: Springer-Verlag, 1999.

[10] P. Dagum and M. Luby, "Approximately probabilistic reasoning in Bayesian belief networks is NP-hard," *Artificial Intell.*, pp. 141–153, 1993.

[11] N. Dawes, J. Aloft, and B. Pagurek, "Network diagnosis by reasoning in uncertain nested evidence spaces," *IEEE Trans. Commun.*, vol. 43, pp. 466–476, 1995.

[12] R. Dechter, "Bucket elimination: A unifying framework for probabilistic inference," in *Uncertainty in Artificial Intelligence*, E. Horvitz and F. V. Jensen, Eds. San Mateo, CA: Morgan Kaufmann, 1996, pp. 211–219.

[13] R. Dechter and I. Rish, "A scheme for approximating probabilistic inference," in *Uncertainty in Artificial Intelligence*, D. Geiger and P. Shenoy, Eds. San Mateo, CA: Morgan Kaufmann, 1997, pp. 132–141.

[14] E. Decker, P. Langille, A. Rijsinghani, and K. McCloghrie, "Definition of managed objects for bridges," IETF Network Working Group, Paper RFC 1493, 1993.

[15] R. H. Deng, A. A. Lazar, and W. Wang, "A probabilistic approach to fault diagnosis in linear lightwave networks," in *Integrated Network Management III*, H. G. Hegering and Y. Yemini, Eds. Amsterdam, The Netherlands, 1993, pp. 697–708.

[16] M. Fecko and M. Steinder, "Combinatorial designs in multiple faults localization for battlefield networks," in *IEEE Military Commun. Conf. (MILCOM)*, McLean, VA, 2001.

[17] R. Gopal, "Layered model for supporting fault isolation and recovery," in *Proc. Network Operation Management Symp.*, J. W. Hong and R. Weihmayer, Eds., Honolulu, HI, Apr. 2000, pp. 729–742.

[18] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *Proc. IEEE INFOCOM*, 2000, pp. 1371–1380.

[1]The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

[19] M. Hasan, B. Sugla, and R. Viswanathan, "A conceptual framework for network management event correlation and filtering systems," in *Integrated Network Manage. VI*, M. Sloman, S. Mazumdar, and E. Lupu, Eds., 1999, pp. 233–246.
[20] D. Heckerman and M. P. Wellman, "Bayesian networks," *Commun. ACM*, vol. 38, no. 3, pp. 27–30, Mar. 1995.
[21] H. G. Hegering and Y. Yemini, Eds., *Integrated Network Management III* Amsterdam, The Netherlands, 1993.
[22] *Proc. Network Operation Management Symp.*, J. W. Hong and R. Weihmayer, Eds., Honolulu, HI, Apr. 2000.
[23] C. S. Hood and C. Ji, "Proactive network management," in *Proc. IEEE INFOCOM*, Kobe, Japan, 1997, pp. 1147–1155.
[24] G. Jakobson and M. D. Weissman, "Alarm correlation," *IEEE Trans. Networking*, vol. 7, pp. 52–59, Nov. 1993.
[25] J. F. Jordaan and M. E. Paterok, "Event correlation in heterogeneous networks using the OSI management framework," in *Integrated Network Mangament III*, H. G. Hegering and Y. Yemini, Eds. Amsterdam, The Netherlands, 1993, pp. 683–695.
[26] G. Kar, A. Keller, and S. Calo, "Managing application services over service provider networks," in *Proc. Network Operation Management Symp.*, J. W. Hong and R. Weihmayer, Eds., Honolulu, HI, Apr. 2000.
[27] S. Kätker, "A modeling framework for integrated distributed systems fault management," in *Proc. IFIP/IEEE Int. Conf. Distributed Platforms*, C. Popien, Ed., Dresden, Germany, 1996, pp. 187–198.
[28] I. Katzela and M. Schwartz, "Schemes for fault identification in communication networks," *IEEE Trans. Networking*, vol. 3, pp. 733–764, Nov. 1995.
[29] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo, "A coding approach to event correlation," in *Integrated Network Management IV*, A. S. Sethi, F. Faure-Vincent, and Y. Raynaud, Eds. London, U.K.: Chapman and Hall, 1995, pp. 266–277.
[30] L. Lewis, "A case-based reasoning approach to the resolution of faults in communications networks," in *Integrated Network Management III*, H. G. Hegering and Y. Yemini, Eds. Amsterdam, The Netherlands: North-Holland, 1993, pp. 671–681.
[31] G. Liu, A. K. Mok, and E. J. Yang, "Composite events for network event correlation," in *Integrated Network Management VI*, M. Sloman, S. Mazumdar, and E. Lupu, Eds. Amsterdam, The Netherlands: North-Holland, 1999, pp. 247–260.
[32] B. Lowecamp, D. R. O'Hallaron, and T. R. Gross, "Topology discovery for large ethernet networks," in *Proc. ACM SIGCOMM*, 2001, pp. 239–248.
[33] K. McCloghrie and M. Rose, "Management information base for network mangement of TCP/IP-based internets: MIB-II," IETF Network Working Group, Paper RFC 1213, 1991.
[34] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of Pearl's "belief propagation algorithm," *J. Select. Areas Commun.*, vol. 16, pp. 140–151, Feb. 1998.
[35] M. Novaes, "Beacon: A hierarchical network topology monitoring system based in IP multicast," in *Services Management in Intelligent Networks*, A. Ambler, S. B. Calo, and G. Kar, Eds. New York: Springer-Verlag, 2000, vol. 1960, Lecture Notes in Computer Science, pp. 169–180.
[36] Y. A. Nygate, "Event correlation using rule and object based techniques," in *Integrated Network Management IV*, A. S. Sethi, F. Faure-Vincent, and Y. Raynaud, Eds. London, U.K.: Chapman and Hall, 1995, pp. 278–289.
[37] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.
[38] R. Perlman, *Interconnections, Second Edition: Bridges, Routers, Switches, and Internetworking Protocols*. New York: Addison Wesley, 1999.
[39] S. Ramanathan, D. Caswell, and S. Neal, "Auto-discovery capabilities for service management: An ISP case study," *J. Network Syst. Manage.*, vol. 8, no. 4, pp. 457–482, 2000.
[40] I. Rish and R. Dechter, "On the impact of causal independence," Dept. Information Computer Science, Univ. California, Irvine, 1998.
[41] S. H. Schwartz and D. Zager, "Value-oriented network management," in *Proc. Network Operation Management Symp.*, J. W. Hong and R. Weihmayer, Eds., Honolulu, HI, Apr. 2000.
[42] A. S. Sethi, F. Faure-Vincent, and Y. Raynaud, Eds., *Integrated Network Management IV*. London, U.K.: Chapman and Hall, 1995.
[43] R. Siamwalla, R. Sharma, and S. Keshav, "Discovering internet topology," Cornell Univ., Ithaca, NY, 1998.
[44] M. Sloman, S. Mazumdar, and E. Lupu, Eds., *Integrated Network Management VI*, 1999.
[45] P. Smyth, "Markov monitoring with unknown states," *J. Select. Areas Commun.*, vol. 12, pp. 1600–1612, Dec. 1994.
[46] S. Srinivas, "Building diagnostic models from functional schematics," Knowledge Systems Laboratory, Dept. Computer Science, Stanford Univ., Stanford, CA, Tech. Rep. KSL 95-15, 1994.
[47] M. Steinder and A. S. Sethi, "Distributed fault localization in hierarchically routed networks," in *13th Int. Workshop Distributed Systems: Operations and Management*, vol. 2506, Lecture Notes Computer Science, M. Feridun, P. Kropf, and G. Babin, Eds., Montréal, Canada, Oct. 2002, pp. 195–207.
[48] ——, "Increasing robustness of fault localization through analysis of lost, spurious, and positive symptoms," in *Proc. IEEE INFOCOM*, New York, 2002.
[49] ——, (2002) Non-Deterministic fault localization in communication systems using belief networks. CIS Dept., Univ. Delaware. [Online]. Available: http://www.cis.udel.edu/~steinder/PAPERS/TR-2003–03.pdf
[50] ——, "End-to-end service failure diagnosis using belief networks," in *Proc. Network Operation Management Symp.*, R. Stadler and M. Ulema, Eds., Florence, Italy, Apr. 2002, pp. 375–390.
[51] C. Wang and M. Schwartz, "Fault detection with multiple observers," *IEEE Trans. Networking*, vol. 1, pp. 48–55, Feb. 1993.
[52] ——, "Identification of faulty links in dynamic-routed networks," *J. Select. Areas Commun.*, vol. 11, pp. 1449–1460, Dec. 1993.
[53] P. Wu, R. Bhatnagar, L. Epshtein, M. Bhandaru, and Z. Shi, "Alarm correlation engine (ACE)," in *Proc. Network Operation Management Symp.*, New Orleans, LA, 1998, pp. 733–742.
[54] S. A. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie, "High speed and robust event correlation," *IEEE Commun. Mag.*, vol. 34, pp. 82–90, May 1996.

Małgorzata Steinder (M'99) received the M.S. degree in computer science from AGH University of Science and Technology, Poland, in 1994 and the Ph.D. degree in computer and information sciences from the University of Delaware, Newark, in 2003.

She was a Junior Faculty Member at AGH from 1994 to 1998. In 2003, she joined the Service Management Middleware Department, IBM T. J. Watson Research Center, Hawthorne, NY, as a Research Staff Member. She is currently working on dynamic resource management for the WebSphere on-demand operating environment, leading the effort in autonomic server and application provisioning.

Dr. Steinder received the Allan P. Colburn Prize for the Outstanding Doctoral Dissertation in Mathematical Sciences and Engineering from University of Delaware for her work on probabilistic fault localization techniques. She also serves as a TPC member for IEEE INFOCOM.

Adarshpal S. Sethi (M'85) received the M.S. degree in electrical engineering and the Ph.D. degree in computer science, both from the Indian Institute of Technology, Kanpur, India.

He is a Professor in the Department of Computer Information Sciences, University of Delaware, Newark. He has served on the faculty at IIT Kanpur, was a Visiting Faculty Member at Washington State University, Pullman, and a Visiting Scientist at IBM Research Laboratories, Zurich, Switzerland, as well as the U.S. Army Research Laboratory, Aberdeen, MD. His research interests include architectures and protocols for network management, fault management, quality-of-service and resource management, and management of wireless networks.

Dr. Sethi is on the Editorial Advisory Board for the *Journal of Network and Systems Management*, an Editor for the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, and an Associate Editor for the *Electronic Commerce Research Journal*. He was Co-Chair of the Program Committee for ISINM '95 and was the General and Program Chair for DSOM '98. He has also been on the program committees of numerous conferences.